CSE 6220 Term Project

1. Title:

Parallel Numerical Methods for American Type Option Pricing

2. Participants:

Jian Lu, Ph.D. student, School of Building Construction, Jian.Lu@gatech.edu

Chenhao Liu, Ph.D. student, School of Civil and Environmental Engineering, chenhaoliu@gatech.edu

3. Goal:

Options are financial instruments that give the holder the right, but not obligation, to buy or sell an asset at a specified price at some future time. An American option has the feature that the holder can exercise at any time before maturity. By now, there is no analytic representation for the value of American put options, and therefore their values can only be obtained by numerical methods. The Projected Successive Over Relaxation (PSOR) method has been proven to be an effective strategy to price American options. However, the PSOR method is serial and highly computation intensive. Therefore, as a motivation for this project, we wanted to implement a parallel version of the PSOR method and evaluate its performance in terms of computational time consumption. This idea was successfully implemented in the CUDA GPU computing environment. As we expected, the results of the parallel PSOR method showed promising potential in terms of reducing the computation time consumption while maintaining results' accuracy, compared to the benchmark of the serial version of the algorithm.

4. Approach:

1). Model Problem:

Options are financial instruments that give the holder the right, but not obligation, to buy or sell an asset at a specified price (strike price) at some future time. An American option has the feature that the holder can exercise at any time before maturity. By now, there is no analytic representation for the value of American put options. Their values are usually obtained by numerical methods.

We start with Black-Scholes model with constant volatility for option pricing.

$$dS = \mu S dt + \sigma S dW$$

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} + rS\frac{\partial u}{\partial S} - ru$$

Consider an American put option, which can be exercised at any time up to expiry T. It is not possible for the option price below the payoff value, hence we must have

$$u(S,t) \geq g(S)$$

Where g(S) = K − S, S is stock price, K is the strike price, and g(S) is the payoff function. At every point of time, the American option holder is confronted with the decision that either exercising or holding the option, so

$$\mathcal{L}u = \frac{\partial u}{\partial \tau} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} - rS\frac{\partial u}{\partial S} + ru = 0 \quad \vee \quad u(S,t) = g(S)$$

Combining these conditions, we get the linear complementarity problem (LCP) for American put option.

$$\begin{aligned} \mathcal{L}u &\geq 0 \\ u &\geq g \\ \mathcal{L}u = 0 \vee (u - g) &= 0 \end{aligned}$$

Where the notation $Lu = 0 \vee (u-g) = 0$ denotes that either $Lu = 0$ or $(u - g) = 0$ everywhere in the solution domain. The boundary condition associated with American put option is

$$u(S,\tau) = 0, \ as \ S \to \infty$$

Applying implicit method of finite difference discretization to the backward Black-Scholes equation, we get

$$\frac{u_i^m - u_i^{m-1}}{\Delta\tau} = \theta\left(\frac{1}{2}\sigma^2 S_i^2 \frac{u_{i+1}^m - 2u_i^m + u_{i-1}^m}{(\Delta S)^2} + rS_i \frac{u_{i+1}^m - u_{i-1}^m}{2\Delta S} - ru_i^m\right)$$

By rearranging the above equation we have

$$a_i^m u_{i-1}^m + b_i^m u_i^m + c_i^m u_i^m = u_i^{m-1}$$

$$\begin{aligned} a_i^m &= -\tfrac{1}{2}\theta\Delta\tau\left(\sigma^2 S_i^2 \tfrac{1}{(\Delta S)^2} - rS_i \tfrac{1}{\Delta S}\right) \\ b_i^m &= 1 + \theta\Delta\tau\left(\sigma^2 S_i^2 \tfrac{1}{(\Delta S)^2} + r\right) \\ c_i^m &= -\tfrac{1}{2}\theta\Delta\tau\left(\sigma^2 S_i^2 \tfrac{1}{(\Delta S)^2} + rS_i \tfrac{1}{\Delta S}\right) \end{aligned}$$

According to boundary conditions, the value of boundary parameters can be set as following.

$$\begin{aligned} b_N = 1, \ a_N = 0 \\ B_N = 1, \ A_N = 0 \end{aligned}$$

We can represent the whole matrix equation in short form.

$$Au^m = u^{m-1}$$

Where A is triangle matrix take the following form

$$A = \begin{bmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_1 & b_2 & c_2 & \dots & 0 \\ 0 & a_2 & b_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & c_{n-1} \\ 0 & \dots & 0 & a_{n-1} & b_n \end{bmatrix}$$

The corresponding discrete linear complementarity problem becomes

$$\begin{aligned} Au^m - u^{m-1} &\geq 0 \\ u^m &\geq g \\ \left(Au^m - u^{m-1}\right)\left(u^m - g\right) &= 0 \end{aligned}$$

Where g is value of payoff, the inequalities and multiplication are under-stood to be element-wise.

2). The PSOR Algorithm:

The above problem can be solved by Projected SOR (PSOR) which is suggested by Cryer (1971). Its idea is to apply maximization constraint to SOR algorithm.

Considering the general form of problem: $Av = f$

For i = 1, 2, …, n:

$$(1) \quad \begin{aligned} t_i^{(k+1)} &= \frac{1}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij} v_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} v_j^{(k)} + a_{ii} f_i\right) \\ v_i^{(k+1)} &= \max\{v_i^{(k)} + \omega(t_i^{(k+1)} - v_i^{(k)}), g\} \end{aligned}$$

Pseudo Code for PSOR Algorithm

Initialization

Financial variables: (K, T, σ, r) as needed.

Grid variable: Smax, Smin, e.g, Smax = 2K, Smin = 0. M as maximum of time steps, N as maximum of nodes, Δt = T/M, ΔS = (Smax – Smin)/N.

Algorithm variables : Relaxation parameterω ∈ [1,2) e.g. ω = 1.3; PSOR Convergence error tolerance, e.g. epsilon = 1.e-9; Set up matrix A, as in (2.6) together with corresponding boundary conditions.

Core Calculations

Initial iteration vector $V^{(M)} := (g(S_0), g(S_1), \ldots, g(S_N))$, where $g(\cdot) = K - S_i$ is the payoff function at maturity.

Time loop: for m = M, M-1, …, 1

Calculate $b^{(m)} = V^{(m+1)}$

Set $v^{(0)} := \max\{A^{-1}b^{(m)}, g(S)\}$

PSOR loop: while $||v^{(k+1)} - v^{(k)}|| > \varepsilon$

Do equation (1)

end while

$V^{(m)} = v^{(k+1)}$

End for

Final result: V := $V^{(0)}$

3). Optimal Omega

In SOR algorithm, the value of Omega influences the speed of convergent. To optimize the speed of the PSOR algorithm, we use the optimal value of Omega as specified by the following equation.

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(G_{Jac})^2}}$$

Please note the optimal Omegas are pre-computed by the Matlab in order to benchmark the computing efficiency of PSOR algorithm only.

4). Parallel PSOR Algorithm

Equation (1) can be implemented in parallel using the red-black ordering technique. The node is denoted red or black according to whether *i* is odd or even. If ***i*** is odd, the (a) phase 1 node (*i*) is marked red, and if *i* is even, the node (*i*) is marked black. The red-black ordering technique is illustrated in Figure 1. The evaluation of each *v(i),* corresponding to red nodes, involves the values of black neighbor nodes only and itself, and vice versa.

Figure 1: Red Black Ordering Technique

Based on the red-black ordering technique, the approximation *v(i)* can be updated in a different order suggested by Equation (1). Figure 2 illustrate the iteration steps. Each iteration of this method consists of two phase:

(1) Updating all the black values with even i first:

a. calculate the t_i with the neighboring red nodes

b. update the black nodes by taking the maximization as stated in Equation (1)

(2) Updating all the red values with odd i:

a. calculate the t_i with the neighboring black nodes that updated in step (1)

b. update the red nodes by taking the maximization as stated in Equation (1)

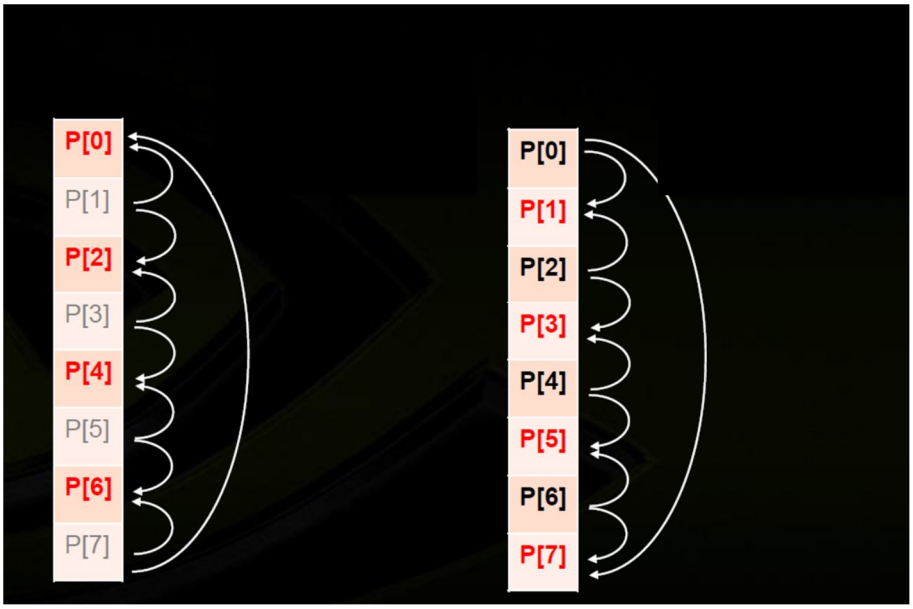

Figure2: Red-Black iterations

In phase 1, the computation of all red nodes depends on the values of all black nodes which are already available in the phase 2 at the last iteration. Thus, the computation can be partitioned into a number of independent tasks and performed by GPU. In phase 2, the computation of all black nodes depends on the values of all red nodes which have been computed in the phase 1. Similarly, it can also be partitioned into a number of independent tasks and performed by GPU.

5). Implementation.

To fully understand the functional performance of the PSOR algorithm in different computing environments, we proposed three versions of computing measures: a serial version (a built-in serial function) using MATLAB, a serial version using C (sor.c) and a parallel version using CUDA C (psor.cu).

To run the psor.cu file, type in the follow commands (place the N.txt and om.txt in the same folder with scripts before running):

nvcc -arch sm_13 psor.cu -o psor

Similarly, to run the sor.c file, type in the follow commands:

gcc –o sor sor.c -lm

5. Evaluation and Metrics:

In this project, we experimented the pricing of an American put option of a stock asset with the following characteristics: current price of the stock is 50, strike price is 50, maturity is in 5 months, interest rate is 10%, and sigma is 0.4. For this specific situation, we proposed the discrete step of the finite difference method to be uniformly ranging from 500 to 19500, with a gap of 500 in each partition. For each N in the range, we also calculated the optimal omega corresponding with N, which is a desirable measure in practice and pass it into the PSOR algorithm as well.

We applied this setting to both the serial and parallel version of the PSOR algorithm, with the output being the runtime consumption and option price. Each run corresponds to a distinct set of N and omega. The experiment results are summarized in Table 1 and Table 2 as follows. In particular, there are 10 runs for the parallel version of strategy, and there are 5 runs for the serial version of strategy.

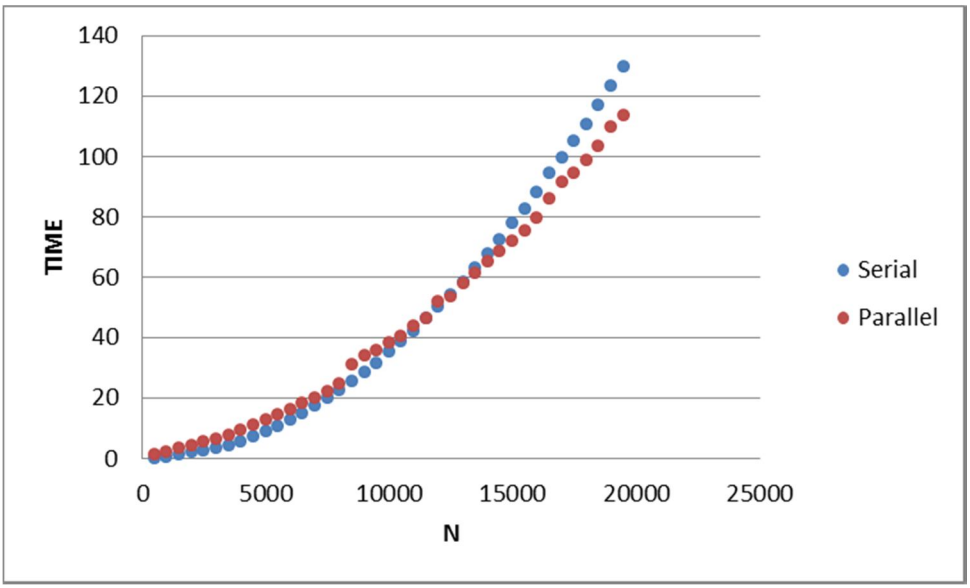

Figure 3. Computational Speed Performance

N	OM	PRICE	TIME1	TIME2	TIME3	TIME4	TIME5	TIME6	TIME7	TIME8	TIME9	TIME10	TIMEAVERAGE
500	1.780101271	4.274737301	1.17	1.22	1	1.13	1.25	1.11	1.15	1.15	1.13	1.15	**1.15**
1000	1.883205774	4.274862648	2.26	2.2	1.91	2.35	2.33	2.16	2.32	2.22	2.12	2.25	**2.21**
1500	1.92053348	4.27488476	3.44	2.77	3.34	3.53	3.46	3.19	3.38	3.30	3.40	3.38	**3.32**
2000	1.939786606	4.27489266	3.9	3.81	4.3	4.3	4.28	4.47	4.20	4.18	4.29	4.29	**4.20**
2500	1.951531648	4.274896612	5.2	5.48	5.26	5.32	5.43	5.56	5.35	5.37	5.34	5.40	**5.37**
3000	1.959443248	4.274898554	6.47	6.47	6.23	6.64	6.69	6.76	6.61	6.55	6.46	6.62	**6.55**
3500	1.965134689	4.274899723	7.39	8.2	7.53	7.68	8.02	8.23	7.78	7.83	7.78	7.89	**7.83**
4000	1.969425441	4.274900551	9	9.25	9.01	9.48	9.9	9.41	9.43	9.35	9.45	9.51	**9.38**
4500	1.972775891	4.274901112	10.63	10.96	10.6	10.89	11.29	10.79	10.90	10.87	10.94	10.95	**10.88**
5000	1.975464593	4.274901481	12.47	12.48	12.18	12.71	12.96	12.35	12.66	12.54	12.57	12.63	**12.56**
5500	1.977669971	4.274901734	14.13	14.71	13.94	14	14.55	14.43	14.14	14.27	14.25	14.27	**14.27**
6000	1.979511598	4.274901939	16.22	16.21	15.73	15.63	16.3	16.08	15.75	15.99	16.02	15.96	**15.99**
6500	1.981072608	4.27490212	18.16	18.18	17.86	18.27	18.85	18.11	18.25	18.24	18.36	18.35	**18.26**
7000	1.982412598	4.274902275	20.14	20.15	19.38	19.97	20.28	20.03	20.01	19.99	19.83	20.02	**19.98**
7500	1.983575404	4.280424819	21.77	22.1	21.65	22.25	22.81	22.91	22.22	22.24	22.23	22.44	**22.26**
8000	1.984593989	4.274902461	24.18	25.23	23.81	24.5	24.56	25.65	24.65	24.65	24.18	24.70	**24.61**
8500	1.985493617	4.274902535	30.97	31.15	30.39	31.05	31.01	30.82	31.07	30.92	30.70	30.93	**30.90**
9000	1.986293976	4.274902593	34.48	34.09	33.17	33.32	35.11	34.43	33.47	34.01	34.14	34.08	**34.03**
9500	1.987010638	4.27490265	35.69	35.26	34.35	35.47	36.53	35.74	35.43	35.50	35.44	35.68	**35.51**
10000	1.98765608	4.274902685	37.74	37.98	37.43	37.74	38.69	38.26	37.79	37.95	38.06	38.08	**37.97**
10500	1.988240414	4.278846724	40.28	39.78	39.63	40.01	42.01	40.43	39.96	40.30	40.82	40.59	**40.38**
11000	1.988771928	4.274902765	42.76	43.12	42.33	43.69	44.84	43.99	43.58	43.47	43.59	43.86	**43.52**
11500	1.989257472	4.274902799	45.11	45.94	45.3	47.26	46.95	47.05	47.00	46.37	46.13	46.79	**46.39**
12000	1.989702765	4.274902813	51.82	52.56	51.05	51.44	52.08	52.39	51.66	51.86	51.56	51.83	**51.83**
12500	1.99011261	4.274902845	52.77	54.75	52.2	53.24	54.49	54.59	53.54	53.65	53.35	53.81	**53.64**
13000	1.99049108	4.274902859	57.28	57.65	56.71	58.43	58.79	59.04	58.27	58.02	57.75	58.38	**58.03**
13500	1.990841644	4.274902869	60.72	61.77	60.18	61.78	62.34	62.59	61.78	61.59	61.26	61.89	**61.59**
14000	1.991167279	4.274902894	64.77	66.38	64.47	65.3	66.09	65.62	65.52	65.45	65.28	65.54	**65.44**
14500	1.991470553	4.274902903	67.58	69.31	67.28	68.63	70.16	69.25	68.77	68.71	68.72	69.04	**68.74**
15000	1.991753692	4.277663396	70.86	72.44	71.1	72.2	73.93	72.24	72.25	72.15	72.52	72.55	**72.22**
15500	1.992018637	4.274902934	74.9	76.5	75.2	74.94	76	76.31	75.25	75.59	75.60	75.61	**75.59**
16000	1.992267088	4.274902941	78.34	80.93	78.45	79.97	81.23	79.71	80.16	79.83	79.84	80.12	**79.86**
16500	1.992500538	4.274902953	84.42	87.63	86.67	86.03	84.87	85.49	86.35	85.92	85.77	85.74	**85.89**
17000	1.992720305	4.274902965	90.97	92.18	91.82	91.58	91.14	92.01	91.70	91.63	91.48	91.59	**91.61**
17500	1.992927559	4.274902971	94.08	94.31	94.24	94.20	94.11	95.09	94.22	94.32	94.18	94.35	**94.31**
18000	1.993123338	4.274902975	98.21	99.09	98.83	98.65	98.33	100.5	98.74	98.91	98.58	98.95	**98.88**
18500	1.993308571	4.274902981	103.36	103.79	103.66	103.58	103.42	103.86	103.62	103.61	103.54	103.60	**103.60**
19000	1.993484086	4.274902987	109.23	109.96	109.74	109.60	109.33	111.53	109.67	109.87	109.54	109.92	**109.84**
19500	1.993650629	4.274902996	112.62	113.76	113.42	113.19	112.78	115.05	113.30	113.45	113.10	113.48	**113.41**

Table 1. Experimental Results of Parallel Computing of PSOR Algorithm

N	OM	PRICE	TIME1	TIME2	TIME3	TIME4	TIME5	TIMEAVERAGE
500	1.780101271	4.274737	0.18	0.18	0.18	0.12	0.165	**0.17**
1000	1.883205774	4.274863	0.72	0.72	0.67	0.48	0.648	**0.65**
1500	1.92053348	4.274885	1.48	1.6	1.37	1.06	1.378	**1.38**
2000	1.939786606	4.274893	1.99	2.68	1.62	1.6	1.973	**1.97**
2500	1.951531648	4.274897	2.29	2.92	2.29	2.29	2.448	**2.45**
3000	1.959443248	4.274899	3.28	3.28	3.27	3.31	3.285	**3.29**
3500	1.965134689	4.2749	4.44	4.43	4.43	4.43	4.433	**4.43**
4000	1.969425441	4.274901	5.76	5.77	5.77	5.77	5.768	**5.77**
4500	1.972775891	4.274901	7.26	7.27	7.26	7.27	7.265	**7.27**
5000	1.975464593	4.274901	8.94	8.95	9.06	8.95	8.975	**8.98**
5500	1.977669971	4.274902	10.79	10.78	10.95	10.78	10.83	**10.83**
6000	1.979511598	4.274902	12.79	12.81	12.79	12.79	12.8	**12.80**
6500	1.981072608	4.274902	14.99	14.99	14.95	15.09	15.01	**15.01**
7000	1.982412598	4.274902	17.32	17.36	17.32	17.33	17.33	**17.33**
7500	1.983575404	4.280425	19.84	19.96	19.82	19.85	19.87	**19.87**
8000	1.984593989	4.274902	22.6	22.63	22.64	22.53	22.6	**22.60**
8500	1.985493617	4.274903	25.39	25.37	25.36	25.68	25.45	**25.45**
9000	1.986293976	4.274903	28.38	28.39	28.37	28.39	28.38	**28.38**
9500	1.987010638	4.274903	31.57	31.58	31.56	31.58	31.57	**31.57**
10000	1.98765608	4.274903	35.12	34.95	35.03	35.01	35.03	**35.03**
10500	1.988240414	4.278847	38.41	38.43	38.63	38.43	38.48	**38.48**
11000	1.988771928	4.274903	42.11	42.11	42.22	42.11	42.14	**42.14**
11500	1.989257472	4.274903	46.45	45.93	46.17	45.94	46.12	**46.12**
12000	1.989702765	4.274903	50.09	49.94	49.93	49.96	49.98	**49.98**
12500	1.99011261	4.274903	54.11	54.16	54.61	54.11	54.25	**54.25**
13000	1.99049108	4.274903	58.43	58.44	58.62	58.44	58.48	**58.48**
13500	1.990841644	4.274903	62.95	63.05	62.88	62.99	62.97	**62.97**
14000	1.991167279	4.274903	67.71	68.96	67.54	67.61	67.96	**67.96**
14500	1.991470553	4.274903	72.39	72.41	72.37	72.54	72.43	**72.43**
15000	1.991753692	4.277663	78.06	78.72	77.39	77.57	77.94	**77.94**
15500	1.992018637	4.274903	82.53	82.66	82.8	82.96	82.74	**82.74**
16000	1.992267088	4.274903	87.84	87.86	88.49	87.86	88.01	**88.01**
16500	1.992500538	4.274903	93.56	95.51	94.44	94.01	94.38	**94.38**
17000	1.992720305	4.274903	98.96	100.4	99.28	99.43	99.52	**99.52**
17500	1.992927559	4.274903	104.71	105.31	105.63	104.79	105.11	**105.11**
18000	1.993123338	4.274903	110.8	110.86	110.77	110.7	110.79	**110.79**
18500	1.993308571	4.274903	117.54	116.9	116.84	116.96	117.06	**117.06**
19000	1.993484086	4.274903	123.58	123.19	123.56	123.44	123.44	**123.44**
19500	1.993650629	4.274903	129.49	129.69	129.87	129.59	129.66	**129.66**

Table 2. Experimental Results of Serial Computing of PSOR Algorithm

Figure 3 shows that the advantage of parallel computing does not reveal itself until N becomes sufficiently large (>= 13000 in this case). As N gets larger the computation time consumption of the parallel strategy becomes shorter than that of the serial version, which shows promising potential of the parallel strategy for exquisite setting of discrete step partitions.

Based on the computation results, we also calculated the speedup and efficiency which are shown in Figure 4 and Figure 5. Note that the number of processors (p) in this context is calculated to be 16, because the total number of threads on the M2090 GPU is 512, and each processor has 32 threads, therefore p = 512/32 = 16.

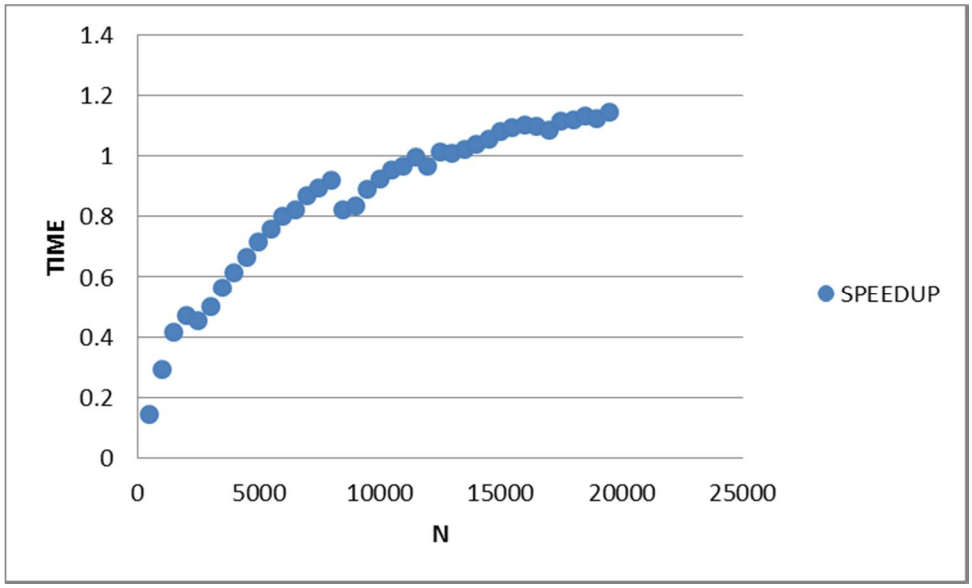

Figure 4. Speedup

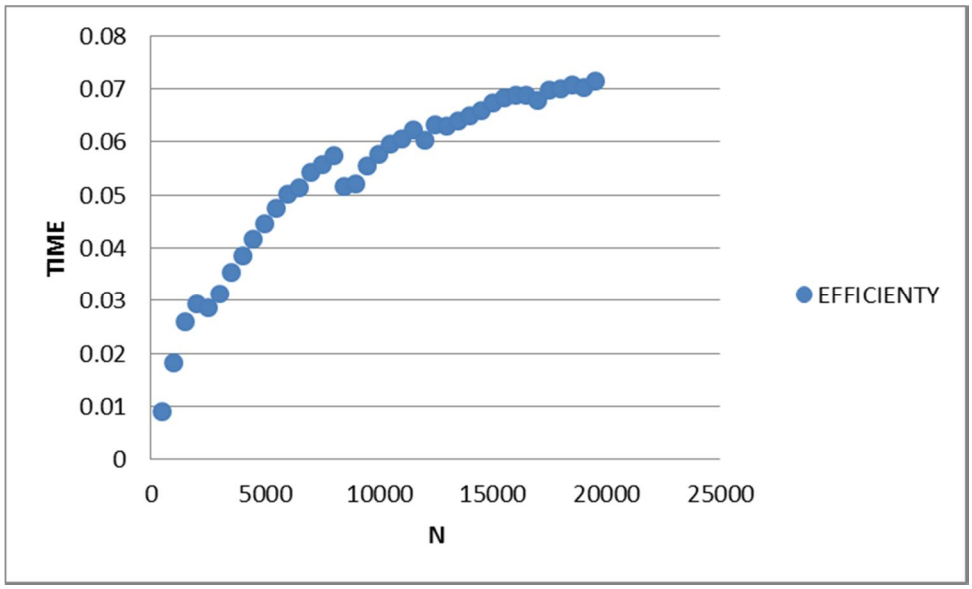

Figure 5. Efficiency

Based on the current results of speedup and efficiency computation results, the performance improvement of the parallel strategy is not significant, due to the fact that the parallel version's advantage is not revealed until N is sufficiently large, and our experiment could only go up to N = 19500. This is due to the limitation of computing the optimal value of omega, whose value is obtained through MATLAB and is highly resource-demanding and time-consuming. However, should we have obtained results for even larger dimension of N (we can already observe a trend from current results), the performance difference between the parallel strategy and serial computation will be much more appealing and promising. We believe that the parallel strategy (and more importantly, the thinking process behind the development of the strategy) will be of great value in the financial industry.

6. Schedule and Milestones:

We managed to make our progress stick to the schedule and milestones. However, the development of the parallel strategy and the debugging of the CUDA C program did cost us longer than we anticipated.

7. Challenges:

1). The PSOR algorithm: The PSOR algorithm, in its essence, makes use of the numerical finite difference method to solve the pricing formula of American type option, represented by Partial Differential Equations (PDE). We needed to study the full context of related mathematics subjects to successfully program the PSOR algorithm using C and CUDA C.

2). Parallel the PSOR algorithm in CUDA C: The most challenging part in this step is to determine how to parallel a naturally serial algorithm. As discussed in the "Approach" section, we saw potential in the algorithm's iterative matrix computation part. By referencing the Red-Black SOR algorithm's logic, we decomposed the rectangle computation domain into two partitions: one containing odd-numbered index elements and the other containing even-numbered index elements. The computations of the two partitions' elements are independent from each other, and therefore were paralleled as two separate steps in our program.

8. Resources:

Software: XCODE, GEDIT, MATLAB, GNU-GCC, CUDA NVCC

Hardware: GT CoC JINX Cluster

Source Code: CUDA SDK

Data Sets: None

9. Related work/projects:

This is the project only for the CSE 6220 HPC class.

10. References:

(1) Jouni Kerman. Numerical Methods for Option Pricing: Binomial and Finite difference Approximations. Publication of Courant Institute of Mathematical Sciences, New York University, January 2012.

(2) Manfred Gilli. Instruction material of course “Numerical Methods in Finance, University of Genève, Spring 2008.

(3) http://www.mathfinance.cn/psor-lcp/, Accessed in March 2012.

(4) Longstaff, F.A., and Schwartz, E. S. (2001). “Valuing American options by simulation: A simple least-squares approach.” Rev. Financ. Stud., 14(1), 113-147.

(5) NVIDIA CUDA C Programming Guide, http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf, Accessed in March 2012.

(6). Liu, Peng (2008). Numerical Methods for American Option Pricing. Numerical Methods for American Option Pricing. Masters thesis, University of Oxford.

(7). Zhang, C., Lan, H. and Estrade, B. D., "Parallel SOR iterative algorithms and performance evaluation on a Linux cluster," Proceedings by the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA"05), CSREA Press, vol. I, 263-268, 2005.