PARALLEL CALIBRATION OF SPATIAL DYNAMIC MODELS IN TERRAME

Saulo Henrique Cabral Silva, Joubert de Castro Lima and Tiago Garcia de Senna Carneiro
Departamento de Computação - Universidade Federal de Ouro Preto
Morro do Cruzeiro, Ouro Preto, MG, Brazil

ABSTRACT

Two of the main problems in calibrating spatial dynamic computational models are: (i) the huge runtime of a simulation, (ii) and few variables that can be simulated in a sequential calibration process. There are several methods to calibrate spatio-temporal models. The three most popular are: Monte Carlo Method, Genetic Method and Least Squares Method. Spatio-temporal computational simulators like Swarm, Stella, TerraME, Dinamica-Ego, Repast and Vensim have one or more calibration methods, but none of them can execute models or calibrations in parallel, i.e., none of them can be executed efficiently in shared memory computer architectures. In this paper, we extend TerraME, by introducing parallel calibration of spatial dynamic models using Monte Carlo and Genetic Methods. Our results demonstrate that parallelism can increase the number of variables being calibrated and reduce the runtime of calibration process. In general, TerraME with parallel calibration has 75-80% of linear speedup when the number of cores is equal the number of disks in a unique machine and 50-60% of linear speedup, otherwise.

KEYWORDS

Parallel calibration, spatial dynamic models, TerraME.

1. INTRODUCTION

One of the main problems in earth systems modeling is the model calibration process. A dynamic spatial model is an abstract representation of a phenomenon that evolves over time and space. This way, the number of model variables can be high and the necessity of multiple runs can make the calibration process expensive in terms of CPU and memory. We cannot avoid calibrating and validating a model, since the use of empirical data to calibrate a model gives realism to its behavior. Depending on the model complexity and on the calibration method, the calibration process can take hours, days or even months to finish.

In the literature, there are many methods to calibrate spatial dynamic models, including Monte Carlo, genetic algorithms, least squares, and others methods (Wu 2002; Straatman, White et al. 2004; Tellinghuisen 2010; Zhang, Pu et al. 2010). However, only few modeling and simulation platforms like Swarm (Minar, Burkhart et al. 1996), SME with Stella (Costanza and Voinov 2001), TerraME (Carneiro 2006), Dinamica-EGO (Soares, Cerqueira et al. 2002), and Repast (North, Collier et al. 2006) provide model calibration services. Some of them provide more than one calibration method, but none of them can simulate models or calibrations in parallel, i.e., none of them can take advantage of shared memory computer architectures.

Due to this limitation, we extend TerraME modeling platform, enabling parallel calibration of spatial dynamic models using Monte Carlo and genetic methods. The parallel calibration methods are tested with the Aedes aegypti dynamic population model proposed by (Ferreira and Yang 2003) and extended by (Lana, Carneiro et al. 2010). The results demonstrate that parallel calibration of models can deal with a higher number of variables being calibrated and can still reduce the runtime of calibration process. In general, TerraME with parallel calibration has 75-80% of linear speedup when the number of cores is equal the number of disks in a unique machine and 50-60% of linear speedup, otherwise. We use a multi-core machine with eight cores and four disks to run the experiments.

The rest of this paper is organized as follows: in Section 2, we detail the TerraME features and the TerraME calibration methods, including the parallel versions. In Section 3, we describe our experiments and results. Finally, in Section 4 we conclude our work, pointing out some future directions.

2. TERRAME CALIBRATION METHODS

This section describes the calibration methods implemented in TerraME. TerraME is an earth system modeling and simulation platform (Carneiro 2006). It enables the integration of spatially-explicit simulations with TerraLib geographical databases (Câmara, Souza et al. 2000). TerraME provides a user/modeler level language that extends the Lua programming language (Ierusalimschy, Figueiredo et al. 1996) with novel types for spatial dynamic modeling. It supports the development of multi-scale models based on the combined use of several conceptual approaches such as cellular automata, agent-based, general system theory and discrete event simulation. TerraME also provides hybrid automata (Henzinger 1996) for simulating mixed discrete and continuous behavior. In addition, irregular cellular spaces (Carneiro, Maretto et al. 2008) are provided to represent anisotropic spaces. Among the selected simulators, TerraME is the unique with such improvements.

2.1. Monte Carlo Method

Monte Carlo method is a very general expression used to designate a class of algorithms that rely on exhaustive random sampling to perform a computation (Metropolis and Ulam 1949). In model calibration, outcomes are produced and reproduced systematically by applying random parameters to the model's equations. The set of parameters that produces the best fitness between model outcomes and observed measures is sad to calibrate a model. Monte Carlo method is one of many methods to analyze uncertainty propagation (Narita, Eberlz et al. 1996; Tellinghuisen 2010). They are useful for problems with high dimensionality or high stochastic parameters, which cannot be understood analytically.

2.2. Genetic Method

A genetic algorithm (GA) mimics the process of natural evolution, producing better solutions from a previously known population of solutions (Mitchell 1998). In model calibration, parameter values go through operations as mutation and crossover, producing new values that are passed as parameters to model's equations. If these new values improve the fitness between model outcomes and observed data, they are selected as possible solution to the calibration problem. Otherwise, these values are discarded. This way, at each generation the population of known solutions is improved. The process continues until a certain level of fitness is achieved or until a maximum number of generations are produced. The individual solution that produces the best level of agreement between model outcomes and measurements is sad to calibrate the model. When a model is stochastic, the simulation result varies according to the same set of parameters, turning the fitness function of such a simulation a function with noise. More details of TerraME Genetic Calibration Method can be found in (Fraga, Carneiro et al. 2010).

2.3 Parallelization Strategy

We have adopted a data decomposition technique, i.e., we decompose the input set of parameters to be evaluated into several subsets and simultaneously execute the same calibration method on each of them. We implement the bag of tasks as our parallel algorithm model and our mapping technique (Ananth Grama,et.al. 2003). We start a calibration task for each machine core. This way, if a sequential calibration method has $nMax$ values of parameters to evaluate, the parallel version will evaluate approximately $\frac{nMax}{cores}$ parameters on each core. The simulation of one model with a given parameter value is encapsulated into a single calibration task that does not interact with other tasks, resulting in no communication overhead. Starting more than one calibration task per core might result into speedup degradation, since the concurrency level and the memory consumption increase as the number of parallel tasks increases.

In our solution each calibration task receives a parameter to evaluate and runs the simulation. After the simulation, the same calibration task asks for a new parameter to be evaluated. The parallel calibration finishes when there are no parameters to be evaluated. Instead of a fixed number of calibration tasks in a bag, the number of tasks might vary depending on the simulation characteristics. Using more calibrations tasks

than the number of available cores might be advantageous whether the simulated model executes several input and output operations.

The IO is also a problem in parallel calibrations. Model outcomes can be stored after each simulation or one single time after all simulations. As simulations deal with a huge amount of data, the second alternative can be prohibitive. The former alternative allows several IO operations to be executed simultaneously along the calibration process. We have implemented a bag of IO tasks to provide parallel IO, with one IO task per disk.

When we calibrate a model using a genetic method, the parameter values produced at each simulation are used to evaluate the subsequent simulations. Basically, we can implement two alternatives: (i) first, each calibration task can evolve independently, i.e., without using results values produced by others tasks, (ii) and each calibration task can use parameter values produced by other tasks. Both alternatives are easy to be implemented in TerraME. Due to the limited space, we describe only the first alternative, since in this paper our goal is to build a scalable solution and not a more accurate one.

```
function Model(x, x1, ..., xn)
        MODEL IMPLEMENTATION
end

paramRange  = {100, 1000};  -- range of parameter values
precision
maxInteractions

--@ParallelCalibration
GeneticCalibration (model, paramRange, precision, maxInteraction , ...);

--@ ParallelCalibration
MonteCarloCalibration(model, lowerBound, upperBound, maxInteraction);
```

Figure 1. Annotations needed for parallel genetic and Monte Carlo calibration in TerraME

We have used annotations to instrument the original model source code. This fact, gives portability to TerraME, since we can execute an annotated model in sequential or parallel TerraME version without recoding. Figure 1 illustrates both genetic and Monte Carlo annotations. The reserved word *@ParallelCalibration* should be inserted as a comment immediately before a call to the calibration method.

We have implemented an interpreter between the user model and the TerraME kernel. Initially, the interpreter reads the user model, identifies the annotations to parallelize the calibration process and identifies the calibration parameters. After the identification phase, the interpreter calls the parallel TerraME kernel, which instantiates as many sequential TerraME kernel instances as possible in a unique machine. Normally, we instantiate one sequential TerraME kernel instance per machine core. The parallel TerraME kernel compares the result of parallel instances and returns the calibration results with minimal error.

3. EXPERIMENTS

In this paper we are interested in analyze the runtime when we increase the number of simulations in a calibration process and the benefit of a multi-core machine running TerraME parallel calibration methods. We tested TerraME running model Pronex-Dengue (Lana, Carneiro et al. 2010). Sequential versions of these methods have already been implemented in TerraME and tested in a real-world study case (Lana 2009; Fraga, Carneiro et al. 2010). The models and data produced in Lana et al. work have been made available to this work. For this purpose, we have used oviposition data collected by (Honório, Codeço et al. 2009), who weekly monitored the Aedes aegypti population in Higienópolis, a neighborhood of Rio de Janeiro, RJ, Brazil, during 1.5 years. The air temperature was obtained from the nearest meteorological station, located at the Rio de Janeiro's international airport.

The machine used in our experiments is a dual Intel Xeon with eight cores, 2GHz each core, and four disks with 7200rps and 32MB buffer. Operating System is Windows server 2003 and the machine has 16GB RAM DDR2 667MHz. We have tested sequential and parallel calibrations in TerraME. Each Monte Carlo calibration was tested with 10000, 15000, 20000, 25000, 30000 and 60000 simulations. Each task is

implemented as a light process/thread. With Genetic method, each calibration has 500, 1000, 1500, 2000, 2500 and 5000 simulations.

Figure 2 illustrates the Monte Carlo method speedup when we increase the concurrence level. In general, Monte Carlo using two threads reduces the runtime in 40%. The speedup with two threads is 80% of the linear speedup. As the number of threads increase the speedup decreases. With four threads and four used disks the speedup is 77% of linear speedup and with eight threads and four used disks the speedup is 50% of linear speedup. As the number of simulations increases the speedup gets better, as we can observe with 60000 simulations. Initially, the speedup with 60000 simulations and two threads is similar to the remaining experiments, but the speedup with four and eight threads are slightly better with 60000 simulations.

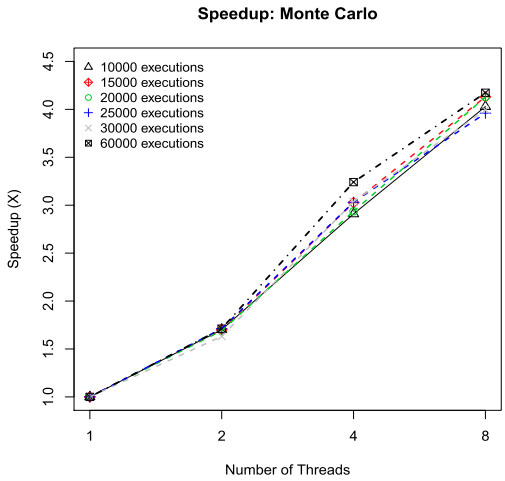

Figure 2. Parallel Monte Carlo calibration speedup

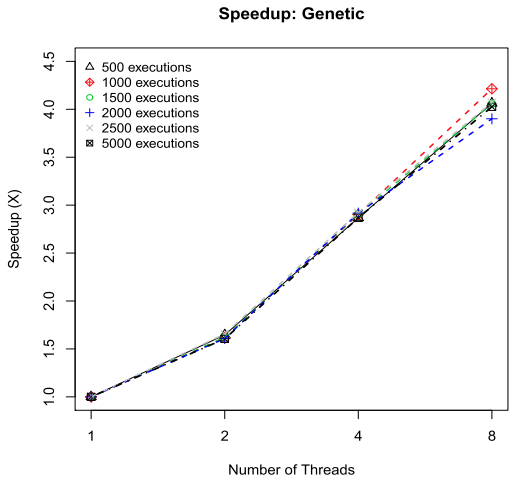

Figure 3. Parallel Genetic calibration speedup

Figure 3 illustrates the speedup of Genetic method with two, four and eight threads. The results are similar to the Monte Carlo method, i.e., with two threads the speedup is 80% of linear speedup, with four threads around 75% of linear speedup and with eight threads 50% of linear speedup. In general, the results of both Genetic and Monte Carlo methods are expected, since the number of disks is lower than the number of cores. The deterioration of speedup from two threads to four threads is higher than the deterioration of the speedup from four threads to eight threads.

IO becomes the bottleneck and we assume it is the main reason of only 50% of linear speedup with eight threads in a machine with eight cores. Each parallel calibration stores the results in a specific disk after each simulation, but in our experiments we have only four disks, so we can have two IO tasks competing to write in a single disk. We experiment both Monte Carlo and Genetic methods with ten threads, but the results, as we expected, deteriorates the speedup even more than eight threads.

4. FINAL REMARKS

The main reason to parallelize calibration methods is the large number of simulations needed in the calibration process (Hadjidoukas, Bousis et al. 2010). In the past, such a calibration process could only be executed on very expensive supercomputers.

In this paper we extend TerraME modeling platform, by parallelizing the existing methods to calibrate a model. The parallelization strategy used here is simple, efficient, and represents the initial studies of a more ambitious plan. In general, we achieve 75-80% of linear speedup in the calibration process of real case study when the number of cores is identical to the number of disks. The model fitness produced by sequential and parallel calibration methods are similar. We do not investigate the benefits of parallelism in producing better model fitness.

The main drawbacks of our solution are both simulations cannot fit in a single machine and each simulation runs sequentially without taking advantage of several available cores or processing nodes of a

cluster. In this work, we have developed only a shared memory version of TerraME calibration methods. The distributed version is not so hard to implement, due to the simple design of our solution. Each processing node and not each core can process many simulations without communication. We believe that the speedup of the distributed version can be closer to linear than the parallel version. This phenomenon is normally observed in distributed versus parallel systems. Note that, we always need to optimize communications in distributed systems and our solution achieves this requirement.

We have started the development of the TerraME HPA (TerraME - High Performance Architecture). The TerraME HPA goal is to parallelize or distribute a single simulation efficiently. It is a hard task, but the results will solve a fundamental problem, i.e., the simulation or the calibration of complex models with many variables and huge amount of data in a parallel or distributed machine. We are also interested in investigating the use of GPUs (Graphical Processing Unit) in simulating and calibrating models with TerraME.

ACKNOWLEDGEMENT

This work is funded by the Federal University of Ouro Preto and CAPES pro-equipaments program.

REFERENCES

Ananth Grama,et.al, 2003. *Introduction to Parallel Computing (2nd Edition)*. Addison-Wesley, Massachusetts,USA.

Câmara, G., R. Souza, et al . (2000). *TerraLib: Technology in Support of GIS Innovation*. II Brazilian Symposium on Geoinformatics, GeoInfo2000, São Paulo.

Carneiro, T. (2006). *Nested-CA: a foundation for multiscale modeling of land use and land change. PhD Thesis in Computer Science* (avaliable at www.dpi.inpe.br/gilberto/teses/nested_ca.pdf). . Doctorate Thesis in Computer Science. Computer Science Department. Sao Jose dos Campos, INPE.

Carneiro, T. G. S., R. V. Maretto, et al, 2008. *Irregular Cellular Spaces: Supporting Realistic Spatial Dynamic Modeling over Geographical Databases*. Brazilian Symposium of GeoInformatic, Rio de Janeiro, RJ, Brazil.

Costanza, R. and A. Voinov, 2001. *Modeling ecological and economic systems with STELLA: Part III.* Ecological Modelling. 143(1-2): 1 - 7.

Ferreira, C. and H. Yang, 2003. *Dinâmica da população de mosquito aedes aegypti.* TEMA - Tendencias em Matemática Aplicada e Computacional 4(2): 187-196.

Fraga, L., T. G. S. Carneiro, et AL, 2010. *Calibração em Modelagem Ambiental na Plataforma TerraME usando Algoritmos Genéticos*. 42º Simpósio Brasileiro de Pesquisa Operacional, Bento Gonçalves, RS, Brazil.

Hadjidoukas, P., C. Bousis, et al, 2010. *Parallelization of a Monte Carlo particle transport simulation code.* Computer Physics Communications 181(5): 928-936.

Henzinger, T. A, 1996. *The Theory of Hybrid Automata*. IEEE Symposium on Logic in Computer Science (LICS'96), New Brunswick, NJ , USA, IEEE.

Honório, N., C. Codeço, et al, 2009. *Temporal distribution of aedes aegypti in different districts of rio de janeiro, brazil, measured by two types of traps*. Journal of Med. Entomology 46(4).

Ierusalimschy, R., L. H. Figueiredo, et al, 1996. *Lua-an extensible extension language.* Software: Practice & Experience 26(6): 635-652.

Lana, R. M, 2009. *Coupled dynamic models for the simulation of Aedes aegypti ecology.* Postgraduate Program in Tropical Biomes Ecology. Ouro Preto, Federal University of Ouro Preto. Master's thesis.

Lana, R. M., T. G. S. Carneiro, et AL, 2010. *Change allocation in spatially-explicit models for Aedes aegypti population dynamics*. Brazilian Symposium of GeoInformatics, Campos do Jordão, SP, Brazil.

Metropolis, N. and S. Ulam, 1949. *The Monte Carlo Method.* Journal of the American Statistical Association 44(247): 335–341.

Minar, N., R. Burkhart, et al, 1996. The *Swarm Simulation System: A Toolkit for Building Multi-Agent Simulation*, SFI Working Paper 96-06-042.

Mitchell, M, 1998. *An Introduction to Genetic Algorithms. Complex Adaptive Systems*. Massachusetts, US, MIT Press.

Narita, Y., S. Eberlz, et al, 1996. *Monte Carlo and experimental evaluation of accuracy and noise properties of two scatter correction methods for SPECT.* Physics in Medicine and Biology: 2481 - 2496.

North, M. J., N. T. Collier, et al, 2006. *Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit.* ACM Transactions on Modeling and Computer Simulation 16(1): 1-25.

Soares, B. S., G. C. Cerqueira, et al, 2002. *DINAMICA - a stochastic cellular automata model designed to simulate the landscape dynamics in an Amazonian colonization frontier.* Ecological Modelling 154(3): 217-235.

Straatman, B., R. White, et al, 2004. *Towards an automatic calibration procedure for constrained cellular automata.* Computers, Environment and Urban Systems 28(1-2): 149-170.

Tellinghuisen, J, 2010. *Least-squares analysis of data with uncertainty in x and y: A Monte Carlo methods comparison*. Chemometrics and Intelligent Laboratory Systems 103(2): 160-169.

Wu, F, 2002. *Calibration of stochastic cellular automata: the application to rural-urban land conversions.* International Journal of Geographical Information Science 16(8): 795-818.

Zhang, F., L. Pu, et al, 2010. *Calibration of cellular automata model with adaptive genetic algorithm for the simulation of urban land-use*. Geoinformatics, 2010 18th International Conference on Beijing, China.