文章编号：0253-374X(2013)05-0792-07 DOI:10.3969/j.issn.0253-374x.2013.05.027

高性能计算中的亚式期权蒙特卡罗加速方法

姜广鑫[1]，徐承龙[1]，寇大治[2]，徐 磊[2]
(1.同济大学 数学系，上海 200092；2.上海超级计算中心，上海 201203)

摘要：研究蒙特卡罗控制变量方法在CPU(central processing unit)集群和GPU(graphic processing unit)计算环境中的实现问题.以离散取样的随机波动率下的算术平均亚式期权为例，选取合适的控制变量，分别研究了在CPU集群和GPU计算中算法与硬件并行加速两者的运算效率，并讨论了模型参数的变化对计算结果的影响.数值试验表明采用算法与硬件加速相结合的方法可以极大提高计算效率、缩短运算时间.

关键词：蒙特卡罗方法；随机波动率；控制变量；CPU(central processing unit)集群计算；GPU(graphic processing unit)计算
中图分类号：O242.1；O246 **文献标志码**：A

Monte Carlo Acceleration Method for Pricing Asian Options in High Performance Computation

JIANG Guangxin[1], *XU Chenglong*[1], *KOU Dazhi*[2], *XU Lei*[2]
(1.Department of Mathematics, Tongji University, Shanghai 200092, China; 2.Shanghai Supercomputer Center, Shanghai 201203, China)

Abstract: An investigation was made into the control variate method of Monte Carlo simulation to price Asian options by stochastic volatility model with central processing unit(CPU) cluster and graphic processing unit(GPU) devices. By taking arithmetic average Asian options with stochastic volatility under discrete monitoring time as example, an efficient control variate was chosen, and the computing efficiencies between algorithm accelerating method and devices accelerating method in CPU cluster and GPU were studied respectively. The relationship between the computation results and the parameters of the model was explored. Numerical results show that an integration of the two accelerating methods can shorten the computation time a lot.

Key words: Monte Carlo method; stochastic volatility; control variate; CPU cluster computing; GPU computation

金融衍生产品的定价问题一直是金融中关注的热点问题之一.1973年美国经济学家Black等[1]提出期权的定价模型，该模型为衍生金融工具的合理定价奠定了基础.但是随着金融衍生产品模型越来越复杂，相应定价问题的封闭解也越来越难以求出，因此越来越多的问题采用数值计算的方法来解决，而蒙特卡罗方法是常用的数值方法之一，它有着通用性强、可以解决高维问题且并行化较容易实现等优点.然而蒙特卡罗方法收敛速度较慢，因此如何进一步通过并行化提高计算效率显得格外重要.

近几年高性能计算发展迅速，在生物工程、石油勘探、化学计算和核物理中有着比较广泛的应用，但在金融领域的应用相对较少.因此如何有效地将高性能计算方法与金融模型相结合是值得研究的热点问题.高性能金融计算在国外的发展较早，Zenios[2]在1999年对高性能计算应用于金融计算领域中的已有成果进行总结，并对未来的研究方向进行了展望.其他的学者如Li等[3]、Thulasiram等[4]分别将拟蒙特卡罗模拟和快速傅里叶变换(FFT)等金融计算中常用的方法移植到高性能计算机中，并用于对衍生产品的定价.从2004年开始，GPU(graphic processing unit)计算的发展尤为迅速，相应的研究成果也有很多.文献[5]研究了在GPU上执行蒙特卡罗模拟的效率问题，并和CPU上执行的结果做了比较，为应用蒙特卡罗模拟金融问题奠定了基础.其他的方法如傅里叶变换方法[6-7]也可以应用在GPU计算中.

本文所关注的是亚式期权定价问题.亚式期权是一种路径相关期权，它根据合同所规定的期限内标的资产价格的平均值来决定是否实施期权.对于确定性波动率模型下亚式期权定价问题的一些理论

收稿日期：2012-04-26
基金项目：国家自然科学基金(11171256)；上海市教委科学计算E-研究院课题(E03004)
第一作者：姜广鑫(1987—)，男，博士生，主要研究方向为计算金融. E-mail：1987_gxjiang@tongji.edu.cn
通讯作者：徐承龙(1963—)，男，教授，博士生导师，理学博士，主要研究方向为计算数学、金融数学. E-mail：clxu@tongji.edu.cn

研究可以参考文献[8-10]等. 本文标的资产模型采用文献[11]提出的随机波动率模型,它将经典的 Black-Scholes 模型中的常数波动率修正为某一扩散过程的函数,以更好反映真实的市场情况. 由于没有解析表达式,文献[12-13]对随机波动率模型进行了进一步的定性和定量的研究. 本文结合控制变量与并行计算技术将普通串行计算机中的蒙特卡罗加速模拟方法移植到 CPU(central processing unit)集群和 GPU 计算中.

1　高性能计算与蒙特卡罗模拟

1.1　高性能计算简介

在高性能计算领域,CPU 集群计算发展比较早. 国内外比较成熟的超级计算机大多采用成百上千的 CPU 列阵. 目前,MPI(message passing interface)是最重要的并行编程实现途径,它定义了一组具有可移植性的编程接口,因此程序员只需要设计好并行算法,调用 MPI 库中的相关函数,就可以实现在多个计算单元上的运算. 在本文中采用 C 语言调用 MPI 库对程序进行并行化.

GPU 计算近年来得到很大的发展. GPU 作为 CPU 的协处理器,拥有众多计算单元,单块的 GPU 卡的计算能力可以比传统的 CPU 高上千倍. GPU 较传统的 CPU 计算能力高的主要原因在于两者设计理念完全不同. CPU 优化的主要目的是提高串行代码的性能,并采用复杂的逻辑控制将指令进行串行化处理. 而 GPU 通常采取多线程来提升运算速度和吞吐量,对于硬件充分利用,减少逻辑控制线程具体内容可见文献[14]. 于是,由 GPU 负责大规模的密集型数据并行计算、CPU 负责执行不适合并行计算的部分的想法随之产生. 2007 年,NVIDIA 推出 CUDA 编程模型,以主机端调用 GPU kernel 的方式使用 GPU 进行计算. 基于 CUDA 的编程模型的计算过程可参见文献[15],它包括在主机(CPU)上执行的部分和在设备(GPU)上执行的部分.

如果建立同等计算性能的超级计算系统,与 4 核 CPU 相比,GPU 以大概 1/10 的成本和 1/20 的功耗即可实现. 但是对于 GPU 计算系统,应用软件的缺乏是其面临的重大问题;在程序设计上,很长时间内缺乏统一的标准,这导致编程的难度增大. CUDA 的出现在一定的程度上解决了这一问,使得 GPU 编程得到一定的简化. 但是随着处理的问题不断增多,对于程序的设计、算法的编写的要求越来越高;处理问题规模的不断增大也促使编程人员对 CUDA 程序进行优化,从而更好地进行计算.

1.2　蒙特卡罗模拟中的控制变量方法

控制变量方法是一种被广泛使用的方差减小技术,它充分利用已知量的估计误差来降低未知量的估计误差. 假设 $V_1,V_2,\cdots,V_i,\cdots,V_m$ 是期权到期回报贴现的 m 次独立模拟值,那么期权价格的蒙特卡罗估计值 $\bar{V}=\frac{1}{m}\sum_{i=1}^{m}V_i$.

假设得到 V_i 的同时能得到另一个随机变量 X 的样本 X_i,且期望 $E[X]$已知,样本(X_i,V_i)相互独立,对于确定的常数 b,令 $V_i(b)=V_i-b(X_i-E[X])$,则期权价格的控制变量估计值$\bar{V}(b)$即为

$$\bar{V}(b)=\frac{1}{m}\sum_{i=1}^{m}(V_i-b(X_i-E[X]))=\bar{V}-b(\bar{X}-E[X]) \tag{1}$$

所谓的"控制"是指$(\bar{X}-E[X])$,$\bar{X}$是 m 个样本 X_i 的均值. 显然控制变量估计是无偏估计量,$V(b)$的方差为

$$D(V(b))=D(V)+b^2D(X)-2b\sqrt{D(V)}\sqrt{D(X)}\rho_{XV} \tag{2}$$

式中:$D(V)$和 $D(X)$分别为 V 和 X 的方差;ρ_{XV}为 X 和 V 的相关系数. 当最优控制系数 $b^*=\frac{\mathrm{Cov}[X,V]}{D(X)}$ 时,$\mathrm{Cov}[X,V]$为 X 和 V 的协方差,上述方差达到最小值为

$$D_{\min}(V(b^*))=(1-\rho_{XV}^2)D(V). \tag{3}$$

控制变量估计的误差减小情况取决于 ρ_{XV}的大小,相关性越大,方差减小,效果越好. 要注意的是,当$|\rho_{XV}|$越接近于 1 时,方差减小的倍数增加得越快,因此选取合适的控制变量对于解决方差减小问题起着重要的作用. 比较典型的例子是文献[16],用连续几何平均亚式期权作为控制变量研究了算术平均亚式期权的定价问题. 文献[17]研究了在未定权益的模拟定价中使用对冲工具作为控制变量来减小模拟的方差.

2　随机波动率下亚式期权的控制变量蒙特卡罗模拟方法

模型建立在风险中性测度下,标的资产 S_t,1($t\geqslant 0$,t 为时间)和波动率 σ_t($t\geqslant 0$)适合如下随机微分方程:

$$\frac{dS_{t,1}}{S_{t,1}} = rdt + \sigma_t dW_{1t}, \quad \sigma_t = \sqrt{Y_t} \tag{4}$$

$$\frac{dY_t}{Y_t} = \mu dt + \sigma_Y dW_{2t} \tag{5}$$

式中:r 为无风险利率;Y_t 为波动率随时间的变化;μ 为波动率的增长率;σ_Y 为波动率的波动率;W_{1t},W_{2t} 为标准布朗(Brown)运动,且 $\mathrm{Cov}(dW_{1t}, dW_{2t}) = \rho dt$. 该标的资产的离散取样算术平均亚式期权的收益函数 V_1 为

$$V_1 := V_1(\{S_{1,t}: t \in [0,T]\}) = \max\left\{\left(\frac{1}{N}\sum_{i=1}^{N} S_{t_i,1} - K\right), 0\right\} \tag{6}$$

式中:T 为期权的到期时间;K 为期权的敲定价;t_i $(i=1,2\cdots N)$是观测时间. 通常取等距的观测时间,即 $t_{i+1}-t_i=\Delta t=T/N$. 目的是通过蒙特卡罗方法求出 V_1 在风险中性条件下的贴现期望值 $C_1 := e^{-rT} \cdot E[V_1]$.

为了提高计算效率,本文采用控制变量技巧来进行加速. 由于控制变量与目标变量要有很高的相关性并且可以快速求出控制变量的期望,这里选取波动率为常数的离散取样几何平均亚式期权作为控制变量. 该控制变量的标的资产 $\{S_{t,2}, t \geqslant 0\}$ 满足下列随机微分方程:

$$\frac{dS_{t,2}}{S_{t,2}} = rdt + \sigma_c dW_{1t}. \tag{7}$$

式中,σ_c 为控制变量标的资产的波动率,根据文献[18]的结论,当 σ_c 满足 $\sigma_c^2 = Y_0(e^{\mu T}-1)(\mu T)^{-1}$(当 $\mu=0$时,$\sigma_c^2=Y_0$)时,控制变量的相关性最大,它相应的收益函数

$$V_2 := V_2(\{S_{t,2}: t \in [0,T]\}) = \max\left\{\left(\left(\prod_{i=1}^{N} S_{t_i,2}\right)^{\frac{1}{N}} - K\right), 0\right\} \tag{8}$$

在风险中性条件下的期望 $E[V_2]$ 可以由 Black-Scholes 公式得出.

根据式(1)可以构造控制变量的估计值 V

$$V = V_1 - b^* (V_2 - E[V_2]) \tag{9}$$

最优的系数 $b^* = \frac{\mathrm{cov}[V_1, V_2]}{D[V_1]}$,可以采用数值模拟得到近似值. 带有控制变量的蒙特卡罗模拟的算法步骤如下:参数 m 为模拟次数,波动率的初值 Y_0,$E=E[V_2]$,最优系数 b^*. 目标是计算 V 的期望. ①等距离散时间点列 $\{t_i\}_{i=1}^{n}$,包含所需的观测点 $\{T_i\}_{i=1}^{N}$ 在内,$T_0=t_0=0$,设置初始值 $S(t_0)=S_0$,$Y(t_0)=Y_0$. ②开始进行 m 次循环. ③采用欧拉离散格式,基于式(4)和式(5)生成随机波动率 σ_{t_i} 和 $S_{t_i,1}$,$i=1,2\cdots N$. ④令 $C_{j,1} = e^{-rT}\max\left\{\left(\frac{1}{N}\sum_{i=1}^{N} S_{t_i,1} - K\right), 0\right\}$. ⑤基于式(7)生成标的资产 $S_{t_i,2}$,$i=1,2\cdots N$. ⑥令 $C_{j,2} = e^{-rT}\max\left\{\left(\left(\prod_{i=1}^{N} S_{t_i,2}\right)^{\frac{1}{N}} - K\right), 0\right\}$;⑦构造控制变量的估计值 $C_j = C_{j,1} - b^*(C_{j,2} - E)$. ⑧结束循环. ⑨返回计算出的平均值 $C = \frac{1}{m}\sum_{j=1}^{m} C_j$.

3 高性能计算环境中的蒙特卡罗加速计算模拟结果

3.1 CPU 集群中计算结果

CPU 集群的计算环境:上海超级计算中心的魔方超级计算机,该平台采用了 QUAD-Core AMD™-8347/Barcelona 处理器. 使用的 MPI 版本为 mvapich-1.0. 随机数生成器采用的是 MT19937[19]. 在 MPI 消息传递中,为了比较计算效果一共有 5 个变量需要传递,具体如表 1.

表 1 传递的参数

Tab.1 Message passing parameters

变量	说明
MPI_Send(&price, 1, MPI_DOUBLE, 0, tag1, MPI_COMM_WORLD)	price 是目标期权价格
MPI_Send(&price2, 1, MPI_DOUBLE, 0, tag2, MPI_COMM_WORLD)	price2 是目标期权二阶矩
MPI_Send(&priceCon, 1, MPI_DOUBLE, 0, tag3, MPI_COMM_WORLD)	priCone 是控制变量价格
MPI_Send(&priceCon2, 1, MPI_DOUBLE, 0, tag4, MPI_COMM_WORLD)	priceCon2 是控制变量二阶矩
MPI_Send(&covsum, 1, MPI_DOUBLE, 0, tag5, MPI_COMM_WORLD)	covcon 是协方差中间量

数据参数如下:$S_0=20$,$K=19$,$T=1.0$,$r=0.02$,$N=12$,每 2 个观测点之间取离散点数 $k=5$,$\mu=0$,$\sigma_Y=0.01$,$Y_0=0.01$,$\rho=0.5$. 这里误差减小倍数 R 定义为 $R=E_1/E_0$,其中 E_0 为控制变量的蒙特卡罗模拟的误差,E_1 为普通蒙特卡罗模拟的误差,$E_0=\sqrt{\frac{D(V)}{m-1}}$,$E_1=\sqrt{\frac{D(V_1)}{m-1}}$. 首先研究运算的核心数目对计算结果的影响,改变运算的核心数目 C 从 1 到 256. 表 2 分别列出普通的蒙特卡罗方法和带控制变量蒙特卡罗方法模拟次数为 2^{26} 次的数值结果.

从表 2 中可以看出,运算的核心数越多,计算的时间越短,而核心数的变化对运算结果值影响不大,因此在以后的计算中采用 16 个核来进行 CPU 集群计算. 单核与 16 核的运算时间见图 1,可见单核的运

表 2　并行计算的核心数与计算结果的关系

Tab.2　Relationship between numbers of CPUs and computation results

C	运算时间/s	V	E_0	V_1	E_1	R
1	2 787.82	1.291 03	0.000 001 643 7	1.290 93	0.000 131 6	80.059 87
2	1 396.88	1.291 03	0.000 001 643 4	1.290 75	0.000 131 6	80.064 11
4	699.87	1.291 03	0.000 001 649 0	1.290 77	0.000 131 5	79.767 45
8	352.56	1.291 03	0.000 001 646 8	1.290 65	0.000 131 5	79.859 67
16	177.47	1.291 05	0.000 001 647 4	1.290 81	0.000 131 5	79.838 97
32	88.27	1.291 04	0.000 001 646 9	1.290 37	0.000 131 5	79.847 65
64	44.43	1.291 03	0.000 001 643 2	1.289 91	0.000 131 4	79.974 62
128	22.17	1.291 00	0.000 001 637 6	1.287 83	0.000 131 3	80.165 66
256	11.17	1.291 01	0.000 001 634 0	1.287 01	0.000 131 4	80.435 27

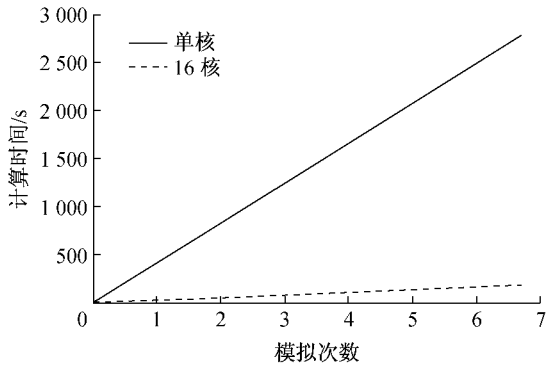


图 1　单核计算与 16 核并行计算的时间与模拟次数的关系

Fig. 1　Relationship between computing time and simulation times with 1 CPU and 16 CPU

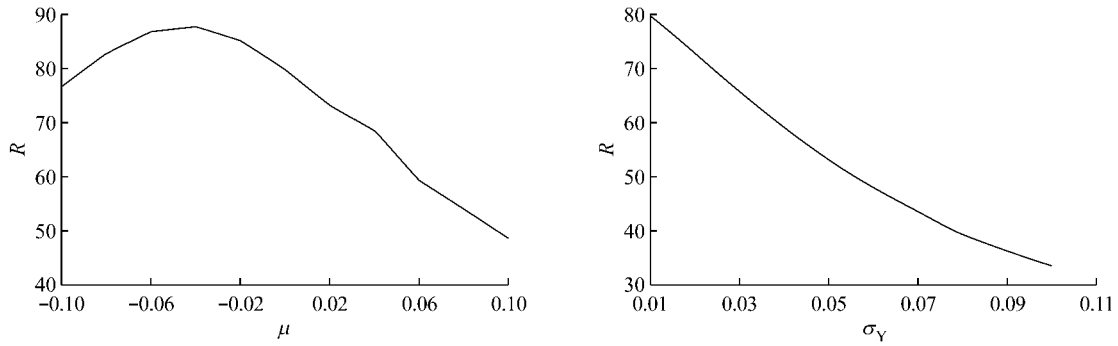
算时间远远高于 16 核,16 核与单核的加速比约为 16.观察不同模拟次数 m 的计算结果,如表 3.

从表 3 中可以看出,R 在 80 左右.在普通的蒙特卡罗法中,当模拟次数达到 2^{24} 次左右才开始精确到小数点后第 3 位.用普通的蒙特卡罗方法对该模型模拟 2^{30} 次得到结果 1.290 93 看作是精确解,应用控制变量进行加速后模拟次数在 2^{12} 次左右精度就达到小数点后第 3 位.接下来研究参数的改变对 R 的影响.分别改变 μ 和 σ_Y.图 2 改变 μ 从 −0.10 到 0.10,其他参数与上文相同,模拟次数采用 2^{20} 次,得到 16 核并行计算中 R 与 μ 的关系.改变 σ_Y 从 0.01 到 0.10,其他参数与上文相同,模拟次数采用 2^{20} 次,

表 3　16 核并行计算中模拟次数与计算结果的关系

Tab.3　Relationship between simulation times and computation results in 16 CPUs

m	V	E_0	V_1	E_1	R
2^{12}	1.290 16	0.000 171 578 1	1.298 75	0.017 488 618 4	81.928 03
2^{14}	1.290 29	0.000 103 950 5	1.299 70	0.008 301 557 8	79.860 66
2^{16}	1.290 73	0.000 054 223 7	1.287 61	0.004 167 206 8	76.852 08
2^{18}	1.290 92	0.000 026 296 6	1.280 18	0.002 097 049 8	79.746 17
2^{20}	1.291 06	0.000 013 223 8	1.292 40	0.001 054 801 0	79.765 42
2^{22}	1.291 01	0.000 006 536 1	1.287 01	0.000 525 735 6	80.435 27
2^{24}	1.291 03	0.000 003 286 3	1.289 91	0.000 262 823 1	79.974 62
2^{26}	1.291 05	0.000 001 647 4	1.290 81	0.000 131 529 1	79.838 97
2^{28}	1.291 04	0.000 000 822 5	1.290 72	0.000 065 789 1	79.986 81
2^{30}	1.291 03	0.000 000 410 9	1.290 93	0.000 032 897 9	80.059 87



a R 与 μ 的关系　　b R 与 σ_Y 的关系

图 2　16 核并行计算中 R 与 μ,σ_Y 关系

Fig.2　Relationship between μ,σ_Y and R in 16 CPUs

得到16核并行计算中 R 与 σ_Y 的关系.

综上所述,在并行计算环境中应用蒙特卡罗方法对期权进行定价同样可以采用控制变量的技巧来减小误差,从而大幅节省计算时间.

3.2 GPU计算结果

GPU计算环境:NVIDIA Tesla™ S2050计算系统,1U机箱内配备了4颗基于Fermi架构的Tesla计算处理器,每颗GPU最高可实现515 Gigaflop双精度峰值性能.计算采用1颗Tesla处理器,CUDA toolkit版本为4.0RC2.在GPU上采用蒙特卡罗方法计算时很重要的一个问题是随机数的生成,由于计算量巨大,随机数的相关性可能影响计算结果的可靠性.这里随机数采用的是CUDA Library中的函数.

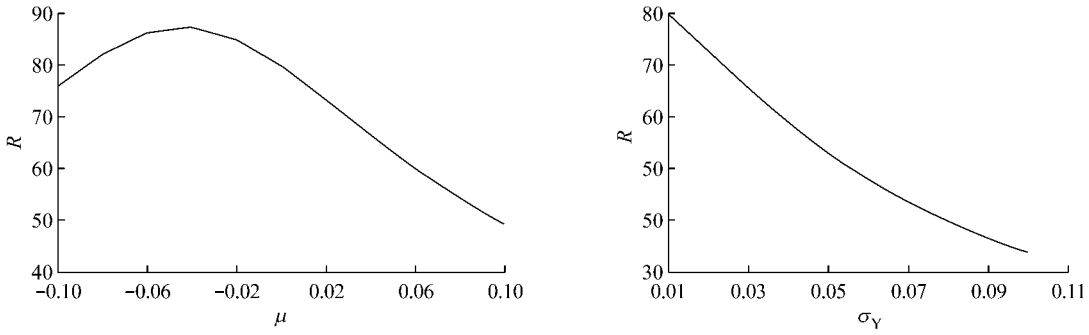
改变模拟次数m,讨论其对计算结果的影响,见表4.

表4 GPU计算中模拟次数与计算结果的关系

Tab.4 Relationship between simulation times and computation results in GPU

m	V	E_0	V_1	E_1	R
2^{12}	1.290 84	0.000 163 312 7	1.305 42	0.013 812 387 8	82.576 30
2^{14}	1.291 04	0.000 098 771 9	1.288 65	0.007 931 041 8	80.296 53
2^{16}	1.291 06	0.000 051 564 3	1.292 53	0.004 156 417 0	80.606 50
2^{18}	1.291 04	0.000 026 205 1	1.292 52	0.002 096 669 6	80.009 88
2^{20}	1.291 04	0.000 013 163 9	1.289 96	0.001 051 314 6	79.863 70
2^{22}	1.291 04	0.000 006 588 7	1.291 59	0.000 526 194 6	79.863 60
2^{24}	1.291 03	0.000 003 291 3	1.290 96	0.000 263 096 3	79.937 97
2^{26}	1.291 03	0.000 001 645 7	1.290 97	0.000 131 567 2	79.944 65
2^{28}	1.291 03	0.000 000 822 6	1.290 99	0.000 065 797 1	79.982 09
2^{30}	1.291 03	0.000 000 411 3	1.291 02	0.000 032 898 3	79.985 12

从表4可见GPU下的计算值与用16个CPU集群计算值几乎相同,随机波动率的亚式期权的精确解为1.291 03,R 为80左右.同样在普通的蒙特卡罗法中,当模拟次数达到 2^{24} 次左右才开始精确到小数点后第3位,应用控制变量进行加速后,模拟次数在 2^{12} 次左右精度就达到小数点后第3位.在图3a中 μ 从−0.10到0.10,在图3b中 σ_Y 从0.01到0.10,其他的参数不变,模拟次数采用 2^{20} 次,得到GPU计算中 R 与 μ 和 σ_Y 的关系与3.1节中相同.



a R 与 μ 的关系　　b R 与 σ_Y 的关系

图3 GPU计算中 R 与 μ,σ_Y 关系

Fig.3 Relationship between μ,σ_Y and R in GPU

研究不同观测点数 N 对计算结果的影响,其他参数保持不变,模拟次数为 2^{20} 次,结果如表5.

从表5可以看出,随着观测点的数目增加,期权的价格逐渐减小,但是 R 没有变化.

3.3 CPU集群计算与GPU计算时间的比较

计算环境和参数选取不变,改变 m,得到CPU集群与GPU计算的运算时间如图4所示.

从图4可见用GPU计算的时间远低于16核的CPU集群计算.此外,控制变量的蒙特卡罗方法的运算时间只比普通的蒙特卡罗方法运算时间多一些,但是采用控制变量的蒙特卡罗方法达到相同精确值的模拟次数远低于普通蒙特卡罗方法.

当达到一定的误差值时,比较2种不同方法所需时间.假定误差值为0.000 05.单核的普通蒙特卡罗方法要进行大约 2^{28} 次以上的运算,时间约为10 753.99 s,单核的控制变量方法蒙特卡罗方法要

表 5　GPU 计算中观测点数与计算结果的关系

Tab.5　Relationship between observation points and computation results in GPU

m	V	E_0	V_1	E_1	R
12	1.291 04	0.000 013 163 9	1.289 96	0.001 051 315	79.863 70
24	1.275 71	0.000 012 863 5	1.275 02	0.001 025 462	79.718 72
36	1.270 62	0.000 012 757 5	1.269 80	0.001 016 344	79.666 47
48	1.268 07	0.000 012 633 5	1.268 10	0.001 012 853	80.172 28
60	1.266 57	0.000 012 645 0	1.266 22	0.001 010 277	79.895 53
72	1.265 54	0.000 012 629 4	1.265 69	0.001 007 621	79.783 69
84	1.264 82	0.000 012 599 5	1.264 34	0.001 006 047	79.848 24
96	1.264 28	0.000 012 608 9	1.264 50	0.001 005 038	79.708 84
108	1.263 86	0.000 012 602 6	1.263 26	0.001 003 552	79.630 71
120	1.263 52	0.000 012 578 9	1.262 55	0.001 002 578	79.703 19

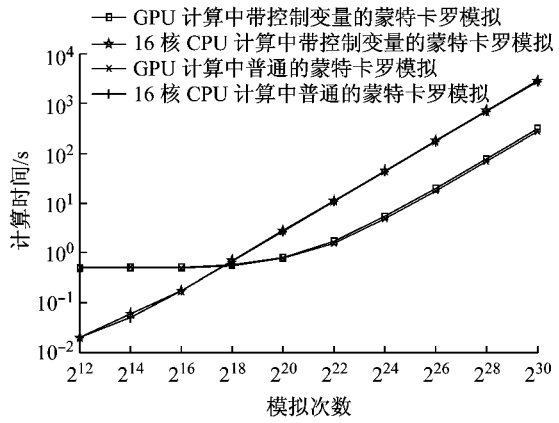


图 4　计算时间与模拟次数的关系

Fig.4　Computation time against simulation times

进行大约 2^{16} 次运算,时间约为 2.75 s;在应用 16 核 CPU 进行并行计算时,进行的运算次数与单核一样,普通的蒙特卡罗方法要进行大约 689.28 s,控制变量的蒙特卡罗方法要进行大约 0.17 s;在应用 GPU 进行计算时,普通的蒙特卡罗方法要进行大约 69.04 s,控制变量的蒙特卡罗方法要进行大约 0.51 s.为达到相同的误差值,在硬件加速方面,16 核 CPU 集群计算与 GPU 计算的速度要远快于单核的 CPU,加速比分别为 15.6 与 155.8;而在算法加速方面,控制变量的蒙特卡罗方法对于计算时间的加速更为明显,效果好于硬件加速的效果.但值得注意的是在 GPU 上进行蒙特卡罗模拟时,应用的是 CUDA Library 中的函数,初始化阶段需要一定的时间,因此在进行次数较低的运算时,运算的时间比 16 核 CPU 集群慢.但是进行大运算量计算时,GPU 上的控制变量蒙特卡罗方法比 16 核 CPU 快很多.

4　结论

将控制变量的蒙特卡罗方法应用在 16 核 CPU 集群计算和 GPU 计算中,选取了波动率为常数的离散取样几何平均亚式期权作为控制变量来计算随机波动率下的离散取样算术平均亚式期权.选取合适的参数,计算的 R 均达到 80 左右.进而讨论参数对计算结果的影响,R 随着波动率的漂移项 μ 及波动率 σ_Y 的增大而减小.其他参数,例如观测点数的改变对 R 几乎无影响.

在计算时间方面,算法加速比硬件加速更节省时间.相同的误差精度,采用普通的蒙特卡罗方法,16 核 CPU 集群计算与 GPU 计算对单核 CPU 计算的加速比分别为 15.6 与 155.8.而在单核 CPU 计算中,采用了控制变量的蒙特卡罗方法与普通的蒙特卡罗方法的加速比可达到 4 000 以上.在 16 核 CPU 集群与 GPU 计算的比较中,GPU 在运算量较大时,比 16 核 CPU 计算得更快,但是由于 GPU 在初始化随机数时需要一定的时间,因此在计算量较小时计算的效率没有 16 核 CPU 高.综上所述,控制变量技巧可以应用于 CPU 集群计算和 GPU 计算环境中,采用算法与硬件加速相结合的方法,可以极大缩短运算时间.控制变量的选取一个很重要的原则是控制变量的期望要容易得到.本文选取了波动率为常数的几何平均亚式期权,它的期望可以由 Black-Scholes 公式很快求出,因此对于计算时间的影响很小.此外,还可以选取波动率为确定性函数的几何平均亚式期权、波动率为确定性函数的算术平均亚式期权和波动率为常数的算术平均亚式期权作为控制变量.虽然对于方差减小的效果会更好,但是其期望很难得出,而且求解期望的方法不适合并行化.

参考文献:

[1] Black F, Scholes M. The pricing of options and corporate liabilities[J]. Journal of Political Economy, 1973, 81(3):637.

[2] Zenios A S. High-performance computing in finance: the last 10 years and the next[J]. Parallel Computing, 1999, 25(13): 2149.

[3] Li X J, Mullen L G. Parallel computing of a quasi-Monte Carlo algorithm for valuing derivatives[J]. Parallel Computing, 2000, 26(5):641.
[4] Thulasiram R K, Thulasiraman P. Performance evaluation of a multithreaded fast Fourier transform algorithm for derivative pricing[J]. Journal of Supercomputing, 2003, 26(1): 43.
[5] Tomov S, McGuigan M, Bennett R, et al. Benchmarking and implementation of probability-based simulations on programmable graphics cards[J]. Computers & Graphics, 2005 29(1):71.
[6] Vladimir S. Parallel option pricing with Fourier space time-stepping method on graphics processing units[J]. Parallel Computing, 2008, 36(7):372.
[7] Zhang B W, Oosterlee C W. Option pricing with COS method on graphics processing units[C]//23rd IEEE International Parallel and Distributed Processing Symposium. Rome:[s.n.], 2009:1-8.
[8] Kemma A G Z, Vorst A C F. Bessel processes, Asian options and perpetuities[J]. Mathematical Finance, 1993, 3(4):349.
[9] Carverhill A P, Clewlow L J. Flexible convolution[J]. Risk, 1990, 3(4):25.
[10] Rogers L C G, Shi Z. The value of an Asian option[J]. Journal of Applied Probability, 1995, 32(4): 1077.
[11] Hull J, White A. The pricing of options on assets with stochastic volatilities[J]. Journal of Finance, 1987, 42(2):281.
[12] Scott L. Option pricing when the variance changes randomly: theory, estimation and an application[J]. Journal of Financial and Quantitative Analysis, 1987, 22(4): 419.
[13] Heston, Steven L. A closed-form solution for options with stochastic volatility, with applications to bond and currency options[J]. Review of Financial Studies, 1993, 6(2):327.
[14] Kirk D B, Hwu W W. Programming massively parallel processors: a hands-on approach[M]. Burlington: Morgan Kaufmann, 2010.
[15] Tian Yu, Zhu Zili, Klebaner C F, et al. Option pricing with the SABR model on the GPU[C]//High Performance Computational Finance (WHPCF), 2010 IEEE Workshop. Los Angeles: IEEE, 2010:1-8.
[16] Kemna A G Z, Vorst A C F, A pricing method for options based on average asset values[J]. Journal of Banking and Finance, 1990, 14(1):113.
[17] Les Clewlow, Carverhill A. On the simulation of contingent claims[J]. Journal of Derivatives, 1994, 2(2):66.
[18] Ma J, Xu C. An efficient control variate method for pricing variance derivatives[J]. Journal of Computational and Applied Mathematics, 2010, 235: 108.
[19] Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator[J]. ACM Transactions on Modeling and Computer Simulation, 1998, 8(1): 3.

(上接第 791 页)

以下求解一个球约束下非凸又非凹的的全局优化问题.

例 3 考虑以下球约束下关于 u_1, u_2 的二维非凸全局优化问题:(Q^*):$min\ Q(\boldsymbol{u})=\frac{1}{2}u_1+\frac{1}{2}u_2+2u_1u_2+u_1-u_2$, s.t. $u_1^2+u_2^2\leqslant 1$.

解 $\boldsymbol{A}=\nabla^2 Q=\begin{pmatrix}1 & 2\\ 2 & 1\end{pmatrix}$, $\boldsymbol{f}=\begin{pmatrix}-1\\ 1\end{pmatrix}$, $\boldsymbol{A}$ 的特征值为 $\lambda_1=3, \lambda_2=-1$, 可见 $Q(\boldsymbol{u})$ 为非凸又非凹的函数. 选取 $\alpha^*=\frac{-\lambda_2}{1-\lambda_2}=\frac{-(-1)}{1-(-1)}=\frac{1}{2}$. 易见 $\mathrm{rank}(\boldsymbol{A}+\boldsymbol{I})=1\neq 2=\mathrm{rank}(\boldsymbol{A}+\boldsymbol{I}, \boldsymbol{f})$. 应用定理 3,需要求取 $\rho^*>-\lambda_1=1$,使得 $\boldsymbol{f}^{\mathrm{T}}(\boldsymbol{A}+\rho^*\boldsymbol{I})^{-2}\boldsymbol{f}=1$. 以下对 $\boldsymbol{A}$ 进行正交分解. 类似例 1 利用正交变换 $\boldsymbol{Q}=\begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{pmatrix}$, $\boldsymbol{A}=\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{Q}=\boldsymbol{Q}^{\mathrm{T}}\begin{pmatrix}3 & 0\\ 0 & -1\end{pmatrix}\boldsymbol{Q}$. 由于 $\hat{\boldsymbol{f}}^{\mathrm{T}}=\boldsymbol{f}^{\mathrm{T}}\boldsymbol{Q}^{\mathrm{T}}=(-1,1)\begin{pmatrix}\frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{pmatrix}=(0,\sqrt{2})$, 由 $\boldsymbol{f}^{\mathrm{T}}\cdot(\boldsymbol{A}+\rho^*\boldsymbol{I})^{-2}\cdot\boldsymbol{f}=\hat{\boldsymbol{f}}^{\mathrm{T}}(\boldsymbol{\Lambda}+\rho\boldsymbol{I})^{-2}\hat{\boldsymbol{f}}=\frac{2}{(\rho-1)^2}=1$ 得到 2 个参数解 $\rho=1\pm\sqrt{2}$,由于 $\rho=1+\sqrt{2}>-\lambda_1=1$, 应取

$$\rho^*=1+\sqrt{2} \tag{21}$$

于是得到全局优化问题(Q^*)的最优点 $\boldsymbol{u}^*=(\boldsymbol{A}+\rho^*\boldsymbol{I})^{-1}\boldsymbol{f}=\begin{pmatrix}\sqrt{2}+2 & 2\\ 2 & \sqrt{2}+2\end{pmatrix}^{-1}\begin{pmatrix}-1\\ 1\end{pmatrix}=\begin{pmatrix}\frac{-\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2}\end{pmatrix}$.

参考文献:

[1] Gurman V I, Ukhin M Yu. The extension principle in control problems-constructive methods and applied problems[M]. Moscow: Fizmatlit, 2005.
[2] ZHU Jinghao, WANG Chao, GAO David. Global optimization over a box via canonical dual function[J]. Journal of Computational and Applied Mathematics, 2011, 235(5):1141.
[3] 朱经浩. 最优控制中的数学方法[M]. 北京: 科学出版社, 2011.
ZHU Jinghao. Mathematics in optimal control[M]. Beijing: Science Press, 2011.
[4] Krotov V F. Global methods in optimal control[M]. New York: Marcel Dekker, 1996.
[5] Pardalos P M. Global optimization algorithms for linearly constrained indefinite quadratic problems[J]. Computers & Mathematics with Applications, 1991, 21(6/7):87.
[6] Floudas C A, Visweswaran V. Quadratic optimization[C]// Handbook of Global Optimization. Dordrecht/Boston/London: Kluwer Academic Publishers, 1995:217-270.