章节 1

背景综述

高性能计算(High Performance Computing)一直是计算机科研领域的重要主题。伴随近20年来超级计算集群 性能和数量的指数式增长，我们的超算能力已经迈进Petaflop时代（10^{15}次浮点运算), 更强大的 exaflop时代(10^{18}次浮点运算)也即将到来。但是目前能够有效充分利用超算集群的应用程序还为数 不多，因此超算应用是制约HPC界发展的一大瓶颈。

在超算的应用领域，科研活动占据了一大部分，如天文学，粒子物理，计算化学等基础学科；另外在工程和 商业领域也有重要的应用，如汽车，飞机设计中得计算流体力学模拟仿真，油田勘探中的海量信息处理和开采 方案设计以及在商业投资领域的高频交易等。此外随着云计算和大数据相关研究和商业应用的快速崛起， 怎样将高性能计算和数据挖掘分析等技术结合起来也成为了当前超算领域的研究热点。本文的主题是探讨 高性能计算在金融领域的一个重要应用以及其在Intel最新的MIC架构上的实现。首先在中我们将简单介绍 高性能计算和金融领域超算应用的一些基本情况。

1.1 高性能计算的诞生和发展

高性能计算的概念和实现依赖于算法理论和硬件制造的发展。在算法理论方面，并行运算(Parallel computing) 的研究在70年代就开始，但是受制于当时硬件领域的发展瓶颈，并行运算方面的很多理论没有得到充分地发展。 如图1.1 所示，Intel的处理器的在2000年之前晶体管的数量和时钟频率都能依照摩尔定律增长。 但是当单个处理器的制造工艺到达纳米级别后，受制于量子效应已经无法进一步提高单个处理器的时钟频率。 随后业界开始转而使用多核系统(Multi-core system)，直至目前出现的众核系统(Manycore system)来提升超算的性能。目前最权威的HPC超算 集群排行榜Top500给出的结果显示，

1.2 金融领域的超算应用

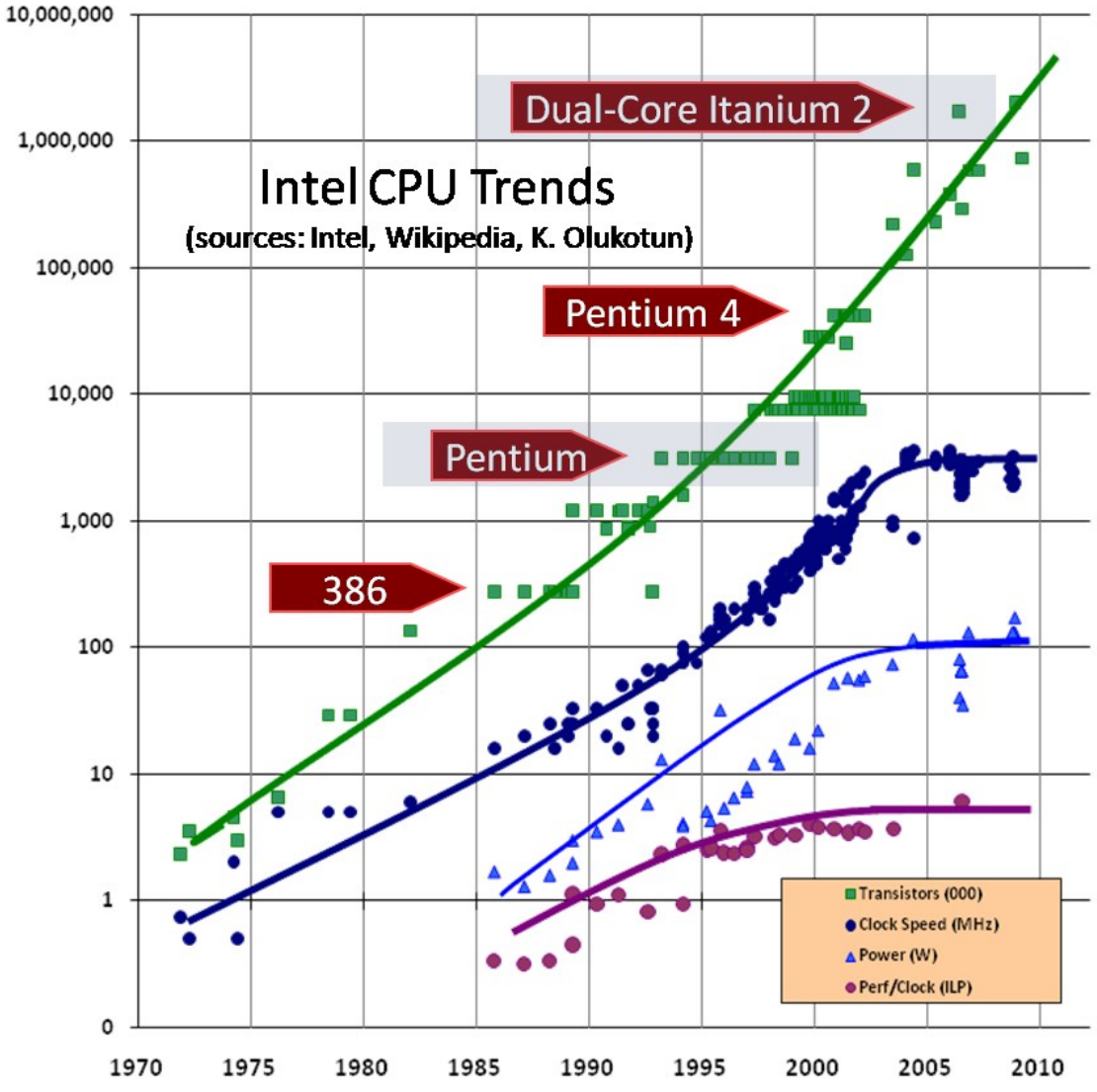

Figure 1.1: Intel处理器的发展趋势

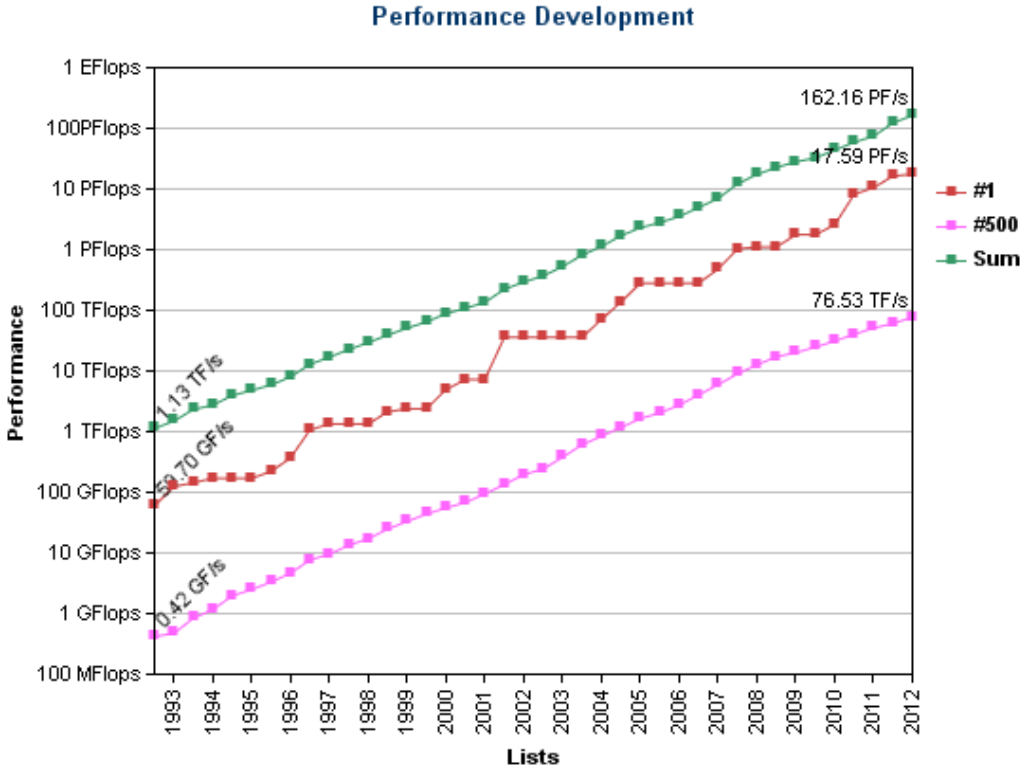

Figure 1.2: Top500超算集群的性能发展

章节 2

课题陈述

本文所要研究的金融问题是基于欧洲式期权的离散时间对冲策略的风险控制。欧式期权(European Option) 是期权的一种类型。另外两种期权类型是美式期权 (American Option)和亚洲期权(Asian Option)。期权本质上是一种金融合约，它授予期权的拥有者（买方）一种在某一特定时间点或之前，以某一个在合约中给出的价格 (strike price) 购买或者出售某一种金融产品(underlying asset) 的权利。买家在执行自己的权利时，卖家必须按照合约规定的价格进行交易， 但买家同时需要付给买家一定的额外交易手续费(premium)。期权的研究主要集中在对其价值的评估。期权的价值一般可以分为两个部分。一是期权的内部价值 (intrinsic value)，主要由该期权基于的金融产品在交易时的市场价格和合约定价的差值决定。二十期权的时间价值(Time Value)，这一部分取决于交易前的其他因素如 利息，增值等。期权的研究从19世纪就开始了，但目前的研究一般都是基于1973年提出的Black-Scholes模型。我们的课题也是基于Black-Scholes模型，并对欧式期权 对冲策略的风险控制进行研究。

2.1 欧式期权

欧式期权的特点在于买方必须在合约规定的时间到期时才能行使权力。通常将欧式期权的合理价格价定为其期望价值。 相比与美式期权，二者在执行时间上有所不同。 美式期权合同在到期日前的任何时候或在到期日都可以执行合同，结算日则是在履约日之后的一天或两天， 大多数的美式期权合同允许持有者在交易日到履约日之间随时履约，但也有一些合同规定一段比较短的时间可以履约，如“到期日前两周”。 欧式期权合同要求其持有者只能在到期日履行合同，结算日是履约后的一天或两天。目前国内的外汇期权交易都是采用的欧式期权合同方式。 通过比较，结论是：欧式期权本少利大，但在获利的时间上不具灵活性；美式期权虽然灵活，但付费十分昂贵。因此，国际上大部分的期权交易都是欧式期权。 中国国内银行设立的外汇期权业务都是采用的欧式期权交易。

2.2 Black-Scholes模型

Black-Scholes 是一个70年代由Fischer Black和Myron Scholes 提出的用于金融衍生品交易的数学模型。这个模型可以推出Black-Scholes公式， 并用于预测欧式期权定价的理论值。70年代至今在众多的实际交易中，该公式的预测估值被认为是非常接近实际观测到的价值。 这个模型的成功也直接导致了期权交易市场的繁荣，并能使芝加哥期权交易市场等大的期权交易场所得到科学的有效的监管。 Black-Scholes 公式是一个偏微分方程，用于预测期权价格随时间的变化。而该模型背后的策略则是通过合理地买卖期权来对冲风险，这种对冲策略被 成为"Delta Hedging"，这也是许多更复杂的对冲机构策略的基础。此后Robert C. Merton 进一步发展了该模型，并和Schole分享了1997年的诺贝尔经济学奖。

2.3 课题表述

本课题研究的对象就是欧式期权利用对冲策略来降低交易的风险。 我们考虑基于Black-Scholes模型的Delta 对冲策略，该策略为一个连续时间策略，也即策略所要求的交易行为发生在每时每刻，关于时间轴连续。 我们知道现实生活中，交易只能发生在一些离散的时间点上, 即交易员只能在一系列相继的时刻点上交易。 一个总所周知的事实是，实际中想要完美的复制实现一个给定的连续时间策略是不可能的。 本课题研究的就是离散时间版本的Delta 对冲策略与理论上的Delta 对冲策略之间的误差及风险控制。

让我们简单的回忆下一般化的 Black-Scholes 模型。 假设在时刻 t, 期权所基于的风险资产价格为 X_t， 另外有一无风险资产价格为 S_t^0。 X_t 满足如下的一维随机偏微分方程:

$$dX_t = \mu(X_t)X_t dt + \sigma(X_t)X_t dB_t, \tag{2.1}$$

$X_0 = x_0 > 0, \sigma(x) > 0, \forall x > 0$, 而无风险资产价格满足一个常微分方程:

$$dS_t^0 = rS_t^0 dt \tag{2.2}$$

。 我们引进一个新的随机过程W_t:

$$W_t = B_t + \int_0^t \frac{\mu(X_s) - r}{\sigma(X_s)} ds \tag{2.3}$$

， 在中性风险的概率空间下，这个随机过程是一个标准的布朗运动。 我们在此中性风险概率空间下考虑问题。 现在风险资产所满足的随机微分方程变为如下形式:

$$dX_t = rX_t dt + \sigma(X_t)X_t dW_t \tag{2.4}$$

若欧洲期权的回报函数为 $f \in L^2(X_t)$, 则该期权在0 时刻的价格由下面公式给出:

$$h(f)(x_0) = \mathbb{E}(exp(-rT)f(X_T)|X_0 = x_0). \tag{2.5}$$

令

$$u(t,x) = \mathbb{E}(e^{-r(T-t)}f(X_{T-t}^x)), \tag{2.6}$$

则u 是如下方程的解:

$$-\frac{\partial}{\partial t}u(t,x) = \frac{1}{2}\sigma^2(x)x^2\frac{\partial^2}{\partial x^2}u(t,x) + rx\frac{\partial}{\partial x}u(t,x) - ru(t,x) \tag{2.7}$$

这里 $(t,x) \in [0,T) \times (0,\infty)$ 且 $u(0,x) = h(f), \quad u(T,x) = f(x)$。

一个广为人知的期权定价公式如下:

$$e^{-rT}f(X_T) = h(f) + \int_0^T \xi_t d\widetilde{X}_t \tag{2.8}$$

这里 $\widetilde{X}_t = e^{-rT}X_t$ 是风险资产的折旧贴现价格。 Ito's 公式给出了Delta 对冲策略ξ:

$$\xi_t = \frac{\partial u}{\partial x}(t, X_t) \tag{2.9}$$

换句话说，为了达到完美对冲，交易员需要在 $t \in [0,T]$ 的时间内无时不刻的交易， 并且使自己在时刻 t 持有恰好 ξ_t 单位的风险资产。 这在现实中是不可能达到的。

一个替代的解决方案是用在离散的时刻点交易的方法来逼近原先的对冲策略。假设交易员在$t \in [0,T)$ 时间内交易N 次，交易时间被定义为$t_k = kT/N$, $(k \in \{0,...,N\})$, 在 $t \in [t_k, t_{k+1})$时间内， 交易员使自己持有ξ_{t_k}单位的风险资产X_t。在此离散化策略的逼近下，在期权到期时刻T, 完美对冲和离散化对冲的差值为:

$$\begin{aligned} M_T^N &= e^{-rT}f(X_T) - (u(0,x) + \int_0^T \frac{\partial u}{\partial x}(\varphi(t), X_{\varphi_t}))d\widetilde{X}_t \\ &= \int_0^T \frac{\partial u}{\partial x}(t, X_t)d\widetilde{X}_t - \int_0^T \frac{partialu}{\partial x}(\varphi(t), X_{\varphi(t)})d\widetilde{X}_t \end{aligned} \tag{2.10}$$

这里 $\varphi(t) = sup\{t_i|t_i \leq t\}$.

由于实际交易中每一个时间点上T_i的交易都要支付一笔手续费，所以在一定的交易风险范围内，如果能找到一个最小的N，就可以最优化交易成本。 假设在$t = T$时，交易的误差可表示为$|P(X) - P_{T_N}(X)|$, 而可接受的最大误差为ϵ，我们将采用蒙特卡洛模拟算法，通过M次模拟，计算出 $|P(X) - P_{T_N}(X)| < \epsilon$的概率$Prob(M, N)$, 当$Prob(M, N)$的值大于一个预先设定的概率值时，我们认为$N$是一个可接受的抽样次数。 算法 1 给出了我们计算验证一个N值的串行算法伪代码。

在获得一个可接受的N值后，我们可以继续寻找满足条件的更小的N值，一般初始的N值取的较大，所以这种最优化的寻找可以通过一个二分法的搜寻算法来实现。 在算法 2 中我们设置的收敛条件为$|nU - nL| \leq \lambda$ 这是因为蒙特卡洛模拟本身具有一定的随机性。在N的收敛过程中会出现波动，导致无法 收敛到一个单一的值上，在$BSERROR$中给定了置信概率$Prob$的情况下，我们很自然地就能利用置信区间这一概念来表述我们取到的最优值 N。在实际应用中， N的初始值通常需要取得很大，然后通过二分法来逐步缩小范围取得最优值。此外蒙特卡洛模拟得次数 M也需要取一个很大的值来保证其精确性。如果N和M的取值 都较大，就会极大地增加计算时间和内存空间需求。这时串行程序在普通的计算机上就难以满足问题的需求，我们将需要转而开发并行程序并使其运行在超级计算机上。

2.4 BSERROR算法的并行化

Algorithm 1 欧式期权对冲策略误差控制的串行算法

参数 $X0$ ▷ 金融产品在$t=0$时的初始价格
参数 σ ▷ 市场波动性
参数 K ▷ 期权合约价格
参数 T ▷ 期权到期时间
参数 ϵ ▷ 可接受的误差上限
参数 $Prob$ ▷ 离散策略误差可接受概率下限
procedure BSERROR(M,N) ▷ M 是蒙特卡洛模拟次数，N是离散抽样次数
    $\delta t \leftarrow T/N$ ▷ 时间间隔
    $count \leftarrow 0$ ▷ 在M次模拟中N被接受的次数
    for all $m = 1 : M$ **do** ▷ 外部循环实现多次蒙特卡洛模拟
        $NRV[N]$ ▷ 一组高斯分布的随机数
        for all $NRV[j]$ **do** $NRV[j] \leftarrow GaussienNumGenerator()$
        end for
        $BM[N]$ ▷ 一个数组模拟布朗随机过程
        $BM[0] \leftarrow NRV[0] \cdot \sqrt{\delta t}$
        for all $BM[j]$ **do** $BM[j] \leftarrow BM[j-1] + NRV[j] \cdot \sqrt{\delta t}$
        end for
        $PX[N+1]$ ▷ 离散化的期权估价
        $PX[0] \leftarrow X0$
        for all $j = 1 : N+1$ **do**
            $PX[j] \leftarrow X0 \cdot exp(-0.5 \cdot \sigma^2 \cdot j \cdot \delta t + \sigma \cdot BM[j-1])$
        end for
        $error \leftarrow 0$ ▷ 离散估价和连续估价的误差
        for all $j = 0 : N-1$ **do**
            $Upper \leftarrow (\log(PX[j]/K) + 0.5 \cdot \sigma^2 \cdot (T - T_j))/(\sigma \cdot \sqrt{T - T_j})$ ▷ 积分上界
            $error \leftarrow error - 1/(\sqrt{2\pi}) \cdot (PX[j+1] - PX[j]) \cdot \int_{-\infty}^{Upper} e^{-t^2/2} dt$
        end for
        if $PX[N] > K$ **then** ▷ 只考虑在合约到期时间买入的情况
            $error \leftarrow error + (PX[N] - K)$
        end if
        $Upper1 \leftarrow (log(X0/K) + 0.5 \cdot \sigma^2 \cdot T)/(\sigma\sqrt{T})$
        $Upper2 \leftarrow (log(X0/K) - 0.5 \cdot \sigma^2 \cdot T)/(\sigma\sqrt{T})$
        $error = error + K/(\sqrt{2\pi} \cdot \int_{-\infty}^{Upper2} e^{-t^2/2} dt - X0/(\sqrt{2\pi}) \cdot \int_{-\infty}^{Upper1} e^{-t^2/2} dt$
        if $error < \epsilon$ **then**
            $count \leftarrow count + 1$
        end if
    end for
    $prob \leftarrow count/M$
    if $prob < Prob$ **then**
        $N \leftarrow N + 1$
    else
        $N \leftarrow N - 1$
    end if
    return N
end procedure

Algorithm 2 最优N值的搜索算法

参数 $M0$ ▷ 蒙特卡洛模拟的次数，$M0$越大模拟结果可信度越高
procedure NBSECT($N0$, λ) ▷ $N0$是初始化抽样次数，λ是最优解的置信区间
    $nL \leftarrow 1$ ▷ 设置n值的初始下界
    $nU \leftarrow N0$ ▷ 设置n值的初始上界
    $n \leftarrow 0.5 \cdot (nL + nU)$
    while $|nU - nL| > \lambda$ **do**
        if $BSERROR(M0, n) < n$ **then** ▷ 缩小搜索范围
            $nU \leftarrow n$
            $n \leftarrow 0.5 \cdot (nL + nU)$
        else ▷ 扩大搜索范围
            $nL \leftarrow n$
            $n \leftarrow 0.5 \cdot (nL + nU)$
        end if
    end while
    return n, nU, nL
end procedure

章节 3

模型验证及实验结果分析

为了验证我们的模型及算法在HPC金融领域的实际应用效果，我们将在Intel MIC集群上展开多种实验。