Radio Wave Propagation Simulation on the Cray T3D

Frédéric Guidec, Pierre Kuonen and Patrice Calégari

Swiss Federal Institute of Technology,
Computer Science Theory Laboratory,
CH-1015 Lausanne, Switzerland.
E-Mail: {guidec|kuonen|calegari}@di.epfl.ch

The ParFlow method permits the simulation of outdoor wave propagation in urban environment, describing the physical system in terms of the motion of fictitious microscopic particles over a lattice. This paper begins with a brief introduction to the ParFlow method. It then reports the design, the implementation in C++, and the experimentation on a Cray T3D of ParFlow++, an object-oriented parallel irregular implementation of the ParFlow method, primarily targeted at MIMD-DM platforms.

1. INTRODUCTION

Computer-based simulation of radio wave propagation is of great interest to telecommunication operators, since it makes it possible to predict the geographical areas covered by any potential radio network. The objective of the European project STORMS (Software Tools for the Optimization of Resources in Mobile Systems) is to develop a software tool to be used for the design and the planning of the future Universal Mobile Telecommunication System (UMTS). This software tool, which is planned to be used mostly interactively, shall include fast radio wave propagation simulation algorithms for both urban and rural environments.

This paper reports the development of ParFlow++, a parallel irregular implementation of the ParFlow method targeted at MIMD-DM platforms (Multiple Instruction stream, Multiple Data stream, Distributed Memory), and its experimentation on a Cray T3D. The ParFlow method was designed at the University of Geneva by Chopard, Luthi and Wagen [4,5]. According to the principle of Huygens, a wave front consists of a number of spherical wavelets emitted by secondary radiators. The 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:

$$\Delta t = \frac{\Delta r}{C_0\sqrt{2}}$$

Space is modeled by a grid with a mesh size of length Δr, and flow values are defined on the edges connecting neighboring grid points. The flows entering a grid point at time t are scattered at time $t + \Delta t$ among the four neighboring points. This model, characterized by

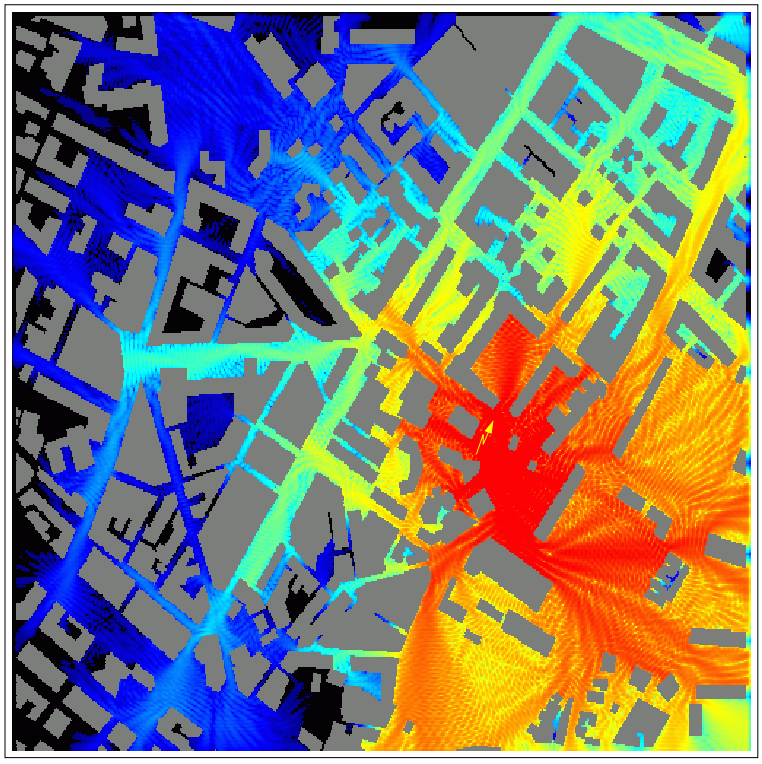


Figure 1. Results of a radio wave propagation simulation on a district of the city of Geneva. The broken arrow shows the position of the transmitter.

a simultaneous independent dynamics and by a very simple numerical scheme, suggests a cellular automaton based implementation.

The ParFlow method thus permits 2D radio wave propagation simulations, using a digital city map, and assuming infinite building height (further information can be found in [3,4]). Figure 1 shows the path-loss map obtained after running a wave propagation simulation on a 500×500 point simulation zone modeling a 1×1 km^2 district of the city of Geneva.

2. IRREGULAR IMPLEMENTATION

In the ParFlow model wall points are perfectly reflecting points that return any incident wave with opposite sign. Consequently the model does not allow for radio wave propagation through buildings. It can thus be most convenient not to process indoor points. Indeed, when modeling an urban area, buildings can represent up to 30 % of the surface considered. Excluding indoor points from the model helps saving memory space, as well as computational power. However, such an approach inevitably leads to the creation and the management of an irregular data structure. This is the reason why ParFlow++ was developed in C++.

All kinds of non-indoor points (namely, outdoor points, wall points, and the source

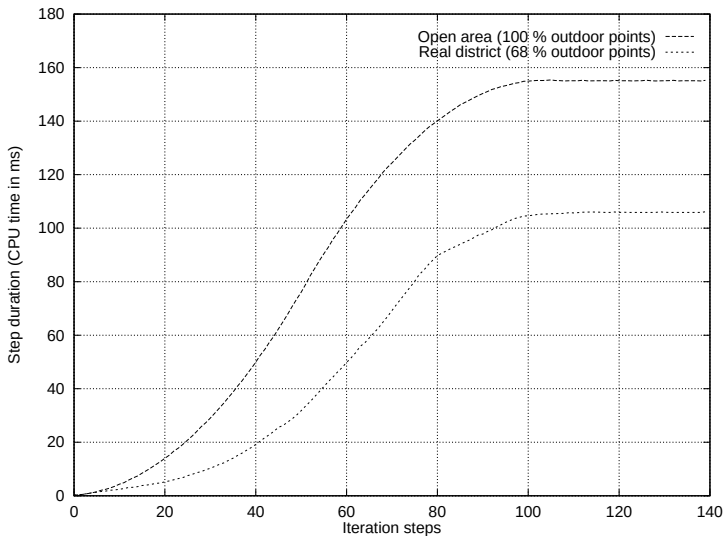


Figure 2. Evolution of the workload per simulation step during a sequential simulation on a 100×100 point simulation zone. These results show the workload evolution observed when achieving a simulation on an open area (upper curve), and on a real district of the city of Geneva (lower curve). The CPU times reported were measured on one processor of a Cray T3D.

point) are described in a hierarchy of C++ classes. Instances of these classes are assembled at runtime so as to constitute an irregular structure that models only the non- indoor part of the simulation zone considered.

A wave radiated by the source point propagates step by step throughout the simulation zone. Consequently the amount of computation required during a simulation step is not constant. ParFlow++ was designed so as to exploit this characteristic. The amount of points implied during any computation step is always kept at a minimum. At runtime each object modeling a non-indoor grid point is either active or inactive. Initially all points are inactive but the source point. Other points get activated as the wave propagates throughout the simulation zone (see [2] for further details).

Experiments confirm the advantage of activating points dynamically. Figure 2 shows how the workload per simulation step increases as the wave propagates and covers a growing surface, and how it reaches a ceiling as soon as the simulation zone is completely covered. The figure 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 wave during its propagation, and the maximal workload is proportional to the amount of outdoor points in the simulation zone.

3. PARALLEL IMPLEMENTATION

Because of the large amount of outdoor points that must be considered in a simulation (typically, several thousands of outdoor points to model a single city district), an

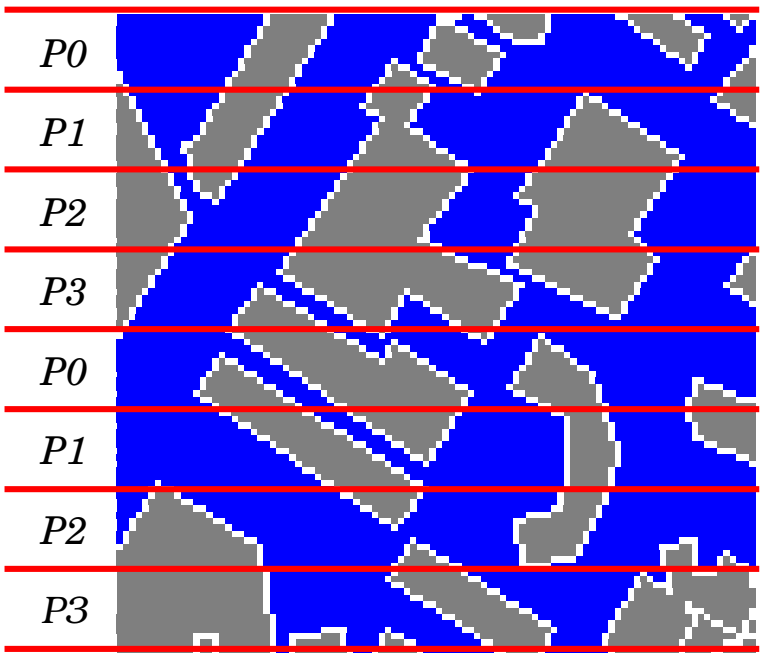


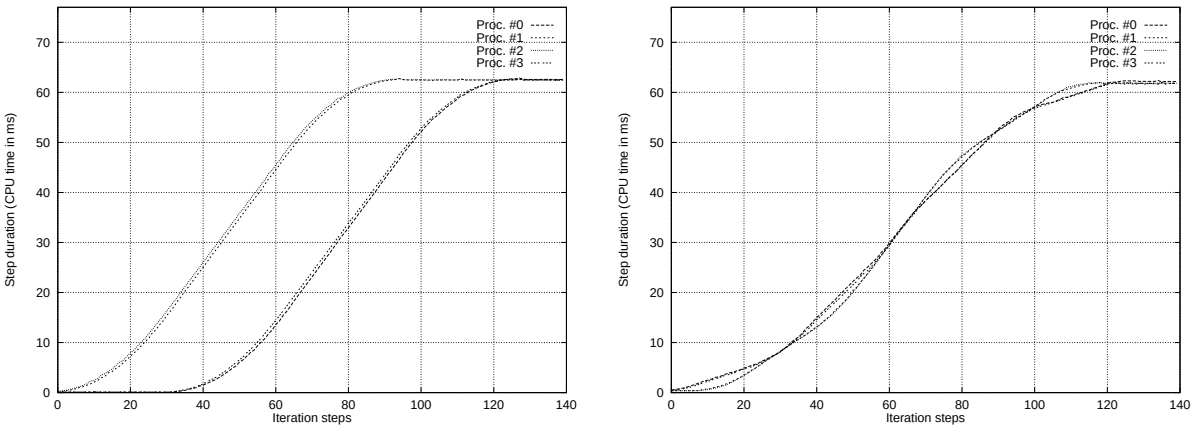
Figure 3. Simulation zone partitioning and mapping on 4 processors.

appropriate policy must be chosen to allocate each point to a processing element of the target platform. Many partitioning policies can be considered for an irregular structure such as that ParFlow++ operates on. Yet, exotic partitioning policies usually require costly mechanisms for locating remote data, and for ensuring efficient data exchanges. This is the reason why the current version of ParFlow++ implements a quite simple data distribution policy. The simulation zone is split in horizontal stripes of equal height. These stripes are then assigned to processors in a round-robin fashion, as shown in Figure 3.

Partitioning the simulation zone in stripes has some advantages. In ParFlow++, mechanisms for data location and message vectorization were readily implemented using facilities of the PVM library [1]. Moreover, as adjacent stripes are allocated to adjacent processors, communications are only required between neighboring processors. However, the irregular structure ParFlow++ operates on, and the fact that the workload is not constant during a simulation (as explained in Section 2), make it more difficult to obtain a good load balancing.

Actually, when partitioning the simulation zone ParFlow++ could allow for the actual distribution of buildings in this zone, hence adjusting the height of each stripe so that all stripes roughly get the same amount of non-indoor points. Yet this is not done in the current implementation, because it is assumed that buildings are distributed homogeneously in the simulation zone. Experiments confirm that this asumption is usually verified.

Although all stripes have roughly the same height in the current implementation of ParFlow++, the number of stripes assigned to each processor –and, as a consequence, their height– can be adjusted easily. Since the wave radiates from a single source point, and



(a) (b)

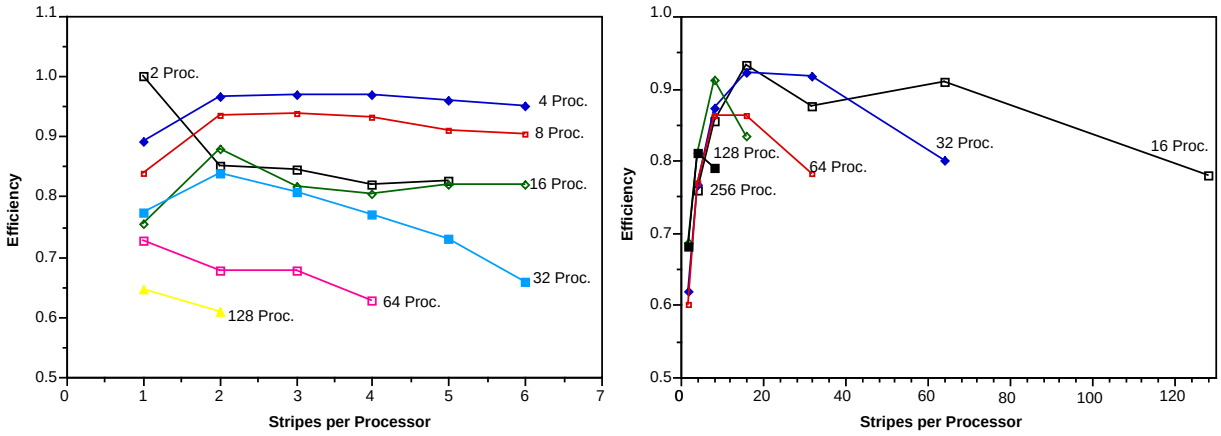
Figure 4. Influence of the number of stripes per processor on the workload per processor, during a simulation achieved on 4 processors of a Cray T3D. The figures show the workload per processor observed when the simulation zone is split (a) in 4 stripes, and (b) in 16 stripes.

since non-indoor points get activated only when they are reached by the wave, partitioning the simulation zone in thin stripes helps ensuring that all processors get their share of work early after the wave started propagating. It is thus possible to obtain a fair distribution of the workload among the processors, simply by adjusting the number of stripes per processor, as confirmed by the results reproduced in Figure 4.

These results were observed when achieving a parallel wave propagation simulation on a 128×128 point 'open' area (no obstacles, source point located in the middle of the simulation zone), on 4 processors of a Cray T3D. They show how the workload per processor depends on the partitioning policy. Figure 4(a) shows the workload observed when the simulation zone was split in 4 stripes (only one stripe per processor), whereas the workload in Figure 4(b) was obtained with 16 stripes (4 stripes per processor). In the latter case the workload increased almost similarly on all processors. These results confirm that partitioning the simulation zone in many thin stripes permits a better balancing of the *computation workload*. Yet, this approach also leads to a greater *communication overhead*, because of the higher number of frontiers between stripes.

This is confirmed in Figure 5, which shows how the number of stripes per processor influences the global performances of the simulation. The figure shows the efficiency observed, for different hardware configurations (from 2 to 256 processors), as a function of the number of stripes per processor, for 512×512 point and 2048×2048 point simulation zones.

Figure 5(a) shows that, for each machine size, there usually exists an optimal partitioning policy. On the one hand, when using only two processors, there is no point in having more than one stripe per processor. Since, in this particular simulation case, the



(a) (b)

Figure 5. Efficiency of parallel simulations on a Cray T3D as a function of the number of stripes per processor. (a) shows the results observed with a 512×512 point simulation zone, and (b) those observed with a 2048×2048 point zone.

source of the wave is located in the middle of the simulation zone, both processors get exactly the same share of work. Partitioning the zone in more than two partitions results in a greater communication overhead, hence a lower efficiency. On the other hand, when using more than two processors it can be interesting to have several stripes per processor (although 2 or 3 stripes usually give the best efficiency). This is not true any more, though, when using more than 32 processors. This is because, when partitioning a 512×512 point zone in more than 32 processors, the height of each stripe then gets so small that the ratio of the computation time over the communication time is not good enough.

The advantage of allocating several stripes per processor can also be observed with the 2048×2048 point simulation zone (Figure 5(b)), except that it can then be interesting to have up to 8 stripes per processor.

Figure 6 shows the speedups we observed when running radio wave propagation simulations on the Cray T3D, for various district sizes. (For the 1024×1024 and 2048×2048 simulation zones, the simulation was not possible on a single processor because of memory limitation, so we had to estimate the sequential reference times).

4. CONCLUSION

ParFlow++ is a new parallel, irregular code that permits the prediction of radio wave propagation in urban environments, based on a two-dimensional simulation over a digital city map. It can be used for simulating radio wave propagation when transmitters are placed below rooftops. This is the case in urban radio networks composed of micro- cells (e.g., cellular phone networks).

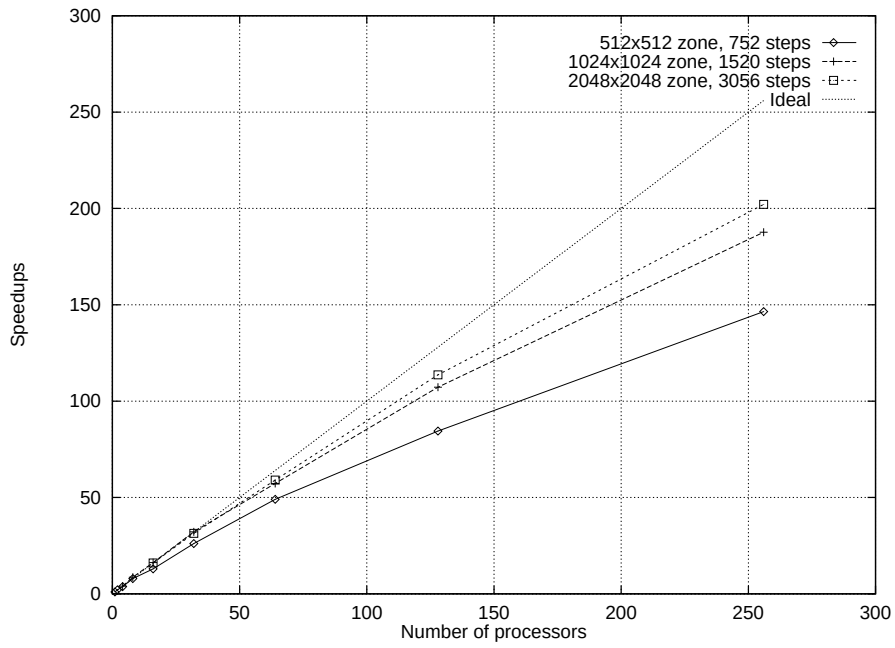


Figure 6. Speedups observed on a Cray T3D, when achieving simulations for various district sizes.

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

Experiments on a Cray T3D confirm the performance of the code: the computation of Figure 1 took about 18 minutes on a single processor of the Cray T3D, which is quite short compared to the computation times required by other urban wave propagation simulation algorithms). They also show the scalability of the implementation, since the efficiency of the parallel simulations discussed in Section 3 was never less than 60 %. Yet these experiments also suggest that finding the 'best' partitioning policy for a given problem case is a real issue. Many parameters must be accounted for, such as the simulation zone size, the location of the radiating source, and the characteristics of the target parallel machine (raw performances, topology, number of processors, etc.). Work is now in progress to build a mathematical model of the workload and of communications in ParFlow++. This model should help select the best partitioning policy for any simulation problem case. We would also like to experiment with a heterogeneous partitioning policy. Since many stripes are required near the source point in order to give better workload balancing, and since too many stripes lead to a degradation of communication performances, a satisfactory compromise must be found for each problem case. Allocating thinner stripes in the area around the source point should be an interesting step towards such a compromise. This shall be investigated in the near future.

Acknowledgements

STORMS (Software Tools for the Optimization of Resources in Mobile Systems) 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. A. Geist and al. *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994.
2. F. Guidec, P. Calégari, and P. Kuonen. Parallel Irregular Software for Wave Propagation Simulation. In *Proceedings of the High-Performance Computing and Networking (HPCN Europe '97) Conference*, April 1997. To be published. Conference to be held in Wien, Austria, April 28-30.
3. W. J. R. Hoeffer. The Transmission-Line Matrix Method: Theory and Applications. *IEEE Transactions on Microwave Theory and Techniques*, 33(100):882–893, oct 1985.
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. Radio Wave Propagation Simulator on Scalable Parallel Computers. In N. Droux, editor, *Proceedings of the SIPAR'95 Workshop, Parallel and Distributed Systems*, pages 63–66. Biel School of Engineering, CS Department, CH-2501 Biel-Bienne, Switzerland, October 1995.