Grid solution for market and counterparty risk calculation. Real life problems from the point of view of the system developer.

Daniel Crespo[*]
Indizen Technologies
Av. Pablo Iglesias, 2. Madrid, Spain
E-mail: daniel.crespo@indizen.com

Jesus Gil
E-mail: jesus.gil@indizen.com

Alberto Gómez
E-mail: alberto.gomez@indizen.com

Enrique Mota
E-mail: enrique.mota@indizen.com

We present a market and counterparty risk calculation system that has been developed for one of the biggest Spanish banks. This system calculates VaR and credit exposures using Monte Carlo simulation. For a bank of this size, the number of portfolios and traded securities involved is so high that calculation time becomes a major concern. We describe the solution that has been adopted, using a commercial grid platform to reduce total calculation time and increase the performance of the system. With the adopted strategy, total calculation time could be reduced from several hours to a few minutes, but it would be desirable to reduce it even further to the point where the system could calculate in real time. The aim of this work is to outline the main problems that need to be addressed in terms of parallelisation, concurrent data access and network capacity in order to increase the throughput of the risk platform to reach the desired performance. We hope that it could be a valuable input for the academic community working on this field.

Grid Technology for Financial Modeling and Simulation
February 3/4, 2006 – Palermo, (Italy)

[*] Speaker

1. Introduction

The authors have been involved in the development and deployment of a risk calculation system for one of the main banks in Europe, and the biggest bank of Spain. The main functionality of this system is to calculate market risk (VaR) and credit exposures for the global treasury department of the bank. The system calculates VaR by Monte Carlo simulation, and given the large number of positions and portfolios, the total calculation process consumes a lot of time and hardware resources.

For this system, total calculation time is a key issue. VaR calculation must be done every day, and total calculation time should not be more than 4 hours. Further than that, it would be desirable to reduce this calculation time to a minimum, and reach a point where the system calculates nearly in "real time". A system that could calculate risk for a whole portfolio in real time could be used for risk-oriented trading, and help the bank make buy-sell decisions on the trading floor, knowing how it would affect its total risk in real time.

When this system was first developed, it was clear that total calculation time would be well over 4 hours unless some effort was made to parallelize the calculations. We decided to implement a grid solution, based on a commercially available grid platform, to divide the total calculation into independent parallel processes run on different machines. Using a grid infrastructure with 30 machines, we have reduced total calculation time by a factor of 20 .

This work describes the grid solution that has been adopted to reach this increase in performance. But our main objective is to outline the problems that still remain unsolved in order to increase the throughput of the system even further, up to the desired real-time solution. We try to pose the questions that need to be answered, from the point of view of a system developer, in order to increase the total performance by yet 2 orders of magnitude. We hope that this can be a valuable input for the academic community working in this field.

2. System Description

The system under consideration has a wide functionality, but in order to keep this work clear, we will consider it as a system that just calculates market risk VaR by Monte Carlo simulation. This calculation is performed in three consecutive steps: simulation, valuation and aggregation.

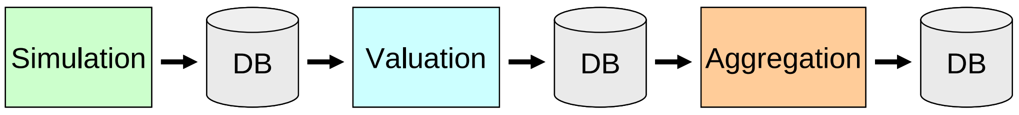

Figure 1. Schematic view of the risk calculation process.

The first step, simulation, consists in generating pseudo-random future scenarios of the economy for a given number of future time horizons, or tenors. For this discussion, it is just important to know that the simulation process generates, and stores, a tri-dimensional matrix of values s_{irt} where *i* denotes the number of scenario, *r* denotes the risk factor, and *t* denotes the time-horizon. The simulation is a key process for risk calculation, because it encapsulates all the economical assumptions about the future evolution of the markets, but it is not a heavy process in terms of calculation, so we won't be concerned about it in this work.

In the second process, the valuation process, the market value of all of the bank's assets and liabilities is calculated in each of the scenarios. The valuation process can be schematically represented as the process of calculating, and storing, a four-dimensional matrix

$$v_{ijtg} = f_j\left(s_{irt}\big|_g\right), \tag{1}$$

where *i* denotes the number of scenario, j denotes the instrument, *t* the time horizon, and *g* denotes the risk group. A risk group is a risk calculation that takes into account the influence of only a given set of risk factors. For example, in the Equity risk group only stock prices will be treated as random variables, the rest of risk factors will be considered fixes. And $f_j(s)$ is the valuation function of instrument j , it calculates the market value of the particular instrument, that will depend of the risk factors. The typical figures for this process in the system once it is in production would be: 10^5 instruments, 10^4 scenarios, 10 tenors and 10 risk groups. This gives a total size of the valuation matrix of 10^{11} values that need to be calculated, and stored.

The last step, the aggregation process, consists in the calculation of the probability distribution of future values for portfolios. A portfolio is just a vector of instruments, and the aggregation can be considered as a matrix-vector product of the form

$$a_{iktg} = v_{ijtg}\, p_{jk}\,, \tag{2}$$

where p_{jk} indicates the position of instrument *j* contained in portfolio *k*. Once we have the probability distribution of the value for each portfolio, we can give risk measures for each portfolio on each time horizon and risk group. A risk measure like VaR is just a percentile of the obtained probability distribution. The typical number of portfolios in this system is 10^3, so the typical size of the matrix of aggregated values a_{iktg} is 10^9.

Valuation and aggregation are both heavy processes. For example, total calculation time using only 10^4 instruments is approximately 4 hours for valuation and 2 hours for the aggregation. The valuation and aggregation processes can both be pictured in terms of data flows as shown in figure 1. They both imply reading some input data from a database, calculating a matrix of values and writing it to the database. The good thing is that, in both cases, each value of the matrix can be calculated, and stored, independently. This means that both valuation and aggregation can be easily divided into independent sub-processes running on parallel in different machines, and one would expect a first-order linear gain in system throughput.

3 Process distribution

For the sake of simplicity, we will discuss only the distribution of the valuation process, the same results apply to the aggregation process. As we commented in the previous section, the valuation process implies calculating a four-dimensional matrix of values v_{ijtg} . Each value in the matrix is independent of the rest, so each of them can be calculated independently.

Assuming that the total calculation time when running as a single process is *C*, if we have a grid infrastructure with *n* calculation nodes, the calculation time can be simplistically assumed to be

$$T = o_c + \frac{C}{n}, \qquad (3)$$

where $o_c << C$ would be the overhead time to set-up each calculation process. Under this approximation, if we wanted to reduce a calculation time of 4 hours (14400 sec) to 15 seconds, we would have to use approximately 1000 processors. And one could think of having as many processors as elements there are in the matrix v_{ijtg} , and reach a minimum calculation time.

Obviously, the assumption in (3) is too simplistic. To start with, if the *n* sub-processes that run simultaneously are all writing (or reading) to the same data repository, there will be an overhead time in establishing data connections that will grow as *n* grows. We could reduce this effect by distributing also the data into *m* independent data repositories, then by increasing accordingly the number of calculation processes *n* , and the data repositories *m*, we could ideally reach the desired level of performance. But then, we also need to take into account the limitations imposed by network band-with. The number of simultaneous messages travelling through the network will grow, at least, proportional to the number of simultaneous calculation processes *n*. Each of these messages will have an overhead associated with it, therefore the total amount of information that needs to be transported by the network grows with *n*, and it could reach a point where band-with starts to be the limiting factor for system throughput. Therefore, assuming that band-with for a single network is limited, it will be necessary to distribute as well the network infrastructure, having different parts of the calculation and storage being performed on independent networks.

Therefore, there are three types of components that need to be distributed accordingly: calculation nodes, data repositories, and network infrastructure. This situation is schematically shown on figure 2.

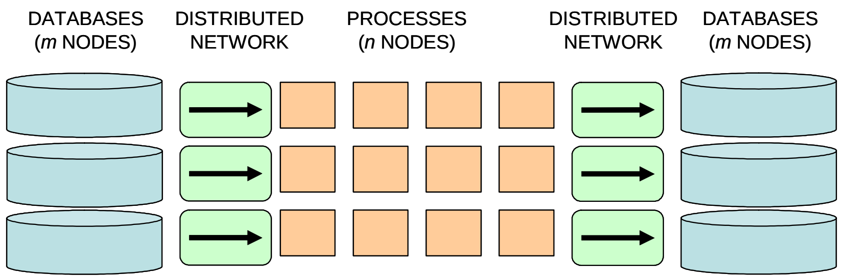

Figure 2. Schematic view of the distributed valuation process, with *n* distributed calculation nodes, *m* data repositories, and a distributed network infrastructure.

4 Key Issues for system design

We have already implemented a GRID infrastructure for this system. But for the time being, we have only distributed the calculation nodes, not the data repository nor the network infrastructure. We have adopted a commercial GRID platform [1], and we have set-up a system with 30 calculation nodes. Each calculation node is a standard PC with 1GB RAM memory and a 2.4 GHz Pentium processor. The valuation process is an application written in C++ installed on each node, and data from every node is stored in the same MySQL server running on a similar PC.

We have run some performance tests for a valuation process, with 10^4 instruments, 10^4 scenarios, 6 tenors and 5 risk groups. Each node performs the total calculation for a particular tenor and a risk group, so we have 30 completely independent processes, one for each node. The gain in performance observed was a factor of 20, and total calculation time passed from 4 hours to 15 minutes. Calculation time is not inversely proportional to the number of nodes; this implies that there are overheads somewhere: in calculation process set-up, in data writing or reading, in network communications, or even in the GRID platform book-keeping of sub-processes.

It would be desirable to reduce calculation time by approximately 2 orders more of magnitude. Now is the moment to design what the GRID infrastructure should be for that full-size system. But, how do we do this? We need to estimate the optimum number of processing nodes, the distribution of data repositories, and the distribution of network infrastructure. We need tools to know in advance the performance of a given GRID infrastructure.

It would be ideal to have a parametric model that would predict performance for a GRID infrastructure. It could take into account things like the overheads in calculation processes, data connections and network communications. It would be ideal to have protocols to measure the values of the parameters of the model on a test GRID infrastructure. Therefore, building a small test system for a particular application, one could estimate the properties of the full-size GRID infrastructure needed to reach a certain performance.

Is this possible? Does it make any sense? What is the correct way to design a GRID infrastructure? Maybe there are already some ideas about this in the academic community, but they are still not reaching the developer community. We hope that this work will help to promote communication between both worlds, and maybe also encourage some research work that can help us design and build better GRID infrastructures.

References

[1] The grid platform currently used is InnerGrid Nytia(tm), by Grid Systems. www.gridsystems.com