MULTILEVEL PARALLELISM APPLIED TO THE OPTIMIZATION OF MOBILE NETWORKS

P. Kuonen, F. Guidec, P. Calégari
Computer Science Department
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
E-mail: {kuonen, guidec, calegari}@di.epfl.ch

Keywords: Parallel programming, meta-computing, mobile networks.

Abstract: In this paper we present the general architecture of a radio network optimization software we developed. This application greatly uses the concept of multilevel parallelism in order to optimize the use of available parallel computing resources. Details of the implementation of each level are presented as well as the performances achieved on several hardware platforms.

1 INTRODUCTION

Today's parallel programs are typically large, non-interactive, stand-alone pieces of software solving very specific problems. On the contrary, in order to satisfy the commercial market, today's software products need to be adaptable and interactive. Therefore, in order to make modern software products benefit from the performance achieved with parallel programming, one has to make parallel applications more versatile and interactive.

One way to achieve this goal is to use 'multilevel parallelism'. In this article we describe the architecture of the parallel application we are developing for helping engineers deploy mobile telecommunication networks. In order to achieve sufficient performance, and to optimize the use of the available computing resources, this application greatly uses the concept of multilevel parallelism.

2 PARALLELISM IN THE STORMS PROJECT

2.1 Objective

The objective of the European project STORMS[1] is to produce a set of flexible and partially automated tools to aid in the design and the optimization of UMTS[2] networks. One major problem telecommunication companies must face when deploying a UMTS is the *optimization of the radio network*. The problem comes down to finding out the best possible sites for Base Transceiver Stations (BTSs), while guaranteeing that all —or at least a given percentage of the surface of— the area is covered, and that the global cost of the resulting radio network is kept at a minimum. Assuming that a set of potential sites is available, our goal is to select the best subset of sites capable of satisfying the coverage requirements.

2.2 Radio wave propagation simulation

The first operation achieved by the radio network optimization software is the computation of the area that each BTS can cover. A geographical location is said to be *covered* when it can receive the signal broadcast by a BTS with a given quality of service. Considering that a single radio network may consist of hundreds of BTSs, and because the STORMS software tool is planned to be used mostly interactively, radio wave propagation simulations must be as fast as possible. In the STORMS project the radio wave propagation simulation for urban environment is achieved by a software module called ParFlow++ (Guidec, Calégari and Kuonen. 1997). ParFlow++ models the physical system in terms of the motion of fictitious microscopic particles over a lattice. The logical underlying data structure of ParFlow++ is a grid and its computational model is that of a cellular automaton. Therefore, by distributing the data structure among several processors, data-parallelism can be exploited to speed up the computation. A parallel version of ParFlow++ targeted at MIMD-DM[3] platforms was implemented and tested on a cluster of Unix workstations and on a Cray T3D. Details of this implementation are given in Section 4.1.

2.3 Computation of the coverage cells

Once a propagation simulation is complete for a given BTS, the geographical area in which the coverage is acceptable for mobile communication must be delimited. The set of geographical locations

[1] 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).

[2] UMTS: Universal Mobile Telecommunication System.

[3] Multiple Instruction stream, Multiple Data stream, Distributed Memory

covered by a BTS is called the *coverage cell* of the BTS. A threshold value is determined based on different radio quality criteria (predicted traffic demand, etc.). Each grid point where the power of the received signal is bigger than the threshold value belongs to the coverage cell.

Radio wave propagation simulation and coverage cell computation must be achieved for all potential BTSs. Since these operations only depend on the BTS considered, the calculations required for distinct BTSs are independent operations that can be done concurrently. In order to exploit this source of parallelism we implemented parallel versions of the software modules that achieve radio wave propagation simulations and coverage cells computation. These pieces of parallel code are based on the master-slaves model. They are detailed in Section 4.2.

2.4 Creation of an coverage graph

When coverage cells have been computed the next objective of the radio network optimization software is the selection of the best subset of BTSs sites. This problem can be modeled using graph theory (Kuonen, Ubéda and Zerovnik. 1996). The relationship between each pixelized covered location and BTSs is naturally modeled as a bipartite graph whose nodes represent either BTSs or geographical locations (pixels). When many geographic locations must be allowed for, such a graph tends to be huge. A smart way to reduce its size without losing any useful information is to build a bipartite graph whose nodes represent either BTSs or *intercells*. An intercell is defined as a set of geographical locations that are covered by exactly the same potential BTSs (see Figure 1). For each intercell node one only needs to encode the number of geographical locations it contains. The weighted bipartite graph hence obtained is called the *intercell graph*. It can be smaller than the former one by at least one order of magnitude.

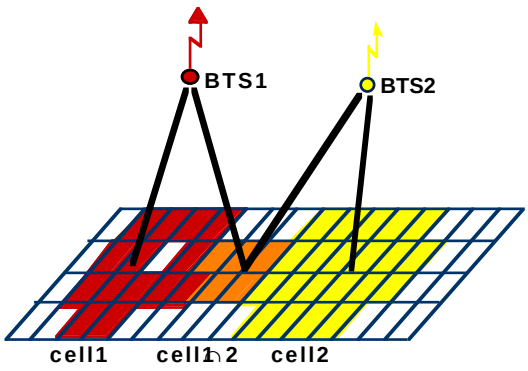

Figure 1: The intercell graph

The drawback of this approach is the computing time required by the algorithm that creates the intercell graph. In order to reduce this time we developed a parallel version of this algorithm. The resulting parallel algorithm is presented in Section 4.3.

2.5 Selection of the best BTS sites

With the graph-based modeling presented in the former section, the problem of finding the best selection of BTSs sites compares to finding a set cover of minimum size. Unfortunately, finding a set cover of minimum size is an NP-complete problem. Hence, unless P=NP, if we want a polynomial time algorithm, we must use an approximation algorithm and look for possibly sub-optimal —yet satisfactory— solutions. To achieve this goal we investigate approaches, which are said to be 'bio-inspired' because they use heuristics that have some analogies with natural or social systems. *ParaGene* (Calégari et al.1998) is a parallel implementation of a genetic algorithm for the selection of the best BTS sites. It uses an island-based approach and runs on a network of workstations. It is presented in Section 4.4.

2.6 Computation chain overview

The sequence of computation phases in the radio network optimization software can be summarized as follows:

1. Radio wave propagation simulation for each potential BTS.
2. Computation of the coverage cell for each potential BTS.
3. Creation of the coverage graph.
4. Selection of the best BTS sites.

These phases can of course be executed sequentially. However it can be noticed that one does not need to wait until all the radio wave propagation simulations have been completed for starting the computation of coverage cells. Actually, as soon as the radio wave propagation simulation of one BTS has been completed one can start the computation of the associated coverage cell, while the radio wave propagation simulation is still in progress for the remaining BTSs. In other words these two operations can be *pipelined*. Likewise, since the module that builds the coverage graph reads coverage cells iteratively, the computation of coverage cells can also be pipelined with the creation of the coverage graph. In addition, as the STORMS tool is planned to be used mostly interactively, images to be displayed on a Graphical User Interface (GUI) are produced so as to show the result of each computation phase. The modules that compute such images were inserted in the pipelined computation chain. Section 4.5 presents how the whole pipeline was implemented in a UNIX environment.

2.7 Graphical user interface

Finally, in order to have a confortable access to the STORMS application, a Java based GUI allowing remote launch of complex computations was developed. As the STORMS application implies several complex and partially independent computations, we are investigating the possibility of using the Java applet-servlet approach for distributing the main computing modules of STORMS on a wide area network of computers.

3 LEVELS OF PARALLELISM IN THE RADIO NETWORK OPTIMIZATION SOFTWARE

At least four levels of parallelism have been identified in the radio network optimization software. Each of them implies different software tools, different algorithms and requires different hardware characteristics.

1. Fine grain to coarse grain parallelism using parallel supercomputers or clusters of workstations (COW). This level applies to radio wave propagation simulation for one BTS and to the creation of the coverage graph.
2. Coarse grain parallelism using tightly coupled networks of workstations (NOW). This level applies to the master-slaves approach for distributing the radio wave propagation simulation and the computation of coverage cells for the BTSs.
3. Coarse grain to large grain parallelism using local area networks of computers. This level applies to the pipelining of the computation chain.
4. Large grain parallelism using loosely coupled wide area networks of computers. This level applies to the distribution of the main components of the application over a wide area network.

Figure 2 illustrates how these four levels of parallelism are exploited in our radio network optimization software.

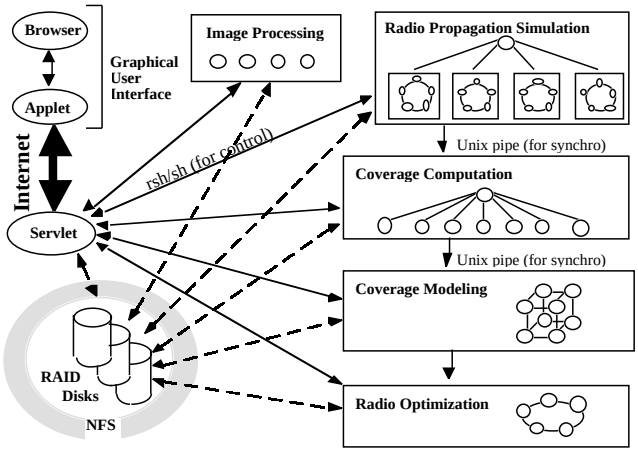

Figure 2: General architecture of the radio network optimization software

4 IMPLEMENTATION DETAILS

4.1 Data-parallel ParFlow++

In our data-parallel implementation of the ParFlow++ software, the 2D grid modeling the geographical area over which the radio wave propagation simulation must be achieved is split in stripes. These stripes are allocated to available processing elements (PEs) based on a round-robin policy. Each PE is thus in charge of part of the simulation. Since the execution model of the ParFlow method is that of a cellular automaton, the simulation is an iterative process. In the data-parallel version of ParFlow++ neighboring stripes must thus exchange information after each iteration step. This makes the parallel software highly communicative. In order to give good performances, an efficient communication library is required, and ParFlow++ must be run on a MIMD supercomputer or a tightly coupled Cluster of Workstations (COW).

Our data-parallel version of ParFlow++ was implemented using the PVM library (Geist et al. 1994). It was experimented on the Cray T3D and SGI Origin 2000 MIMD supercomputers, as well as on a cluster of bi-processor UltraSPARC workstations interconnected through a combination of Fast-Ethernet and FDDI links. On the Cray T3D we could observe an efficiency of about 80% when running ParFlow++ on 256 PEs of the T3D, for real-size problem cases (that is, parallel executions with this configuration were about 200 times shorter than sequential ones). On the SGI Origin 2000 we could only use up to 32 PEs. Yet we observed an equivalent efficiency of parallel executions in that case (performances were less stable, though, because the Origin 2000 we used is a multi-tasking, multi-user platform).

On the cluster of workstations, parallel executions of ParFlow++ showed an efficiency of about 62% on 32 PEs. It is interesting to note that these good performances could not be observed on a more `standard' network of SPARCstations, because the Ethernet trunk could not support the high bandwidth required by ParFlow++. This observation confirms that our data-parallel ParFlow++ requires an efficient, tightly coupled parallel platform in order to give good performances.

4.2 Master-slaves modules

A major characteristic of the master-slaves modules considered in the radio network optimization software is that all significant pieces of data are systematically archived in a file system. A consequence of this characteristic is that all tasks must be able to access the same file system. This is readily offered when all PEs can share files through a Network File System (NFS).

Moreover, contention due to multiple simultaneous accesses can be reduced if the network is fast enough, and if files are stored on parallel Redundant Array of Independent Disks (RAID) systems.

Another consequence of the aforementioned characteristic is that little information needs to be exchanged between a master task and the associated slave tasks. Since all tasks access a common file system for storing or retrieving significant data (e.g., geographical databases, pathloss maps, coverage cells), the information transmitted between a master and its slaves mostly consists of control data (e.g., jobs requests and assignments). Such a model only requires a low bandwidth, but if slave tasks are to be kept busy low latency is required as well.

We implemented the master-slaves modules of the radio network optimization software using the PVM library. Any master and its slaves are thus PVM tasks. A master is simply in charge of feeding its slaves with new jobs as and when they request it. A job assignment simply consists in the identity of a BTS for which some calculation must be done (either a radio wave propagation simulation, or the computation of the resulting coverage cells). Each slave task can retrieve from the file system the data it needs to perform its job, and it can likewise store the results of this job to the file system. As soon as a slave task has run its job to completion, it requests another assignment from the master task.

Our master-slaves modules were recently implemented and tested on a cluster of tightly-coupled UltraSPARC workstations sharing a pool of 6 RAID systems (with 5 or 6 disks per system). Although these modules have not been fully tested yet, preliminary experiments confirm that a master can control up to 30 slaves without any perceptible contention on the file system.

In any case, the number of slaves is bounded by the number of BTSs in the radio network considered (typically, a few hundreds of BTSs), for the creation of a slave task can be justified only if it assigned at least one job during its lifetime.

Although our master-slaves modules have only been tested with mono-processor slaves so far, we plan to experiment with multi-processor slaves in the near future. A slave task will thus be a parallel task running on several PEs. Such a configuration does not really make sense when slaves must compute coverage cells, as this process is sequential in essence. On the other hand, when slaves must achieve radio wave propagation simulations, each simulation can imply the cooperation of several PEs, as has been shown in Section 4.1.

4.3 Coverage graph computation

We developed a data-parallel version of the coverage graph computation module. In this version the geographical area considered for the target radio network is partitioned in sub-areas. The computation of the graph can thus be perceived as a two-phase process.

In the first phase the sub-areas are allocated to distinct PEs. Each PE is in charge of computing a partial coverage graph for the sub-area it has been assigned. This phase of the process is inherently parallel. Moreover it does not imply any communication, for all partial graphs can be computed concurrently and independently.

The second phase of the process consists in a reduction of the distribution information resulting from the first phase. All partial graphs must be merged into a single graph that models the coverage obtained with the radio network considered. This is performed concurrently following a binary tree pattern. Such an implementation requires a tightly-coupled parallel platform (e.g., COW or MIMD supercomputer), for the cost of communications is not neglectable (the data to be exchanged consist of partial graphs).

In any case the amount of potential parallelism in this coverage graph computation is high: the geographical area considered in the radio network may be partitioned in hundreds of sub-areas. As a consequence the computation may be achieved concurrently on as many PEs.

To date the development of the data-parallel coverage graph computation module is still in progress, using the PVM communication library. In the near future it should be ported and tested on clusters of UltraSPARC workstations, as well as on the SGI Origin 2000 supercomputer.

4.4 Radio network optimization

In ParaGene the population of individuals modeling potential radio networks is split in sub-populations called islands, which are assembled along an oriented ring. Islands evolve independently during a generation. Each island transmits a copy of its best individual to the next island in the ring after the new generation has been computed.

It has been observed that the island-based model permits faster convergence of a genetic algorithm towards good individuals. Another advantage of this model is that its parallel implementation is straightforward.

In our data-parallel implementation of ParaGene several tasks are created at startup. Each task can be in charge of one or several islands. Tasks exchange

information with their neighbors periodically (i.e., after each generation) in a quasi-synchronous communication phase.

The number of tasks is limited by the population size, for each task must be in charge of at least one island, and an island must consist of at least two individuals. In any case, the order of magnitude of the population sizes we consider is usually in the hundreds for individuals, while for the number of islands (hence the number of tasks) it is rather in the tens.

The parallel version of ParaGene was implemented using the PVM communication library. Experimentation shows that with this library the cost of communications is almost neglectable compared to that of computations. This is because the amount of data exchanged between tasks is quite low (typically a short bit string). As a consequence, ParaGene can run efficiently on any medium to loosely coupled parallel platform, such as a Network Of Workstations (NOW). Of course it can run on a COW or a MIMD supercomputer as well, but this will not necessarily lead to better performances.

ParaGene was implemented and tested on a network of 80 UltraSPARC workstations. Experiments were performed in order to choose the best selection of BTS sites among a set of 600 potential ones. A very good efficiency (76%) was observed when using up to 80 PEs.

4.5 Pipelining

In order to handle the distribution and the synchronization of the different modules (components) that constitute our radio network optimization software, we use a combination of standard Unix facilities, such as pipe `|' inter-process channels, and local and remote shell scripts. Pipe inter-process channels are mostly used as synchronization channels, as all significant data in the software can systematically be stored to and retrieved from the file system (which, thanks to NFS, transparently gives access to centralized or decentralized storage devices). A pipe channel between any two modules A and B thus only carries little amounts of synchronization information. Generally a piece of information consists of the identity of the last BTS that has been treated by A, and that can now be considered by B.

Hence, as soon as a radio wave propagation simulation has been performed for a given BTS k, the availability of the resulting path-loss cell is signaled to the coverage computation module, which can start the production of the corresponding coverage map. Likewise, the availability of a coverage cell for BTS k will immediately trigger its integration by the graph computation module.

Besides the synchronization of pure computation modules, the pipeline also permits to control the production of images that are built iteratively during each computation phase, and that are sent to a remote Java-based GUI as soon as a computation phase is over. To produce these images, small sequential programs are inserted in the pipeline. These programs do not slow down the whole computation process significantly, as they are executed on one or several dedicated workstations. They nevertheless contribute to make the whole process more `interactive', for they give the user on-the-fly information about the progress of the computation chain.

Although we have not tried to combine in the same computation chain fully heterogeneous resources yet (e.g., workstations and supercomputer), this should be done in the near future. Experiments achieved on 60 COWs have confirmed that the distribution of computation modules and their synchronization through a pipeline lead to a very good (almost optimal) exploitation of the available resources.

To date, starting a given computation module on any available PE (e.g., single workstation, cluster of workstations, MIMD supercomputer) is achieved through the Unix `rsh' (remote shell) command, and the distribution of the different modules over the available PEs is managed in traditional shell-script files. This rather simplistic implementation makes it possible to experiment the whole parallel chain at little cost, but it is not versatile enough. In the future we plan to develop a more flexible and adaptive control software for managing the available resources dynamically (e.g., adding or deleting a PE), and for mapping computation sessions on these resources, taking into accounts the particular needs of each part of the computation chain. This control software shall also permit that several users share the radio optimization software simultaneously.

4.6 Towards meta-computing

Our future developments aim at the distribution of computation modules over the computational resources offered on the EPFL site, combining heterogeneous parallel platforms such as loosely-coupled pools of workstations, medium to tightly-coupled clusters of workstations (or PC-stacks), and tightly-coupled MIMD supercomputers (e.g., Cray T3D and SGI Origin 2000).

Once on-site distribution has been obtained, the next step will consist in the distribution of computation modules over remote sites. This shall require further developments in order to ensure transparent (and secure) data and control exchanges between the

selected sites. To achieve this goal we could rely on standard networking facilities such as remote file copy (Unix command `rcp'), or automated FTP-based file transfers.

5 CONCLUSION

In this paper we presented how different types of parallelism can be exploited in order to optimize the use of available parallel computing resources. In the application presented (taken from the telecommunication field) four levels of parallelism were exploited, using hardware platforms ranging from parallel supercomputers to wide area networks. Thanks to this approach we could realize an adaptable, interactive parallel software solving a complex multi-disciplinary problem.

Further studies need to be undertaken, especially toward parallel 'meta-computing' applications. In our opinion, future commercial software products will greatly make use of multilevel parallelism in order to achieve good performances together with a user-friendly interface.

REFERENCES

Calégari, P.; F. Guidec; P. Kuonen; and D. Kobler. To be published in Spring 1998. Parallel Island-Based Genetic Algorithm for Radio Network Design. *Journal of Parallel and Distributed Computing (JPDC): special issue on Parallel Evolutionary Computing*. Academic Press.

Geist, A.; A. Beguelin; J. Dongarra; W. Jiang; R. Manchek; V. Sunderam. 1994*. PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*. Scientific and Engineering Computation Janusz Kowalik, Editor . MIT Press.

Guidec, F.; P. Calégari;. and P. Kuonen. 1997. Object-Oriented Parallel Software for Radio Wave Propagation Simulation in Urban Environment. in *EuroPar'97 Parallel Processing (Third International EuroPar Conference, Passau, Germany, August 1997, Proceedings*), S. Lengauer, M. Griebl, and S. Gorlatch, Eds. Sept. 1997, vol. 1300 of Lecture Notes in Computer Science, Springer, 832-839.

Kuonen, P.; S. Ubéda; and J. Zerovnik. 1996. Graph Theory Applied to Mobile Network Optimisation. In *Electronical Review,* Lubjana, Slovenia. Publised by the Electronical Society of Slovenia, Lubiana, Vol 63, No.2, pp.65-144.

BIOGRAPHIES

Pierre KUONEN was born in Switzerland in July, 1958. He obtained a degree in Electrical Engineering from the Swiss Federal Institute of Technology (EPFL) in 1982. After six years of experience in industry he joined the Computer Science Theory Laboratory at the EPFL in 1988. He then started working in the field of parallel computing, and received his Ph.D. degree in 1993. Since 1994 he has been a scientific collaborator, heading the Parallel Computing Research Group of the Computer Science department. He is a lecturer in parallel computation courses.

Frédéric GUIDEC was born in France in March, 1966. He received his Ph.D. in Computer Science from the University of Rennes, France, in 1995. Since then he has been a member of the Parallel Computing Research Group at the Swiss Federal Institute of Technology (Lausanne, Switzerland). His research interests include parallel computing, software engineering, as well as object-oriented technology applied to telecommunications and supercomputing.

Patrice CALÉGARI was born in France in November, 1969. He received the M.S. degree in Computer Science from the University of Lyon, France, in 1994. He is now pursuing a Ph.D. in Computer Science at the Swiss Federal Institute of Technology (Lausanne, Switzerland). His research interests are focused on parallel computing, evolutionary algorithms and combinatorial optimization.