# 使用 CUDA 平行計算機制運用到高頻套利交易—以台指期權市場為例

徐宇薇[1] 林俞君[1] 潘長華[1] 吳現任[1] 戴天時[1] 王釧茹[2]
國立交通大學資訊管理與財務金融學系
台北市立大學資訊科學系

## Abstract

*高頻交易，指的是透過極高速且精密的電腦程式，在極短時間內下達大量的交易指令，以捕捉商品瞬時價格波動，賺取利益，故可高速執行的交易資訊設備與以及有效率的交易演算法對高頻交易十分重要。而平行計算可透過將複雜的難題拆解成許多的小問題，再分配給不同的平行計算單元同時處理，以提升計算效率，減少整體計算時間，所以本文將研究如何使用平行計算提高高頻交易的效能。*

*本研究將以期權市場的套利為例，採用期貨箱型價差、賣權買權期貨平價理論、買權賣權價差及蝶狀價差等四種套利策略，將期交所提供的委託單的歷史資料輸入虛擬交易所程式，還原市場當時的交易狀況，並且加入使用傳統循序計算的程式與使用 GPU(Graphic Process Unit)，請見後文解釋)執行平行計算的下單程式，利用虛擬交易所揭露的最佳的五檔買賣價資訊，互相競爭尋找套利機會，並利用二分逼近法計算各履約價的隱含波動度找尋錯價時機，比較錯價與套利策略的關係。藉此驗證 GPU 強大的運算能力，同時證明平行運算對於高頻交易有很大的幫助。*

## 1 研究動機與研究問題

隨著交易技術進步，交易時從委託到成交的延遲時間不斷縮短，高頻交易即是利用極短時間內的價格變動創造利潤。由於人工交易不易捕捉到瞬時價格變動，故高頻交易常需仰賴自動化的程式交易。大多數人認為高頻交易降低了買賣價差，提高了市場流動性，且降低了市場波動率，亦有反對者指出其對全球金融體系構成威脅。2009 年時，根據美國一財金新聞網站「FT/alphaville」一篇名為「The Cold War in high frequency trading」的文章，表示當時全球已有 70%的股票買賣活動是由高頻交易的買程式賣系統所完成。故我們認為不論高頻交易帶來的影響好壞與否，都有深入了解之必要。

高頻交易的關鍵在於交易資訊設備與交易演算法，設備或演算技術贏過其他高頻交易商就有較高的獲利機會， 而硬體的效能進化有助於演算法的演算效率與速度。故本文將以期權市場的套利為例，觀察並分析平行計算利用於高頻交易上所獲得的效益。

套利策略的獲利基礎，在於利用實際價格與理論價格的偏離，賺取其間的價差。而現今的國際金融與衍生性金融產品市場如此發達，套利機會可說是無所不在。但在「效率市場」的假說之下，一個成熟的期權市場資訊流通往往十分迅速，套利機會稍縱即逝，故非常適合利用高頻交易作迅速捕捉。各個券商無不借助計算機的高速運算，欲成為最早捕捉到套利機會的勝利者，因為若稍慢送出套利單，就幾乎沒有獲利機會了。

一般而言，電腦裡的資料運算都是由 CPU(Centeral Process Unit，請見後文介紹)來循序處理。而隨著科技的發展，過去主要負責圖形影像處理的 GPU，現今也多被用來進行一般的運算程式。GPU 憑著其強大的平行運算能力，大幅提升了程式的計算效能。翁輝澤(2011)中指出高度資料平行的演算法，GPU 能夠獲得比 CPU 高出將近 8-10 倍的計算效益。Zhang, N., J. Wang and Y. Chen (2010)中也指出在 Sobel edge detection( 邊緣檢測 ) 與 homomorphic filterin(同態濾波)兩圖形處理演算法下,GPU 的圖形處理速度相較於 CPU 快了約 25-49 倍。皆證明了 GPU 的運算效能大大勝過了 CPU。若能將 GPU 運用在尋找套利機會上，勢必也能獲得比使用 CPU 更好的效果。

林威辰(2011)使用 CUDA 的平行計算功能來提升尋找套利策略的速度，實驗中，他利用自行建構的虛擬交易所及期交所的委託單資料，可還原出 2007~2008 年間的臺灣期權市場的任一時間點的交易狀況(如成交價量及未成交委託單的價量)，利用 GPU 上的 CUDA 平行計算程式與 CPU 分別檢驗市場揭露的未成交委託單資訊以尋找有無套利機會，結果顯示出 GPU 搜尋套利機會的能力及獲利效率均優於傳統非平行計算的程式，證明了套利策略經由平行運算後可大大提升真實市場的獲利率。其中他使用了 Put-Call-Future parity 及 Convexity of Option Prices 尋找套利機會。耿世鈞(2008)指出選擇權具有靈活多變化的特性，僅使用 Put-Call-Future parity 及 Convexity of Option Prices 無法真正檢測出市場效率性，故其利用期貨箱型價差、賣權買權期貨平價理論、買權賣權價差及蝶狀價差等四種套利策略，檢測台指選擇權的效率

性，了解套利機會出現的頻率及報酬。因此本研究欲採用上述四個套利策略，利用林威辰(2011)的虛擬交易所程式，同樣在其中加入 CPU 與 GPU 兩虛擬交易商，利用程式交易互相競爭尋找套利機會。更進一步的，我們也將檢測隱含波動度與套利機會之間的相關性。代表標的物風險大小的「隱含波動度」是國內券商多用來觀察選擇權市場是否有錯價產生以判斷有無獲利機會工具之一。廖宏盛(2005)中利用買權賣權平價理論進行台指選擇權的套利，已證實隱含波動率與獲利率有正向關係。因此我們可以利用前述之四套利策略所找出之套利機會，與標的物買賣價報價的隱含波動度作比較，利用更完整的模型來檢測，以獲得更精確的結果。
本研究將期交所委託單資料輸入虛擬交易所程式，以還原當時市場交易環境，檢視台指期權市場的套利機會，並驗證 GPU 擁有較佳計算能力，同時期許未來能將平行計算推廣應用到各種需處理大量及時運算問題的高頻交易，獲得更大的效益。

## 2 文獻回顧與探討

隨著科技進步，GPU 憑著其強大的平行計算能力已逐漸取代 CPU 使用於程式運算上。林威辰(2011)利用平行運算可提升計算速度的特性，建構出一個高效能的衍生性金融商品的套利交易系統，藉由台灣期貨交易所提供歷史委託單來模擬真實世界的期貨和選擇權市場，接著在虛擬交易所的環境中加入兩個互相競爭的虛擬交易商，其中一個使用 CPU 執行傳統循序計算的程式，而另一個使用 CUDA 來平行計算尋找套利機會，其中使用的套利策略分別是「Convexity of Option Prices」和「Put-Call-Future Parity」，而研究結果證明 CUDA 成功的在套利交易競賽中打敗 CPU。
圖 1 為林威辰(2011)建構出的一個讀取委托單的虛擬交易平台(Virtual future exchange)，該平台會自動搓合可成交的委託單，整理未成交委託單，揭露最佳五檔買賣價，此外，我們另加入兩個自動電子下單系統，分別使用 CPU 和 GPU 尋找套利策略。所以由圖可看出主程式將同時執行三個 Thread，第二個 Thread 功能是虛擬真實的交易所，第一個 Thread 與第三個 Thread 核心功能是在找尋套利機會的自動交易系統，其中的差異在尋找套利機會時計算是用哪個核心。第一個 Thread 是以 CPU 單執行緒作尋找套利的機會；第三個 Thread 在找套利機會時是交給 GPU 作平行運算。
虛擬交易所主要是還原交易日的各時間點的交易狀況，最大的特色是使用一個 Timer 來作計時動作，以 250 (ms)周期讀取委託單資料，並處理委托單的撮合，找出尚未撮合契約的最佳五檔買賣價，盡可能將該日的交易完整呈現。
自動交易系統則是一開始讀取交易所中即時報價的資料，將期貨及不同履約價的買權和賣權的最佳五檔買賣價讀取出來。分別交給 CPU 或

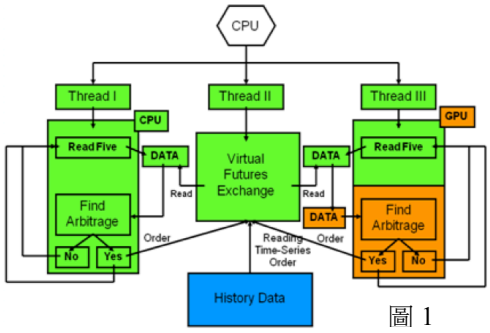

圖 1

GPU 來作尋找套利策略的動作，若是有找到套利機會，即立刻作下單的動作，之後再循環到一開始讀取最佳五檔部份；若是沒有找尋到套利機會，即直接將前述動作不斷重覆直到該日交易時間結束。
以下將介紹本研究所使用的四種套利策略：
耿 世 鈞 (2008) 根 據 Capelle-Blancard and Chaudhury(2001)和 Ackert and Tian(2000)的無套利關係式，做為實證模型。符號如下表：
TXO—臺指選擇權
TX—臺股期貨契約
C：TXO 買權價格
P：TXO 賣權價格
F：TX 指數期貨價格
K：選擇權之履約價格
(T-t)：選擇權距到期日之時間(單位：年)；t 為現在；T 為到期日
rL：貸款利率
rB：借款利率
M：保證金
m：在距標的物到期期間之保證金的利息 ($m = M(e^{-rL(T-t)} - 1)$)
Hi：i = c,p,f 分別表示買權、賣權及期貨之交易稅及交易成本
一、買賣權價差(spread strategy)
買權的多頭價差為買進一個 約價為 $K_1$ 的買/賣權，同時賣出一個履約價為 K2 的買/賣權，其中 $K_1 < K_2$；若在未考慮交易成本之無套利假設下，歐式買權價差組合、賣權價差組合的合理價格關係式應為：

$$(K_2 - K_1)e^{-r(T-t)} \geq C_2 - C_1 \quad (1A)$$

$$(K_1 - K_2)e^{-r(T-t)} \geq P_2 - P_1 \quad (1B)$$

在考慮交易手續費及期交稅、到期結算相關費用及期交稅和保證金等交易成本後，欲執行套利交易需滿足下列關係式：(π 為期初獲益)

$$\pi \equiv (K_2 - K_1)e^{-rL(T-t)} - (C_2 - C_1) - 2H_c - m > 0 \quad (2A)$$

$$\pi \equiv (K_1 - K_2)e^{-rL(T-t)} - (P_2 - P_1) - 2H_p - m > 0 \quad (2B)$$

若(2A)成立，表示低履約價的買權相對於高履約價的買權有高估的情形，因此，可賣一個買權多頭價差並借出$(K_2 - K_1)e^{-r(T-t)}$套利；同理，若(2B)成立時，表示高履約價的賣權相對於低履約價的賣權有高估的情形，可賣一個賣權空頭價差並借出$(K_2 - K_1)e^{-r(T-t)}$來套利。但是，選擇權策略組合部位的到期日現金流量並非定值，其決定於最後結算價。放空買權價差套利，在結算價$F_T \geq K_2$時現金流入最低；放空賣權價差套利，在結算價$F_T \leq K_1$時現金流入最低。因此，在形成套利投資組點後的任何時間點 $\tau$，若提前平倉利益$\pi_\tau > 0$，則應提前平倉。

買權價差組合提前執行平倉條件可用下列不等式表述：

$$\pi_\tau = (K_2 - K_1)e^{-rL(T-\tau)} - (C_{2,\tau} - C_{1,\tau}) - 2H_c > 0$$

且 $F_\tau \geq K_2$ (3A)

賣權價差組合提前執行平倉條件：

$$\pi_\tau = (K_1 - K_2)e^{-rL(T-\tau)} - (P_{2,\tau} - P_{1,\tau}) - 2H_p > 0$$

且 $F_\tau \geq K_2$ (3B)

二、箱型價差

第一位使用箱型價差套利關係檢測選擇權市場的效率性為 Billingsley and Chance(1985)。一個多頭(或空頭)箱型價差部位包含買進(或賣出)一個買權多頭價差以及賣出(或買進)一個賣權空頭價差，其選擇權的履約價分別為 K2 與 K1(其中$K_1 < K_2$)，創造一個無風險部位，並於到期日得到無風險報酬$K_2 - K_1$(或$K_1 - K_2$)理論上，如不考量交易手續費等費用，其支付(或得到)權利金淨額應等於該無風險報酬的現值，關係式如下：

$$C_1 - C_2 + P_2 - P_1 = (K_2 - K_1)e^{-r(T-t)} \quad (7)$$

在考慮借入利率(rB)與貸出利率(rL)，無套利價格區間成為

$$(K_2 - K_1)e^{-rB(T-t)} \leq C_1 - C_2 + P_2 - P_1 \leq (K_2 - K_1)e^{-rL(T-t)} \quad (8)$$

若考慮成本與費用，買進箱型價差套利交易獲利(π)為

$$\pi = (K_2 - K_1)e^{-rB(T-t)} - (C_1 - C_2 + P_2 - P_1) - 2H_c - 2H_p - m \quad (9A)$$

再者，放空箱型價差套利交易獲利(π)為

$$\pi = C_1 - C_2 + P_2 - P_1 - (K_2 - K_1)e^{-rL(T-t)} - 2H_c - 2H_p - m \quad (9B)$$

而形成套利投資組合後的任何時點 $\tau$，若提前平倉利益$\pi_\tau > 0$，則應提前平倉：

多頭箱型價差平倉的獲利為：

$$\pi_\tau = C_{1,\tau} - C_{2,\tau} + P_{2,\tau} - P_{1,\tau} - (K_2 - K_1)e^{-rB(T-\tau)} - 2H_c - 2H_p > 0 \quad (10A)$$

空頭箱型價差平倉的獲利為：

$$\pi_\tau = (K_2 - K_1)e^{-rL(T-\tau)} - (C_{1,\tau} - C_{2,\tau} + P_{2,\tau} - P_{1,\tau}) - 2H_c - 2H_p > 0 \quad (10B)$$

三、蝶狀價差(Convexity strategy)

蝶狀價差包含三種不同履約價的買權或賣權組合蝶狀價差包含三種不同履約價的買權或賣權組合，也就是，同時買進w單位及$(1 - w)$單位履約價分別為K1和K3的買權(或賣權)並賣出一單位履約價為K2的買權(或賣權)，其中

$K_1 < K_2 < K_3$；

$w = (K_3 - K_2)/(K_3 - K_1)$。若標的物價格低於K1或高於K3，該投資組合期末報酬為零，反之，該組合期末報酬為正。在無套利之前提下，期初買進蝶型價差的投資成本應大於零，在不考慮交易成本之下，選擇權權利金的關係式應滿足：

$$wC_1 + (1 - w)C_3 \geq C_2 \quad (4A)$$

$$wP_1 + (1 - w)P_3 \geq P_2 \quad (4B)$$

若考慮交易手續費及期交稅、到期結算相關費用及期交稅、和保證金的機會成本，無套利交易條件的關係式為：(π為利益)

$$\pi \equiv ב_2 C_2 - ב_1 C_1 - (ב_2 - ב_1)C_3 - (2ב_2)H_C - m < 0 \quad (5A)$$

$$\pi \equiv ב_2 P_2 - ב_1 P_1 - (ב_2 - ב_1)P_3 - (2ב_2)H_P - m < 0 \quad (5B)$$

其中，$ב_1$和$ב_2$為互質的整數，且$w = ב_1/ב_2$。

若(5A)或(5B)不成立，表示履約價格為K2的買權或賣權被高估，若設$w = \frac{1}{2}$，可賣出兩個K2的買權或賣權，並買入一個K1和K3的買權或賣權進行無風險套利，其到期日報酬恆大於等於零。為了追求更大獲利，在形成套利投資組點後的任何時間點 $\tau$，若符合下面條件，且提前平倉利益：

$\pi_\tau > 0$，則應提前平倉：

買權蝶狀價差組合提前執行平倉條件：

$$\pi_\tau = \text{ב}_1 C_{1,\tau} + (\text{ב}_2 - \text{ב}_1)C_{3,\tau} - \text{ב}_2 C_{2,\tau} - C_{2,\tau} - (2\text{ב}_2)H_C > 0$$

且

$$F_\tau \geq K_3 \text{ or } F_\tau \leq K_1 \quad (6A)$$

賣權蝶狀價差組合提前執行平倉條件：

$$\pi_\tau = \text{ב}_1 P_{1,\tau} + (\text{ב}_2 - \text{ב}_1)P_{3,\tau} - \text{ב}_2 P_{2,\tau} - P_{2,\tau} - (2\text{ב}_2)H_p > 0$$

且

$$F_\tau \geq K_3 \text{ or } F_\tau \leq K_1 \quad (6B)$$

四、賣權買權平價關係式(Put-Call-Futures-Parity)

賣權買權平價關係式最早是由Stoll(1969)所建立的，而Tucker(1991)證明了一般的賣權買權期貨平價關係，在買賣權的平價關係式中，用期貨取代現貨指數。以小台指、大台指選擇權及台指期貨進行說明，若不考慮股利發放和交易成本或賣空限制，其關係式為：

$$(F - K)e^{-r(T-t)} = C - P \quad (11)$$

其中K代表選擇權的履約價格。若考慮借入利率(rB)與貸出利率(rL)，其無套利價格區間為：

$$(F - K)e^{-rB(T-t)} \leq C - P \leq (F - K)e^{-rL(T-t)}$$

當 $F - K \geq 0$ (12) or

$$(K - F)e^{-rB(T-t)} \leq P - C \leq (K - F)e^{-rL(T-t)}$$

當 $K - F \geq 0$ (13)

若考慮成本與費用，則需先考慮契約規格比率θ，如：標的物為大台指和小台指選擇權之投資組合，其$\theta = 4$，因為大台指一點為200元；小台指一點為50元。

在此定義套利組合Z（放空買權、買進賣權、買進期貨、存入$(F - K)e^{-rL(T-t)}$）及其期初交易獲利 (π)。

若套利組合Z欲套利成功，需滿足以下條件關係式：

$$\pi = \theta(C - P) - (F - K)e^{-rL(T-t)} - \theta H_c - \theta H_p - H_F - (m) > 0$$

當 $F - K \geq 0$ (14A) or

$$\pi = (K - F)e^{-rB(T-t)} + \theta(C - P) - \theta H_c - \theta H_p - H_F - (m) > 0$$

當 $K - F \geq 0$ (15A)

而定義套利組合Y（放空賣權、買進買權、放空期貨、借出(F-K)e-rL(T-t)）及其期初交易獲利 (π)。

若套利組合Y欲套利成功，則需滿足以下條件關係式：

$$\pi = (F - K)e^{-rB(T-t)} - \theta(C - P) - \theta H_c - \theta H_p - H_F - (m) > 0$$

當 $F - K \geq 0$ (14B) or

$$\pi = \theta(C - P) - (K - F)e^{-rL(T-t)} - \theta H_c - \theta H_p - H_F - (m) > 0$$

當 $K - F \geq 0$ (15B)

上述投資組合在到期日的報酬恆為0，如果在到期日之前任何時點 τ的提前平倉利益$\pi_\tau > 0$，則應提前平倉以追求更大利潤。

具體而言，投資組合Z若符合以下條件則提前平倉：

$$\pi_\tau = (F_\tau - K)e^{-rL(T-\tau)} - \theta(C_\tau - P_\tau) - \theta H_c - \theta H_p - H_F > 0$$

當 $F - K \geq 0$ (16A) or

$$\pi_\tau = \theta(P_\tau - C_\tau) - (K - F_\tau)e^{-rB(T-\tau)} - \theta H_c - \theta H_p - H_F > 0$$

當 $K - F \geq 0$ (17A)

投資組合Y若符合以下條件則提前平倉：

$$\pi_\tau = \theta(C_\tau - P_\tau) - (F_\tau - K)e^{-rB(T-\tau)} - \theta H_c - \theta H_p - H_F > 0$$

當 $F - K \geq 0$ (16B) or

$$\pi_\tau = (K - F_\tau)e^{-rL(T-\tau)} + \theta(C_\tau - P_\tau) - \theta H_c - \theta H_p - H_F > 0$$

當 $K - F \geq 0$ (17B)

## 3 研究方法與步驟

本研究將利用林威辰(2011)的虛擬交易平台讀取 2007~2008 年台灣期交所日內買賣委託單的資料，依其委託時間送入虛擬交易所的程式進行搓合，並加入兩個電子交易系統，作為兩個互相競爭的虛擬交易商，分別在 CPU 與 GPU 上進行運算、找尋套利機會，藉由兩者取得套利機會次數來比較兩者運算效能。

為了更精確的模擬出真實的交易環境，我們將採用四種套利策略－箱型價差、賣權買權平價理論、買權賣權價差及蝶狀價差，交由 CPU 與 GPU 尋找套利機會。而由於台灣期權市場僅揭示最佳五檔的委託價量資訊，故我們的套利程式也將利用各檔選擇權的最佳五檔報價資訊來

尋找套利機會。

被視為投資者恐慌指標的隱含波動度是學界研究市場投資者行為的重點。我們可將選擇權的市價帶回 Black-Scholes 評價公式，使用二分逼近法反求出隱含波動度。當某一履約價的買(賣)報價所對應的隱含波動度偏離波動度曲線時，代表發生錯價，套利機會產生。此時我們即可利用前述使用四套利策略找尋出的套利機會與此作比較，以得知兩者的相關性。相關的研究議題詳述如下。

### 3.1 CUDA 的平行運算

在文獻探討與回顧當中得知 CUDA 平行運算是建構在 GPU 產生的 Thread 上。GPU 上的程式可以分成三個層次，最上層的是 Grid，同一個 Grid 中分為多個 Block，一個 Block 中又可分為多個 Thread。而同一個 Block 裡的 Thread，會以不同的資料來執行相同的指令。因此如何根據「最有效率的將資料共用」來分配 Thread 給每一個 Block 是平行運算最佳化的關鍵。以 Put Call Future Parity 為例，因不同履約月份無法彼此進行套利，故適合用 Block 來切割不同月份的選擇權委託單。舉例來說，現在選擇權與期貨所能交易的月份為 2、3、6、9 和 12 月，故我們可宣告 Block 的個數為 5，再利用 Block 中的 Thread 平行檢驗每一個套利機會是否可行。

除此之外，檢查套利策略所需要計算的量會隨著選擇權履約價檔次增加而大幅增加，再加上檢驗每組套利策略時，必須考慮相對應的選擇權未成交委託單的價量，因此需要的計算複雜度也大幅增加，適合採用 CUDA 的技術將大量問題分給不同的 thread 平行處理，以減少計算時間，爭取下單的速度。舉例來說，假設目前市場上有履約價為 5600，委買價為 50 元的選擇權一張，履約價為 5700，委賣價為 71 元與 68 元的選擇權各一張，以及履約價為 5800，委買價為 90 元的選擇權一張。以 Convexity Strategy 為例，投資者將會賣一張 5600、賣一張 5800、買兩張 5700 的選擇權。如不考慮未成交委託單的分佈，只用最佳的買賣價計算可得期初總成本為 50+90-71*2= -2，可以執行套利。但真實成本則為 50+90-71-68 =1，不能套利。由此例可知，若考慮策略執行時配合套利策略後的買賣量，整體的運算機制將變得更複雜，因此需藉由 GPU 的平行運算技術負責龐大的運算以加快執行套利策略的效率。

### 3.2 負載平衡

負載平衡指的是透過適當的機制將工作量平均分配給不同運算單元，以達到整體計算時間的最佳化。當委託單依履約月份分配到不同 Block 後，下一個需要考量的重點即是 Block 中各個 Thread 的工作量該如何分配。而 CUDA 平行計算的特性是必須等最大工作量的 Thread 作完後才能下套利單。因此我們依照 Thread 數量與總工作量，計算出每個 Thread 的平均工作量，利用負載平衡的概念將工作量平均分配，使平行運算發揮最大效用。舉例來說，若履約月份為一月的選擇權的履約價從 5000 點到 5900 點總共有 10 檔履約價，以 Convexity Strategy 為例，一個套利策略組合包含三種不同履約價的買權或賣權組合，$C_3^{10} = 120$，共有 120 組套利策略組合。假如一個 Block 中有 32 個 Thread，則一個 Thread 將檢驗 3 或 4 個套利策略(也就是工作量)。

### 3.3 Warp 的上限

在研究的過程中，我們發現了 Warp 有上限的特性，以 NVIDIA GeForce GT 640 為例，如圖 2 所示，大約是 128 個 Warp。只要一超過 128 個 Warp，我們可以發現計算時間直接跳到大約兩倍的時間。

### 3.4 Naïve Load v.s. Dynamic Programming Load Balance

Load Balance 的目的是將工作平均分配給各個計算單元，以達最佳的運算時間。因為 CUDA 上的計算是平行的關係，若讓每個 Warp 分配到差不多的工作量，就不會有互相等候的情況，可以達到整體計算時間的最佳化。在此我們有兩種方式，第一種是 Naïve Load Balance，我們假設不同策略執行時所花的 Latency Time 都一樣，將所有工作平均分配給各個 Warp。每個 Warp 都會執行到四種不同的策略。但事實上每個策略的 Latency Time 都不盡相同，因此可能會發生 Warp 閒置或等待的情況。為了解決上述的問題，我們使用了 Dynamic Programming Load Balance。在計算單元有限制的情況下，針對不同的策略有不同的計算量以及分配給每種策略

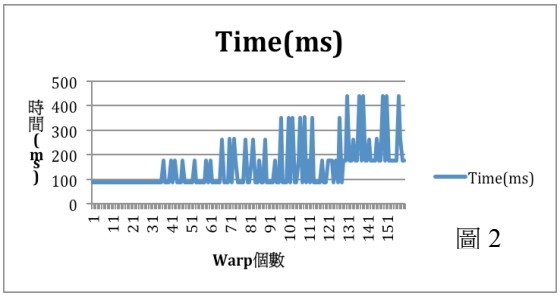


圖 2

不同數量的計算單元。相較於 Naïve Load Balance，較不會有 Warp 閒置狀況，也能解決不同策略下 Latency Time 不同的問題。

### 3.5 Dynamic Programming Load Balance

為了達到最佳的運算時間，我們要分配最適合數量的策略給 Warp。我們定義 $f_i^{exe}(w_i)$ 為第 i 個交易策略，在給 $w_i$ 個 Warps 之下的 GPU Latency Time。如圖 3 所示，分配給交易策略越多的 Warp，Latency Time 就會越小，但是超過一個極限之後，分配太多的 Warp 反而會有較大的 Latency Time。而如圖所示，wi 為策略 i 的最佳 Warp 配置，可使得 GPU Latency time 最小化。

我們假設最適的 Warp 上限為 W，並假設總共有 k 個交易策略。會產生兩種情況：

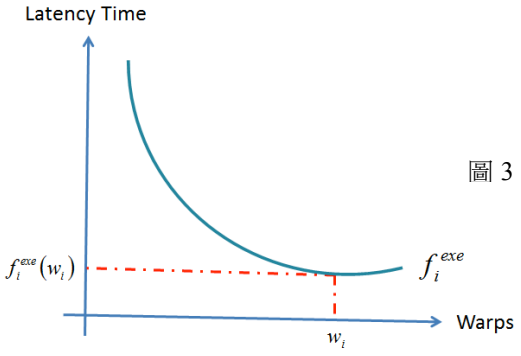


圖 3

(1)

最佳 Warp 配置的總數量小於 Warp 上限。在這種情況下，最佳的 Warp 分配就是將最佳 Warp 數量分配給各個策略。

(2) $\sum_{i=1}^{k} w_i > W$

最佳 Warp 配置的總數量大於 Warp 上限。在這種情況下，我們必須個別配置

$p_1, p_2, \ldots\ldots, p_k$ 個 Warp 給策略 *1,2,...,k*。

我們定義

$$L_k^{Opt}(W) = \begin{cases} \min\limits_{1\le p_k\le W} \max\left[f_k^{exe}(p_k), L_{k-1}^{Opt}(W-p_k)\right] & k>1 \\ \min\left[f_1^{exe}(W), f_1^{exe}(w_1)\right] & k=1 \end{cases}$$

$$\begin{aligned} L_l(W') &= \max_{1\le i\le l} f_i^{exe}(p_i) \\ &= \max\left[f_l^{exe}(p_l), \max_{1\le i\le l-1} f_i^{exe}(p_i)\right] \\ &\ge \max\left[f_l^{exe}(p_l), L_{l-1}^{Opt}(W'-p_l)\right] \quad \text{由以上} \\ &\ge \min_{1\le p_l\le W} \max\left[f_l^{exe}(p_l), L_{l-1}^{Opt}(W'-p_l)\right] \\ &= L_l^{Opt}(W') \end{aligned}$$

得證，最適的 Warp 上限為 W，有 k 個交易策略，Optimal latency time 為

$$L_k^{Opt}(W) = \begin{cases} \min\limits_{1\le p_l\le W} \max\left[f_l^{exe}(p_l), L_{l-1}^{Opt}(W-p_l)\right] & k>1 \\ \min\left[f_1^{exe}(W), f_1^{exe}(w_1)\right] & k=1 \end{cases}$$

# 4 實證結果與分析

## 4.1 Naïve Load Balance 與 Dynamic Programming Load Balance 的差異

在分配適當個數 Warp 給各個策略以達到最佳執行時間的問題上，我們採用兩種不同的分配方法。一種是 Naïve Load Balance，將工作量平均分配給每一個策略。而另一種是 Dynamic Programming Load Balance，則是依據各策略所需的最佳分配來配置。理論上來說，採用 Dynamic Programming Load Balance 的總執行時間應該優於 Naïve Load Balance 的總執行時間。但由圖 4 可以看出，兩種方式的執行時間並沒有太大的差異。這很有可能是因為本研究所執行的套利策略筆數不夠多，且利用 Dynamic Programming Load Balance 時必須做前置的 Warp 個數配置運算，以至於無法看出兩者差異。

## 4.2 隱含波動度偏移與套利次數的關係

我們定義隱含波動度偏離發生的時間跟套利發生時間相差五秒內以內為相關。由圖 5 圖 6 可得，1 月 8 日有 75%相關，1 月 21 日有 78%相關，1/22 則有 75%相關。因此我們可以推斷，隱含波動度發生偏離的時候，往往會有套利機會的產生。證實我們先前所做的假設。

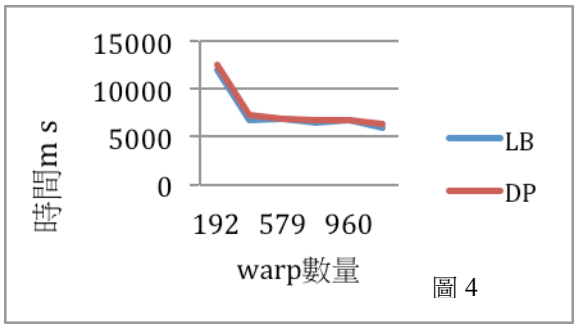


圖 4

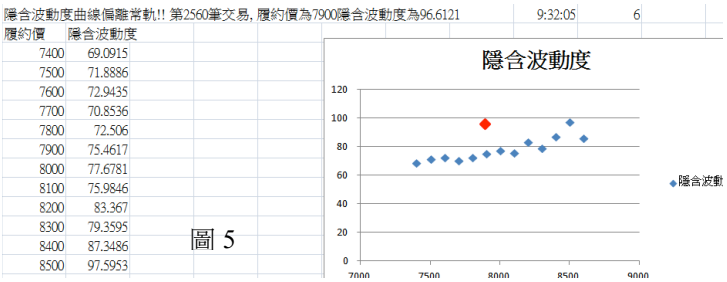


圖 5

圖 6

| 隱含波動度偏離時間 | 套利發生時間 | 相差秒數 |
|---|---|---|
| 9:10:33 AM | 9:10:32 AM | 1 |
| 9:34:01 AM | 9:34:01 AM | 0 |
| 9:35:16 AM | 9:34:18 AM | 2 |
| 9:35:20 AM | 9:35:07 AM | 13 |
| 9:35:59 AM | 9:36:58 AM | 1 |
| 9:41:41 AM | 9:41:54 AM | 13 |
| 9:48:00 AM | 9:48:02 AM | 2 |
| 9:59:24 AM | 9:59:24 AM | 0 |

| 隱含波動度偏離時間 | 套利發生時間 | 相差秒數 |
|---|---|---|
| 09:04:24 | 09:04:28 | 4 |
| 09:32:05 | 09:32:00 | 5 |
| 09:33:40 | 09:33:41 | 1 |
| 09:49:10 | 09:49:15 | 5 |
| 09:52:00 | 09:51:58 | 2 |
| 09:57:00 | 09:56:57 | 3 |
| 09:57:17 | 09:57:06 | 11 |
| 09:58:08 | 09:58:12 | 4 |
| 09:59:13 | 09:59:04 | 7 |

| 隱含波動度偏離時間 | 套利發生時間 | 相差秒數 |
|---|---|---|
| 9:01:08 AM | 9:01:36 AM | 28 |
| 9:01:42 AM | 9:01:42 AM | 0 |
| 9:01:47 AM | 9:01:46 AM | 1 |
| 9:03:58 AM | 9:03:58 AM | 0 |

# 參考文獻


[1] Ackert, Lucy F. and Yisong S. Tian, 2001“Efficiency in Index Options Markets and Trading in Stock Basket”, Journal of Banking and Finance, pp. 1607-1634.

[2] Billingsley, R. S., and Chance, D. M., 1985. “Option Market Efficiency and the Box Spread Strategy”, The Financial Review, 20, pp. 287-301.

[3] Black, F. & Scholes, M.,1973“The Pricing of Options & Corporate Liabilities,” Journal of Political Economy, Vol.18, pp. 637-654.

[4] Capelle-Blancard, G. and M. Chaudhury, 2001,“Efficiency Tests of the French Index(CAC40) Options Market,” Working Paper, University of Paris.

[5] Hans R. Stoll, , 1969“The Relationship Between Put and Call Options Prices,” Journal of Finance, Vol.21, pp.801-804.

[6] Tucker, Alan L., 1991Financial Futures, Options, and Swaps, MN: West Publishing.

[7] MacBeth, J. D., and L. J. Merville. ,1979 “An empirical examination of the Black-Scholes call option pricing model.”Journal of Finance. Vol.34, No.5, pp. 1173-1186.

[8] Tracy Alloway, 2009, The Cold War in high frequency trading, Available at: http://ftalphaville.ft.com/2009/07/08/60761/the-cold-war-in-high-frequency-trading/?, Accesed 08 July 2009.

[9] Yang, Z., Y. Zh, and Y. Pu, 2008, Parallel Image Processing Based on CUDA, IEEE International Conference on Computer Science and Software Engineering, Wuhan, China, Vol. 3, pp. 198–201.

[10] NVIDIA CORPORATION. CUDA Programming Guide Version 2.1, Santa Clara, 2009.

[11] 王啟明(2010)，”波動率估計法對選擇權價格預測能力之比較探討－以台指選擇權為例”，實踐大學資訊科技與管理學系碩士班，碩士論文。

[12] 林威辰(2011)，“平行運算用於即時套利策略交易系統”，國立交通大學應用數學系，碩士論文。

[13] 耿世鈞(2008)，”臺指選擇權與臺指期貨間對價關係與套利機會之檢測”，國立銘傳大學財務金融學系碩士在職專班，碩士論文。

[14] 翁輝澤(2010)，” GPU 應用於圖演算法之計算效益分析”，國立中山大學資訊工程學系研究所，碩士論文。

[15] 廖宏盛(2005)，”隱含波動率之研究”，臺中健康暨管理學院經營管理研究所，碩士論文。

[16] 戴天時(2005)，C++ 財務程式設計，證期會。