Multi-parametric intensive stochastic simulations for hydrogeology on a computational grid

J. Erhel[1] *, J.-R. de Dreuzy[2], E. Bresciani[1]

[1] INRIA,35042 Rennes Cedex, France
[2] Geosciences Rennes, UMR CNRS 6118, France

Abstract. Numerical modelling is an important key for the management and remediation of groundwater resources. Numerical simulations must be performed on domains of a large size, at a fine resolution to take into account the scale of geological heterogeneities. Numerical models are based on probabilistic data and rely on Uncertainty Quantification methods (UQ). In this stochastic framework, non intrusive methods require to run multiple simulations. Also, each simulation is governed by multiple parameters and a complete study requires to carry out analysis for more than 50 sets of parameters. We have identified three levels of distributed and parallel computing: subdomain decomposition in one simulation, multiple simulations for UQ methods, multiparametric studies. Our objective is to use the computing and memory resources of computational grids to deploy these multiple large-scale simulations. We discuss our implementation of these three levels, using an object-oriented approach. We present some preliminary results, with a strategy to choose between the first and second level.

1 Introduction

Numerical modelling is an important key for the management and remediation of groundwater resources. Natural geological formations are highly heterogeneous, leading to preferential flow paths and stagnant regions. The contaminant migration is strongly affected by these irregular water velocity distributions. In order to account for the limited knowledge of the geological characteristics and for the natural heterogeneity, numerical models are based on probabilistic data and rely on Uncertainty Quantification methods. In this stochastic framework, non intrusive methods require to run multiple simulations. Also, numerical modelling aims at studying the impact of various physical parameters, such as the Peclet number. Therefore, each simulation is governed by multiple parameters and a complete study requires to carry out analysis for more than 50 sets of parameters. The hydraulic simulations must be performed on domains of a large size, at the scale of management of the groundwater resource or at the scale of the homogeneous medium type in terms of geology. This domain must be discretized at a fine resolution to take into account the scale

* this work was partly funded by the Grid'5000 grant from the French government

D. Tromeur-Dervout (eds.), *Parallel Computational Fluid Dynamics 2008*,
Lecture Notes in Computational Science and Engineering 74,
DOI: 10.1007/978-3-642-14438-7_41, © Springer-Verlag Berlin Heidelberg 2010

of geological heterogeneities. Characterization of transport laws requires simulating advection and dispersion on very long times and in turn in very large domains. Our objective is to use the computing and memory resources of computational grids to deploy these multiple simulations.

A first level of parallelism is used in each simulation. Indeed, in order to reach the target of large scale domains, it is necessary to run each simulation on a parallel computer with enough memory and with enough computing power. A second level of parallelism comes from Uncertainty Quantification. A third level of parallelism is the study of different sets of parameters. These multiparametric simulations are clearly independent and are thus very well-suited to techniques inspired from peer-to-peer. However, it should be kept in mind that each study is in itself a heavy computation involving a large number of random simulations, requiring high performance computing for each simulation. Our objective is to use current middleware developed for grid architectures, in order to make the most of the three levels of parallelism. Several difficulties arise, ranging from basic software engineering (compatibility of systems, libraries, compilers) to scheduling issues.

2 Existing work

2.1 Parallel Monte-Carlo

The Monte-Carlo method is heavily used in physical simulations, either to evaluate integrals or in the framework of stochastic models. In general, a run is composed of more or less independent simulations, so that a run is embarassingly parallel. The main difficulty is to generate random numbers correctly. Also, since the flowchart of a Monte-Carlo run is identical for many applications, it is natural to design a generic software. For example, the ALPS project (Algorithms and Libraries for Physics Simulations) is an open source effort aiming at providing high-end simulation codes for strongly correlated quantum mechanical systems as well as C++ libraries for simplifying the development of such code (http://alps.comp-phys.org/). The ALPS software contains a module for parallel Monte-Carlo runs and parallel multiparametric simulations [14]. It is based on a distributed memory paradigm and uses MPI, with clusters as target computers. Currently, each simulation is sequential.

2.2 Distributed multiple simulations and grid applications

Multiparametric experiments arise in many application domains, for example in biology, chemistry and earth science. Computational grids provide interesting power and memory resources. Many initiatives of grids are built around the world. For example, DEISA is a grid gathering several supercomputing centers in Europe (http://www.deisa.org/). Other grids build an infrastructure with several partners to create an integrated, persistent computational resource. Some examples are Teragrid in USA (http://www.teragrid.org/about/), EGEE in Europe (http://www.eu-egee.org/)

and Grid'5000 in France (http://www.grid5000.fr). These infrastructures aim at developing e-science applications using global resources provided by the grid. For example, the GEON web portal (http://www.geongrid.org) provides tools to a network of partners in earth science, like SINSEIS, a synthetic seismogram computation toolkit, built as a grid application. Some applications are multiparametric scientific simulations; for example, in earth science, the footprint project (http://www.eu-footprint.org/) develops tools for pesticide risk assessment and management, based on Monte-Carlo and multiparametric simulations for dealing with uncertainty [10]. Whereas most grid initiatives use computing resources of research laboratories or computing centers, another approach rely on Internet to run scientific software on desktop computers. The platform BOINC [2] is a distributed computing tool which allows to run computationally expensive projects by using the aggregate power of desktop computers. The project climateprediction.net [13] uses BOINC to run millions of simulations of a climate model coupling atmosphere and ocean.

2.3 Middleware on grids

Computational grids are often built as a network of several clusters located in different geographical places. Multiple simulations can in principle run on these clusters by using the grid infrastructure. However, scheduling tools are required to distribute the simulations and to communicate data between the clusters. Some projects enable scientists and engineers to seamlessly run MPI-conforming parallel application on a Computational Grid, such as a cluster of computers connected through high-speed networks or even the Internet. For example, MPICH-Madeleine [6] is a free MPICH-based implementation of the MPI standard, which is a high-level communication interface designed to provide high performance communications on various network architectures including clusters of clusters. Another solution is to provide a tool specifically devoted to distributed computing. Nimrod/G is an example of such software and has been successfully used with different grids [1]. The Grid'5000 project provides the software Oaregrid [7] and Taktuk [12], which can also help to deploy parametric simulations. Other approaches are based on a software component paradigm [8].

3 Our work

We are developing a scientific platform H2OLab for hydrogeology. Our platform is designed in order to ensure integration of new modules and to facilitate coupling of existing modules. We use C++ development environments and software engineering tools. We pay a lot of attention to test generation, non regression issues and numerical validation. Modularity and genericity are essential for a scientific platform of this size. These requirements are satisfied by a rigorous design inspired from UML techniques and by an object-oriented approach. We have identified three levels of distributed and parallel computing. At the simulation level, we choose to define distributed memory algorithms and to rely on the MPI library for communications

between processors. These parallel deterministic simulations are operational in the software H2OLab and we are investigating scalability issues [9]. The intermediate level is the Uncertainty Quantification non intrusive method, currently Monte-Carlo. We apply a paragim similar to the software ALPS and have developed a generic Monte-Carlo module. It differs from ALPS in a number of ways including the use of Monte-Carlo, random number generation, the physical application, the memory and CPU intensive simulations and the development tools used. Our objective is to design a facility for running this run of Monte-Carlo by choosing either a parallel approach with MPI or a distributed approach with a grid middleware. We use a specific random number generator in order to guarantee independent simulations. At the multiparametric level, we choose only the distributed approach as is done in most projects on computational grids.

3.1 Generic tools in H2OLab

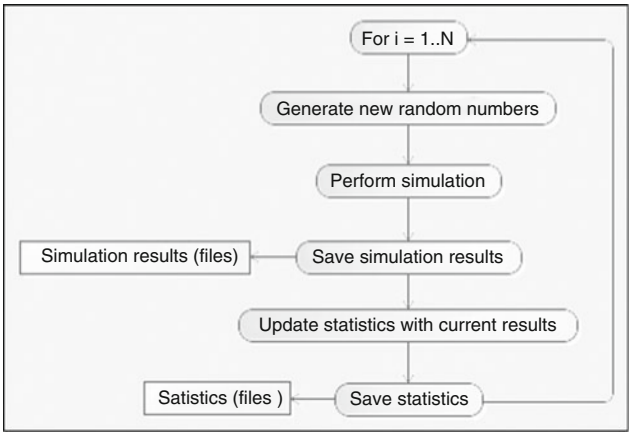

Fig. 1. Monte-Carlo simulations, saving results and collecting statistics.

We have implemented a generic Monte Carlo method where a loop performs N simulations and computes first and second statistical moments of the results. This loop contains a checkpoint at every iteration with a restore point. This recovery facility allows to resume simulations in case of failure or to complete already existing statistics with new results. The generic loop is depicted in Figure 1. Some operations are always done by any executable program : reading inputs, creating results directories, initializing random number generators, launching the simulation, writing the results, initializing visualization tools, etc. All those operations are factorized and performed by a Launcher class. This is lot-of-time saving for the user. In order to be generic to any application, the Launcher and the Monte Carlo modules need a common interface. We thus developed a Simulation virtual class, which owns all necessary objects to perform a simulation: parameters, results, random number streams.

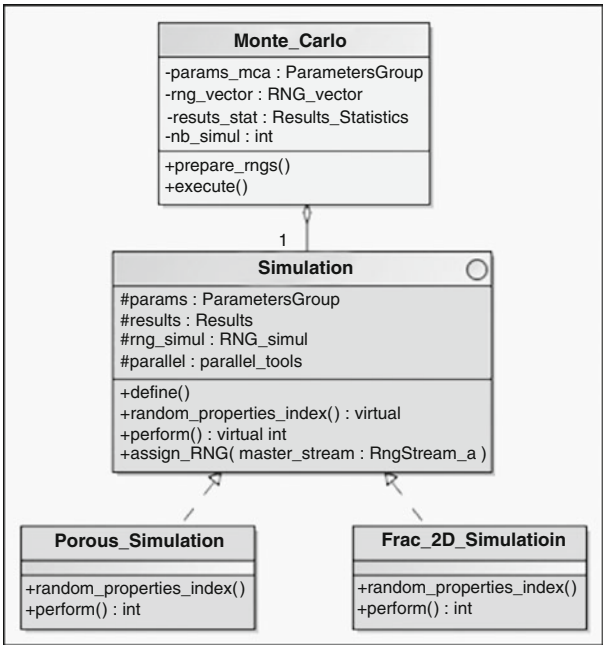

Fig. 2. The virtual class simulation, and its use for Monte-Carlo with various applications.

Practically, the user only has to override two functions to define the specific random properties associated to the simulation and to write the very job of the simulation. The virtual class is depicted in Figure 2.

We have opted for the XML standard language to define a generic structure for input parameters. The use of this standard has allowed us to easily define an associated user interface and develop C++ read/write methods thanks to already existing tools. In our scheme, the parameters are defined by four fields: name, value, default value and description, and can be organized in groups in a recursive manner. This structure facilitates the development of non-conflicting C++ packages.

The simulation results are stored in a generic structure (C++ class) which can contain scalars, vectors and matrices, organized in categories. This class also provides a method to write the results in files in an appropriate format. Application-specific results and categories are defined in XML files.

Regarding random number generation, we have to deal with several difficulties: a simulation has several random properties, a random property can require several random numbers and this quantity is not fixed in advance, depending on the studied medium or phenomenon. Moreover, the run must be reproducible for validation and composed of independent simulations for distributed computing. We have designed a set of classes, based on the RngStream package [11], to generate random numbers streams that solve these difficulties. These classes are generic to any application.

3.2 Deployment on grid architectures

Multiparametric simulations require more than 50 sets of data and generate as many results. We have developed a tool to automatically generate a multiparametric study: from a given range of parameter values, the tool generates all corresponding input data files and an associated batch file to run the complete study. This tool is now ready to be deployed on a computational grid using an adapted middlware.

Thanks to our generic module and our random number generation, a run of Monte Carlo contains an embarassingly parallel loop of simulations, which can be readily distributed on a computational grid. We have currently implemented a parallel version using the MPI standard. It can be generalized to a version with an extended MPI library or to a distributed version with a grid service. Also, the Monte Carlo module can be extended to any non intrusive UQ method.

Finally, each simulation is memory and CPU intensive. The platform relies on free software libraries such as parallel sparse solvers which use MPI. Thus we choose to develop parallel software also with MPI. Each simulation is fully parallel with data distributed from the beginning to the end.

3.3 Experiments on clusters and conclusion

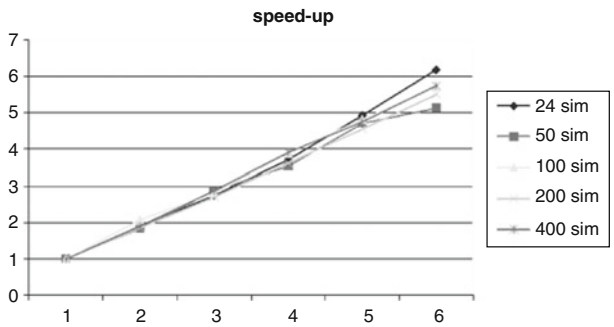

Fig. 3. Speed-up versus the number of nodes (2 subdomains per node).

We use the different clusters available thanks to the french grid project Grid'5000. We have made a thorough performance analysis of our parallel simulations applied to flow and solute transport in heterogeneous porous media [3, 5]. We have also done some experiments with parallel simulations applied to flow in Discrete Fracture of Networks [4]. This analysis allows us to find out in advance the number of processors necessary for a given set of data. Thus we can rely on a static scheduling

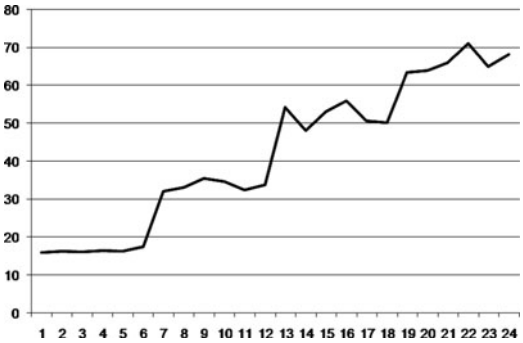

Fig. 4. CPU time versus the number of simulations, with 6 nodes.

of resources for each simulation. For a run of Monte Carlo, we can also define the number of simulations and use a static scheduling.

Here, we give the results for several experiments combining parallel Monte Carlo runs of parallel simulations. In a first step, we run parallel Monte-Carlo simulations of moderate size so that each simulation can run on one node. We use a cluster of nodes with a Myrinet network where each node is one-core bi-processor, with 2GB memory. We apply our method to flow and solute transport in heterogeneous porous media, with a mesh of 1024 times 1024 cells. We have done several measures with varying parameters of the model with similar results, so we plot here the results with one set of parameters. In Figure 3, we plot the speed-up in function of the number of processors. For a small number of nodes, the speed-up is almost linear, as could be expected since parallel Monte-Carlo does not induce communication. In Figure 4, we plot the CPU time in function of the number of simulations, using 6 nodes of the same cluster. As could also be expected, we get a stairwise function, due to the distribution of simulations which is obtained simply by dividing the number of simulations by the number of processors.

In a second experiment, we use a two-level parallelism. We run the same application on a cluster of four nodes and try three configurations, distributing both the subdomains of one simulation and the Monte-Carlo simulations. Results are given in Table 3.3. Clearly, Monte-Carlo parallelism is more efficient since subdomain decomposition involves communications. Therefore, the limiting resource is here the memory available and the best strategy is to choose the smallest number of subdomains so that each subdomain fits in the memory of one core, defining chunks of processors. Then Monte-Carlo simulations are distributed among the different chunks.

These results are preliminary but show clearly that what we get is what we expect. So we can now adopt the same strategy for larger computational domains with a larger number of nodes. Also, in a next future, we plan to use middleware available on grid'5000, in order to run the three levels of distributed computing.

[1] D. Abramson. Applications development for the computational grid. In *Applications Development for the Computational Grid*, volume 3841 / 2006 of *Lecture Notes in Computer Science*, pages 1–12, 2006.

Number of subdomains	CPU time
2	129
4	199
8	299

Table 1. Two-level parallelism with 4 nodes (bi-processors): CPU time according to the number of subdomains.

[2] David P. Anderson. Boinc: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, November 2004.

[3] A. Beaudoin, J-R. de Dreuzy, and J. Erhel. An efficient parallel particle tracker for advection-diffusion simulations in heterogeneous porous media. In A.-M. Kermarrec, L. Boug, and T. Priol, editors, *Euro-Par 2007, LNCS 4641*, pages 717–726. Springer-Verlag, Berlin, Heidelberg, 2007.

[4] A. Beaudoin, J-R. de Dreuzy, J. Erhel, and H. Mustapha. Parallel simulations of underground flow in porous and fractured media. In G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, and E. Zapata, editors, *Parallel Computing: Current and Future Issues of High-End Computing*, volume 33 of *NIC Series*, pages 391–398, Jlich, Germany, 2006. NIC.

[5] A. Beaudoin, J. Erhel, and J.-R. de Dreuzy. A comparison between a direct and a multigrid sparse linear solvers for highly heterogeneous flux computations. In *Eccomas CFD 2006*, volume CD, 2006.

[6] Darius Buntinas, Guillaume Mercier, and William Gropp. Data Transfer in a SMP System: Study and Application to MPI. In *Proc. 34th International Conference on Parallel Processing(ICPP 2006)*, Colombus, Ohio, August 2006.

[7] Nicolas Capit, Georges Da Costa, Yiannis Georgiou, Guillaume Huard, Cyrille Marti n, Grgory Mouni, Pierre Neyron, and Olivier Richard. A batch scheduler with high level components. In *Cluster computing and Grid 2005 (CCGrid05)*, 2005.

[8] A. Denis, C. Prez, and T. Priol. Achieving portable and efficient parallel corba objects. *Concurrency and Computation: Practice and Experience*, 15(10):891–909, August 2003.

[9] Jocelyne Erhel, Jean-Raynald de Dreuzy, Anthony Beaudoin, Etienne Bresciani, and Damien Tromeur-Dervout. A parallel scientific software for heterogeneous hydrogeology. In *Parallel CFD*, Antalya (Turkey), May 2007. Invited plenary talk.

[10] Stenemo F. and Jarvis N.J. Accounting for uncertainty in pedotransfer functions in vulnerability assessments of pesticide leaching to groundwater. *Pest Management Science*, 63(9):867–875, 2007.

[11] Pierre L'ecuyer, Richard Simard, E. Jack Chen, and W. David Kelton. An object-oriented random-number package with many long streams and substreams. *Operations Research*, 50(6):1073 – 1075, 2002.

[12] Vincent Martin, Jrme Jaffr, and Jean E. Roberts. Modeling fractures and barriers as interfaces for flow in porous media. *SIAM Journal on Scientific Computing*, 26:1667–1691, 2005.

[13] N. Massey, T. Aina, M. Allen, C. Christensen, D. Frame, D. Goodman, J. Kettleborough, A. Martin, S. Pascoe, and D. Stainforth. Data access and analysis with distributed federated data servers in climateprediction.net. *Advances in Geosciences*, 8:49–56, 2006.

[14] Matthias Troyer, Beat Ammon, and Elmar Heeb. Parallel object oriented monte carlo simulations. In *ISCOPE '98: Proceedings of the Second International Symposium on Computing in Object-Oriented Parallel Environments*, pages 191–198, London, UK, 1998. Springer-Verlag.