Portfolio and investment risk analysis on global grids

Rafael Moreno-Vozmediano [b], Krishna Nadiminti [a], Srikumar Venugopal [a],
Ana B. Alonso-Conde [c], Hussein Gibbins [a], Rajkumar Buyya [a],*

[a] *Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering,
The University of Melbourne, VIC 3010, Australia*
[b] *Department of Computer Architecture, Universidad Complutense de Madrid, 28040 Madrid, Spain*
[c] *Department of Business Administration (Finance), Universidad Rey Juan Carlos, 28032 Madrid, Spain*

Abstract

The financial services industry today produces and consumes huge amounts of data and the processes involved in analysing these data have large and complex resource requirements. The need to analyse the data using such processes and get meaningful results in time, can be met only up to a certain extent by current computer systems. Most service providers attempt to increase efficiency and quality of their service offerings by stacking up more hardware and employing better algorithms for data processing. However, there is a limit to the gains achieved by using such an approach. One viable alternative would be to use emerging technologies such as the Grid. Grid computing and its application to various domains have been actively studied by many groups for more than a decade now. In this paper we explore the use of the Grid in the financial services domain; an area which we believe has not been adequately looked into.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Grid computing; e-Science; Gridbus resource broker; Finance application

1. Introduction

Investments in stocks almost always involve a risk-reward trade off. To get higher returns on investment, an investor must be prepared to take on a higher level of risk. Investors aim to optimise their investment portfolio in order to minimise the risk and maximise the returns. However, there are many variables involved in portfolio optimization and therefore, it is a very computationally-intensive process. In this paper we explore the use of Grid technology to implement a distributed version of a portfolio optimization method, based on Value-at-Risk (VaR) estimation by means of Monte Carlo simulation.

The computational issues of common finance industry problems, such as option pricing, portfolio optimization, risk analysis, etc. requires the use of high-performance computing systems and algorithms. Traditional solutions to these problems involve the utilization of parallel supercomputers [1,13,24]. This approach has several drawbacks such

---

* Corresponding author.
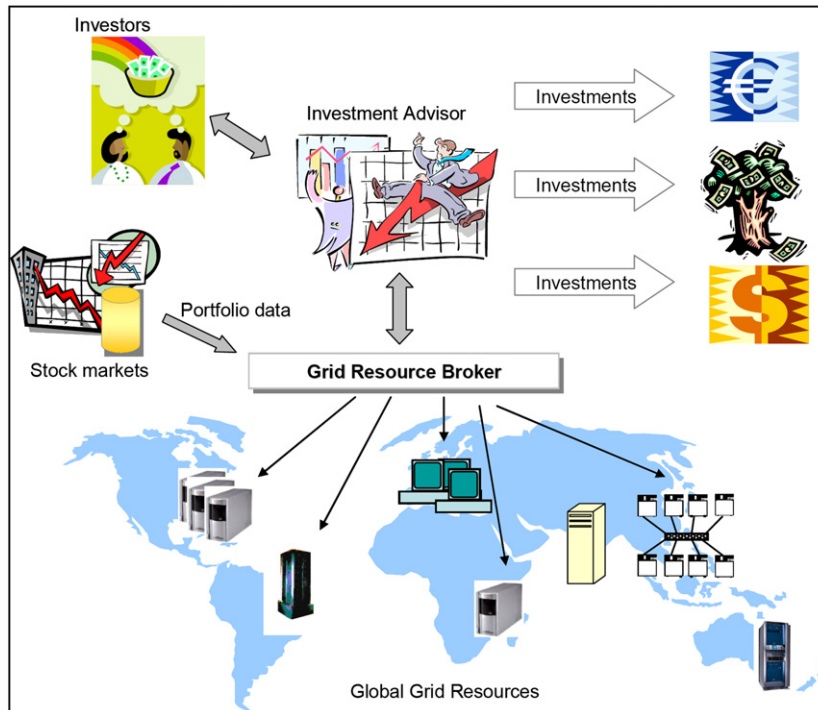 *E-mail address:* raj@csse.unimelb.edu.au (R. Buyya).

Fig. 1. Simple scenario illustrating the use of the Grid in financial markets for portfolio analysis.

as high cost of the systems, highly qualified personnel for administration and maintenance and difficult programming environments (distributed memory or message passing).

In this context, Grid computing [9] is emerging as a promising model for the next generation of distributed computing solutions. This technology is based on the efficient sharing and cooperation of heterogeneous, geographically distributed resources such as CPUs, clusters, multiprocessors, storage devices, databases and scientific instruments. Computational grids have been successfully used for solving grand challenge problems in science and engineering. However, the use of this technology for computationally demanding applications in economics and finance has not been deeply explored.

A simple scenario for using the Grid in financial markets is shown in Fig. 1. As more data is produced by stock markets, this data is fed into the Grid, and analysed using the various Grid resources. A Grid resource broker acts as an access point to the Grid for various investors who wish to carry out portfolio analysis to help them optimize their financial portfolio, make better investment decisions and eventually reap the benefits.

The rest of the paper is organized as follows. Section 2 describes the portfolio optimization method and the VaR application. Section 3 presents a brief outline of some efforts to apply distributed computing to finance problems and also related work in the area of applied Grid computing in other fields such as science. Section 4 talks about the Grid technologies that were used in our experiments. Section 5 deals with how the "VaR" application was Grid-enabled. In Section 6, we describe the experimental setup we used for evaluating the benefits of Grid-enabling the optimization process. In Section 7, we present the results of the experiments conducted. Finally, Section 8 concludes with a reflection on the whole experiment and the lessons learnt therein.

2. Application description

2.1. Value-at-Risk based portfolio optimization

The aim of this section is to describe the VaR application, identify its computational complexity, and illustrate how it can benefit from being Grid-enabled.

Value-at-Risk (VaR) [6] is an important measure of the exposure of a given portfolio to different kind of risks inherent in financial environments, which can be used for portfolio optimization purposes.

Given a portfolio P composed by k assets $\overline{S} = \{S_1, S_2, \ldots, S_k\}$, and $\overline{w} = \{w_1, w_2, \ldots, w_k\}$ the relative weights or positions of the assets in the portfolio, the price of the portfolio at time t is given by:

$$P(t) = \sum_{i=1}^{k} w_i \cdot S_i(t).$$

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

$$\text{Prob}\{\Delta P(\Delta t) < \text{VaR}\} = 1 - c,$$

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.

In this context, the portfolio optimization problem can be stated either in terms of wealth maximization or in terms of risk minimization. If we consider the wealth maximization criteria, the optimization problem is finding the portfolio composition vector $\overline{w}$ which maximizes the expected portfolio yield $\Delta P(\Delta t)$, subject to a given constraint on VaR:

$$\text{VaR} \leqslant V$$

and

$$\sum_{i=1}^{k} w_i = 1.$$

On the other hand, if we consider the risk minimization criteria, the optimization problem is finding the portfolio composition vector $\overline{w}$ which minimizes the expected portfolio VaR, subject to a given constraint on yield:

$$\Delta P(\Delta t) \geqslant Y$$

and

$$\sum_{i=1}^{k} w_i = 1.$$

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 portfolio assets, 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 simulations.

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

$$dS(t) = \mu S(t)\, dt + \sigma S(t)\, dW(t),$$

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

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

$$d\big(\ln S(t)\big) = \big(\mu - \sigma^2/2\big)\, dt + \sigma\, dW(t).$$

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

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

where η is a standard normal random variable.

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

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

where Z_i are k correlated random variables with covariance

$$\mathrm{cov}(Z_i, Z_j) = \mathrm{cov}(S_i, S_j) = \rho_{ij}.$$

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

$$R = AA^T,$$

where

$$R = \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1k} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{k1} & \rho_{k2} & \cdots & \rho_{kk} \end{pmatrix}$$

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 $\bar{\eta}$ generates the new correlated random variables $\bar{Z}$

$$\bar{Z} = A\bar{\eta}.$$

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:

$$S_i(t + \delta t), S_i(t + 2\delta t), \ldots, S_i(t + \Delta t) = S_i(t + m\delta t), \quad \forall i = 1, 2, \ldots, k,$$

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

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

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

$$\Delta P_j(\Delta t) = P_j(t + \Delta t) - P(t), \quad \forall j = 1, \ldots, N.$$

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

The problem of portfolio optimization problem is a complex computational consuming problem, since this MC simulation must be achieved for different portfolio compositions vector $\overline{w}$, in order to find that one which maximizes yield or minimizes risk. There are several techniques for limiting the solution space, and shortening the overall simulation time, although many times they fall on local minima solutions.

So, in practice, it could be necessary to simulate different weight compositions (several thousand scenarios), more complex portfolios (several hundred assets), more price paths (several millions), or longer holding periods. However, increase in the number of parameters also increases the simulation time significantly and running several scenarios could potentially take several hours or even days on a single computer.

Thus, the long turnaround time of the simulations motivates the use of High-Performance Computing (HPC) resources within the domain of portfolio analysis. However, the variable nature of such workloads makes it difficult to provision the right amount of resources for running them. Therefore, on demand allocation of resources is required to handle expansions and contractions in the workload.

3. Related work

In recent times, the promise of Grid computing has led researchers and developers to apply the technology on different scales to a wide range of domains such as Bio-informatics [12], High Energy Physics [5,16], Neurosciences [4], Language Processing [14], and Earth Sciences [2]. A lot of groups that have made efforts towards scaling up their applications from Clusters to Grids come from the scientific community. In the commercial world, the area of financial services can benefit hugely from distributed computing. Some companies in the finance business have already reaped good benefits from distributing their analysis and other resource intensive applications across enterprise clusters [17–19].

Grids are the next logical step beyond clusters, and provide a better solution for large-scale compute and data intensive applications, spanning across multiple organisations with different policies and varying types of resources. The sharing of such heterogeneous resources in a service-oriented market paradigm will only benefit all involved parties due to the vastly higher potential of the Grid. Also, a owning and managing a dedicated cluster may not always be feasible in all cases. One solution to obtaining resources only when needed is to use a Grid which provides services on-demand.

One of the many different approaches to achieving performance gains is to actually rewrite an application using Message Passing Interface (MPI) or similar paradigms to distribute the work across multiple processors. In the context of computational economics and finance, one such work is described by Abdelkhalek and Bilas [1]. However, this involves a lot of effort and time and the application cannot adapt itself well to changing conditions as are found in Grids. The approach presented in this paper of composing the application as a bag of independent tasks and letting a resource broker execute them not only eliminates the need to rewrite applications but also offloads the parallelization logic on to the broker thus isolating the application developer from the need to factor in the heterogeneous Grid environments. Also, the resource broker is capable of allocating resources depending on varying application requirements thus enhancing scalability and adaptability of the process.

4. Background grid technologies

The computational grid is enabled by the use of software services known as Grid-middleware. These services make possible secure and uniform access to heterogeneous resources to execute applications. There are many technology options, today for running applications on remote computers that are part of a Grid. These include low-level middleware such as Globus [8], UNICORE (UNiform Interface to COmputing REsources) [20] and Alchemi [15] and user-level middleware or brokers which perform aggregation of Grid services and meta-scheduling, such as the Gridbus broker [16], Nimrod [3], Condor [11] and GRUBER (Grid Resource Usage SLA Broker) [7], etc.

For the purpose of Grid-enabling portfolio optimization, our requirements included a system which automates or makes it easy to conduct the process of distributing the application, deploying and running it on Grid nodes, monitor the progress, handle failures and collate the results of execution. Globus is a good choice of middleware as it is one of the most widely used low-level Grid middleware systems today in both research and commercial areas and has wide community support and an active development group. The Gridbus broker, a user-level middleware that supports the Globus middleware, was chosen for this application as it provides simple mechanisms for rapidly formulating the application requirements and meets the requirements mentioned previously. A brief description of Globus and the Gridbus broker follows.

4.1. The Globus Toolkit

The open source Globus Toolkit is a set of software services and libraries for resource monitoring, discovery, and management, plus security and file management. It facilitates construction of computational Grids and Grid-based applications, across corporate, institutional and geographic boundaries. The toolkit is developed and maintained by the Globus Alliance, which includes the Argonne National Laboratory, USA and others. It allows secure access to remote computers via GSI (Grid-security infrastructure) and makes the node a part of the Grid, while preserving the autonomy of the node by using locally set policies to decide who can access the services offered and when. The toolkit includes software for security, information infrastructure, resource management, data management, communication,

fault detection, and portability. It is packaged as a set of components that can be used either independently or together to develop applications.

4.2. The Gridbus Broker

The Gridbus service broker [16] is a flexible open-source platform-independent resource brokering system, implemented in Java, which provides brokering services for distributed execution of applications on various low-level middleware systems including Globus, UNICORE, Alchemi, XGrid [23], and queuing systems such as PBS (Portable Batch System) [21], and SGE (Sun Grid Engine) [22]. It hides the complexity of the Grid by translating a bag-of-independent-tasks or parameter-sweep type application into jobs that can be scheduled to be executed on resources, monitoring those jobs and collating the results of the execution when finished. The broker acts as a user-agent and makes scheduling decisions on where to place the jobs on the Grid depending on the computational resources charac-

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xpml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="XMLInputSchema.xsd">
        <parameter name="scenario" type="integer" domain="range">
                <range from="0" to="99" type="step" interval="1"/>
        </parameter>
        <requirement type="node">
                <copy>
                        <source location="local" file="cholesky.dat" />
                        <destination location="node" file="cholesky.dat" />
                </copy>
                <copy>
                        <source location="local" file="volat.dat" />
                        <destination location="node" file="volat.dat" />
                </copy>
                <copy>
                        <source location="local" file="input.dat"/>
                        <destination location="node" file="input.dat"/>
                </copy>
                <copy>
                        <source location="local" file="var" />
                        <destination location="node" file="var" />
                </copy>
        </requirement>
        <task type="main">
                <copy>
                        <source location="local" file="positions\_\$scenario.dat"/>
                        <destination location="node" file="positions\_\$scenario.dat"/>
                </copy>
                <execute location="node">
                        <command value="./var"/>
                        <arg value="\$scenario"/>
                </execute>
                <copy>
                        <source location="node" file="output\_\$scenario"/>
                        <destination location="local" file="output\_\$scenario"/>
                </copy>
                <copy>
                        <source location="node" file="var\_\$scenario"/>
                        <destination location="local" file="var\_\$scenarioyy"/>
                </copy>
        </task>
</xpml>
```

Fig. 2. Application description in XML.

teristics (such as availability, capability, and cost), the users' quality of service requirements such as the deadline and budget, and the proximity of the required data or its replicas to the computational resources.

5. Grid Enabling the VaR optimization application

The VaR application is written in the C language, and is a simple program that is not directly aware of the Grid by itself, that is it was not designed to run as a distributed application.

A single run of the VaR application computes the value-at-risk for a portfolio of k assets, by simulating N price-paths, of the stock movements over a holding period Δt, using a basic time-step of δt. The k assets are defined in a data file, *volat.dat*, with their volatility and drift information. The *cholesky.dat* input data file contains the Cholesky portfolio composition matrix $\overline{w}$. The input parameters N, Δt, and δt are contained in another data file, *input.dat*. The output it produces is a frequency distribution, which is used to get a measure of the portfolio VaR by taking the $1 - c$ percentile of the distribution, where c is the level of confidence.

Grid enabling the VaR application involves running the same application over multiple data sets or input parameters, for simulating different scenarios of stock movements. Each run of VaR is independent of another run and exhibits coarse-grained data parallelism and hence, this application fits nicely into the parameter-sweep paradigm.

To run the application on the Grid using the Gridbus broker, we described the application using the declarative XML-based eXtensible Parametric Modelling Language (XPML) provided by the broker, as it offered an easy way to vary the parameters and rerun the application. XPML allows us to specify the inputs, executable files and outputs generated by the VaR application. The XPML file shown in Fig. 2 describes the application to be consisting of a parameter i ranging from 0 to 99 (i.e. 100 scenarios for computing VaR). The task performed by each job in the application is described by a sequence of commands which copy files and execute the VaR program. The XPML specification largely simplifies and accelerates the process of describing an application to be deployed and run on distributed resources. XPML allows the user to focus on the specifics of the application tasks, and leave the execution details to the resource broker. More details about the specific experiment runs conducted are given in the next section.

6. Experiments and evaluation

To evaluate the benefits the Grid brings to this finance application, we conducted three sets of experiments as shown in Table 1. For our experiments we varied the input parameters Δt (holding period) and δt (time-step) and used $k = 76$ assets and $N = 500,000$ price-paths in which the stocks could vary. The assets were derived from a real investment product and are companies trading on the Madrid Stock Exchange in Spain.

The first set involved running one scenario on one computer, varying the holding period parameter (Δt), with number of simulations $N = 500,000$, number of assets $k = 76$, and a basic time step of $\delta t = 1$ day. These aimed to investigate the effect of varying input parameters on the output VaR computed

Table 2a shows the input parameters of the three experiments from the first set. These simulations were run on a single computer, with Intel P4 processor at 2.5 GHz, 512 MB RAM, and Linux OS.

The second set of experiments conducted aimed to simply confirm that Grid-enabling the VaR application was useful in terms of application performance. Four experiments with varying number of Grid-nodes were performed, keeping the application parameters k, N, Δt, and δt constant. The parameter values used in this set of experiments is shown in Table 2b.

Table 1
Description of experiments

	No. of experiments	Description
Set 1	3	Computes VaR on a single computer, running a single scenario with different values for the Δt (holding period) parameter.
Set 2	4	Evaluates application performance in terms of speed, with fixed job-size (i.e. using same parameters) and varying number of Grid nodes.
Set 3	3	Evaluates application performance with varying job size and same set of Grid nodes by computing VaR on a Grid of 5 nodes, running 100 different scenarios with different values for the Δt (holding period), and enables comparison of the outputs with those from experiment Set 1.

Table 2a
Parameters for simulation experiments 1–3 (set 1)

Set 1	No. of assets (k)	No. of scenarios	No. of simulations (N)	Holding period (Δt) (day)	Basic time step (δt) (day)	No. of time steps $(m) = (\Delta t)/(\delta t)$
Exper. 1	76	1	500,000	1	1	1
Exper. 2	76	1	500,000	5	1	5
Exper. 3	76	1	500,000	10	1	10

Note: Total Investment (USD) = 160.8 million.

Table 2b
Grid application parameters used for the performance experiment with varying number of grid nodes (set 2)

Set 2	No. of assets (k)	No. of scenarios	No. of simulations (N)	Holding period (Δt) (day)	Basic time step (δt) (day)	No. of time steps $(m) = (\Delta t)/(\delta t)$	No. of grid nodes
Exper. 1	76	100	100,000	1	1	1	1
Exper. 2	76	100	100,000	1	1	1	2
Exper. 3	76	100	100,000	1	1	1	3
Exper. 4	76	100	100,000	1	1	1	4

Note: Total Investment (USD) = 160.8 million.

Table 2c
Parameters for simulation experiments 1–3 (set 3)

Set 3	No. of assets (k)	No. of scenarios	No. of simulations (N)	Holding period (Δt) (day)	Basic time step (δt) (day)	No. of time steps $(m) = (\Delta t)/(\delta t)$	No. of grid nodes
Exper. 1	76	100	500,000	1	1	1	5
Exper. 2	76	100	500,000	5	1	5	5
Exper. 3	76	100	500,000	10	1	10	5

Note. Total Investment (USD) = 160.8 million.

Table 3
Resources used in the experiments

Server name	Owner organisation	Configuration	Grid middleware
belle.cs.mu.oz.au	GRIDS Lab, The University of Melbourne	IBM e-Server with 4 CPUs	Globus v.2.4
belle.anu.edu.au	Australian National University, Canberra	IBM e-Server with 4 CPUs	Globus v.2.4
belle.physics.usyd.edu.au	School of Physics, The University of Sydney	IBM e-Server with 4 CPUs	Globus v.2.4
lc1.apac.edu.au	APAC, Canberra	154 node, 156 CPU 2.8 GHz Dell P4 Linux cluster	Globus v.2.4
manjra.cs.mu.oz.au	GRIDS Lab, The University of Melbourne	x86 Linux cluster with 13 nodes	Globus v.4.0

Finally, a third set of experiments, similar to those in the first was conducted on a Grid of 5 nodes. These involved running 100 different scenarios on Grid nodes by varying the input parameter—Δt (holding period). In addition to serve as an indication of application performance with varying simulation parameters, these tests were also useful to get outputs, from distributing the VaR application on the Grid, which could be compared with the outputs obtained running one scenario on a single computer (set 1). The application parameters used for set 3 of experiments is shown in Table 2c.

For the Grid experiments (set 2 and set 3), the Belle analysis test bed Grid [16]—consisting of resources distributed around Australia including Melbourne, Adelaide and Canberra—was used. These systems are interconnected via GrangeNet (Grid and Next generation Network) which is a multi-gigabit network supporting Grid and advanced communication services across Australia.

The broker was deployed on a PC at the GRIDS lab (bart.cs.mu.oz.au), at the University of Melbourne, and the agents were dispatched to other resources at runtime by the Gridbus broker. The performance tests aimed to determine the effect of increasing number of Grid nodes for a fixed job size and number of jobs. The test bed resources are shown in Table 3.
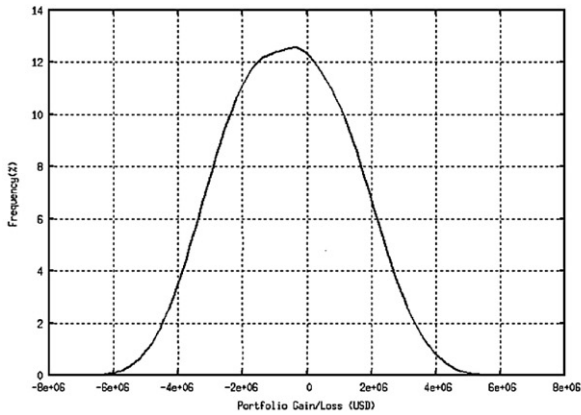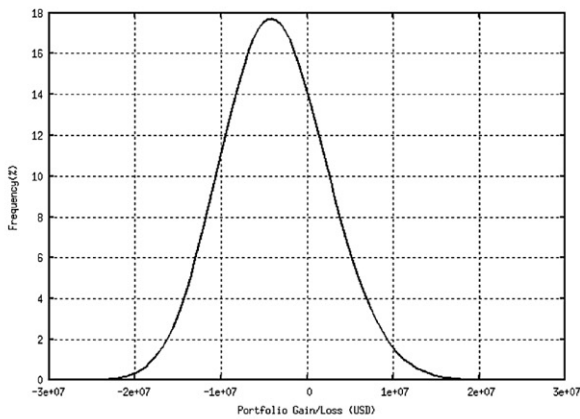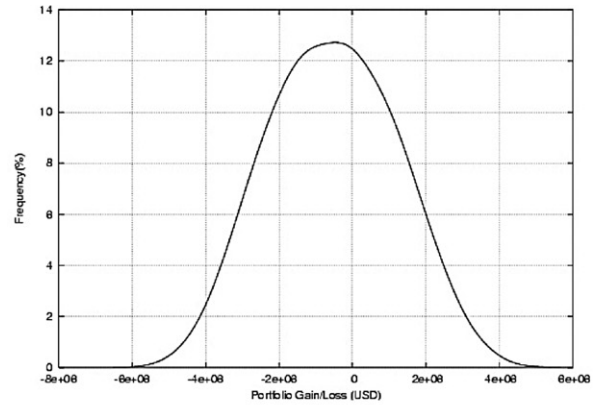
(a) Holding period = 1 day
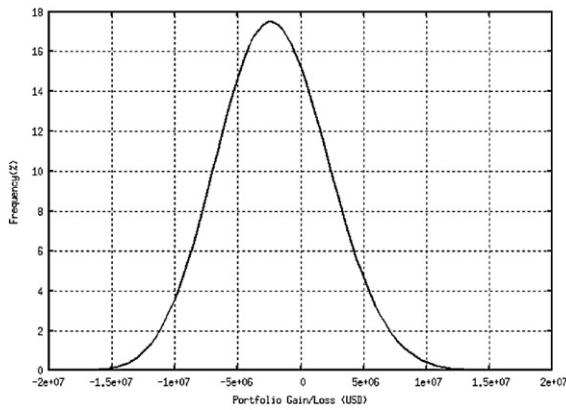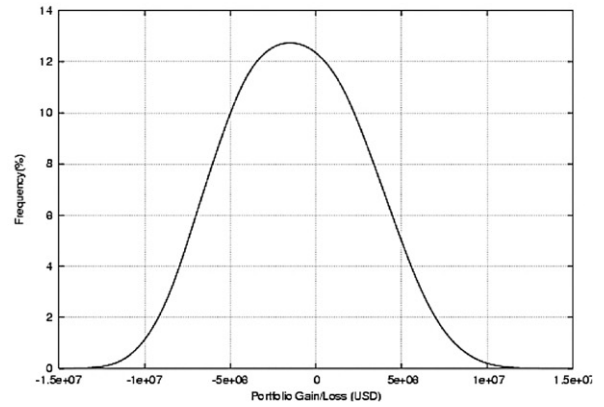

(b) Holding period = 5 days
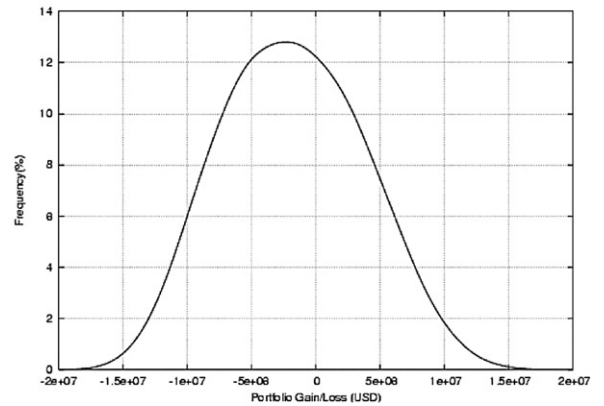

(c) Holding period = 10 days

Fig. 3. Frequency graph for set 1: Experiments 1–3 (# of scenarios = 1).

(a) Holding period = 1 day

(b) Holding period = 5 days

(c) Holding period = 10 days

Fig. 4. Frequency graph for set 3: Experiments 1–3 (# of scenarios = 100).

7. Results

Figures 3(a), (b), and (c) plots the frequency distribution graphs resulting from the simulations of the set 1 experiments 1, 2, and 3 respectively, and Table 4 summarizes some VaR estimation values for different levels of confidence c,

Table 4
VaR values for the three simulation experiments from set 1

Set 1	VaR (USD) $c = 90.0\%$ (million)	VaR (USD) $c = 95.0\%$ (million)	VaR (USD) $c = 97.0\%$ (million)	VaR (USD) $c = 99\%$ (million)
Experiment 1	2.8	3.4	3.8	4.4
Experiment 2	6.7	8.2	9.1	10.8
Experiment 3	10.1	12.1	13.4	15.8

Table 5
Application performance results (set 2)

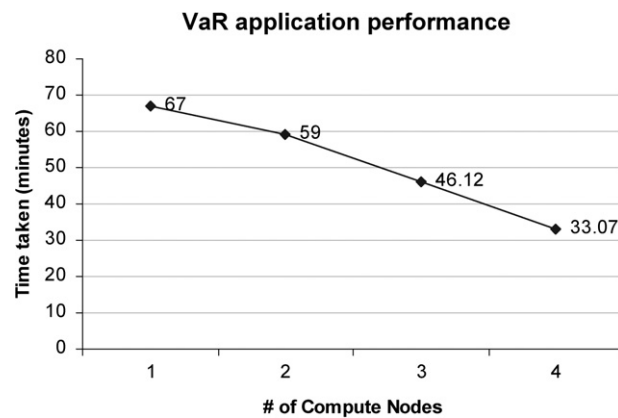| Set 2	No. of simulations (N)	Holding period (Δt)	Basic time step (δt)	No. of grid nodes	Time taken (min)
Exper. 1	500,000	1	1	1	67
Exper. 2	500,000	1	1	2	59
Exper. 3	500,000	1	1	3	46
Exper. 4	500,000	1	1	4	33

Fig. 5. VaR application performance on a Grid with varying Grid number of nodes (set 2).

obtained from the frequency graphs. For example, if we hold the portfolio investment for 1 day the probability of losing more than 5 million dollars is lower than 1% ($c = 99\%$). For 5 days, the probability of losing more than 10 million dollars is around 1% ($c \approx 99\%$), however if we hold the portfolio investment for 10 days, the probability of losing more than 10 million dollars is 10% ($c \approx 90\%$).

The results for the second set of experiments are shown in Fig. 5. This shows the performance of distributing the simulation over different Grid nodes. The main parameters of this simulation are summarized in Table 5. In this case we have simulated 100 different scenarios over a holding period (Δt) of 1 day, with a basic time step (δt) of 1 day, and 500,000 price paths per scenario (N). As we can see, the simulation of 100 scenarios on a single computer takes around 67 min. If we distribute these simulations over different Grid nodes, we can obtain a significant time reduction, for example using 4 computing nodes, the resulting simulation time is halved (33 min).

The results for experiment set 3, shown in Fig. 4(a)–(c), plot the frequency distribution graphs resulting from the simulations of the set 3 experiments 1–3, respectively. These results are similar to those in set 1, as the application input parameters were varied in the same way, except that the experiment was conducted over 100 scenarios in each case, over a Grid. Table 6 summarizes the VaR estimation values for different levels of confidence (c), obtained from the frequency graphs obtained from results of experiments 1–3 of set 3 (running the VaR on the Grid). The values that are produced from running the VaR application on the Grid testbed for 100 scenarios are given in Table 6. This was done by computing 100 different frequency distributions (one for each scenario), and obtaining 100 different VaR values (for a given level of confidence). Then, the lowest (absolute) value of VaR is selected as the scenario with this value is likely to be the best one, because the loss of money of the investment is likely to be lower. Comparing the

Table 6
VaR values for the three simulation experiments from set 3

Set 1	VaR (USD) $c = 90.0\%$ (million)	VaR (USD) $c = 95.0\%$ (million)	VaR (USD) $c = 97.0\%$ (million)	VaR (USD) $c = 99\%$ (million)
Experiment 1	2.5	3.1	3.5	4.1
Experiment 2	5.7	6.9	7.7	9.0
Experiment 3	8.1	9.8	10.9	12.7

Table 7
Application performance results (set 3)

Set 3	No. of simulations (N)	Holding period (Δt) (day)	Basic time step (δt) (day)	No. of grid nodes	Time taken (min)
Exper. 1	500,000	1	1	5	46
Exper. 2	500,000	5	1	5	58
Exper. 3	500,000	10	1	5	134

values in Table 4 (for 1 scenario) and Table 6, we see that those in the latter are lower than the former. While the values are still probabilistic, they are better estimates of the VaR as more scenarios were considered in the evaluation.

Table 7 shows the application performance when run on a Grid of 5 nodes simulating 100 scenarios (constituting 100 Grid jobs), while varying the holding period. Here, we see that there is a sharp increase in the time required to complete the experiment for a holding period of 10 days from that of 5 days. This suggests that the time required increases exponentially rather than linearly with the increase in the holding period.

8. Summary and conclusion

In this paper, we have explored the application of Grid technologies within financial services domain by executing a portfolio optimization application that estimates the Value-at-Risk for a given asset portfolio through Monte Carlo simulation. We have utilised readily available Grid technologies and have shown how with the use of a simple, declarative interface and without rewriting the application, it is possible to execute a sequential, single machine application on aggregated Grid resources.

From the results of our execution, it is evident that running on a Grid reduces the time of execution significantly. Also, a user is able to run the application for more scenarios and receive a better estimation of VaR in a shorter period of time.

However, this is only one of the ways in which Grid technologies can be applied in this domain. While, in our evaluation, the asset values have been provided in a static file, it is possible to visualise a service that will aggregate information from various stock quote providers and perform VaR analysis for a given portfolio over a Grid. This will be able to make use of emerging Service-Oriented Architecture (SOA) paradigm that has been realized in Grid computing through Grid services [10].

References

[1] A. Abdelkhalek, A. Bilas, Parallelization, optimization, and performance analysis of portfolio choice models, in: Proceedings of the 30th International Conference on Parallel Processing, Valencia, Spain, September 3–7, 2001.

[2] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach, D. Williams, High-performance remote access to climate simulation data: A challenge problem for data grid technologies, in: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing, SC'01, Denver, CO, USA, ACM Press, November 2001.

[3] R. Buyya, D. Abramson, J. Giddy, Nimrod-G resource broker for service-oriented grid computing, IEEE Distrib. Syst. Online 2 (7) (November 2001).

[4] R. Buyya, S. Date, Y. Mizuno-Matsumoto, S. Venugopal, D. Abramson, Neuroscience instrumentation and distributed analysis of brain activity data: A case for eScience on global grid, J. Concurrency and Computation: Practice and Experience 17 (15) (December 2005).

[5] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, S. Koranda, GriPhyN and LIGO: Building a virtual data grid for gravitational wave scientists, in: Proceedings of the 11th IEEE international Symposium on High Performance Distributed Computing, HPDC-11, July 24–26, 2002, IEEE Comput. Soc., Washington, DC, 2002.

[6] D. Duffie, J. Pan, An overview of value at risk, J. Derivatives 4 (1997) 7–49.

[7] C. Dumitrescu, I. Foster, GRUBER: A grid resource usage SLA-based broker, in: Proceedings of EuroPar 2005, August 30–September 2, 2005, Lisbon, Portugal.

[8] I. Foster, C. Kesselman, Globus: A metacomputing infrastructure toolkit, Int. J. Supercomputer Appl. 11 (2) (1997) 115–128.

[9] I. Foster, C. Kesselman (Eds.), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann, USA, 1999.

[10] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, Int. J. High Performance Computing Appl. 15 (2001) 200–222.

[11] J. Frey, T. Tannenbaum, I. Foster, M. Livny, S. Tuecke, CondorG: A computation management agent for multiinstitutional grids, in: International Symposium on High Performance Distributed Computing, San Francisco, CA, 2001, pp. 55–67.

[12] H. Gibbins, K. Nadiminti, B. Beeson, R. Chhabra, B. Smith, R. Buyya, The Australian BioGrid portal: Empowering the Molecular Docking Research Community, in: Proceedings of the 3rd APAC Conference and Exhibition on Advanced Computing, Grid Applications and eResearch, APAC 2005, September 26–30, 2005, Gold Coast, Australia.

[13] J. Gondzio, R. Kouwenberg, High-performance computing for asset-liability management, Oper. Res. 49 (6) (2001) 879–891.

[14] B. Hughes, S. Bird, Grid-enabling natural language engineering by stealth, in: Proceedings of HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems (SEALTS) Edmonton, Canada, 2003, pp. 31–38.

[15] A. Luther, R. Buyya, R. Ranjan, S. Venugopal, Alchemi: A .NET-based enterprise grid computing system, in: Proceedings of the 6th International Conference on Internet Computing, ICOMP '05, June 27–30, 2005, Las Vegas, USA.

[16] S. Venugopal, R. Buyya, L. Winton, A grid service broker for scheduling e-Science applications on global data grids, J. Concurrency and Computation: Practice and Experience 18 (6) (2006) 685–699.

[17] An overview of Grid computing in financial services, http://www.jayeckles.com/research/grid.pc, September 2005.

[18] What's so great about Grid?, http://www.banktech.com/features/showArticle.jhtml?articleID=21400554&pgno=5.

[19] Texas Tech University performs stock price analysis in hours instead of days [SAS Institute inc.], http://support.sas.com/rnd/scalability/grid/ttu.html, 2005.

[20] Almond, D. Snelling, UNICORE: Uniform access to supercomputing as an element of electronic commerce, Future Generation Computer Systems 15 (5–6) (October 1999).

[21] Portable Batch System, http://www.openpbs.org/.

[22] Sun Grid Engine, http://www.sun.com/software/gridware/index.xml.

[23] Apple XGrid, http://www.apple.com/server/macosx/features/xgrid.html.

[24] S.A. Zenios, High-performance computing in finance: The last 10 years and the next, Parallel Comput. 25 (13–14) (1999) 2149–2175.