Efficient High Performance Computing Framework for Short Rate Models

T.P. Dampahala, H.D.D.D. Premadasa, P.W.W. Ranasinghe, J.N.P. Weerasinghe, K.A.D.N.K. Wimalawarne
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka

Abstract— **Many mathematical calculations in the field of computational finance consume a lot of time and resources for processing. Some of the Short rate models used in quantitative finance which have been taken into consideration in this paper have been optimized for performance within a cluster computing environment. The back-end cluster has been seamlessly integrated with an easy-to-use front-end which can be used by finance professionals who are not aware of the details of the computational and database cluster. Furthermore, many techniques that have been utilized to improve the efficiency of the models have also been described. This paper also describes the generalization of a High Performance Computing Cluster designed for One-factor Short rate models and how it can be used easily to be further extended for other mathematical models in quantitative finance. The ultimate objective is to come up with a generalized framework for quantitative finance.**

I Introduction

The field of high performance computing and computational finance is of special interest to computer scientists as well as economists. Current research is focused on maki ng use of parallel computing to build high performance clusters to tackle the heavy workload of quantitative mathematical models. This research was started with the setting up of a cluster computing environment for the parallel execution of mathematical models used for short rate modeling, namely Vasicek Model [1] and Cox-Ingersoll-Ross Model [2]. Each algorithm along with its calibration routine has been parallelized. Several measures have been adopted to further enhance the level of optimization. Client-server architecture has also been implemented to enable any client from any location to invoke tasks within the cluster, whose services are exposed via a server application. The client was further customized by developing a plug-in for Microsoft Excel 2007 for the benefit of quantitative analysts who will be making use of the system.

The project was steered towards forming a generalized framework where most of the commonly used components are made reusable. This enables the user to extend the framework for other mathematical models used in quantitative finance. Currently, most of the available High Performance Computing implementations for quantitative finance are based on web-services and require the users to possess a certain level of programming capabilities. This initiative distinguishes itself by relying on message parsing and C++ implementations of the models for sheer performance coupled with seamless integration with the MS Excel Plug-in for ease-of-use. This means that the cluster implementation and its intricacies will be hidden from the users.

II Background

A. Parallel processing in Finance

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently. In the case of this research initiative, optimization has been achieved by parallelizing Monte Carlo simulations that are needed in most quantitative models in finance. An advantage here is that most serial Monte Carlo algorithms are readily adaptable to a parallel environment. Furthermore, calibration routines can be carefully analyzed to incorporate parallel processing. This has been discussed in detail under the mechanisms adopted to increase efficiency of the framework.

B. Mathematical Models in Quantitative Finance

1) Geometric Brownian Motion Model

A stochastic process St is said to follow a GBM [3] if it satisfies the following stochastic differential equation (SDE):

$$dS_t = \mu S_t dt + \sigma S_t dW_t \qquad (1)$$

where Wt is a Wiener process or Brownian motion and μ (percentage drift) and σ (percentage volatility) are constants.

Weiner process is a continuous-time stochastic process commonly used in mathematics characterized by the following three facts:

i. $W_0 = 0$
ii. W_t is almost surely continuous
iii. W_t has independent increments with distribution $W_t - W_s \sim N(0, t-s)$ (for $0 \leq s < t$).

2) Vasicek Model

Vasicek model [2] is a mathematical model describing the evolution of interest rates. The model specifies that the instantaneous interest rate follows the stochastic differential equation:

$$dr_t = a(b - r_t)dt + \sigma dW_t \quad (2)$$

Where W_t is a Wiener process modeling the random market risk factor. The standard deviation parameter, σ, determines the volatility of the interest rate. 'b' represents the long-term mean and 'a' the mean reversion speed. This model is called an Ornstein-Uhlenbeck stochastic process.

3) Cox-Ingersoll-Ross Model

The CIR model [3] can also be used in the valuation of interest rate derivatives. The model specifies that the instantaneous interest rate follows the stochastic differential equation:

$$dr_t = a(b - r_t)dt + \sigma\sqrt{r_t}dW_t \quad (3)$$

where W_t is a Wiener process modeling the random market risk factor. The standard deviation parameter 'σ', long-term mean 'b' and mean-reversion speed 'a' have the same meanings as in the Vasicek model.

C. Discretization of Models

The models used above need to be discretized before running Monte Carlo Simulations since they are continuous stochastic processes. There are many methods of discretization available. The exact solution of the stochastic process has been used in all three cases because they provide a high level of accuracy compared to the other available methods.

Exact solution of GBM model:

$$S_t = S_0 \exp\left[\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma\sqrt{t}N_t\right] \quad (4)$$

Exact Solution of Vasicek model:

$$r(t_i + 1) = r(t_i)e^{-a(t_{i+1}-t_i)} + b\left(1 - e^{-a(t_{i+1}-t_i)}\right) + \sigma\sqrt{\frac{1}{2a}(1 - e^{-2a(t_{i+1}-t_i)})}Z \quad (5)$$

Algorithm to arrive at exact solution to CIR model: [4]

$$let\ d = \frac{4ba}{\sigma^2}$$

$$if\ d > 1$$

$$for\ i = 0\ to\ (n-1)$$

$$c \leftarrow \frac{\sigma^2\left(1 - e^{-a(t_{i+1}-t_i)}\right)}{4a}$$

$$\lambda \leftarrow \frac{r(t_i)\left(e^{-a(t_{i+1}-t_i)}\right)}{c}$$

$$generate\ Z \sim N(o,1)$$

$$generate\ X \sim \chi_{d-1}$$

$$r(t_{i+1}) \leftarrow c\left[\left(Z + \sqrt{\lambda}\right)^2 + X\right]$$

$$end$$

$$if\ d \leq 1$$

$$for\ i = 0\ to\ (n-1)$$

$$c \leftarrow \frac{\sigma^2\left(1 - e^{-a(t_{i+1}-t_i)}\right)}{4a}$$

$$\lambda \leftarrow \frac{r(t_i)\left(e^{-a(t_{i+1}-t_i)}\right)}{c}$$

$$generate\ N \sim Poisson\left(\frac{\lambda}{2}\right)$$

$$generate\ X \sim \chi_{d+2N}$$

$$r(t_{i+1}) \leftarrow cX$$

$$end$$

D. Calibration Routines

The calibration of the Geometric Brownian Motion is very straightforward and thus the logarithmic return method has been used to calculate both the standard deviation and the annual growth rate. In the case of Vasicek and CIR models, there are many available calibration methods and it remains an active research area. Some of the popular methods are the use of General Method of Moments, Efficient Method of Moments, Least Squares Regression and Maximum Likelihood Estimators. The maximum likelihood estimator method has been used in the implementation described in this paper.

III HPC-CLUSTER

A. Computational Cluster

The Computational cluster is used to parallelize the selected financial models. MPICH2 cluster [2] which is an implementation of Message Passing Interface (MPI) [3] specification is used to setup the computational cluster. The cluster was setup in UNIX environment and comprised a master node and compute nodes. The master node is responsible for submitting jobs to the cluster. The cluster can be expanded dynamically. The main drawback of MPICH2 was that load balancing and fault tolerance

were not supported and had to be handled explicitly. Lack of load balancing lead to serious performance degradation, since all the jobs are submitted to the same set of nodes.

B. Database Cluster

MySQL Cluster has been deployed for the purpose of this research project. It is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions [4] and offers the following benefits.

Automatic and transparent data distribution across data nodes enables automatic replication of data in all nodes when the client application updates a single API node. This avoids the users having to manually update the market data used for calibration in all data nodes to facilitate the parallel calibration routines. MySQL Cluster prevents single point of failure as it ensures an application automatically fails over to another database node that contains a consistent data set, if one or more database nodes fail. Shared nothing architecture enables extendibility of data nodes and SQL API nodes according to the requirement of the application.

C. Parallel algorithms and Calibration

MPI was used to implement parallelization for short-rate models. MPI is a message passing API, together with protocol and semantic specifications for how its features must behave in any implementation. MPI's goals are high performance, scalability, and portability [3].

1) Parallel Calibration Routines

To simulate Vasicek Model and Cox-Ingersoll-Ross Model it is necessary to determine Mean (Drift), Standard Deviation (Sigma) and Mean Reversion Speed (Lambda) values from historical data set. Parameter calibration consumes a large amount of time for instances where large volumes of market data is available. This becomes a serious concern to a cluster which seeks to optimize the execution time.

The solution is to perform the calibration process in parallel. However, the decision to calibrate in parallel or not should be based on the size of the data set. If the size of the dataset remains less than a pre-calculated threshold value, parameter calibration will be done in a single machine. If not calibration is done in the main node and broadcast to the other nodes.

2) Parallel Monte-Carlo simulations

Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling to compute their results. Monte Carlo methods are often used when simulating financial derivatives and tend to be used when it is infeasible or impossible to compute an exact result with a deterministic algorithm. Higher the Monte Carlo simulations lower the standard error. In parallel computing context Monte Carlo methods are extremely parallelizable.

D. Server

A server was implemented in the cluster to communicate with and respond to requests of clients (in this case an MS Excel plug-in, but can be any client implemented in compliance with the protocol of the server) and resides in the master node of the cluster.

The server is capable of handling multiple concurrent requests from clients. The server listens through a port for ticket requests. When it gets a request, a unique identifier is sent back to the client as a reference to the job. As soon as the client acknowledges receipt of it, the request is handed over to a separate thread to continue. This works as a handshake mechanism between the clients and the server. The reference identifier enables the clients to terminate the respective job if required. The thread is taken from a pool of threads [5] which is conFig.d as the server is started and which gives more control over threads that accept requests. The server responsible for the load balancing since it is not covered by the computational cluster itself.

E. Client

The necessary objective of having a separate client is to provide an abstract interface for any non technical user, who might find it messy to control and exploit the application from back end. In this case MS Excel has been used owing to the fact that it is commonly used by quantitative analysts.

The client can basically be used to;

1) Invoke and execute tasks on the cluster
2) Mange data in database cluster (which is overlapped with execution cluster in this application model)

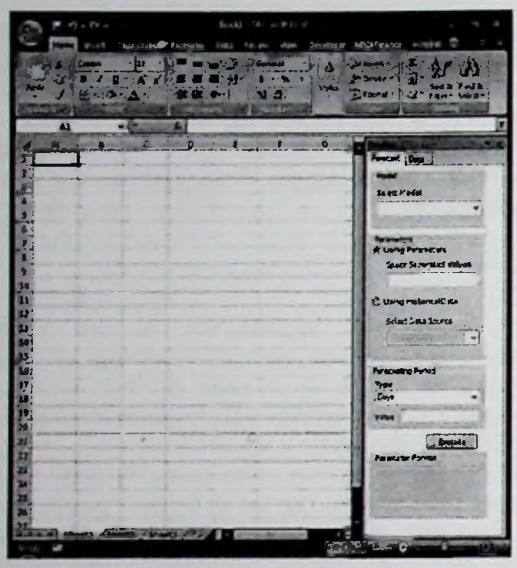

Fig. 1. Screenshot of MS Excel panel

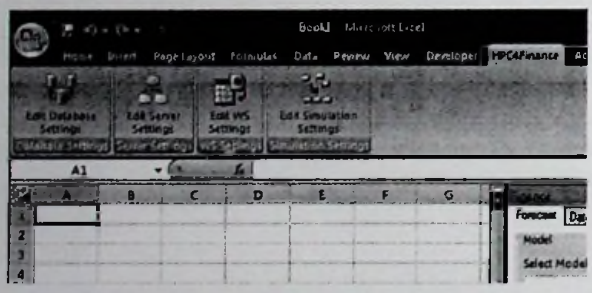

Fig. 2. MS Excel 2007 Ribbon for configuration

Since the communication between the client and the cluster is done via TCP sockets, the client is not expected to be aware about any platform specific details of the cluster; hence would interoperate with any back end which may provide a consistent interface. Configuring the client (see Fig. 2) is possible through Excel itself, where the user can specify database connection strings, server address, service port of the server and credentials for data acquisition. Having conFig.d the above setting accurately, whatever the model or performance extensions done at the server would reflect in the client. For instance, when a new model is implemented and deployed the server, no client side configuration is needed; the extension would reflect in the client automatically.

F. Random Number Generation in Parallel

The issue of random number generation assumes greater significance in the case of parallel computation. This is because; there exists a faint possibility of each node in the computer cluster producing identical or correlated random numbers. Therefore, in order to ensure unique random numbers across all nodes in the cluster a mechanism has been implemented where the individual seeds are obtained by multiplying a global seed generated based on time, multiplied by a prime number determined by the rank of each node. This method has been developed based on the suggestions made in [6]. Reproducibility of results has been achieved by storing the unique seed generated based on current time in the transaction log.

$$Unique\ seed = seed\ based\ on\ time \times (Rank\ of\ node + 4)^{th}\ prime\ numt \quad (7)$$

IV MEASURES TO IMPROVE EFFICIENCY

A. Load Balancing

The computational cluster does not provide load balancing in the current stable version of MPICH2. Therefore as a remedy, load balancing has been implemented explicitly in this HPC cluster.

The simple load balancing mechanism that has been implemented is as follows. The focus is on making the methodology simple and efficient; in order to make sure that additional overhead of load balancing does not diminish the benefits of parallelization. It is to utilize the nodes in the cluster, in a round robin manner. Therefore the jobs are submitted to the next available nodes, ensuring that nodes are not loaded with multiple jobs while other nodes lie idle.

B. Performance Tuning of the Cluster

This concept is more appropriate for a heterogeneous cluster, as is the case in the cluster used for this research initiative.

This algorithm is aimed at exploiting the fact that all the nodes of the cluster may not have the same computational resources. This algorithm ensures that the node with higher computational resources attract a higher proportion of the work load and vice versa.

Initially, the master node will divide the entire number of Monte Carlo simulations equally among the nodes and carry out the simulations. After execution, a ratio will be calculated based on the time consumption of each node. This will be stored in a performance tuning file and subsequent tasks will have the simulations divided among the nodes based on this ratio. This introduces a learning effect and the ratio will improve as time goes on.

V PERFORMANCE RESULTS

The testing environment of the computational cluster comprised the following resources;

TABLE I

DETAILS OF COMPUTATIONAL CLUSTER

	1st Machine	2nd Machine	3rd Machine
Processor	Intel Core 2 Duo 2.13GHz	Intel Pentium 4 HT 2.4 GHz	Intel Pentium 4 2.8GHz
Memory	1GB	512 MB	512MB
O/S	Fedora 8	Fedora 8	Fedora 8

The source code was compiled by gcc version 4.1.2 with '-O3' switch that enables compiler optimization.

A. Vasicek Model

Vasicek model consumes less amount of time compared to the other short rate models. Following results were obtained for Vasicek model. Testing results were obtained for 60 months.

TABLE II

PERFORMANCE RESULTS FOR VASICEK MODEL

Simulations / No of nodes	10000	25000	50000	100000
1	0.430441	1.044148	1.928339	3.605762
2	0.320161	0.655840	1.245749	2.563714
3	0.253804	0.523522	0.997676	1.729426
4	0.295980	0.424379	0.769589	1.531241

The results were graphed as in Fig. 3.

B. CIR Model

CIR model is more resource intensive process in the framework. The results are obtained as follows for 60 months predictions;

TABLE III

PERFORMANCE RESULTS FOR CIR MODEL

Simulations / No of nodes	10000	25000	50000	100000
1	31.32415	77.91261	155.035	310.2927
2	19.79184	47.17689	93.68947	187.0565
3	13.76775	35.81046	63.64923	136.1746
4	11.14291	29.92602	49.94145	106.4778

The results were graphed as in Fig. 4.

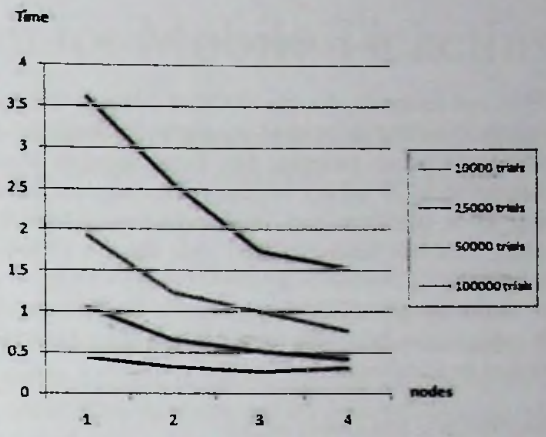

Fig. 3. Graph for Vasicek Model

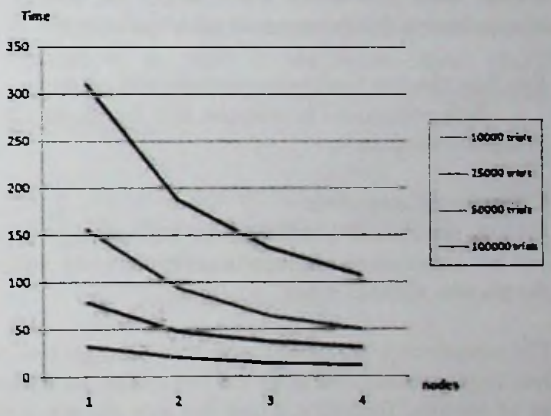

Fig. 4. Graph for CIR model

C. Analysis

The results obtained in Table II and III are with all optimizations enabled (i.e. performance tuning, compiler optimization and load balancing). It can be clearly seen (see Fig. 3 and 4) that the time consumption for both short rate models decrease significantly while increasing the number of nodes of the cluster. Furthermore, the performance gain is much significant for higher simulation counts, due to the increased scope for parallelization. The communication overhead seems to be deteriorating the benefits gained by parallel execution as the node count increases.

The time saving reaches considerable levels in the case of CIR model, due to the fact that it is much more computationally intensive than the Vasicek Model. Furthermore, the memory consumption of each of the models also affects the time consumption. A significant finding of this analysis is that the performance tuning feature can make significant gains by accounting for the non-homogenous nature of the cluster when dividing the workload.

VI Generalization of System

First and foremost, the use of Client-server architecture has generalized the entire implementation in one aspect by allowing any client (whether MS Excel based as in this implementation or other) to access the services of the cluster in a platform and implementation independent manner. This has been achieved with the use of socket programming.

In order to generalize the system to be extended for such models, the following parts of the system have been identified as components;

A. Monte Carlo Simulation engine

This includes the MPI code and will accept certain basic parameters which define the number of simulations to run the required model parameters

B. Random Number Generation with parallel seeding

This component is wrapped into the Monte Carlo Simulation engine

C. Calibration Component

A generalized parallelization methodology which can be extended to add new algorithms while leaving the parallel routines intact.

The components identified above have been generalized in their implementation in order use this system for a wider range of models. Therefore, a user can mix and match the above components along with the relevant algorithm in order to implement a new model. New calibration routines can also be added by extending the existing component. Another important issue is the applicability of this framework for two-factor or three-factor pricing models.

This can be realized by generalizing the code using C++ templating mechanism. Thereafter, we can extend the given models of this framework to create code for the multiple-factor versions of the same models.

VII Conclusion

Based on the results obtained, it is evident that the High Performance Computing Cluster can bring significant time savings to resource intensive processing. This capability is highly desirable in the financial services industry where HPC clusters are required to analyze or formulate various derivative instruments. The high level of performance enables quantitative analysts to obtain more accurate results with less standard error in less time. Furthermore, the seamless integration with the client application provides ease-of-use for finance professionals.

Acknowledgement

The authors wish to acknowledge the active support and advice given by Dr. Naveen Gunawardena and Dr. Ranjiva Munasinghe

References

[1] P. Glasserman, "Monte Carlo Methods in Financial Engineering", Springer-Verlag, New York, 2003, pp. 124-125.

[2] "MPICH2: High Performance and Widely Portable MPI". [Online]. Available: http://www.mcs.anl.gov/research/projects/mpich2/. [Accessed: Jul 3rd 2008].

[3] Message Passing Interface Forum; MPI-2: Extensions to the Message-Passing Interface; pp 122-157 [E-book] Available: http://www.mpi-forum.org/docs/docs.html Dddd

[4] MySQL AB, Sun Microsystems, Inc, MySQL 5.1 Reference Manual, Chapter 17. MySQL Cluster NDB 6x; pp 1344-1592 [EBook] Available: http://dev.mysql.com /doc/

[5] Irfan Pyarali, Marina Spivak, and Ron Cytron, Douglas C. Schmidt, "Evaluating and Optimizing Thread Pool Strategies for Real-Time CORBA", Proceedings of the ACM SIGPLAN Workshop on Optimization of Middleware and Distributed Systems (OM 2001), Snowbird, Utah, June 18, 2001

[6] R. Mirani, "Options: Approach for Parallel Implementation of Boyle's Monte Carlo Method", April 2002.