Object-Oriented Parallel Software for Radio Wave Propagation Simulation in Urban Environment

Frédéric Guidec, Patrice Calégari, Pierre Kuonen

Swiss Federal Institute of Technology Lausanne
Theoretical Computer Science Laboratory
CH-1015 Lausanne (Switzerland)
E-mail: {guidec,calegari,kuonen}@di.epfl.ch

1 Introduction

Because of the rapid growth of cellular phone networks, radio wave propagation simulation is of great interest to telecommunication operators, for it makes it possible to predict the shape of the cells of any potential future networks. The objective of the project STORMS[i] is to develop a software tool to be used for the design and the planning of the future UMTS (Universal Mobile Telecommunication System) network. This software tool, which is planned to be used mostly interactively, shall include radio wave propagation simulation algorithms for both urban and rural environments. Since a single cellular network may consist of thousands of cells, radio wave propagation simulation must be as fast as possible.

The ParFlow method permits the simulation of outdoor radio wave propagation in urban environment, describing the physical system in terms of the motion of fictitious microscopic particles over a lattice. It is appropriate for simulating radio wave propagation when fixed antennas are placed below rooftops, as is the case in urban networks composed of micro-cells.

This paper reports the design and the implementation of ParFlow++, an object-oriented, irregular implementation of the ParFlow method targeted at distributed memory parallel platforms. To date the use of object-oriented programming is not very common in parallel supercomputing. This is the reason why the parallel implementation of the ParFlow method using object-oriented techniques appeared to us as an appealing challenge.

2 The ParFlow Method

The ParFlow method was designed at the University of Geneva by Chopard, Luthi and Wagen [4]. It compares with the so-called Lattice Boltzman Model, that describes a physical system in terms of motion of fictitious microscopic particles over a lattice [1]. According to the Huygens principle, a wave front consists of a number of spherical wavelets emitted by secondary radiators. The

[i] Software Tools for the Optimization of Resources in Mobile Systems.

ParFlow method is based on a discrete formulation of this principle. Space and time are represented in terms of finite elementary units Δr and Δt, related by the velocity of light C_0[ii]. Space is modeled by a grid with a mesh size of length Δr, and flow values are defined on the edges connecting neighbouring grid points. The flows entering a grid point at time t are scattered at time $t + \Delta t$ among the four neighbouring points. Because of the discretization of time, it is convenient to distinguish between the flows coming into a grid point, and those going out of this point. Outgoing flows at time t are a linear combination of incoming flows at that time, and an incoming flow at time t corresponds to the outgoing flow calculated on a neighbouring grid point at time $t - \Delta t$. These rules apply for all grid points but those modelling the source (the transmitter) and obstacles. The source point radiates a signal through its four outgoing flows, but it does not propagate incoming flows. Obstacles (typically, city buildings) are modelled by two kinds of grid points: wall points, and indoor points (see Figure 1). As, in the current ParFlow model, it is assumed that radio waves do not penetrate buildings, indoor points are not involved in the computation. Wall points are perfectly reflecting points that return any incident wave with opposite sign, and whose reflection coefficient and matrix elements can be modified in order to model different kinds of walls [5].

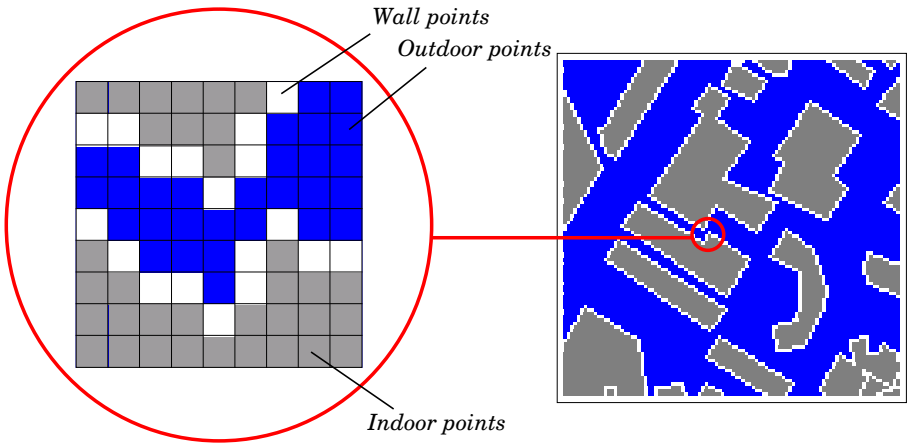

Fig. 1. The ParFlow method operates on a city district described as a bitmap that distinguishes between indoor points, outdoor points, and wall points.

[ii] $\Delta t = \frac{\Delta r}{C_0\sqrt{2}}$.

3 Design and Implementation

3.1 Object-Oriented Irregular Application

Since the ParFlow method does not simulate radio wave propagation through buildings, it can be most interesting not to model indoor points. Indeed, experience shows that when modelling an urban area, buildings can represent up to 30 % of the surface considered. Not modelling indoor points avoids a waste of memory space and a waste of computational power. Such an approach inevitably leads to the creation and the management of an irregular data structure. As they provide powerful features to describe and to manipulate complex, irregular data structures, object-oriented languages are perfectly suited to an irregular implementation of the ParFlow method.

Design and Object-Oriented Implementation ParFlow++ was developed in C++, according to the fundamental principles of software engineering. The preliminary analysis and design of the application were achieved using the Fusion method [2]. The main classes and the relationships that were identified during the analysis phase constitute an object model, that is partially reproduced in Figure 2.

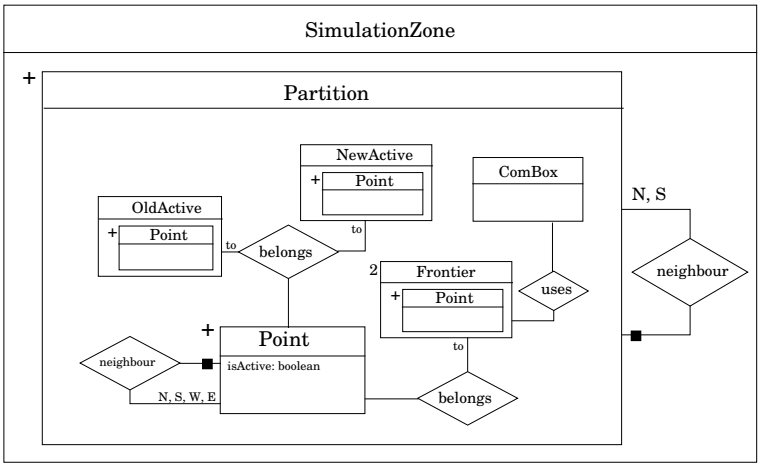

Fig. 2. Fusion object model of ParFlow++.

All kinds of non-indoor points are described in a hierarchy of C++ classes. Instances of these classes can be assembled at runtime so as to constitute an irregular structure that preserves the neighbourhood relationships.

In ParFlow the amount of computation required during a simulation step is not constant. Because of the discretization of time and space in the simulation process, a wave radiated by the source point propagates step by step throughout the simulation zone. ParFlow++ was designed in such a way that the amount of points implied during any computation step is always kept at a minimum. At any time, a point that has not been reached by the wave yet is said to be inactive (initially, all points are inactive but the source point). Points that have just been reached by the wave (during the current simulation step) are said to be *new* active points, whereas those that were reached during former simulation steps are referred to as *old* active points.

At each step of the simulation flow values need only be calculated for active points, and the propagation of outgoing flows is only required from active points to their neighbours. To achieve these goals ParFlow++ manages several structures internally. Every newly activated point is inserted in a list *newActive*, whose role is to permit an efficient propagation of the activity status after each computation step: the only points that are in a position to be activated are those that are neighbours to members of the *newActive* list, and that are still inactive. Iterating through members of the *newActive* list facilitates the identification of new active points. Once a member of *newActive* has activated its neighbours, it is transferred into another list *oldActive*. Hence, this point will never be considered again when looking for new active points, although it will still participate in the computation.

Figure 3 confirms the advantage of activating points dynamically. It shows how the workload per simulation step increases as the radio wave propagates and covers a growing surface, and how it reaches a ceiling when the simulation zone is completely covered.

Figure 3 also confirms the advantage of using a data structure that does not model indoor points. The speed at which the workload increases depends on the amount of obstacles met by the radio wave during its propagation, and the maximal workload depends on the amount of outdoor points in the simulation zone.

3.2 Data Parallel Implementation

A major advantage of the ParFlow method is that, although the calculations made during each iteration step are theoretically synchronous, updates of points require independent computation. The ParFlow algorithm is therefore a good candidate to parallel implementation.

The simulation zone is split in horizontal bands, which are then allocated to processors based on a round-robin policy, as shown in Figure 4.

In ParFlow++ a partition basically consists of a structured collection of points and of the two lists that are used to maintain references to 'new' and 'old' active points (see Section 3.1). Each partition also manages two 'frontiers' internally. A frontier can be perceived as a collection of references to points that are either on the northern edge or on the southern edge of a partition. The behaviour of frontier points differs slightly from that of other points, for they

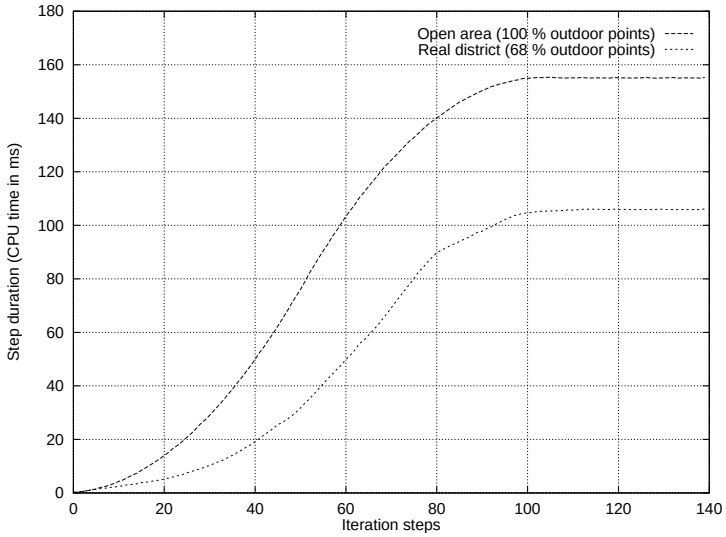

Fig. 3. Evolution of the workload per simulation step during a sequential simulation on a 100×100 points area. These results show the workload evolution observed when achieving a simulation on an open area (upper curve), and on a real district of the city of Geneva (lower curve). The CPU times reported were measured on one processor of a Cray T3D supercomputer.

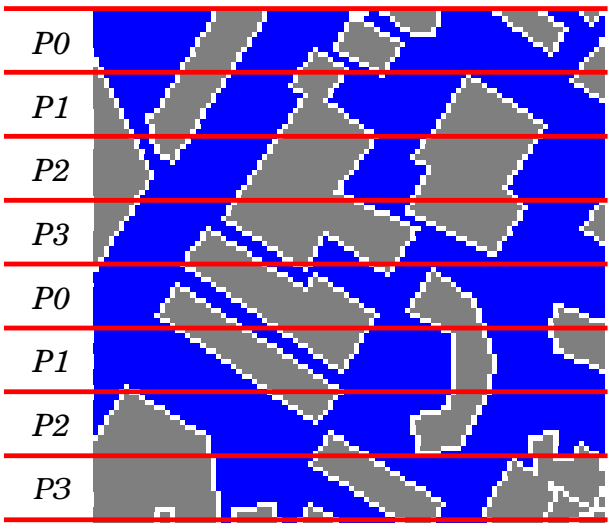

Fig. 4. Simulation zone partitioning and mapping.

have to interact with 'neighbouring' points that are actually stored on remote processors. A frontier thus ensures the propagation of flows between a partition and one of its neighbours. It also ensures the propagation of the activity status, since a newly active point located on a frontier may activate a point in the opposite frontier of a neighbouring partition.

4 Experimental Results

ParFlow++ was compiled on a Cray T3D. We ran a 800 step radio wave propagation simulation on a 500x500 point zone modelling a 1 km^2 district of the city of Geneva. Figure 5 shows the resulting path-loss map.

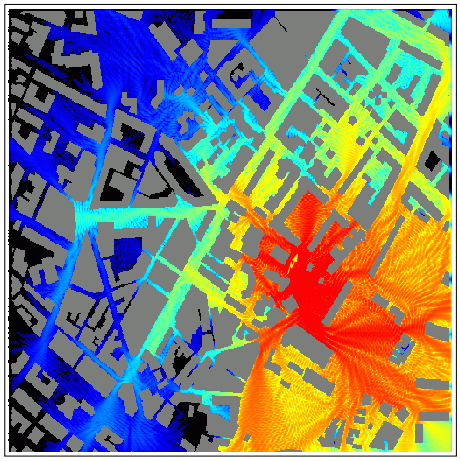

Fig. 5. Results of a radio wave propagation simulation on a district of the city of Geneva. The broken arrow shows the position of the emitter. The computation of such a figure takes about 18 minutes on a single processor of the Cray T3D.

On the Cray T3D we also measured the speedups observed for various simulation zone sizes. Figure 6 confirms the scalability of the parallel implementation. (For the 1024×1024 and 2048×2048 zones, the simulation was not possible on a single processor because of memory limitation, so we had to estimate the sequential reference times).

5 Related Work

The performances reported in the former section are quite satisfactory. Yet experiments have shown that partitions dimensioning is a crucial issue. In the future we would like to experiment with an heterogeneous partitioning policy. Since many bands are required near the source point in order to give better workload

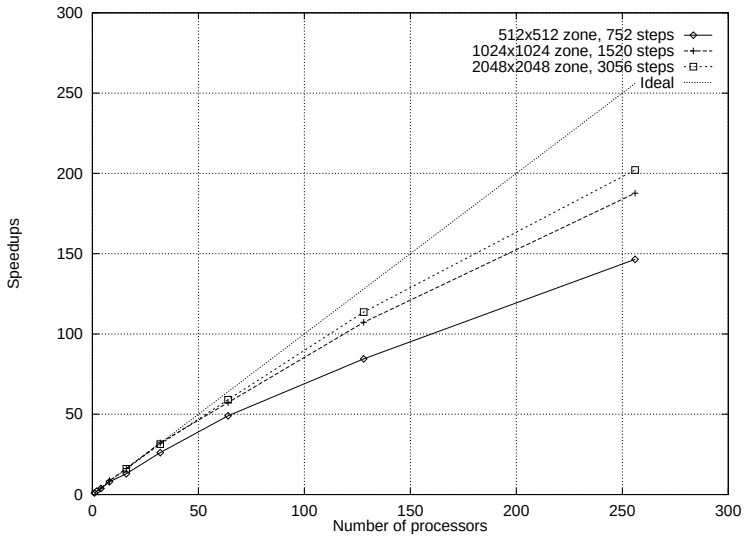

Fig. 6. Speedups observed on a Cray T3D, when achieving simulations for various district sizes.

balancing, and since too many bands lead to a degradation of the communication performances, a compromise would be to allocate thinner bands near the source. Some work has been achieved to build a mathematical model of the workload and of communications in ParFlow++[iii]. This model should help select the best partitioning policy for any simulation problem case and any target platform.

6 Conclusion

In this paper we have reported the development of ParFlow++, a new parallel object-oriented piece of software that permits the prediction of radio wave propagation in urban environments, based on a bidimensional simulation over a digital city map.

The main originality of ParFlow++ with respect to former implementations of the ParFlow method [5] is that it was implemented as an irregular application. Since the method does not allow for radio waves to propagate through buildings, ParFlow++ does not model indoor points. Experience shows that this approach permits a significant reduction of computation times.

ParFlow++ was designed in an object-oriented way, and implemented in C++ for MIMD-DM[iv] platforms. It was developed so as to be highly portable, and its current implementation relies on the PVM library [3]. Although this

[iii] *'Performance Analysis of a Parallel Program for Wave Propagation Simulation'*, paper also published in these proceedings.

[iv] Multi Instruction stream, Multi Data stream, Distributed Memory

implementation can still be improved in many ways, experiments achieved on the Cray T3D at the Swiss Federal Institute of Technology show the scalibility of the code. The results also demonstrate that an object-oriented irregular implementation does not necessarily lead to poor performances on MIMD-DM platforms.

The object-orientation of ParFlow++ brings in much versatility. The set of classes that constitute its source code could easily be extended, or reused in another context. For example, in the near future ParFlow++ will be modified so as to simulate tri-dimensional radio wave propagation. Its classes should also serve as basic building blocks to develop software tools capable of simulating other physical phenomena, such as fluid dynamics or reaction-diffusion processes.

Acknowledgments

The STORMS project is a European ACTS project, funded by the European Community and by the Swiss government (OFES grant). The Cray T3D experimentations were conducted on the machine of the Swiss Federal Institute of Technology in Lausanne.

References

1. R. Benzi, S. Succi, and M. Vergassola. The Lattice Boltzmann Equation: Theory and Applications. *Physics Reports*, 222(3):145–197, 1992.
2. D. Coleman and al. *Object-Oriented Development - The Fusion Method.* Prentice Hall Object-Oriented Series, Englewood Cliffs, NJ, 1995. ISBN0-13-338823-9.
3. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. MIT Press, 1994.
4. P. O. Luthi and B. Chopard. Wave Propagation with Transmission Line Matrix. Technical report, University of Geneva and Swiss Telecom PTT, 1994.
5. P. O. Luthi, B. Chopard, and J.-F. Wagen. Wave Propagation in Urban Micro-cells: a Massively Parallel Approach using the TLM Method. In *Proceeding of PARA'95, Workshop on Applied Parallel Scientific Computing, Copenhagen*, aug 1995. Also in COST 231 TD(95) 33.