High Performance Computing for a Financial Application Using Fast Fourier Transform

Sajib Barua, Ruppa K. Thulasiram*, and Parimala Thulasiraman

Department of Computer Science, University of Manitoba
Winnipeg, MB R3T 2N2 Canada
{sajib,tulsi,thulasir}@cs.umanitoba.ca

Abstract. Fast Fourier Transform (FFT) has been used in many scientific and engineering applications. In the current study, we have applied the FFT for a novel application in finance. We have improved a recently proposed mathematical model of Fourier transform technique for pricing financial derivatives to help design and develop an effective parallel algorithm using a swapping technique that exploits data locality. We have implemented our algorithm on 20 node SunFire 6800 high performance computing system and compared the new algorithm with the traditional Cooley-Tukey algorithm We have presented the computed option values for various strike prices with a proper selection of strike-price spacing to ensure fine-grid integration for FFT computation as well as to maximize the number of strikes lying in the desired region of the asset price.

Keywords: HPC for commercial application; Option pricing; Fast Fourier transform; Mathematical modeling; Parallel algorithm; Data locality.

1 Introduction

The finance industry demands efficient algorithms and high-speed computing in solving many problems [1]. In this research we cut across two historically established, technologically evolving and most importantly traditionally different areas: computing and finance - computational finance. Specifically, this paper addresses the problem of option pricing.

Terminologies: An option is a financial contract where one of the two parties involved, known as *holder*, gets the right (but not obligation) to *buy/sell* a set of underlying financial instruments such as stocks at a preset price (known as *exercise* or *strike* price) at a preset date (known as *exercise/maturity* date) *from/to* the other party known as the *writer*. If the holder decides to exercise the option, the writer is obligated to satisfy the holder's decision. Buying/selling underlying asset through such contract is referred to as *Call/Put* option. If the option can be exercised only at the maturity date, the option contract is known as *European* option, whereas if the option can be exercised any time prior to the maturity, it is known as *American* option. Value of a call/put option (shortly call/put value

* Author for Correspondence: tulsi@cs.umanitoba.ca

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1246–1253, 2005.

or call/put price) depends on the spot price of the underlying asset, strike price among other parameters such as risk-free interest rate, volatility[1] of the asset, period of the contract.

The solution for the optimal exercise policy for a financial option must typically be performed numerically, and is usually a computationally intensive problem. Pricing of options has been traditionally done using either binomial tree approach or using Monte-Carlo simulation or engineering approaches such as finite-differencing (see for example [2]). A recent addition to the numerical techniques for the option pricing problem is the use of Fast Fourier Transform (FFT) [3]. By providing an one-to-one mapping from the mathematics of Fourier space to the computational domain of the FFT, [4] explored the high performance computing for this problem.

In the current study, we develop an improved mathematical model of FFT for option pricing and a new parallel FFT algorithm. While there are many FFT algorithms available, for example, Stockham auto sort algorithm (SAS) [5] (chapter 1.7) and Bailey algorithm [6], we had to develop a new algorithm in the current study especially to satisfy the mathematics of the option pricing problem described in section 2. The structure of our new algorithm behaves similar to the SAS algorithm, however, captures the physics of option pricing closer than the SAS algorithm as explained in section 2. Due to lack of space we do not discuss the SAS or other available algorithms here. Readers are referred to [5] for an in depth look on various FFT algorithms. We leave the work on fine tuning SAS [5] (chapter 1.7) and Bailey's [6] algorithm for option pricing problem as a future study.

The rest of the paper is organized as follows. In section 2, we mention the drawback of one major related work on mathematical model of option pricing problem using Fourier transform and present an improvement to the mathematical modeling with which a finer mapping from mathematics to the FFT computational domain for option pricing is presented. In section 3, we present the new FFT algorithm, which exploits data locality to improve the performance. The results are presented in section 4, with call value results followed by the experimental results. We conclude the current study in section 5.

2 Drawback of an Existing FFT Model and Improved Model for Option Pricing

An important contribution of the current work is to alleviate the drawback in Carr-Madan (CM) model [3]. They developed a FFT model for option pricing in continuous and discrete form as follows: If $M = e^{-\alpha k}/\pi$ and $\omega = e^{-i}$ then

$$C_T(k) = M \int_0^\infty \omega^{vk} \psi_T(v) dv. \tag{1}$$

[1] Variation in the asset prices is generally split into two parts: (i) changes due to known factors affecting the asset price such as periodic changes - known as deterministic changes or *drift* in prices; (ii) changes due to unknown phenomena in the market place - generally known as *volatility*.

If $v_j = \eta(j-1)$ and applying trapezoidal rule for the integral on the right of equation (1), $C_T(k)$ can be written as

$$C_T(k) \approx M \sum_{j=1}^{N} \psi_T(v_j)\omega^{v_j k}\eta, k = 1, \ldots, N, \tag{2}$$

where the effective upper limit of integration is $N\eta$ and v_j corresponds to various prices with η spacing. Here $C_T(k)$ is the call option price; $\psi_T(v)$ is the Fourier transform of this call price given by $\psi_T(v) = \frac{e^{-rT}\phi_T(v-(\alpha+1)i)}{\alpha^2+\alpha-v^2+i(2\alpha+1)v}$; where α is a dampening factor and k is the logarithm of the strike price, $k = log(K)$); r is the interest rate; T is the period of the option contract. The calculation of $\psi_T(v)$ depends on the factor $\phi_T(u)$, where $u = v - (\alpha+1)i$. We derive $\phi_T(v)$ as, $\phi_T(v) = \int_0^{\lambda}(\cos(vk) + i\sin(vk))q_T(s)ds$ where λ is terminal spot price and integration is taken only in the positive axis.

To calculate the call values, equation (1) has to be solved analytically. The discrete form equation (2) is not suitable to feed into the existing FFT algorithms for example,Cooley-Tukey [7], Stockham auto sort [5](chapter 1.7) and Bailey [6]. Hence, the CM model in its current form cannot be used for faster pricing. This is a major drawback of using CM model for practical purposes and for real time pricing we need to improve this mathematical model.

This leads us to state the objectives of the current work as: (1) Improving the mathematical model that will be tractable for parallel computing and for getting accurate solutions quickly; (2a) Designing an efficient parallel FFT algorithm that can map the mathematics from the improved model to the computational domain; and (2b) implementing the algorithm on distributed memory architecture to study the performance.

Improved Mathematical Model: The limits on the integral have to be selected in such a way as to generate real values for the FFT inputs. To generate the closed form expression of the integral, the integrands, especially the function $q_T(s)$, have to be selected appropriately. Without loss of generality, we use uniform distribution for $q_T(s)$. This implies occurrence of a range of terminal log prices at equal probability, which could, of course, be relaxed and a normal or other distribution could be employed. Since the volatility of the underlying asset is assumed constant (low) the variation in the drift is expected to cause a stiffness[2]

[2] Stiffness occurs when two processes controlling a physical phenomenon proceeds at two extremely different rates. It is common in scientific problems such as chemical reactions and high temperature physics. When a system with such physical phenomenon is manifested in mathematics such as differential or integral equations, the mathematical system is known to be *stiff*, where solution of such systems of equations would require special techniques to handle the 'stiffness'. Drift and volatility in the finance systems act as two phenomena affecting the system away from equilibrium hence may induce 'stiffness'. Our assumptions of uniform distribution for the density function to make the integration easier, in conjunction with assumed constant volatility, however, naturally avoids this issue.

in the system. However, since we have assumed uniform distribution for $q_T(s)$, variation in drift is eliminated and hence the stiffness is avoided. Therefore, use of uniform distribution would make the integration easier.

For computation purposes, the upper limit of equation (1) is assumed as a constant value and the lower limit is assumed as 0. The upper limit will be dictated based on the terminal spot price. In other words, to finish the call option in-the-money[3], the upper limit will be smaller than the terminal asset price and hence we arrive at the the modified expression for $\phi_T(v)$ presented earlier.

Without loss of generality, further modifications are required as derived below. The purpose of these modifications is to generate feasible and tractable initial input condition to the FFT algorithm from these equations. Moreover, these modifications make the implementation easier. Due to lack of space we skip the mathematical derivation and present the final improved mathematical model as

$$\psi_T(v) = \frac{A}{\{B\}\{C^2 + D^2\}} \left[\{C\Delta + D\Delta_x\} + i\{C\Delta_x - D\Delta\} \right] \tag{3}$$

where, $A = e^{-rT} q_T(s)$; $B = (\alpha + 1)^2 + v^2$; $C = \alpha^2 + \alpha - v^2$; $D = (2\alpha + 1)v$. We use this final expression for the new parallel FFT algorithm to compute the call price function. The financial input data set for our parallel FFT algorithm is the calculated data points of $\psi_T(v)$ for different values of v. We refer equation (3) as *BTT-CM Model* or *BTT-CM equation.*

We then calculate call value for different strike price values v_j where j will range from 1 to N. The lower limit of strike price is 0 and upper limit is $(N-1)\eta$ where η is the spacing in the line of integration. Smaller value of η gives fine grid integration and a smooth characteristics function of strike price and the corresponding calculated call value. If γ is the spacing in k, then the values for k can be obtained from the equation: $k_u = -p + \gamma(u-1)$, for $u = 1, \ldots, N$. Hence, the log of the ratio of strike and exercise price will range from $-p$ to p where $p = \frac{N\gamma}{2}$. Substitution of previous equation for k_u in equation (2) and replacing v_j with $(j-1)\eta$ in the equation gives ($for\ u = 1, ..., N$)

$$C_T(k_u) \approx \frac{exp(-\alpha k_u)}{\pi} \sum_{j=1}^{N} \{e^{-i\gamma\eta(j-1)(u-1))} e^{ipv_j} \psi_T(v_j)\eta\}. \tag{4}$$

Comparing equation (4) with the basic FFT equation, we note that $\gamma\eta = \frac{2\pi}{N}$. Smaller values of η will ensure fine grid for the integration. But call prices at relatively large strike spacings (γ), few strike prices will lie in the desired region near the stock price [3]. Furthermore, if we increase the values of N, we

[3] In-the-money call option is a situation where underlying asset price of the option is larger than the strike price; at-the-money call means asset price equals the strike price; natural extension is for out-of-the-money call, which corresponds to a situation where the asset price is smaller than the strike price. These definitions are reversed for a put option

will get more intermediate points of the calculated call prices ($C_T(k_u)$) corresponding to different strike prices (v_j). This helps the investor to capture the call price movements of an option for different strike prices in the market. In the experimental result (section(4)) of 1024 (N) numbers of calculated call values, assuming $\eta = 0.25$ with the intuition that it will ensure fine grid integration, γ is calculated as 0.02454. Similar to basic FFT equation, equation (4) can also be parallelized with an efficient parallel algorithm. In the next section we develop a data swapping technique that exploits data locality to reduce communication on a parallel computer and effectively apply our mathematical model. We implement this algorithm with the inputs derived from equation (4).

3 An Effective Parallel FFT Algorithm

Figure 1 illustrates our data swap algorithm. We assume we have N ($N = 2^m$) data elements and P ($P = 2^p$) processors where $N > P$ [8]. In our algorithm, we apply the blocked data distribution and the first ($\log N - \log P$) stages require no communication. However, in the last $\log P$ stages that require communication, we swap some data at each stage and let the data reside in the processor's local memory after swapping. Therefore, the identity of some of the data points in each processor changes at every stage of the $\log P$ stages.

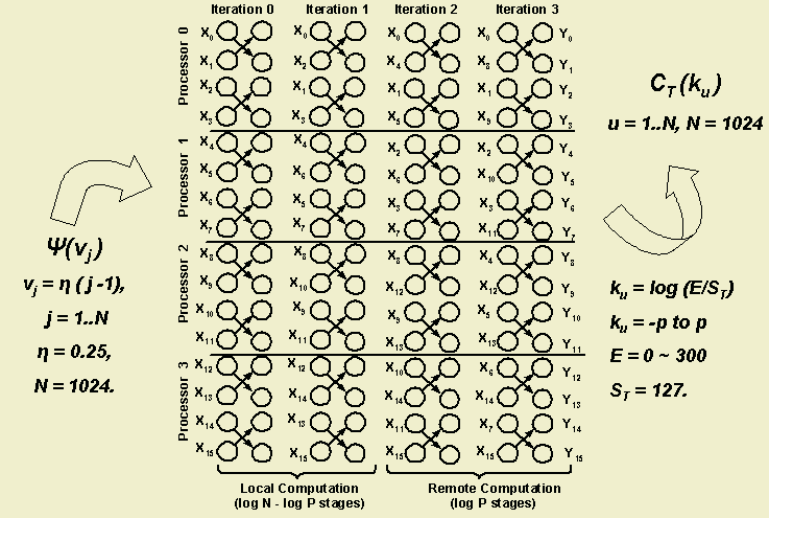

Fig. 1. Data Swap Algorithm

In figure (1), we can see that in iteration 2, processor 0 needs two input data points with index 4 and 5 and these do not reside in the local processor. Hence, we need two send operations to bring these values from processor 1. In general, for an input data point with N/P data in every processor, $N/(2P)$ communication is

required. This is half of what is required in the Cooley-Tukey algorithm. That is, in the new parallel FFT algorithm, the number of communications is reduced by half. We take advantage of the the fact that communication between processors is point to point and swap the data in a similar manner. However, in this case, only $\frac{N}{2P}$ amount of data (message size) is communicated by each processor at every stage. Also note that, data swapping between processors at each location allows both the upper and lower part of the butterfly computations to be performed locally by each processor. This improvement enhances good data locality and thereby providing performance increase in the new FFT algorithm compared to the Cooley-Tukey algorithm. Analytically, the parallel runtime is given by [9] $t_c(N/P)\log N + t_s^{''}\log P + t_w(N/2P)\log P$, where t_s is the start up time; t_w is the per word transfer time; and t_c is the time required for the butterfly computation.

4 Results and Discussions

Option Pricing Results: Figure (1) shows how the data swap algorithm calculates the call values from the input data set generated from the BTT-CM equation. The data swap algorithm calculates N number of call values. When the call option is in-the-money, the investor would prefer to exercise the option (purchasing the option) at the strike price and immediately sell the asset in the market at the terminal spot price. Thus, the holder can profit. Figure (2) depicts the calculated in-the-money call values for different strike prices using the data swap algorithm. In the experiment of call value computation, strike price can be any value between 0 and 300. Our data swap algorithm can calculate (figure 1) call values for in-the-money, at-the-money and out-of-the-money call options. We are considering in-the-money call where the terminal spot price is always greater than the strike price. Therefore, figure 2 plots a portion of the calculated call values (in-the-money) from the output values of the data swap algorithm. The plot shows that the normalized option value is decreasing with the increase of strike price. If X, the strike price, is decreased, the call option value is expected to increase, which can be seen in figure 2. For larger values of N we can get more number of call values computed for the strike price range from 0 to 127, which makes the plot as a continuous function.

Significant Experimental Results: The experiments were conducted on a 20 node SunFire 6800 high performance computing system at the University of Manitoba running MPI. The Sunfire consists of Ultra Sparc III CPUs, with 1050 MHz clock rate and 40 gigabytes of memory and runs Solaris 8 operating system. The data generated in section 2 is used for the FFT input. Due to lack of space, we present only limited number of results.

Figure (3 a) depicts a comparison of the execution time between the swap algorithm and the Cooley-Tukey algorithm. At each iteration $\frac{N}{2P} = \frac{2^{20}}{2^5} = 2^{15}$ data points are swapped on each of the 16 processors. On a 2 processor machine, there are $\log 2^{20} - \log 2 = 19$ local computations and only 1 remote communication. However, there is a significant decrease in execution time in 16 processors. This is

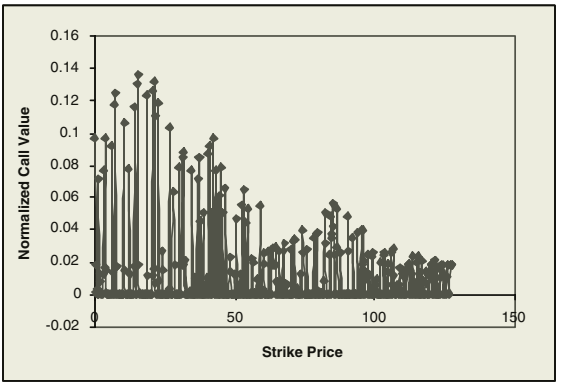

Fig. 2. Computed Call Values

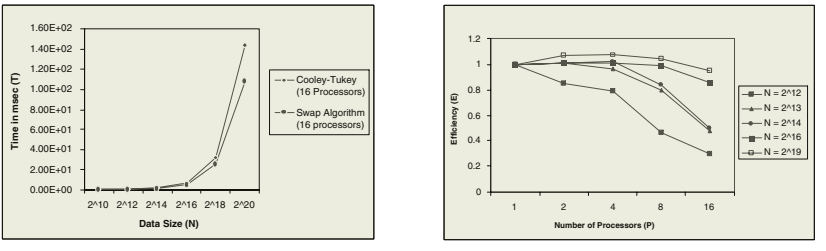

Fig. 3. a)Comparison of the execution times of swap and Cooley-Tukey algorithms, and b) Efficiency of the data swap algorithm

attributed to the fact that in MPI, the packing and unpacking of $\frac{N}{2P} = 2^{18}$ data elements for each of the 2 processors requires significant amount of time. When we compare the swap algorithm to the Cooley-Tukey algorithm in figure (3) on 16 processors, the swap algorithm performs 15% better than Cooley-Tukey algorithm on a data size of 2^{20}.

We calculated the efficiency of the swap algorithm for various processors on a fixed data size as presented in figure (3 b). The efficiency for 16 processors is close to 1. For 4, 8, and 16 processors the efficiency is 90% for data sizes 2^{14}, 2^{16}, 2^{19} respectively. Also for 8 and 16 processors the efficiency is 50% for 2^{12} and 2^{13} respectively. These results illustrate that as we increase the data size and the number of processors, the swap algorithm exhibits very good scalability.

5 Conclusions

Without loss of generality, we have improved the mathematical modeling of FFT for option pricing and we have identified appropriate values for the parameters to generate the input data set for the parallel FFT computations. We have reduced the communication latency by improving the data locality. We have presented the computed call values for various strike prices with a proper selection of

strike-price spacing to ensure fine-grid integration for FFT computation as well as to maximize the number of strikes lying in the desired region of the asset price. Compared to the traditional Cooley-Tukey algorithm, the current algorithm with data swapping performs better by more than 15% for large data sizes.

Acknowledgement

The last two authors acknowledge partial financial support from Natural Sciences and Engineering Research Council (NSERC) of Canada and the University of Manitoba Research Grant Program (URGP). They also gratefully acknowledge the discussions with Prof. Sanjiv R. Das, Department of Finance, Leavey School of Business, Santa Clara University, Santa Clara, CA, USA, on the Fourier transform application for finance problems especially the option pricing problem.

References

1. E. J. Kontoghiorghes, A. Nagurnec, and B. Rustem. Parallel Computing in Economics, Finance and Decision-making. *Parallel Computing*, 26:207–209, 2000.
2. J.C. Hull. *Options, Futures and Other Derivatives*. Prentice Hall, Upper Saddle River, NJ, 5th edition, 2002.
3. P. Carr and D. B. Madan. Option Valuation using the Fast Fourier Transform. *The Journal of Computational Finance*, 2(4):61–73, 1999.
4. R. K. Thulasiram and P. Thulasiraman. Performance Evaluation of a Multithreaded Fast Fourier Transform Algorithm for Derivative Pricing. *The Journal of Supercomputing*, 26(1):43–58, Aug. 2003.
5. C.Van Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM: Frontiers in Applied Mathematics, Philadelphia, PA, 1992.
6. D. H. Bailey. FFTs in External or Hierarchical Memory Fourier. *The Journal of Supercomputing*, 4, 1990.
7. J.W. Cooley, P.A. Lewis, and P.D. Welch. *The Fast Fourier Transform and its Application to Time Series Analysis*. Wiley, New York, 1977. In statistical Methods for Digital Computers.
8. A. Grama and A. Gupta and G. Karypis and V. Kumar. *Introduction to Parallel Computing*. Addison Wesley, New York, NY, Second edition, 2003.
9. S. Barua. Fast Fourier Transform for Option Pricing: Improved Mathematical Modeling and Design of an Efficient Parallel Algorithm. Master's thesis, University of Manitoba, Winnipeg, MB, Canada, July 2004.