Parallel Computing for Option Pricing Based on the Backward Stochastic Differential Equation

Ying Peng, Bin Gong*, Hui Liu, and Yanxin Zhang

School of Computer Science and Technology, Shandong University,
Jinan 250101, P.R. China
pengy@sdu.edu.cn, gb@sdu.edu.cn,
amyliuhui@sdu.edu.cn, zhangyanxintongxue@yahoo.com.cn

Abstract. The Backward Stochastic Differential Equation (BSDE) is a robust tool for financial derivatives pricing and risk management. In this paper, we explore the opportunity for parallel computing with BSDEs in financial engineering. A binomial tree based numerical method for BSDEs is investigated and applied to option pricing. According to the special structure of the numerical model, we develop a block allocation algorithm in parallelization, where large communication overhead is avoided. Runtime experiments manifest optimistic speedups for the parallel implementation.

1 Introduction

Over the past two decades, an increasing number of complex mathematical models have been applied in financial engineering. As most of these models can not be solved by analytical formulas, we must use numerical methods, which need much more computing efforts. Moreover, time constraints should be respected in financial trade-off, therefore effective parallel implementations are in great demand. To decrease the computation time, many parallelization works are presented for solving various numerical models in financial computing area [2,3,4,5,8].

The BSDE was firstly introduced to solve the problem of derivative pricing by El Karoui in [6]. Since then, the BSDE theory was applied in financial area more and more frequently. Comparing with the Black-Scholes formula [1], the BSDE is more robust to fit in the situation of probability model uncertainty, thus can be used to perform more approximated calculations in financial derivatives pricing and risk analysis.

In this paper, we develop a parallel algorithm for the BSDEs with application to option pricing. Though many efforts have been made in the numerical methods for solving BSDEs, few work has been done for the parallel implementation until now. The reminder of this paper is structured as follows: In Sect. 2 we present the application of BSDE in financial engineering. In Sect. 3, we describe the numerical algorithm for BSDE and apply it to option pricing. The parallel implementation is discussed in Sect. 4. Section 5 shows the experimental results. Finally we give the conclusion in Sect. 6.

* Corresponding author.

W. Zhang et al. (Eds.): HPCA 2009, LNCS 5938, pp. 325–330, 2010.

2 BSDEs in Financial Engineering

The general form of a BSDE is shown as following:

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

where W_t is a d-dimensional Brownian motion. $Y_T = \xi$ is the terminal condition of the BSDE and ξ is a F_T measurable random variable. The solution of a BSDE is a couple of variables (Y_t, Z_t), and $g(\cdot)$ is a function of (Y_t, Z_t). In the following we give an example for the application of BSDEs in option pricing and risk hedging. Assuming a bond and a stock traded in a stock market. The bond price p_t and the stock price S_t, respectively, obey:

$$p_t = e^r t\ , \tag{2}$$

$$S_t = S_0 \exp(bt + \sigma W_t - \frac{1}{2}\sigma^2 t)\ , \tag{3}$$

where r is the return rate of the bond, b is the expected return rate of the stock, σ is the volatility of the stock, and $\{W_t\}_{t\geq 0}$ is a standard Brownian motion. S_t follows the geometric Brownian motion.

Now consider a call (or put) option with strike price X on time period $[0, T]$, where the option holder has the right to buy (or sell) a stock of equation (3) from the seller at time T. Then we can use the BSDE of equation (1) to calculate the price Y_0 of the option, which correspond to the value of Y_t at time 0. The terminal condition is given by:

$$\begin{aligned} Y_T &= \xi = \phi(W_T) \\ &= \max(S_T - X, 0) \text{ for a call}\ , \\ &= \max(X - S_T, 0) \text{ for a put}\ , \end{aligned} \tag{4}$$

where $\phi(\cdot)$ is a function of the Brownian motion W_T and represents the payoff of the option holder at time T.

Furthermore, the BSDE of equation (1) helps to determine the risk hedging strategy. For example, as at time T the option holder get the payoff Y_T, the seller of the contract should pay for the lost. In order to hedge the risk, he can use the Y_0 money to do investment at time 0, where Z_0 money is used to buy the stock and $Y_0 - Z_0$ is used to buy the bond. The function $g(\cdot)$ is given as following:

$$g(Y_t, Z_t, t) = -(rY_t + (b - r)Z_t + (R - r)(Y_t - Z_t)^-)\ . \tag{5}$$

Generally, the rate R is larger than r, thus we get a non-linear BSDE, i.e. the function $g(\cdot)$ is non-linear, while the Black-Scholes model can only deal with the linear case when $R = r$.

3 Numerical Algorithm

We study the algorithm of [7] based on time discretization and scaled random walk for non-linear BSDEs solving. We apply it in option pricing, as mentioned in Sect. 2.

In the numerical algorithm, the time period $[0, T]$ is devided into n time steps with interval $\delta = T/n$. Then we backwardly calculate Y_t.

As the state of the Brownian motion at time j can be approximated with scaled random walk:

$$W_t^j = \sqrt{\delta} \sum_{i=1}^{j} \varepsilon_i \ , \tag{6}$$

where $\{(\varepsilon_m)_{1<m<n}\}$ is a Bernoulli sequence, we can use a binomial tree to represent the Brownian motion W_t, as shown in Fig.1.(a). The upper-triangular form of it is shown in Fig.1.(b) for better description, where for a node labeled (i, j), i and j represent respectively the space step and the time step the node locates on.

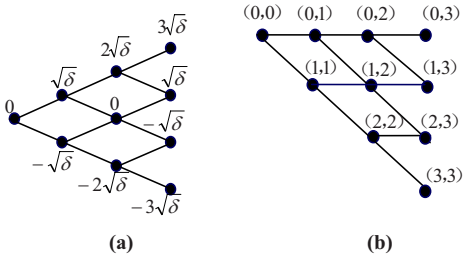

Fig. 1. Movement Approximation of the Brownian Motion

With the binomial tree model, we first calculate all the possible state of Y_T on level n with equation (4) and (6). In the following, an explicit scheme is performed iteratively from level $n-1$ to level 0 of the tree to get the solution (Y_0, Z_0), as shown in equation (7). The total number of nodes to be computed is $(n+1)(n+2)/2$.

$$\begin{cases} Y_{i,\ j} &= \frac{1}{2}(Y_{i,\ j+1} + Y_{i+1,\ j+1}) \\ & \quad + g(\frac{1}{2}(Y_{i,\ j+1} + Y_{i+1,\ j+1}), \frac{1}{2\sqrt{\delta}}(Y_{i,\ j+1} + Y_{i+1,\ j+1}), t_j) \ , \\ Z_{i,\ j} &= \frac{1}{2\sqrt{\delta}}(Y_{i,\ j+1} - Y_{i+1,\ j+1}). \end{cases} \tag{7}$$

4 Parallel Implementation

Parallelization works for option pricing with traditional binomial tree model are discussed in [4] and [5]. According to the numerical model for non-linear BSDEs,

we develop a block allocation algorithm by combining [4] and [5] to avoid large communication overhead.

We assume the number of time steps n and the number of processors p to be power of two. According to the structure of the binomial tree, the number of leaf node is $n+1$. All rows of nodes are distributed evenly among processors but the last processor receives an additional row. Thus each processor except the last one is in charge of $m = n/p$ rows of nodes. Initially option price of the leaf nodes are calculated simultaneously with equations (3), (4), (6) on each processor. For calculating the nodes of other levels, we only need to send the values of boundary nodes to the neighbor processor to perform equation (7). Thus large amount of data transfer is avoided. Furthermore, we introduce a parameter L so that the data transfer is performed every L levels to avoid frequent communication operation. In this way, the computation process is performed with $n/(L-1)$ iterations. During each iteration, every processor operates on a block of $m * L$ nodes except the last one, which has $(m+1)*L$ nodes. Without loss of generality, we assume L to be power of two plus one so that n can be divided by $(L-1)$. Figure 2 shows the block allocation for $n = 8, p = 2, L = 3$.

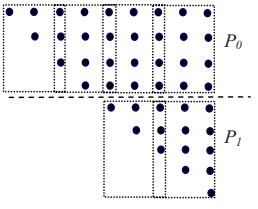

Fig. 2. Block Allocation for n=8, p=2, L=3

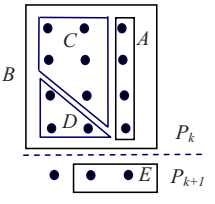

Fig. 3. Node Valuation in a Block

For the iteration valuating an arbitrary block B on processor P_k , we decompose block B into three regions: the inner regions A, C and the outer region D, as shown in Fig.3. The node value of the inner regions can be calculated only with the local data, while that of the outer region depends not only on local data, but also on the data of the neighbor block located on processor P_{k+1}. At the beginning of each iteration, data in A are already calculated before updating B. Thus C nodes can be updated only with local data according to equation (7). In the following, as the calculation of D nodes needs not only A nodes, but also the nodes of the first row (region E) on processor P_{k+1}, the corresponding data are transferred from processor P_{k+1} so that D nodes can be calculated.

To measure the performance of the parallel algorithm, the computation time and the communication time are evaluated. Overall, we obtain the total computing time of $O(n^2/p)$ for the parallel algorithm, with the communication time of $(O(n))$. Low communication overhead is achieved, as for large problem size the parallel communication time seems insignificant.

5 Experimental Results

We implement the parallel program by using C and MPI. Runtime experiments are performed on a LANGCHAO TIANSUO 10000 cluster with 2 Intel Xeon DP 2.8GHZ cpu and 2GB memory on each node. We use 16 nodes with 1 cpu per node for the experiment. The source code is compiled with the -O3 option set for gcc compiler. Table 1 shows the timing results for various number of time steps $n = 8192, 16384, 32768, 65536, 131072$. The block parameter L is fixed to 9.

Table 1. The Parallel Runtimes (in seconds) with the TIANSUO 10000 Cluster

n	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$
8192	2.33	1.52	0.85	0.74	0.7
16384	12.02	7.99	4.28	2.97	1.98
32768	51.35	35.7	18.27	8.72	5.35
65536	211.73	136.72	79.28	38.18	19.2
131072	864.19	575.65	338.94	160.71	83.71

The results in Table 1 seem optimistic. We can see that with a single processor, the running time increases greatly when the number of time steps n grows up. And the parallel runtime for $n = 131072$ decreases much more sharply than that with small n when the number of processors increases. For better analysis of the parallel runtime results, we use the data in Table 1 to produce speedup plots in Fig. 4.

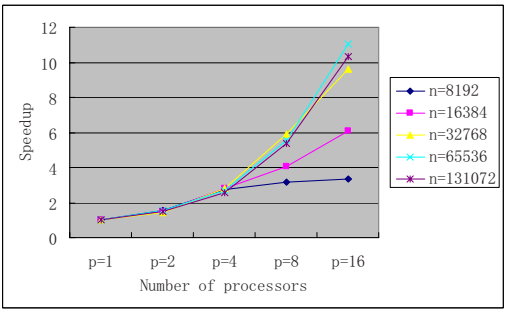

Fig. 4. Speedup Graph with the TIANSUO 10000 Cluster

As observed from Fig.4., for small number of time steps, the speedups increase very slowly while augmenting the number of processors, especially when $n = 8192$, the speedup only achieves 3.33 for $p = 16$. This could be explained that for small problem size, the parallel overhead is relatively large comparing with the computation time. Then for larger number of time steps, as the amount of

computation increase quadratically when n grows up, the parallel overhead tends to be insignificant. Hence with the augmentation of p, the speedups increase much more rapidly for $n = 32768, 65536, 131072$ and achieve respectively 9.60, 11.02, 10.32 when $p = 16$. Therefore large number of time steps offers a better problem for parallel computing.

6 Conclusion

In this paper, a binomial tree based numerical algorithm for non-linear BSDEs is applied in option pricing and studied for parallelization. A parallel algorithm with block allocation is proposed according to the inherent structure of the numerical model. Large communication overhead is avoided on considering both the communication frequency and the amount of data transfer. Parallel run time analysis is performed on a cluster with 16 nodes and shows promising results.

Acknowledgement

This work was supported by a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2006AA01A113).

References

1. Black, F., Scholes, M.: The pricing of options and corporate liabilities. Journal of Political Economy 81, 637–654 (1973)
2. Wan, J.W.L., Lai, K., Kolkiewicz, A.W., et al.: A Parallel Quasi-Monte Carlo Approach to Pricing American Options on Multiple Assets. International Journal of High Performance Computing and Ne tworking 4, 321–330 (2006)
3. Sak, H., Ozekici, S., Boduroglu, I.: Parallel computing in Asian option pricing. Parallel Computing 33, 92–108 (2007)
4. Huang, K., Thulasiram, R.K.: Parallel Algorithm for Pricing American Asian Options with Multi-Dimensional Assets. In: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications, pp. 177–185 (2005)
5. Gerbessiotis, A.V.: Architecture independent parallel binomial tree option price valuations. Parallel Computing 30, 301–316 (2004)
6. El Karoui, N., Peng, S., Quenez, M.C.: Backward Stochastic Differential Equations in Finance. Mathematical Finance 7, 1–71 (1997)
7. Peng, S., Xu, M.: The Numerical Algorithms and simulations for BSDEs (2008), ArXiv: math.PR/08062761
8. Surkov, V.: Parallel Option Pricing with Fourier Space Time-stepping Method on Graphics Processing Units. In: IEEE International Symposium on Parallel and Distributed Processing (2008)