Bul. Ştiinţ. Univ. Baia Mare, Ser. B,
Matematică-Informatică, Vol. XVII(2001), Nr. 1-2, 41-48

MONTE CARLO SIMULATION FOR PRICING OPTIONS USING PVM ON A CLUSTER OF PCs

Adrian RĂBÂEA, Eduard-Marius CRĂCIUN

Abstract. Pricing options often requires use of Monte Carlo methods in financial industries. We describe and analyze the performance of a cluster of personal computers dedicated to Monte Carlo simulation on the evaluation of financial derivatives. Usually, Monte Carlo simulation (MCS) requires too much computer time. This requirement limits most of MCS techniques to use supercomputers, available only at supercomputer centers. With the rapid development and low cost of PCs, PC clusters are evaluated as a viable low-cost option for scientific computing. The free implementation of PVM is used on fast ethernet based systems. Serial and parallel simulations are performed.

MSC: 65C05, 65Y05

Keywords: Monte Carlo simulation, parallel computing

1 Introduction

Among the different numerical procedures for valuing options, the Monte Carlo simulation is well suitable for the construction of powerful pricing models. It is especially useful for single variable European options where, as a result of a non-standard pay-out, a closed-form pricing formula either does not exist or is difficult to derive. In addition, the price of complex options is sometimes difficult to explain intuitively and a simulation can often provide some insight into the factors that determine the pricing.

The commonly used Monte Carlo simulation procedure for option pricing can be briefly described as follows: firstly simulate sample paths for the underlying asset price; secondly compute its corresponding option payoff for each sample path; and finally, average the simulated payoffs and discount the average to yield the Monte Carlo price of an option.

An option is a contract that gives you the right to buy or sell an asset for a specified time at a specified price. This asset can be a "real" asset such as real estate, agricultural products, or natural resources, or it can be a "financial" asset such as stock, bond, stock index, foreign currency, or futures contract. Essentially, by buying the option, you transfer your risk to the entrepreneur selling you the options.

Therefore, an option is a contract between two parties: a buyer and a seller (or option writer). The buyer pays to the seller a price called the premium, and

in exchange, the option writer gives to the buyer the right to buy or sell some underlying securities at some specified price for some specified period of time.

An option to buy is a call option, and an option to sell is a put option. The specified price is called the strike or exercise price, and the option's life is called the time to maturity or time to expiration. If the right to exercise is "any time until maturity" then the option is called an American option. If the right to exercise is "at time of maturity" then the option is called an European option.

2 The Black-Scholes model

We denote by $S(t)$ the stock price at the time t. In the certainty case, the stock price at the time of the option's maturity T, equals the future value of the stock price $S(0)$ when continuously compounded at the risk-free interest rate r, $S(T) = S(0)e^{rT}$. One way to think about this is that the future value is the end result of a dynamic process. That is, the stock price starts at $S(0)$ at the present time 0, and evolves through time to its future value.

The formal expression that describes how the stock price moves through time in the certainty case is: $dS/dt = rS$ (this says that the rate of change of the stock price over time is proportional to the stock price at time t). The expression above describes the dynamic stock price process in a world with certainty. We can rewrite the equation as: $dS/S = rdt$. In this form, r is the instantaneous rate of stock's return (r is also called the drift rate of the stock price process).

If instantaneous return is r, its logarithm is a continuously compounded return. For the case of no uncertainty, the drift rate for the logarithm of this process is the same, but in an uncertain world this is not the case.

In a world with uncertainty and risk-averse investors, we expect that the instantaneous return from the stock, noted μ, will exceed the instantaneous risk-free rate of return (i.e. $\mu > r$). We must add a source of randomness to the instantaneous rate of return which has statistical properties that capture the fact that observed stock prices vary, and that a typical stock price path has variance which increases with time.

Black and Scholes [2] assume a model for stock price dynamics that is formally described as geometric Brownian motion. This model has the following form:

$$\frac{dS}{S} = \mu dt + \sigma dW \quad , \quad t \in [0, T] \tag{1}$$

where the parameters μ and σ are constant with respect to t and S. Here there are two factors that affect the instantaneous rate of return on a stock. The first one is the time. Over the period of time dt, the stock's return changes by the amount μdt. The second factor is uncertainty. The sensitivity to this source of uncertainty is captured by the term σ which is the volatility coefficient for the stock price. The net effect of adding the term σdW to the certainty model is to create a stochastic path for stock prices around the certainty path. Uncertainty in the model is added to let the model better satisfy properties exhibited by real world stock price.

Let $f(S,t)$ denote the value of any derivative security (e.g. a call option) at time t, when the stock price is $S(t)$. Using Ito's lemma,

$$df = \sigma S \frac{\partial f}{\partial S} dW + \left(\mu S \frac{\partial f}{\partial S} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} + \frac{\partial f}{\partial t} \right) dt$$

and assuming that there are no arbitrage opportunities on the market, we get the following parabolic partial differential equation (PDE)):

$$\frac{\partial f}{\partial t} + rS \frac{\partial f}{\partial S} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} - rf = 0 \,.$$

This is called the Black-Scholes partial differential equation. Its solution, subject to the appropriate boundary condition for f, determines the value of the derivative security. For an European call option, the boundary condition is

$$f(S,T) = \max\{0, S(T) - K\} \,,$$

where K is the strike price and T is the time to expiration.

The solution of the above PDE is given by the Black-Scholes formula :

$$C = SN(d_1) - Ke^{-rT} N(d_2) \,, \tag{2}$$

where

$$d_1 = \frac{\log(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \,, \qquad d_2 = d_1 - \sigma\sqrt{T}$$

and N is the cumulative standard normal distribution.

Therefore, there are five parameters which are essential for the pricing of an option: the strike price K, the time to expiration T, the underlying stock price S, the volatility of the stock σ, and the prevailing interest rate r.

In some cases, the type of option is so complicated (for example, in (1) μ or σ are random) that the solution of the PDE is very difficult to be found. When this is the case, it is nearly always possible to obtain the option price by an approximation using an appropriate - maybe computationally intensive - numerical method. The standard methods are discussed in [3].

3 Monte Carlo Simulation

Usually, for solution of financial problems, the Monte Carlo Simulation (MCS) methods are used (e.g. to value European options and various exotic derivatives). But, because such kind of problems are very complicated, the MCS algorithms becomes too computational "expensive". On the other hand, due to the inherent parallelism and loose data dependencies of the above mentioned problems, Monte Carlo algorithms can be very efficiently implemented on parallel machines and thus may enable us to solve large-scale problems which are sometimes difficult or prohibitive to be solved by the other numerical methods. For implementing Monte Carlo method to price European options, the following procedure is used:

- for the life of the option, simulate sample paths of the underlying state variables in the pricing model,
- for each path, calculate the discounted cash flows of the option
- take the sample average of the discounted cash flows over all sample paths.

For pricing European calls, we have to compute $E(S(T)-K)^+$, where $(S(T)-K)^+ = \max\{S(T)-K, 0\}$. Then, we compute the call value C by

$$C = \exp(-rT)E(S(T)-K)^+ . \tag{3}$$

As we already have mentioned in Section 2, the assets follow a geometric Brownian motion and the stock price $S(t), t \in [0,T]$ are log-normally distributed. For a given partition $0 = t_0 < t_1 < ... < t_N = T$ of the time interval $[0,T]$, a discrete approximation to $S(t)$ is a stochastic process $\tilde{S}_t$ satisfying

$$\tilde{S}_{k+1} = \tilde{S}_k \exp((r - \frac{1}{2}\sigma^2)\delta t + \sigma\sqrt{\delta t}Z_k) , \tag{4}$$

for $k = 0, 1, ..., N-1$ (where we used the subscript k instead of the time step subscript t_k), with $\delta t = t_{k+1} - t_k = T/N$, and initial condition

$$\tilde{S}(0) = S(0) , \tag{5}$$

where

- $Z_0, Z_1, ..., Z_{N-1}$ are independent standard normal random variables, and
- $\sigma\sqrt{\delta t}Z_k$ represents a discrete approximation to an increment in the Wiener process of the asset.

The call price estimate is then computed using the discount formula (3). After repeating the above simulation for a large number of time steps, the initial call value is obtained by computing the average of estimates for each simulation. The disadvantage of the Monte Carlo simulation for European options is the need of a large number of trials in order to achieve a high level of accuracy.

4 Cluster Architecture

Clusters of computers (workstations) constructed from low-cost platforms with commodity processors are emerging as a powerful tool in computational science. These clusters are typically interconnected by standard local area networks, such as switched Fast Ethernet. Fast Ethernet is an attractive option because of its low cost and widespread availability. However, communication over Fast Ethernet incurs relatively high overhead and latency. But, for our above described problem, the communication requirements are insignificant.

We used a cluster of 20 PCs, 10 of them having Compaq Deskpro Intel 200MHz Pentium-MMX processor, with 64MB of RAM and another 10 of them having Digital Venturis FX 5133s Intel 133MHz Pentium processor, with 32MB of RAM. For the Fast Ethernet networking we used a 3Com Fast EtherLink XL 10/100Mb TX (Ethernet NIC 3C905B-TX) PCI network cards and a 3COM Super Stack II Baseline 10/100 Switch 3C16464A switch.

The Windows 98 distribution was used.

5 Parallel Implementation

The Monte Carlo method for pricing derivatives is an ideal candidate for the use of PVM software. As stated above, the Monte Carlo can take up to hours to run. Using PVM ([1], [5]), this disadvantage can be eliminated.

We implemented parallel Monte Carlo simulation on a cluster described in Section 4 before under PVM and we applied master/slave approach. As we have already mentioned in the above Section 4, the MCS for pricing option allowed us to have minimal communication, i.e. to pass to each processor only the parameters:

- $S0$ = initial underlying stock price,
- K = strike price,
- σ = volatility of the stock,
- r = prevailing interest rate,
- δt = length of interval in uniform N-period partition for time T,
- T = time to expiration,
- $NSIMP$ = number of Monte Carlo simulations per processor,

to run the algorithm in parallel on each processor by computing the option's value for $NSIMP$ simulations and, at the end, to collect the results from slaves without any communication between sending the parameters and receiving the call value. The only communication is at the beginning and the end of the algorithm execution which allows us to obtain very high efficiency for our parallel implementation.

This algorithm was implemented using the PVM MASTER/SLAVE model. The MASTER program is responsible for sending/receiving parameters to/from slaves and computing the final value. Each SLAVE receives from the MASTER program the parameters it needs for computation, computes and sends back the results.

There is a worker process (task) per node (processor); the master process can either share a node with a worker process or run on a dedicated node.

M. The MASTER program
   1. The MASTER process send to all SLAVE processes the parameters: $S0$, K, σ, r, δt, T, $NSIMP$ which are necessary for calculating C;
   2. The MASTER process receives from each SLAVE process the computed value of C ;
   3. The MASTER process computes the final option price.

S. The SLAVE program
   1. Each SLAVE process receives parameters from the MASTER process for computing initialization;
   2. The SLAVE process performs its local computation (to evaluate C, using (3), (4) and (5)) ;
   3. The SLAVE process sends the results back to the MASTER process.

6 Numerical tests

We performed a simulation study using the Black and Scholes model. The Monte Carlo option price for a given $NSIMP$ and p processors can be computed according to above algorithm.

The numerical tests were made on a cluster of 20 PC, under PVM, for $r = 0.07$, $S0 = 100.00$, $K = 95.00$, $\sigma = 0.20$ and $T = 0.25$. The theoretical value computed according to (2) is $ST = 8.056$. We tested the methods in the European case because the true price can be analytically determined.

For $\delta t \in \{10^{-2}, 10^{-3}\}$ we generated Monte Carlo option price estimates. These prices were computed with $10^1, ..., 10^6$ sample paths in order to examine the impact caused by different numbers of sample paths. For each test we priced options and computed the *error = theoretical value - Monte Carlo simulation value*. The numbers in Table 1 and Table 2 represent the errors for $\delta t = 0.001$ and $\delta t = 0.01$, respectively. The first column in both tables indicates the number of simulation steps per processor.

Table 1. Errors for Monte Carlo Simulation using $\delta t = 0.001$

p NSIM	1	2	3	4	5	6	7	8	9	10
10^1	-3.200	+1.212	-2.122	-0.900	-0.453	+0.819	+0.610	+0.167	-0.978	-1.726
10^2	-0.768	-0.227	-0.285	-0.305	-0.012	+0.153	-0.374	+0.012	-0.229	+0.276
10^3	+0.039	+0.107	-0.054	-0.127	+0.111	+0.049	+0.057	+0.083	-0.129	+0.071
10^4	+0.052	-0.034	+0.041	-0.041	+0.028	+0.002	+0.001	+0.001	-0.009	-0.020
10^5	+0.007	+0.022	+0.005	-0.002	-0.004	+0.007	+0.001	+0.001	-0.014	+0.001
10^6	+0.003	+0.002	-0.001	-0.002	-0.002	-0.001	-0.004	+0.005	-0.002	+0.003

Table 2. Errors for Monte Carlo Simulation using $\delta t = 0.01$

p NSIM	1	2	3	4	5	6	7	8	9	10
10^1	-2.753	-2.117	+0.003	+0.146	-1.624	-0.358	-1.028	+0.022	-1.638	-0.711
10^2	-0.811	-0.149	+0.482	+0.284	+0.248	-0.159	+0.515	-0.343	+0.223	+0.219
10^3	-0.018	-0.107	-0.173	-0.228	+0.211	-0.073	+0.003	-0.027	-0.075	-0.141
10^4	+0.038	+0.043	+0.042	-0.004	+0.053	-0.006	-0.009	-0.007	-0.045	+0.001
10^5	-0.012	-0.027	-0.011	+0.003	+0.001	-0.002	-0.011	-0.023	+0.003	-0.005
10^6	+0.002	+0.004	+0.005	+0.002	-0.002	-0.003	+0.002	-0.001	+0.002	+0.002

We can observe that the error decreases when the number of sample paths increases. Unfortunately, in the same time we have to observe a disadvantage of the Monte Carlo simulation for European options. This consists in the large number of trials necessary to achieve a high level of accuracy.

In all tests we used the *gasdev* routine for generating random deviates with a normal distribution (Box-Muller method) and the pseudo random generator *ran2* as the source of uniform deviates (the long period random number generator of L'Ecuyer with Bays-Durham shuffle and added safeguards) [4]. The results are sensitive to the initial seed. Figure 1 shows the results of Monte Carlo simulation obtained with randomly chosen initial seeds.

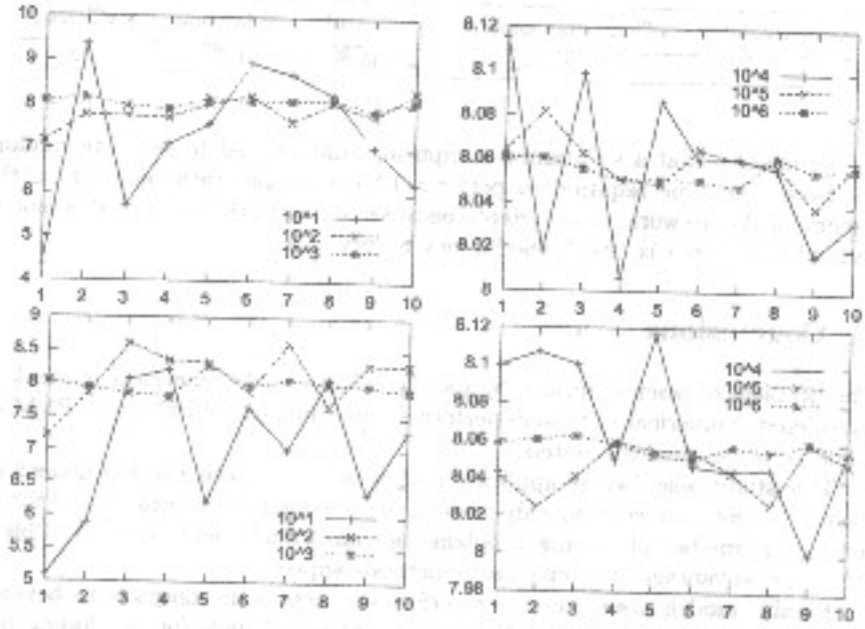

Fig. 1. Computed option price with respect to number of processors, using MCS method for $10^k, k = 1, ..., 6$ sample paths (upper: $\delta t = 0.001$, down: $\delta t = 0.01$)

Theoretically, we can obtain an arbitrary degree of accuracy. But, from a practical view point, higher is the level of accuracy, bigger will be the computational effort for the MCS algorithm. This is, of course, due to the well-known fact that the standard error of Monte Carlo estimate is inversely proportional to the square root of the number of simulated sample paths.

The quality of the random number generator is essential. We used pseudo random generator. This random generator often requires a very large number of simulation repetitions to minimize errors. It is possible to use quasi random generators that are designed simply to fill the space in an interval more uniformly than uncorrelated random points.

In a more formal sense, using quasi random generators we can reduce the error associated with a simulation from $\mathcal{O}(1/\sqrt{N})$ to $\mathcal{O}(1/N)$.

The parallel efficiency E, as a measure that characterizes the quality of the parallel algorithm, is defined as: $E = \frac{T_1}{pT_p}$, where T_p is the value of the computational time for implementation the algorithm on a system of p processors. The parallel efficiency and the percentages of work, random generation and latency, in all Monte Carlo repetitions, are given in Table 3.

Table 3. Computing time and efficiency for 10.000 simulations and $\delta t = 0.001$

Time in seconds	% random generator	% work	% latency	efficiency
15	66 %	33 %	1 %	99 %

We observe that a substantial computing time is used to generate random numbers. The time required to perform 10.000 sample paths depend on the latency of the network; it is variable because the network is a shared resource. But, when network is "free", the latency is very small.

7 Conclusions

The problem of pricing options by parallel Monte Carlo numerical methods is considered. Numerical tests were performed for a number of PCs using PVM on a cluster of personal computers.

This study describes an application of parallel computing in the finance industry. Options are continuously growing more complex and exotic, and for an increasing number of pricing problems, no analytical solutions exist. This is where the advantage of Monte Carlo methods appears.

Parallel models are required for performing large scale comparisons between model and market prices. Parallel models are useful tools for developing new pricing models and applications of pricing models.

In our parallel implementation we calculated one price of the call option. To compute this price by Monte Carlo simulation we need more computational power. Using p processors the execution time is p times small.

References

1. Beguelin, A., et al: A Users' Guide to PVM Parallel Virtual Machine, Oak Ridge National Laboratory, U.S. Department of Energy Contract, DE-AC-05-84OR21400.
2. Black, F., Scholes, M.: The pricing of Options and Corporate Liabilities, Journal of Political Economics 81, 1973, 637-659
3. Hull J.: Options, Futures, and Other Derivative Securities, Prentice-Hall, 1993
4. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C, Second Edition, Cambridge University Press, 1992
5. Scott, S. L. ,Fischer, M., Geist, A.: PVM on Windows and NT Clusters, EuroPVM-MPI98, Lecture Notes in Comp. Science, Springer Verlag, 1998

Received: 15.09.2001

Faculty of Mathematics and Informatics
"Ovidius" University of Constantza
Blvd. Mamaia 124, 8700 Constantza, Romania
{arabaea,ncraciun}@univ-ovidius.ro