X10 implementation of Parallel Option Pricing with BSDE Method [*]

Hui LIU Ying PENG DaiZhen WEI Bin DAI

School of Computer Science and Technology
Shandong University
Jinan Shandong, China
amyliuhui@sdu.edu.cn

Abstract

Option pricing is one of the most important parts in the field of financial derivatives pricing and risk management. To promote precision of pricing, we use an parallel method based on BSDE(Backward Stochastic Differential Equation) model, which is a widely used numerical model in financial computing domain.BSDE model can improve the accuracy and effectiveness of option pricing. X10 is a high level high-performance programming language with new concurrency constructs, namely, places, asyncs, finish, atomic and clock. X10 also provides a rich array language which includes region, distributions and distributed arrays. In this paper, we present how to utilize X10's properties to implement option pricing with BSDE model. Experimental results show that the parallel program implemented in x10 can achieve a superior performance and nicer speedup. It can be widely used in financial area.

Keywords OptionPricing,BSDE,X10

1. Introduction

In the whole field of financial applications, option pricing is one of the most complex issues in mathematic. The most celebrate model for derivatives pricing is the Black-Scholes [1] model, which uses analytic formula to get the explicit solution for option pricing. However, in many conditions the explicit solution can't be derived from analytic formulas, so we must resort to numerical methods, which need much more computing efforts to get enough precise solutions, especially for long dated derivative pricing. In 1997, N. EI Karoui, Shige Peng and Quenez obtained the extending of the Black- Scholes formula [2] by using Backward Stochastic Differential Equations (BSDEs). With the result, we can calculate and analyze the issue of option pricing more accurately. At present, BSDEs cover many scientific fields, especially the financial field. The theory of BSDEs not only can be used on option pricing, but also can help various kinds of investors to hedge and analyze other types of risk[2]. It can be used on incomplete markets to provide a powerful analytical and approximate calculation for all kinds of derivative securities pricing and hedging. When the generating function is linear or other special types of nonlinear functions, with given terminal conditions and an identified generating function, the BSDEs can get explicit solution. But for most of cases, we solve the BSDEs only through numerical method. By solution principle, BSDEs can be divided into two different types. One uses nonlinear Feynman-Kac formula to transform the original problem into similar linear parabolic PDEs. The other constructs numerical format according to the properties of BSDEs. In [3], it replaced the traditional Black-Scholes model computational method with BSDEs to solve option value. The high-precision -scheme is one of the numerical methods of getting solutions, and it contains a scheme named fully discrete -scheme. The method of solving the BSDEs in [3] is achieved by the fully discrete -scheme through the space discretization and Monte Carlo simulation.

X10 is a high level high-performance programming language with new concurrency constructs, namely, places, asyncs, finish, atomic and clock. X10 also provides a rich array language which includes region, distributions and distributed arrays[4]. In this paper, we efficiently utilize X10's properties to implement option pricing with BSDEs. We adopt a high accurate theta scheme method to solve the BSDEs. This method needs large amounts of calculation, and the computation time is very long for getting enough accurate result. Fortunately, the computation process of this method meets the conditions for parallel computing in X10. We analyze the serial algorithm and design the acceleration strategy. We also conduct performance experiments to compare the running time between the parallel implementation and the serial implementation and analyze factors affecting the parallel acceleration.

The remainder of this paper is organized as following. In Section 2 we present the application of BSDE in option pricing and the numerical method for solving the BSDEs. In Section 3 we describe the X10 parallel programming algorithm according to the special structure of the numerical model. Section 4 shows the experimental results. Finally we give the conclusion in Section 5.

2. Option Pricing with BSDEs

The BSDE is proved to be a robust tool for derivatives pricing and risk hedging in financial area. The mathematical formula for a BSDE is shown as following:

$$\begin{aligned} -dY_t &= f(Y_t, Z_t, t)dt - Z_t dW_t,\ t \in [0,T] \\ Y_T &= \xi \end{aligned} \quad (1)$$

[*] This material is based upon work supported by the Science and Technology Research and Development Program of Shandong Province of China,the National Grand Fundamental Research 973 Program of China,the National High-Tech Research and Development Plan of China.

where $W_t = (W_t^1, \ldots, W_t^d)^*$ s a d-dimensional Brownian motion defined on some complete, filtered probability space$(\Omega, F, P, \{F_t\}_{0\leq t\leq T})$, and the notation* is the transpose operator for matrix or vector. $Y_T = \xi$ is the terminal condition of the BSDE and ξ is a F_T measurable random variable. The solution of the BSDE is a couple of variables (Y_t, Z_t), and the generating function $f(\bullet)$ is a function of (Y_t, Z_t).

While applied in option pricing, the solution Y_t represents the option price at time t, and Z_t helps determine the risk hedging strategy. The generating function $f(\bullet)$ is:

$$f(t, y, z) = -ry - \sigma^{-1}(\mu - r)z \tag{2}$$

where r is the return rate, σ represents the volatility, and μ is the expected return rate.

For the European options which can be exercised only at its expiration, the terminal condition of the BSDE is given at the expiration time T by:

$$\begin{aligned} Y_T &= \xi = \phi(W_T) \\ &= \max(S_T - X, 0) \qquad \text{(call option)} \\ &= \max(X - S_T, 0) \qquad \text{(put option)} \end{aligned} \tag{3}$$

where $\phi(\cdot)$ is a function of the Brownian motion W_T and it represents the payoff of the option holder at the expiration time $T.S_T$ is the asset price at the expiration time and X is the strike price.

In order to calculate the option price with the BSDE, we adopt the theta scheme method [3], which is of high accuracy and suitable for the parallel implementation in X10. The θ scheme method discretizes the continuous BSDEs on time-space discrete grids. On each grid point, it uses the Monte Carlo method to approximate mathematical expectations, and uses space interpolations to compute values at non-grid points.

By performing iteratively the full discretization theta scheme method, we could get the value of Y_t and Z_t on every grid point, as shown in equation (4).

$$\begin{aligned} y_i^n = &E_{h,t_n}^{t_n,x_i}[y^{n+1}] \\ &+\triangle t_n(1-\theta_1^n)E_{h,t_n}^{t_n,x_i}[f(t_{n+1}, y^{n+1}, z^{n+1})] \\ &+\triangle t_n \theta_1^n f(t_n, y_i^n, z_i^n) \\ 0 = &E_{h,t_n}^{t_n,x_i}[y^{n+1}\triangle W_{t_{n+1}}] \\ &+\triangle t_n(1-\theta_2^n)E_{h,t_n}^{t_n,x_i}[f(t_{n+1}, y^{n+1}, z^{n+1})\triangle W_{t_{n+1}}] \\ &-\triangle t_n\{(1-\theta_2^n)E_{h,t_n}^{t_n,x_i}[Z^{n+1}] + \theta_2^n z_i^n\} \end{aligned} \tag{4}$$

where i indicates the space layer, n indicates the time layer on the grid. $E[X]$ denotes the conditional mathematical expectation of the random variable X, which can be approximated with the Monte Carlo method. We show in equation (5) the approximation of $E_{h,t_n}^{t_n,x_i}[y^{n+1}]$, other conditional expectations can be gotten similarly, shown in detail in [3].

$$E_{h,t_n}^{t_n,x_i}[y^{n+1}] = \frac{\sum_{k=1}^{N_E} I_h y^{n+1}(\hat{x}_i^k)}{NE} \tag{5}$$

In (5), $\hat{x}_i^k$ can be derived with $\hat{x}_i^k = x_i + \triangle_h^k W_{t_{n+1}}$. For each space point x_i, let $\triangle_h^k W_{t_{n+1}} (k = 1, 2, \ldots, NE)$ be the NE times realizations of $\sqrt{\triangle t_n} N(0,1)$, where $N(0,1)$ is a standard normal distribution with mean 0 and variance $1.I_h u(\hat{x}_i)$ represents an interpolation approximation of the function $u(x)$ at the space point $\hat{x}_i$ by using the values of $u(x)$ at a finite number of the space points X_j's near the space point $\hat{x}_i$.

3. Parallel Algorithm

We first describe the computation process of the algorithm. Then we present the parallel optimization strategy in X10 according to the algorithm process.

3.1 Parallel Algorithm

According to the numerical algorithm represented in Section 2, the whole process for option pricing can be divided into three steps.

Step1: constructing the time-space discrete grid, as shown in Figure 1.

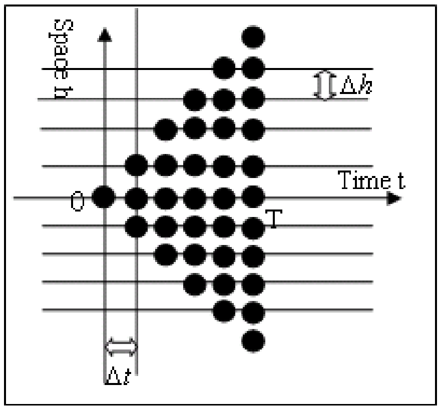

Figure 1. Time-Space Grid

We evenly divide the time period $[0, T]$ into N time steps with interval $\triangle t = T/N$ to get $N+1$ time layers. The value of the space interval $\triangle h$ is supposed to be the same as $\triangle t$. Then for every space point x_i, its value for calculating equation (5) can be derived with the value of $\triangle h$.

Step 2: calculating the terminal conditions which represent the possible option prices at the expiration time T (on time layer N) by using equation (3).

Step 3: calculating option price iteratively from the time layer $N-1$ to the time layer 0 with equation (4), until we get the value of (Y_0, Z_0), which represent (Y_t, Z_t) at the time layer 0. The process for calculating the corresponding option price on an arbitrary grid point is shown in Figure 2.

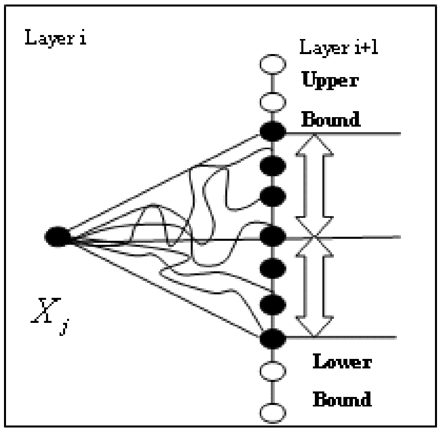

Figure 2. Calculation Process on a Grid Point

As mentioned in Section 2, the Monte Carlo simulation is used to approximate the mathematical expectation $E[X]$. In Figure 2, for each time of MC simulation, the geometric Brownian motion start from the space point X_j on time layer i, along the path from time layer i to time layer $i+1$, in this way we get the projection value of $\hat{x}_i^k$ on time layer $i+1$. By using space interpolation,$I_h u(\hat{x}_i)$ can then be calculated. We use cubic spline interpolation to achieve better accuracy. According to equation (5), after NE times of Monte Carlo simulations, we could get the value of $E[X]$ and use them to compute equation (4). At last the solution of the BSDE (Y_0, Z_0) is achieved.

Therefore, for calculating the option price of a grid point X_j on time layer i, the grid points on time layer $i+1$ is needed. Without loss of generality, we define the upper and lower bound $C = c * \sqrt{\triangle t}$ for the geometric Brownian motion on a single time step, where c is a constant. The upper and lower bound could also be defined by the number of space points $P_s = C/\triangle h$.In this way, the number of space points on time layer i can be gotten by $M_i = i * P_s * 2$, and we define $M_0 = 1$.

3.2 Parallel Implementation

As shown in Section**??**, the algorithm for option pricing with the BSDE can be performed with three steps. For step 1 and step 2, the construction of the time-space discrete grid and the calculation of the terminal conditions, we do not need large amount of calculation. To verify that, we measure the running time of step 1 and step 2. We let N and NE represents respectively the number of time steps and the number of Monte Carlo simulations. When (N, NE) are fixed at $(64, 40000)$, the running time of step 1 and step 2 is only $0.281ms$. Comparing with $2,098,757ms$, which is the running time of all steps, the cost of step 1 and step 2 seems insignificant. Thus for these two steps, we implement them serially.

For the step 3, each option price corresponding to space point X_j at time layer t_i needs NE times of MC simulation. Thus step 3 is the most time-costing part in the serial algorithm. Moreover, on each time layer, the option price of different space points can be calculated completely independently. Thus step 3 is very suitable to be parallelized. We implement the parallel algorithm in X10.

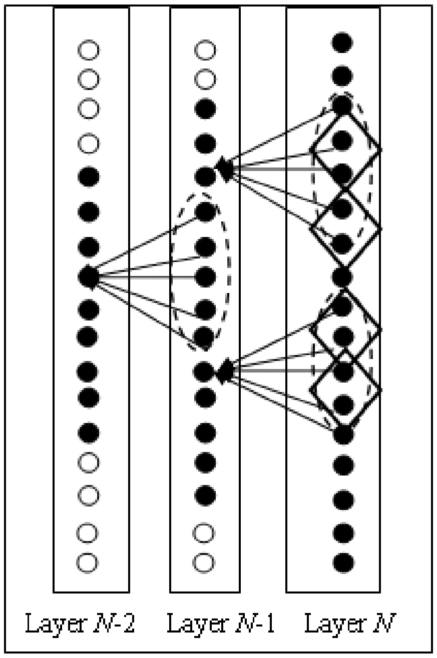

Figure 3. The Calculation Process of BSDE

The X10 language has two levels of parallelization,one is the process level (inter-place) and the other is the thread level (intra-place).In this paper, we use X10 to implement BSDE algorithm by mapping the X10's parallel levels to the two levels of the current clusters (inter-host level and intra-host level). We run one place per host and multiple threads per place. From Figure 3 we can see that the computing of the value of the grid points of the former layer depends on the value of the grid points of the latter layer, and the number of the grid points decreases as the number of time layer decreases, which means that the number of parallel tasks decreases as the time layer moves backwardly. In order to simplify the dependence between the grid points, we suppose that a grid point of the former layer depends on all the grid points of the latter layer, so the value of a grid point of the former layer will be computed after all the values of the grid points of the latter layer being computed. Figure 4 shows the parallel logic of the BSDE algorithm.

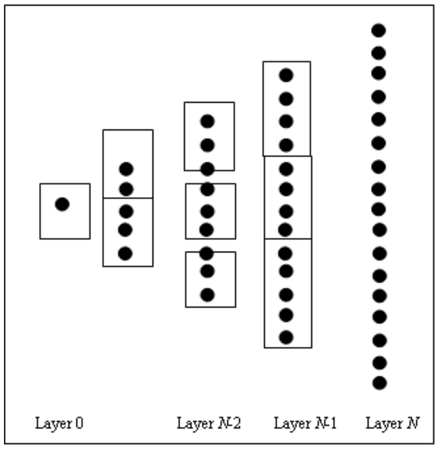

Figure 4. The Parallel Strategy of BSDE in X10

Each rectangle stands for a place, so at the start of each time layer, we balance the work load between all the places. Coding in X10 for parallel BSDE algorithm is much simpler than in MPI, because X10 belongs to PGAS language and X10 has global data structure which makes the communication between places transparent. So in the algorithm**??** we use distribute array to store the value of the grid points which can be referred by all the places.

Algorithm 1 Calculation of Current Layer i in X10

```
Require: val $T$;{the number of time layers.}
    val $f\_Layer$ : $DistArray[Float]$;{store the values of the grid points of the next layer.}
    val $c\_Layer$ : $DistArray[Float]$;{store the values of the grid points of the current layer.}
    initialize ($f\_Layer$);{initialize $f\_Layer$ to store the values of the grid points of the last time layer.}
1: for int $i$=0; $i < T$; $i++$ do
2:     $finish$ for ([$p$] in 0..$NumPlace$ -1) $async$
3:     {
4:         $finish$ for ([$t$] in 0..$numThread$ -1) $at$ ($p$) $async$
5:         {
6:             $BSDE\_Compute(f\_Layer, c\_Layer, i, p, t)$;
7:             $SwapValue(f\_Layer, c\_Layer)$;
8:         }
9:     }
10: end for
```

Function $BSDE\_Compute(f\_Layer, c\_Layer, i, p, t)$ will find the grid points of layer i belonging to the thread t in place p and compute the values of these grid points, and store the values in distribute array c_Layer.

4. Experiment Results

The parallel algorithm is implemented in X10. Runtime experiments are performed on a computing node with 8*8cores/socket Intel(R) Xeon(R) X7550 2. 0GHz CPU and 252G GDDR3 memory. The results are compared with the serial code.

Table 1. The Parallel Runtimes with the LANGCHAO Cluster

Processor	$N = 4$ $NE = 1000$	$N = 8$ $NE = 4000$
1	1456	37309
2	1185	32269
4	1075	25493
8	1219	32865
16	1212	34209

The results of Table1 seem satisfying. We can see that with a single processor, the running time increases greatly when the problem size N and NE growing up. Using more processors helps decrease the swap operation frequency of the system, thus decrease the running time.

The speedup trend is like Figure5 and Figure6.

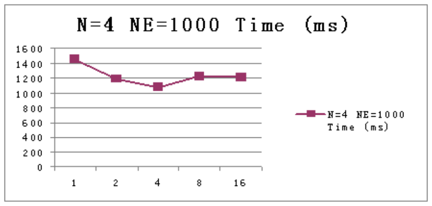

Figure 5. The Speedup of N=4 NE=1000

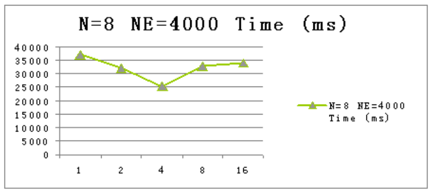

Figure 6. The Speedup of N=8 NE=4000

From the experiment, we know that the computing speed can be promoted and computing time can be decreased in X10, although its performance needed to be improved in the future. If the computing is implemented with larger data set, it will get better speedup.

5. Conclusion

We have presented a parallel implementation for financial derivative pricing. A parallel option valuation algorithm has been proposed according to the inherent structure of high-precision -scheme BSDE model. Large communication overhead has been avoided on considering both the communication frequency and the amount of data transferring. We have evaluated the performance and performed parallel run time analysis, both of them achieve optimistic results.

In the future, more challenging problems such as multi-asset derivative pricing will be considered for parallelization, and more applications commonly used in financial computing will be implemented in x10.

References

[] F. Black, and M.Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 1973, Vol 81, pp. 637-654, 1973.

[] N.El. Karoui, S. Peng, and M.C. Quenez. Backward stochastic differential equations in finance. *Mathematical Finance*, Vol. 7, No. 1, pp. 1-71, 1997.

[] Weidong Zhao, Lifeng Chen, and Shige Peng. New Kind of Accurate Numerical Method for Backward Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, SISC Volume 28 Issue 4, Pages 1563-1581, 2006.

[] `http://x10.codehaus.org/`.