computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209

International Scientific
Journal of Computing

COMPUTATIONAL GRIDS TO SOLVE LARGE SCALE OPTIMIZATION PROBLEMS WITH UNCERTAIN DATA

Chefi Triki, Lucio Grandinetti

University of Calabria
Department of Electronics, Informatics and Systems
87030 Rende (CS) – ITALY
e-mail : chefi@parcolab.unical.it
URL: http://www.parcopab.unical.it

Abstract: *In this paper we discuss the use computational grids to solve stochastic optimization problems. These problems are generally difficult to solve and are often characterized by a high number of variables and constraints. Furthermore, for some applications it is required to achieve a real-time solution. Obtaining reasonable results is a difficult objective without the use of high performance computing. Here we present a grid-enabled path-following algorithm and we discuss some experimental results.*

Keywords: *- Large-Scale Optimization Problems, Two-Stage Stochastic models, Grid Computation, Condor*

1. INTRODUCTION

Many real world applications are characterized by input data which are uncertain, incomplete and/or erroneous. One way to deal with this difficulty is to model these data as random variables with some probability distribution. Stochastic Programming (SP) is an effective tool used to solve this kind of problems giving the optimal solution across the different events that could be observed.

We focus on two-stage SP problems with a finite and discrete distribution of the random variables. The realization of the random variables consists in the occurrence of one of the *N* possible events, known as *scenarios*. In this formulation, decision variables are divided into two groups: *anticipative* decisions taken before knowing the random values, and *adaptive* variables determined after the realization of the random event. For each scenario *l=1, .., N* a corresponding adaptive vector is calculated containing the relative decisions.

Using a mathematical representation the problem can be formulated as following (see [1] for details):

$$\min \quad c^T x + \sum_{l=1}^{N} p_l c_l^T y_l$$

$$\begin{aligned} \text{s.v.} \quad & \boldsymbol{Ax = b} & \\ & \boldsymbol{T_l x + W_l y_l = h_l} & \boldsymbol{l = 1 .. N} \\ & \boldsymbol{xi\ 0} & \\ & \boldsymbol{y_l\ i\ 0} & \boldsymbol{l = 1 .. N} \end{aligned}$$

It can be easily noted that the number of variables and constraints increases considerably as the number of scenarios *N* increases. For most of the real-world applications *N* is very big so the resulting problems can not be solved using conventional sequential systems. The use of parallel machines is necessary to achieve high level of efficiency in reasonable time. The parallelization techniques is based on the idea of splitting the overall problem in order to handle *N* independent sub-blocks of the constraint matrix, each block corresponding to one scenario.

For the implementation of the parallel algorithm on a computational grid we used Condor as a resource management software. After a short description of the parallel algorithm, we will present the motivations behind the choice of Condor and then we describe the basic parallel implementation by using the message-passing paradigm. The paper will be concluded by experimental results and a summary.

2. PARALLEL ALGORITHM

For the solution of the limited recourse model we used a path-following interior point algorithm as described in [1]. The algorithm is based on the iterative solution of a symmetric linear system of the following form:

$$ADA^T \Delta t = \psi \qquad (1)$$

where *A* is the constraints matrix of the optimization problem and D is a diagonal positive definite matrix. Both the right hand side vector ψ and the diagonal matrix D depend on the current solution and are, consequently, updated at each iteration of the path-following scheme. The output of system (1), i.e. the dual step Δt, is used to update the iterate that moves along the central path to reach the optimal solution.

The repeated solution of the system (1) at each iteration of the PF algorithm represents the most expensive task in the solution procedure. Its implementation on parallel platforms should reduce the solution time and increase the size of tractable problems. The parallelization of the other steps of the algorithm will not be discussed in this paper since it is either trivial or impossible.

Since the computation associated with the N scenarios may be performed independently, the proposed approach can fruitfully benefit from a scenario-oriented parallelism. Only a modest amount of data movements is required among processors in order to collect the information needed for the successive iterations.

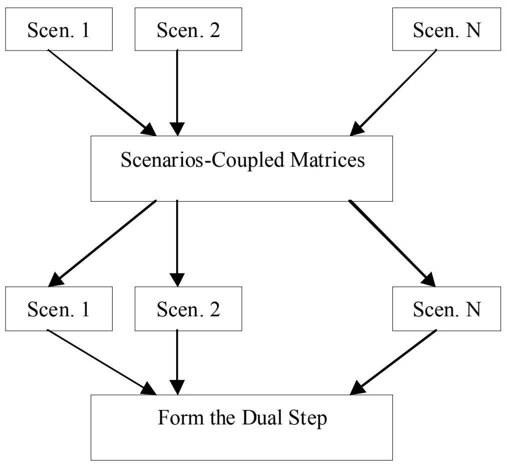

Fig. 1 – *Parallel Scheme of the Path-Following Algorithm.*

More specifically, the parallel scheme proposed in our previous paper [1] can be extended to this case by reporting some slight modifications as follows: a scenario-wise parallelism is performed across the N independent tasks, then a reduction is required to form the scenario-coupled matrices needed by all independent tasks in order to achieve the computation of the dual step's components. The parallelization strategy can be represented schematically as shown by the chart in Fig. 1.

3. CONDOR ENVIRONMENT

In this section we will briefly outline the characteristics of the Condor grid-computing environment, with particular emphasise on the advantages that have encouraged us to choose this software among many other available packages.

Condor is essentially a "specialized batch system for managing compute-intensive jobs" [3] tailored for opportunistic environments. Indeed, Condor is designed to find the free resources in a dynamically changing configuration, and to use them in order to execute the submitted jobs. Several interesting features, such as *Checkpoint, Migration* and *Matchmaking*, makes of Condor a quite effective tool for computational grids. Moreover, Condor is also suitable to be used with dedicated systems.

Some of the more attractive features offered by Condor and from which our implementation can take advantage are the following:

• Condor can be configured to save the state of the running jobs at a given interval of time. In case of failure or unexpected interruption of the process , the state of the process is preserved and the execution of the job can be continued starting from the interruption point;

• the checkpoint data can be used to migrate a process from a computational resource to another. A job should migrate if the resource it was using is needed by a user with higher priority or by the owners;

• Matchmaking is used to find the appropriate resources to execute the submitted jobs. This approach is based on finding all the feasible resource-job pairs and choosing the best matches among them. To perform the matchmaking condor needs indications on the requirements and preferences of both the resource owners and the job submitters. These indications can be described by defining the so-called *ClassAds*. Owners and users use ClassAds as sellers and buyers use classified advertisement on the newspapers: the firsts describe their products (the resources) and the seconds define their (computational) needs. By using Boolean expressions ClassAds permits to find easily the optimal matching on the grid;

• Even though the standard Message Passing Interface [4] is not yet implemented, it is possible to use the Parallel Virtual Machine [10] environment made available within Condor by an independent module called Condor-PVM;

• Condor can share his resources with other flocking and can be used to submit jobs within the Globus package [8].

Finally, it is important to note that for the use of Condor the root privileges are not needed. Indeed, it is enough to have an account with username condor and it is possible to proceed with the installation and the execution of all the facilities of Condor.

4. MESSAGE PASSING IMPLEMENTATION

The parallel PVM implementation is based on the parallelization scheme for an interior point method as described in [1]. In this scheme most of the computation tasks, corresponding to the independent scenarios, are carried out in parallel on the available processors. Some communication and synchronization points are necessary in order to form the scenarios-coupled matrices.

With the aim of developing a grid-enabling solver, a static implementation of the PVM version does not seem to be an appealing choice. It is, indeed, unlikely to have the chance to run programs on dedicated grids for which an ‘‘a priori” splitting of the workload fits well with the static configuration of the system. Furthermore, for real world applications, the number of scenarios is too big to match the available processors of any nowadays systems.

A more attractive alternative is based on a dynamic version that implements the master-slave model. The master task schedules the independent tasks corresponding to the *N* scenarios of the problem. The master spawns also the slaves and assigns to each slave its first task to be processed. Once a slave is idle it requests the master for the next not processed task until all the tasks are assigned.

This scheme is not a classic master-slave model. The master, indeed, does not have the task of collecting the processed data because this will increase the number of communication points within the algorithm and, consequently, deteriorates the performance. Instead, each processor will keep the data corresponding to its processed tasks in order to achieve the reduction operation to form the scenario-coupled matrices. The master program, acting in this case as a scheduler of the tasks, can be ran on a dedicated host machine (workstation, PC, ...).

The reduction should be performed by using two different kinds of PVM subroutines. The broadcast of partial and final results among all the slaves and the sum operator call to form the scenarios-coupled matrices. Once the reduction among processors is performed, each processor will continue to operate on the same data corresponding to the task it had in hand just before the synchronization point. This will allow the reuse of the cached data and will avoid further communications with the master.

This dynamically load balancing implementation combined with a careful tuning and optimization of the algorithm to fit the characteristics of both the single processor and the whole parallel system have shown good performance in terms of speedups on standard stochastic test problems by using an Origin2000 architecture machine.

5. GRID-ENABLED EXPERIMENTS

In this section we will discuss the performance of the parallel implementation of the Path-Following algorithm on a grid. Computational experiments have been carried out by using some standard stochastic test problems collected by Holmes [5]. The experimental mini-grid used has been defined by using two different Origin2000 machines having the characteristics described in Table 1.

Table 1. Origin2000 machines characteristics

Machine	Number of CPU	RAM memory
Origin 1	8	512
Origin 2	4	1024

The two Origin machines are located in two different Departments of the University of Calabria. The latency time for a communication between the two machines is quite high. For example, the roundtrip measured by using the ping command needs some milliseconds.

5.1. Dedicated Grid

In order to evaluate the influence of the overhead caused by the communication between the two hosts with respect to the overhead of the communication within each single machine, we carried out some experiments in a dedicated environment. During the small slice of time in which it was possible to have both the machines idle, we ran the parallel algorithm on two and eight processors, respectively.

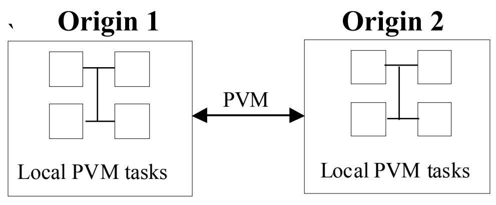

Fig. 2 – *Scheduling of the PVM tasks on the Grid.*

In the first case, we generated only one process on each machine. Only inter-machines communication is involved in this execution. In the second experiment we have both inter-machines and intra-machines communications. Four PVM processes have been spawned on each machine. The critical operation of reduction is done in two different steps: an intra reduction among the four tasks belonging to each machine and then a PVM call among the two machines. Schematically, this approach can be represented as depicted in Fig. 2.

Table 2. Times on dedicated grid (sec.)

Test Problem	2 tasks	8 tasks
Scagr7.432	4.61	2.99
Scsd8.432	22.62	12.13
Sctap1.480	21.44	10.60

The results collected in a dedicated environment are reported in Table 2 and depicted in the Fig. 3. These experiments have been carried out by using a static load balance approach. In a dedicated environment this approach is preferred with respect to the dynamic one because it avoids some communication. In this case, indeed, there is no need to have a master task and the implementation based on the slave-alone model follows the scheme depicted in Fig. 1.

5.2. Non Dedicated Grid

Using a non dedicated system the number of a priori unknown factors that can influence the performance of the code increases. Among these factors the latency of the communication network and the level of use of the different machines of the grid. For this reason, it is preferable in this case to use the dynamic version of the implementation as described in section 3. The allocation of the tasks to the different processors has been done by activating the matchmaking facility of Condor.

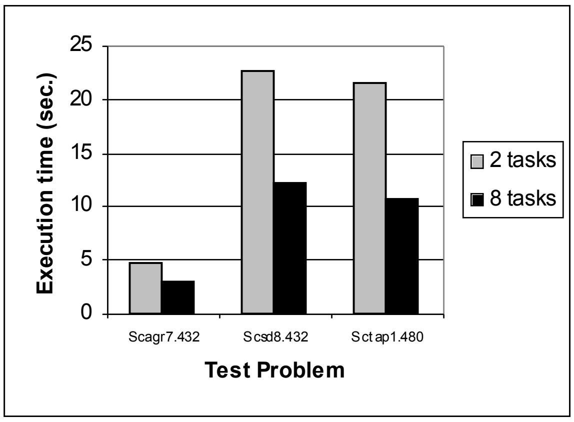

Fig. 3 – ***Results on dedicated Grid.***

The results collected on a set of test problems are summarised in Table 3. We report the execution time (in seconds) by varying the number of slaves spawned by the master task. For the sake of comparison, we report in the same Table the execution time of the same test problems on a single local machine (i.e. Origin 1) without activating Condor environment (italic values).

Table 3. Times on non dedicated grid (sec.)

	# of Tasks			
Test Problem	**1**	**2**	**3**	**4**
Scagr7.432	5.23	3.80	3.24	3.04
	5.20	*3.06*	*2.40*	*2.13*
Scagr7.216	2.75	2.19	1.93	1.84
	2.73	*1.78*	*1.49*	*1.28*
Scsd8.216	12.08	8.62	7.01	6.52
	11.88	*7.46*	*6.04*	*5.13*
Sctap1.216	13.88	9.45	7.30	6.42
	13.80	*8.32*	*6.55*	*5.92*

From these results we can note immediately that the execution time decreases as number of slaves increases. The relative decrease depends on the structure of the test problem and the number of scenarios.

From the other side, it is easy to note the effect of the communication overhead on the algorithm performance. This effect can be calculated as the difference between the execution time on the grid (regular values) and on the local Origin machine without Condor (italic values). The gap increases as the number of tasks increase. By generating only one task the gap is very small and represents the effect of the activation of Condor on the Origin machine. When two tasks are spawned the gap is caused by the activation of Condor and also the inter-machines communication since no communication among the processors of the same machine is needed in this case. In addition to these two reasons, a further increase of the gap can be observed when we are in presence of intra-communication overhead. This is the case of the execution time by using more than two slaves.

6. CONCLUSIONS

In this paper we dealt with the solution of optimization problems with uncertain input data. We modelled the random variables as a set of finite scenarios and we formulated a two-stage stochastic problem. We exploited the discrete representation of the random data to split the overall constraints matrix in small sub-blocks and we developed a scenario-wise parallelization scheme. Computational results on an experimental mini-grid has shown how we can take effective advantage from the use of computational grid to solve difficult optimization problems. We expect that this trend will persist in the solution of huge stochastic problem whenever a real computation grid becomes available.

7. REFERENCES

[1] P. Beraldi, L. Grandinetti, R. Musmanno, and C. Triki, Parallel algorithms to solve stochastic linear programs with robustness constraints. Parallel Computing, 26:1889—1908, 2000.

[2] S. W. Bova et alt. Dual level parallel analysis of harbor wave response using MPI and OpenMP. The International Journal of High Performance Computing, 14(1):384—392, 2000.

[3] Condor Team. Condor Version 6.1.15 Manual. University of Wisconsin, 2000.

[4] Message Passing Interface Forum, MPI-2: Extentions to the Message Passing Interface, 1997.

[5] D. Homes. A collection af stochastic programming problems. Rapporto tecnico 91 - 11, Department of Industrial and Operations Engineering, University of Michigan, 1994.

[6] S. Zenios and Y. Censor. Parallel Optimization: Theory, Algorithms and Application. Oxford University Press, 1997.

[7] Q. Chen, M. C. Ferris, and J. T. Linderoth. FATCOP 2.0: Advanced Features in an Opportunistic Mixed Integer Programming Solver. Technical Report, Computer Science Department, University of Wisconsin, 1999 (to appear on Annals of Operations Research).

[8] The Globus Project. The Globus Toolkit 1.1.3 System Administration Guide. Available on the web from: http://www.globus.org/toolkit/documentation/, 2000.

[9] M. Livny. Personal Condor – Your Window to the Computational Grid. Proceedings of the Advanced Research Workshop on High Performance Computing, Cetraro, Italy, June 2000.

[10] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM: Parallel Virtual Machine, A users' guide and tutorial for networked parallel computing, The MIT Press, 1994.

[11] J.-P. Goux, J. T. Linderoth, and M. Yoder. Metacomputing and the Master-Worker Paradigm. Preprint ANL/MCS-P792-0200, Mathematics and Computer Science Division, Argonne National Laboratory. Available on the web from http://www.mcs.anl.gov/metaneos/publications, 2000.

Chefi Triki is an assistant professor at Mathematics Department of the University of Lecce, Italy. He received his PhD at the University of Calabria in May 1998 in the area of operations reserach. His interests are mainly in the fields of stochastic programming and parallel optimization algorithms with application to energy management problems.

LUCIO GRANDINETTI.

Full Professor at the Department of Electronics, Informatics and Systems of University of Calabria; scientific director of the Parallel Computing Laboratory at the University of Calabria; co-director of NATO ARW on Software for Parallel Computation, Italy (1992) and NATO ARW on High Performance Computing, Italy (1996); member of numerous organising and scientific committees of international conferences on high-performance parallel computing (e.g. EUROPAR, HPCN Europe, PARCO); member of IEEE Technical Committee on Parallel Processing; founding member of the International Society of Computational Engineering and Sciences (ISCES).

His areas of expertise are the design of numerical algorithms for parallel and distributed computer systems, modeling and simulation of large scale systems, numerical optimisation methods for complex problems, software engineering aspects related to parallel processing.

He has been and is currently involved in research projects sponsored and financed by the National Research Council of Italy, by CEC, and by Italian Ministry of Research.

He has been evaluator and reviewer of CEC Research Projects in the IT Programme ESPRIT, during 1993 and 1995. He is currently serving as reviewer of ESPRIT projects in the 4th framework programme and evaluator of research projects in the 5th Framework Programme.

He is co-author of more than 60 papers in refereed journals, and co-editor of several books on numerical methods for nonlinear optimization, computational engineering, parallel algorithms and software for vector and parallel computing. He is member of the editorial board of the following international journals:

Parallel Computing (Elsevier)

Parallel Computing Special Issues on Applications (Elsevier)

Optimization Methods and Software (Gordon and Breach)

He is also co-Editor of the book series "Scientific and Engineering Computation" published by MIT Press (USA).