A High Throughput Solution Portfolio VaR Simulation

ANA BELÉN ALONSO CONDE [†]
RAFAEL MORENO VOZMEDIANO [‡]

[†] Department of Finance, Accounting, and Commerce
Universidad Rey Juan Carlos
Fac. Ciencias Jurídicas y Sociales, P. Artilleros s/n, 28032 Madrid
SPAIN
abac@fcjs.urjc.es

[‡] Department of Computer Architecture
Universidad Complutense de Madrid
Fac. de Informática, Ciudad Universitaria, 28040 Madrid
SPAIN
rmoreno@dacya.ucm.es http://www.dacya.ucm.es/rafa

Abstract: - This paper explores the application of Grid computing technology for solving compute-intensive problems in finance. In particular, we propose a high-throughput parallel version of the Monte Carlo algorithm for portfolio VaR simulation, based on a master-worker paradigm, which runs in a Grid environment an obtains a substantial time reduction with regard to the serial algorithm, by exploiting the idle periods of the existing computational resources, like PC's or workstations.

Key-Words: - Grid Computing, High-Throughput Computing, Computational Finance, Value-at-Risk, Monte Carlo.

1 Introduction

The computational issues of common finance industry problems, such as option pricing, portfolio optimization, or risk analysis, require the use of high-performance computing systems and algorithms. Traditional solutions to these problems involve the utilization of parallel supercomputers [1], which exhibits several drawbacks: high cost of the systems, highly qualified personal for administration and maintenance, difficult programming environments (distributed memory or message passing), etc.

In this context, the Grid is emerging as a new technology for next generation of high-performance computing solutions. This technology is based on the efficient sharing and cooperation of heterogeneous, geographically distributed computing resources, like CPUs, clusters, multiprocessors, storage devices, databases, scientific instruments, etc. Computational grids have been successfully experienced for solving grand challenge problems in science and engineering, however the use of this technology for high computational demand applications in economics and finance has not been deeply explored.

Within the range of computational finance applications, the Monte Carlo simulation based models are used to solve a broad variety of problems. Monte Carlo applications are inherently parallel, since random samples can be generated and evaluated independently. In this paper we explore the use of grid technology to implement a parallel version of a Monte Carlo simulation algorithm for estimating the Value-at-Risk (VaR) of a portfolio. This grid-based implementation of the Monte Carlo algorithm is probed to be highly efficient, since it obtain a significant simulation time reduction with respect to the serial version of the algorithm. Moreover, it can be considered a low-cost solution, because it does not required the utilization of complex and expensive parallel computer, but it makes use of the idle existing computing resources, like PCs or workstations.

2 Grid Computing Technology

According to the definition of Kesselman and Foster [2] "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." In this context, a grid user will have the ability of look for and hire out computational resources around the world, based on his current computing needs. On the other hand, companies or individuals proprietary of computing systems, can benefit from the idle periods of such resources by making them available for renting on the grid.

2.1 Grid components

Conceptually the grid is made up of three key components [3], as shown in figure 1:

- *The grid fabric.* This component embraces all the distributed resources that are available from anywhere on the Internet. The grid fabric can consist of different kinds computer systems (PCs, workstations, clusters, supercomputers, etc.) running various operating systems, as well as other components like specialized computing devices, storage systems, scientific instruments, etc.
- *The grid middleware.* The main functionality of the grid relies on a complex software infrastructure, called grid middleware. This middleware includes different services for resource discovery, remote job execution and monitoring, remote storage access, security and quality of service, etc.
- *Grid applications.* Typical grid applications are computing or data intensive applications, like simulations, parameter searches, NP-complete problems, etc., which demand massive computing power, or accessing to large data sets. Grid enabled applications come from many different areas of science and engineering, like high energy physics, tomography, drug design, climate modeling, aerodynamic simulation, financial analysis, etc.

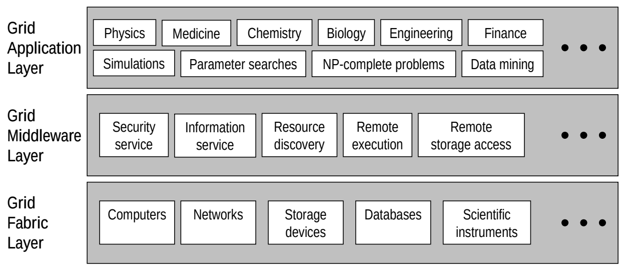


Fig. 1. Grid layered architecture

2.2 The Globus Toolkit

The Globus Toolkit [4] has become a de facto standard middleware in grid computing environments. Globus comprises a set of components that implement the basic grid services for security, information, resource management, and remote data access. Following, we briefly describe the basic functionality of these Globus services.

- *GSI (Globus Security Infrastructure).* Grid operations involve the access of users to remote systems, which can belong to a different organization, and the transmission of information between grid machines. In this context it is necessary a security infrastructure supporting user authentication and encrypted communications.
- *GRAM (Globus Resource Allocation Manager).* GRAM allows the users to run jobs in remote resources. It processes the user requests for remote application execution, allocates the required resources, and manages the active jobs.
- *MDS (Metacomputing Directory Service).* MDS allows the users to discover and obtain information to about grid resources. The user can ask the MDS for static computer information (CPU type, operating system version, number of processors, memory size, etc.), dynamic computer information (load average, free memory, …), storage system information, etc.
- *Global Access to Secondary Storage (GASS).* GASS provides the users basic access to remote files. Operations supported by GASS include remote file read, remote file write.

2.2 The GridWay Framework

Although Globus provides the basic tools for grid operation, the user is responsible for manually scheduling jobs to the grid. In this context, the GridWay framework [5] has been developed as a Globus compatible environment, which simplifies the user interfacing with the Grid, and provides the mechanisms for efficient job scheduling and execution on the Grid with dynamic adaptation to changing conditions. From the user point of view, the GridWay framework consists of two main components (see Figure 2).

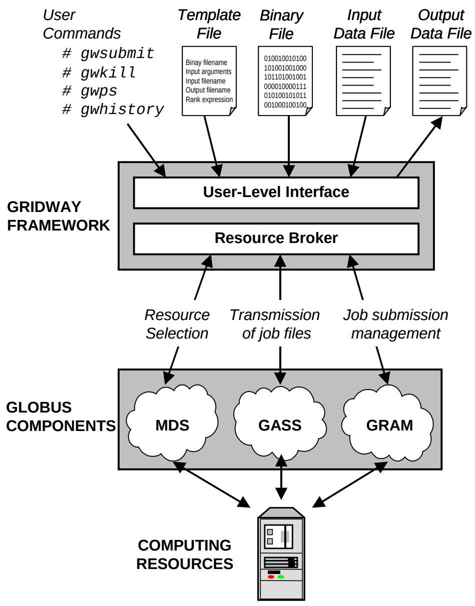


Fig. 2. Gridway and Globus interoperability

- *The command-Line User Interface.* This interface significantly simplifies the user operation on the Grid by providing several user-friendly commands for submitting jobs to the Grid along with their respective input files, controlling, and monitoring the state of the jobs.

- *The personal Resource Broker.* Each user interacts with its own personal resource broker, which communicates with the different Globus components, and is responsible for resource discovering, scheduling and submitting the user jobs, monitoring job performance, and migrating jobs when it is required. The scheduling policy is based on a greedy approach, so that the scheduler tries to maximize the optimization criterion specified by the user for each individual job.

3 Monte Carlo Value-at-Risk

To illustrate the application of grid technology to solve complex computational problems in finance we have implemented a Monte Carlo simulation algorithm for computing the Value-at-Risk (VaR) of a portfolio [6].

The VaR of a portfolio can be defined as the maximum expected loss over a holding period, Δt, and at a given level of confidence c, i.e.,

$$\Pr ob\{\Delta P(\Delta t) < VaR\} = 1 - c \tag{1}$$

where $\Delta P(\Delta t) = P(t + \Delta t) - P(t)$ is the change in the value of the portfolio over the time period Δt.

Several methods for computing VaR have been proposed:

- *Parametric models*, like asset-normal VaR, delta-normal VaR, or delta-gamma-normal VaR.
- *Non-parametric models*, like historical simulation or Monte Carlo (MC) simulation.

The MC approach is based on simulating the changes in the values of the risk factors, and revaluating the entire portfolio for each simulation experiment. The main advantage of this method is its theoretical flexibility, because it is not restricted to a given risk term distribution and the grade of exactness can be improved by increasing the number of simulation.

For simulation purposes, the evolution of a single asset, $S(t)$, can be modeled as a random walk following a Geometric Brownian Motion:

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t) \tag{1}$$

where dW_t is a Wiener process, μ the instantaneous drift and σ the volatility of the asset.

Assuming a lognormal distribution and using the Itô's Lemma, the expression (2) can be transformed in an Arithmetic Brownian Motion:

$$d(\ln S(t)) = (\mu - \sigma^2/2)dt + \sigma dW(t) \tag{3}$$

Integrating the previous expression over a finite time interval, δt, we can reach an approximated solution for estimating the price evolution of $S(t)$:

$$S(t+\delta t) = S(t)e^{(\mu - \sigma^2/2)\Delta\delta + \sigma\eta\sqrt{\delta t})} \tag{4}$$

where η is a standard normal random variable.

For a portfolio composed by k assets, $S_1(t)$, $S_2(t)$, …, $S_k(t)$, the portfolio value evolution can be modeled as k coupled price paths:

$$S_i(t+\delta t) = S_i(t)e^{(\mu_i - \sigma_i^2/2)\delta t + \sigma_i Z_i\sqrt{\delta t})} \tag{5}$$

where Z_i are k correlated random variables with covariance

$$\operatorname{cov}(Z_i, Z_j) = \operatorname{cov}(S_i, S_j) = \rho_{ij} \tag{6}$$

To transform a vector of k uncorrelated normally distributed random variables $\overline{\eta} = (\eta_1, \eta_2, ..., \eta_k)$ into a vector of n correlated random variables $\overline{Z} = (Z_1, Z_2, ..., Z_k)$, we can use the Cholesky descomposition of the covariance matrix, R:

$$R = AA^T \tag{7}$$

where $R = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1k} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2k} \\ \vdots & \vdots & & \vdots \\ \rho_{k1} & \rho_{k2} & \cdots & \rho_{kk} \end{pmatrix}$ is assumed to be symmetric and positive definite, A is a lower triangular matrix and A^T is the transpose of A.

Then, applying the matrix A to $\overline{\eta}$ generates the new correlated random variables $\overline{Z}$

$$\overline{Z} = A\,\overline{\eta} \tag{8}$$

To simulate an individual portfolio price path for a given holding period Δt, using a m-step simulation path, it is necessary to evaluate the price path of all the n assets in the portfolio at each time interval: $S_i(t+\delta t)$, $S_i(t+2\delta t)$,…, $S_i(t+\Delta t)=S_i(t+m\delta t)$, $\forall\, i=1, 2, ..., n$, where δt is the basic simulation time-step, $\delta t = \Delta t/m$.

For each simulation experiment, j, the portfolio value at target horizon is

$$P_j(t+\Delta t) = \sum_{i=1}^{k} w_i S_{i,j}(t+\Delta t), \quad \forall j = 1,...,N \tag{9}$$

where w_i is the relative weight of the asset S_i in the portfolio, and N is the overall number of simulations.

The changes in the value of the portfolio are

$$\Delta P_j(\Delta t) = P_j(t+\Delta t) - P(t) \quad \forall j = 1,...,N \tag{10}$$

The portfolio VaR can be measured from the distribution of the N changes in the portfolio value at the target horizon, taking the $(1\text{-}c)$-percentile of this distribution, where c is the level of confidence.

The convergence error (ε_N) of the MC estimation can be approximated to

$$\varepsilon_N = \sqrt{\frac{1}{(N-1)}\sum_{j=1}^{K}(P_j - \widehat{P})^2} \tag{3}$$

where $\hat{P}$ is the average value of the portfolio at target horizon. Hence, the convergence error of the MC estimation decreases at the order $O(1/\sqrt{N})$.

4 Monte Carlo Simulation on a Grid

The main drawback of Monte Carlo simulation is the computational cost, since a large number of price paths may be necessary to evaluate in order to obtain accurate results, and hence the simulation time may be extremely long. To lessen this problem two kind of techniques are mainly used:

- *Variance reduction techniques*. These techniques try to reduce the number of simulations required to reach a precise result. There is a good variety of variance reduction techniques [7], as for example control variate, importance sampling, stratified sampling, conditional Monte Carlo, etc. Although the efficiency of these techniques has been proved in several works, their use results in a substantial increase of the VaR model complexity. Nevertheless, the analysis of these techniques is beyond the scope of this paper.
- *Parallel Monte Carlo implementation.* The second alternative to reduce the MC simulation time is implementing a high throughput version of the algorithm [7], capable of distributing the portfolio price path computations in parallel over different computing resources. These computer resources can be different processors within a parallel supercomputer, can be different computers within a cluster, or, as we propose in this work, can be different computing resources within a grid environment.

Since MC simulation methods rely on the generation of random number sequences, we first analyze the most common methods for generating random numbers in a parallel environment, and then we propose a methodology for distributing the computations on the grid, based on a master-worker paradigm.

4.1 Parallel random number generation

An ideal random number generator (RNG) should meet with several conditions: the sequences satisfy statistical tests for randomness; are uniformly distributed; are not correlated; have long period; are reproducible; are fast; are portable; can be changed by adjusting seed; and require limited memory. Certainly, it is impossible for a computer to generate random number sequences satisfying all these requirements simultaneously. However, for practical purposes, MC algorithms can use pseudo-random number sequences, provided that the period of the stream is larger than the total number of random numbers needed by the application, and the correlations are sufficiently weak. The most common serial pseudo-random number generators employed in Monte Carlo applications are the Linear Congruential Generator (LCG) and the Lagged Fibonacci Generator (LFG).

LFG has become very popular because it is easy to implement, the computation of the sequence exhibits low computational cost, and it satisfies excellently the statistical tests for randomness.

On the other hand, parallel random number generators should satisfy the same conditions than serial RNGs, and some extra requirements: the streams should not exhibit inter-processor correlation; the algorithm must be scalable, i.e., it must work for any number of processors; the communication between processors must be minimum.

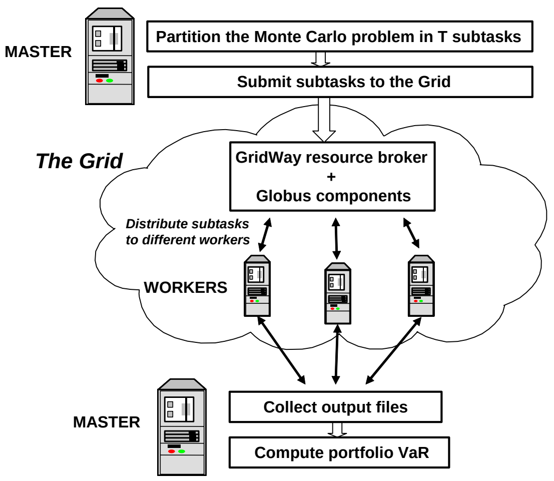
In the last few years, a variety of parallel pseudo-random number generators based on parameterization have been developed and tested. These algorithms have been implemented and freely distributed in the software package SPRNG (Pseudo Random Number Generators Library) [8], which includes different parallel pseudo-random number generators (linear congruential generators, additive and multiplicative lagged Fibonacci generators, and combined multiple recursive generators)

4.2 The master-worker computing paradigm

The Master-Worker (MW) paradigm has been broadly used to solve a variety of large-scale problems in a parallel or distributed environment, like tree-search algorithms, genetic algorithms, or Monte Carlo simulations.

As shown in Figure 3, the master application partitions the problem in *T* identical subtasks or jobs, which are distributed among different workers. The master can monitor the activity of the workers and detect the job completion. As the workers complete their jobs, the master must collect all the output files. In our particular problem, each job computes *N/T* different portfolio values, where *N* is the overall number of simulations.

The master uses the GridWay interface to submit the array of T subtasks to the Grid, trying to maximize some optimization criterion specified by the user. The GridWay framework interacts with Globus to resource discovering, selecting resources for job execution, transmitting job files, and monitoring and controlling job progress.


MASTER
Partition the Monte Carlo problem in T subtasks
Submit subtasks to the Grid
The Grid
GridWay resource broker + Globus components
Distribute subtasks to different workers
WORKERS
MASTER
Collect output files
Compute portfolio VaR
Fig. 3. The master-worker paradigm

The submission of each subtask involves three stages:

- *Prolog*: the application binary file and the input data files are transmitted to the worker.
- *Execution*: the worker performs the number of simulations specified by the master.
- *Epilog*: the worker sends back the output file to the master

When all the sub-tasks are completed, the master merges the output files and computes the portfolio VaR with confidence *c*.

It is important to note that a grid is a heterogeneous environment, where workers can exhibit different characteristics and performance. This fact differs from other parallel environments, like a multiprocessor system or a workstation cluster, where all the processors usually have similar features. Consequently, in a grid context, job scheduling becomes an important challenge [10], since the overall execution time is enforced by the slowest worker.

3 Results

To evaluate the proposed high throughput approach for Monte Carlo VaR, we have chosen a 76-asset portfolio[1], and a time horizon of 10 days. In order to reduce the relative convergence error under 2%, we have performed 4 million price-path simulations using a one day time-step. If we run a serial version of the algorithm on the master computer, a Pentium III at 600Mhz, the 4 million simulations take around 25 minutes in executing.

In order to reduce this time we have implemented a master-worker version of the algorithm, which run on the distributed resources of our heterogeneous grid testbed. The main features of these resources are summarized on Table 1. A multiplicative lagged Fibonacci generator has been used for parallel random number generation.

Host	Processor model	CPU Clock (MHz)	Mem. (MB)	OS	Peak Perform. (GFLOPS)
Master	PC P-III	600	256	Linux	0,6
Worker #1	PC P-IV	2400	512	Linux	2,4
Worker #2	PC P-IV	2400	512	Linux	2,4
Worker #3	Sun Blade 100	502	256	Solaris	1,0
Worker #4	Sun Enterp. 250	250 x 2 CPUs	256	Solaris	1,0

Table 1. Main features of grid resources

We have divided the problem in 40 identical subtasks, each one performing 100,000 simulations, and we have achieved different experiments, whose results are summarized in Fig. 4[2].

In the experiment #1, all the 40 subtasks are submitted to the Worker #1 (PC Pentium IV 2,4 Ghz). Although there is an important time reduction with respect to completion time in the master (6,5 minutes against 25 minutes), thanks to the higher performance of the worker, it is important to note that the *prolog* and *epilog* times are very significant (23% of the overall time, in average), and cannot be ignored. This fact is evidenced by the experiment #2, where the subtasks are distributed between workers #1 and #2 (20 subtasks each). However, even tough both workers are identical machines, the completion time reduction does not reach a 30%, since the *prolog* and *epilog* times do not diminish in the same proportion than the execution time.

Experiments #3 and #4 highlight the relevance of scheduling in a heterogeneous environment. In the experiment #3, scheduling is done without regarding to the worker performance, so that the same number of tasks are distributed among all the four workers (10 subtasks each one). With this sub-optimal scheduling the completion time get worse than the previous experiments, since it is dominated by the slowest workers. In such an environment, it is important to distribute the tasks in proportion to the performance of each worker, as done in the experiment #4, where workers #1 and #2 execute 15 subtasks each one, while workers #3 and #4 execute 5 subtasks each one. As we can observe, this scheduling improves the execution time with regard to the previous experiments.

[1] The composition of the portfolio has been taken from the Citiequity Euroland Fund Euro, managed by Citibank.

[2] All the experiments have been performed assuming

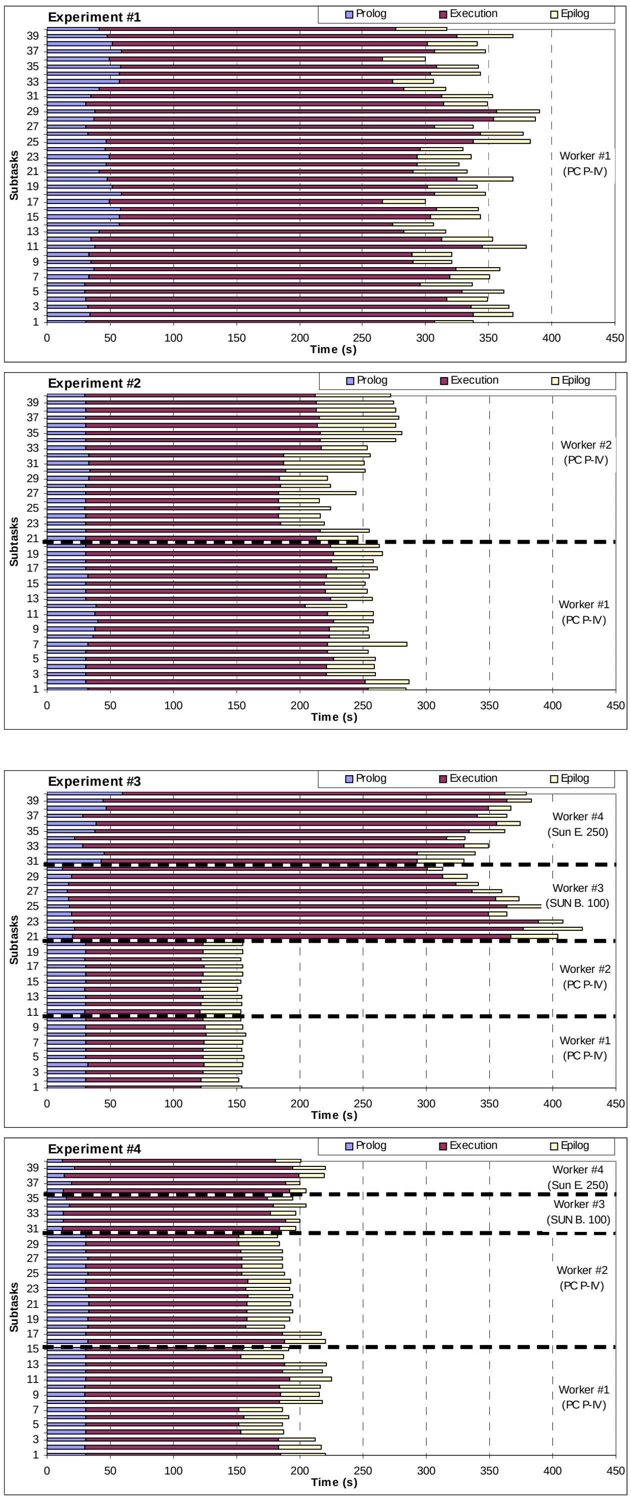


Fig. 4. Experimental results

4 Conclusion and future work

In this paper we have shown how modern grid technologies can be used to provide high-throughput solutions for classical finance problems, like Monte Carlo VaR estimation. The exploitation of the idle periods of the existing computational resources brings new opportunities for solving compute-intensive finance problems at a low-cost.

Several important issues will be explored in future works: adapting the scheduling to dynamic and changing grid conditions, like variations in the workload of the workers, which can reduce their effective performance, or variations in the file transmission times (*prolog* and *epilog* periods) due to different network bandwidths, improving fault tolerance under unexpected worker fails, etc.

References:

[1] S. A. Zenios, High-Performance Computing in Finance: The las 10 years and the next, *Parallel Computing*, No. 25, 1999, pp. 2149-2157.

[2] Foster, I., Kesselman, C., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1998.

[3] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, PhD thesis, Monash University (Melbourne – Australia), 2002, chapter 2, pp. 9-23.

[4] I. Foster, C. Kesselman, The Globus Project: A Status Report. *In Proc. Heterogeneous Computing Workshop,* IEEE Press, 1998, pp. 4-18.

[5] E. Huedo, R.S. Montero, I.M. Llorente, *An Experimental Framework For Executing Applications in Dynamic Grid Environments*, NASA-ICASE Technical Report 2002-43.

[6] P. Jorion, *Value at Risk: The New Benchmark for Managing Financial Risk*, McGraw-Hill (2nd edition), 2000.

[7] P. Glasserman, P. Heidelberger, P. Shahabuddin, Variance Reduction Techniques for Estimating Value-at-Risk, *Management Science*, Vol. 46, October 2000, pp. 1349-1364.

[8] J. Basney, R. Raman, M. Livny, High Throughput Monte Carlo. *Proc. of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, Texas, March 1999.

[9] M. Mascagni, A. Srinivasan, Algorithm 806: SPRNG: a scalable library for pseudorandom number generation. *ACM Transactions on Mathematical Software (TOMS)*, Vol. 26, No. 3, September, 2000, pp. 436-461.

[10] R. Moreno, Job Scheduling and Resource Management Techniques in Dynamic Grid Environments, *Proc. of the 1st European Across Grids Conference* (published in CD-ROM), 2003.