回測套利交易系統之平行化程式運用

Backtesting Arbitrage-Strategy Trading System

張凱翔
資訊工程學系
國立交通大學
台灣新竹市
kai861108@yahoo.com.tw

張維哲
資訊工程學系
國立交通大學
台灣新竹市
destiny10191019@gmail.com

王威堯
資訊工程學系
國立交通大學
台灣新竹市
sf163864@gmail.com

ABSTRACT

近幾年高頻交易在金融市場中顯得越來越重要，也因此能夠及時處理大量金融交易資料的技術也成了其中的關鍵，而此議題也促使我們使用平行運算的技術針對選擇權市場去設計交易策略以及搜索套利機會。在本次研究中，我們將 Convexity Strategy 應用在台指周選選擇權，發現到在選擇權市場上是擁有套利的空間。接著我們使用了 Pthread、OpenMP、MPI 三種平行化模型來平行加速我們的程式，結果發現速度可以快到約 2.33 倍。由於 thread 的計算量不夠多，造成 Pthread 本身的 overhead 佔了很大的影響，本研究也實驗加大運算量來驗證我們的假設。

INTRODUCTION

近年來，衍生性金融商品的發展相當快速，在金融市場中所扮演的角色日益重要，在台灣，其中最備受矚目的即為台灣期貨交易所股價指數選擇權。

選擇權是一種可交易的衍生金融商品，用以確認在未來某一時間段內針對特定標的物及履約價格的買賣雙方權利義務關係。簡單的說，在選擇權買方支付權利金後，即擁有權力在特定時間或特定時間內以契約中載明的履約價格，向選擇權賣方買進或賣出標的物商品。選擇權以權力型態區分又可分為買權和賣權兩種，又若以履約條件區分可分為歐式選擇權，美式選擇權。買權(Call)是指契約之買方有權在履約時依履約價格向契約之賣方買進一定額數的標的物商品；而契約之賣方則有義務在契約之買方履約時依履約價格賣出一定額數的標的物商品予買方。賣權(Put)是指契約之買方有權在履約時依履約價格向契約之賣方賣出一定額數的標的物商品；而契約之賣方則有義務在契約之買方履約時依履約價格買進一定額數的標的物商品予買方。歐式選擇權(European option)是指契約之買方必須等到契約到期日才可行使買方之權利；美式選擇權(American option)是指契約之買方有權於契約到期日前之任一交易日行使買方之權利。本研究中將會使用到台灣指數期貨選擇權，屬於歐式選擇權，交易的標的物為台灣證券交易所發行量加權股價指數，又稱之為台指選擇權，契約乘數為每一點 50 元，英文代碼是 TXO。

Convexity Strategy 中文又稱為蝶狀價差，是 Merton(1973)所提出的選擇權套利策略，內容主要是利用選擇權的履約價格與價格之間的一個 Convex 性質，並從中獲利。蝶狀價差是由三筆不同履約價格的買權或賣權所組成，因此計算量會隨著履約價格檔位的增加指數性的成長，且每一組合都可以獨立被處理，因此相當適合平行處裡。蝶狀價差包括賣出一單位履約價格為 K_2 的買/賣權，同時買進 ω 單位履約價格為 K_1 的買/賣權和 $1-\omega$ 單位履約價格為 K_3 的買/賣權，並滿足 $K_1 < K_2 < K_3$，$\omega = (K_3 - K_2 / K_3 - K_1)$。以買權為例，若對應履約價格

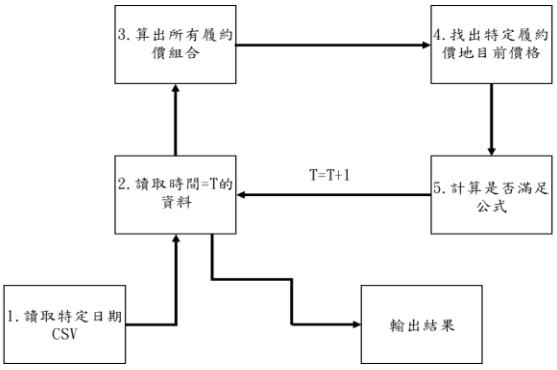
從低到高，買權價格分別以 C_1、C_2、C_3 表示，不考慮交易成本下可得公式：$C_2 \leq \omega * C_1 + (1-\omega) * C_3$

上述公式說明了市場上選擇權組合的合理價格，若有任一組合不滿足上述公式，即會出現套利機會。但是由於在實際交易中，需考慮交易成本 Cs，並為了方便計算將上述公式改寫如下：$(X_3-X_1) * C_2 - (X_3-X_2) * C_1 - (X_2-X_1) * C_3 \leq Cs$。$(X_3-X_2)$、$(X_3-X_1)$、$(X_2-X_1)$分別代表交易量的單位比例。舉例來說，假設 X_1=9000、X_2=9100、X_3=9200，則三筆選擇權交易量分別為 100、200、100，但是為了方便進行市場交易，因此我們將三者除去最大公因數 100，求出最小交易量分別為 1、2、1 口。

由台指選擇權市場的回測實驗中可以發現，台指選擇權是存在 Convexity Strategy 的套利機會，但套利機會通常都稍縱即逝。若要把握住套利機會，除了必須要實現自動化交易，還受兩個因素考量，第一個因素為運算套利速度，第二個因素為網路下單的速度。第二個因素偏向硬體的傳輸部分，因此本篇主要研究如何能夠利用平行計算，提升運算套利的速度。

期許藉由結合資工以及金融之領域，能夠真正提升在選擇權市場套利的效率。

PROPOSED SOLUTION



圖(一) 程式架構

圖(一)為本研究流程圖，在回測過程中，我們總共將過程分成 5 大步驟。

第一步：先將特定日期的所有資料讀進程式裡。

第二步：將所有的資料中，時間為 T 的篩選出來，每日一開始的 T 為 8 點 45 分 0 秒。

第三步：算出所有履約價的組合個數有多少個。假設在時間點 T，買權總共有 8400、8450、9000、9050 這些履約價成交，根據 Convexity Strategy，我們需要選出三個不同的履約價，因此總共會有①8400、8450、9000②8400、8450、9050③8400、9000、9050④8450、9000、9050 這四種可能。

第四步：因為在同一秒內，成交的價格很有可能不會只有一個，因此我們要把這些價格全部都找出來。根據第三步的第一種可能，假設履約價 8400 有 20、20.5、21 元三種價格，履約價 8450 有 50、50.5 元兩種價格，履約價 9000 有 100、105 元兩種價格，並將這些資料傳給第五步。

第五步：根據第四步，總共會有 3 * 2 * 2 = 12 種可能。因此計算履約價 8400 成交價格為 20 元，計算履約價 8450 成交價格為 50 元，計算履約價 9000 成交價格為 100 元是否有滿足公式，接著再計算履約價 8400 成交價格為 20 元，計算履約價 8450 成交價格為 50 元，計算履約價 9000 成交價格為 105 元是否有滿足公式，依此類推。

接著 T=T+1，重複步驟二到步驟五，當 T 為 13 點 45 分 0 秒時(為選擇權收盤時間)，輸出今日總共成功的套利數量。

在步驟五中，所有的運算都是獨立的，且為整個流程圖中最多運算量的地方，非常適合進行平行化。因此我們將利用 Pthread、OpenMP、

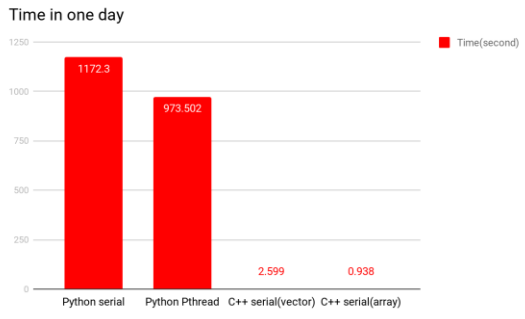
MPI 三種平行化模型，分別平行步驟五，檢驗各個模型所花費的時間是否有比原始的程式好。

EXPERIMENTAL METHODOLOGY

本次實驗使用課堂提供的伺服器來做實驗，使用的資料是從「台灣期貨交易所」網站提供的 2016 年的資料。

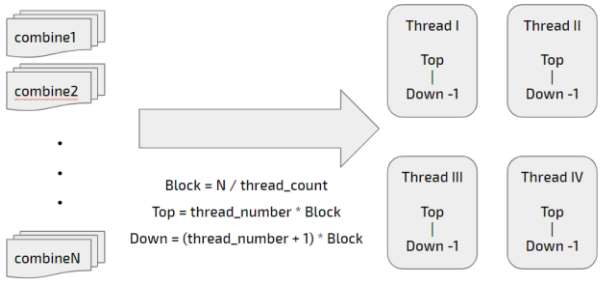
本研究首先在語言上選擇 Python，因為 Python 相較於其他的語言在資料處理上比較方便，而我們需要處理大量的資料(2016 年每天的資料)，故選擇最為便利的語言。但實做完程式碼後，發現 Python 即使在使用 Pthread 加速後，還是需要相當長的時間，於是我們將程式語言改為 C++來做後面進一步的平行化，並分別實作 array 與 vector 兩個不同的儲存方法。

從圖(二)可以看出 Python 計算完一天資料的時間遠大過 C++所需要的時間，Python 在使用 Pthread 加速後計算完一天資料的時間也需要約 973 秒，而 C++的兩個儲存版本皆只需要不到三秒的時間，以 array 為儲存方法更只需要 0.938 秒即可處理完。可能的原因為 Python 是直譯式語言，而 C++是先編譯過的語言，且 Python 因為不需定義資料型態，這些情況導致 C++程式整體使用的時間遠小於 Python 所需要的時間。



圖(二) Python 和 C++ 所需時間比較

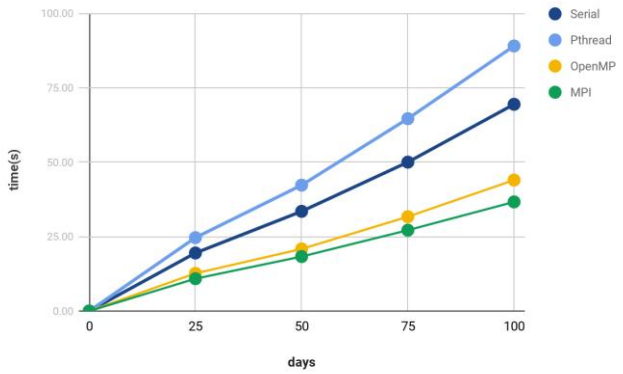
根據 proposed solution 中的第五步，每種組合之間是獨立計算的，也就是它們每一種組合都是個別去計算有無符合套利機會的條件，本研究平行化的方法即是利用第五步資料的特性。假設有 4 個 thread，便將所有組合的資料平均分配給每個 thread 如圖(三)，讓它們分別計算完套利機會的次數後，再將結果傳回加總，成為最終的答案。



圖(三) 資料分割範例

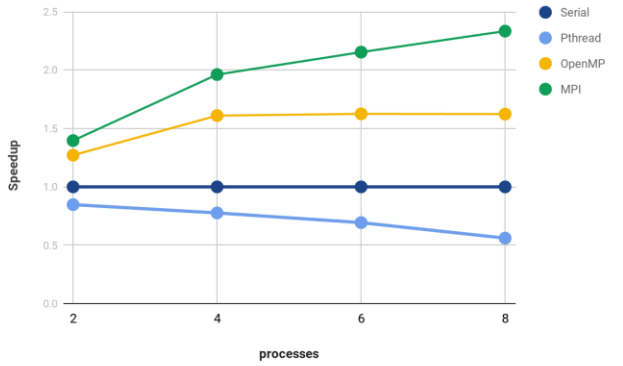
本研究利用三種平行化模型：Pthread、OpenMP 和 MPI 來做平行化，因為旨在接收一個時間點的資料後，快速計算出套利機會，所以平行化的部分只有在計算套利機會。實驗分別將 25 天、50 天、75 天和 100 天為單位當作輸入，個別計算所需要的時間，結果如圖(四)。也利用計算出來的個別時間與不同的核心數量來算出加速倍率如圖(五)和效率如圖(六)。

Days	Serial	Pthread	OpenMP	MPI
25	19.49	24.65	12.66	10.87
50	33.48	42.29	20.85	18.30
75	50.01	64.61	31.68	27.15
100	69.44	89.03	43.99	36.65



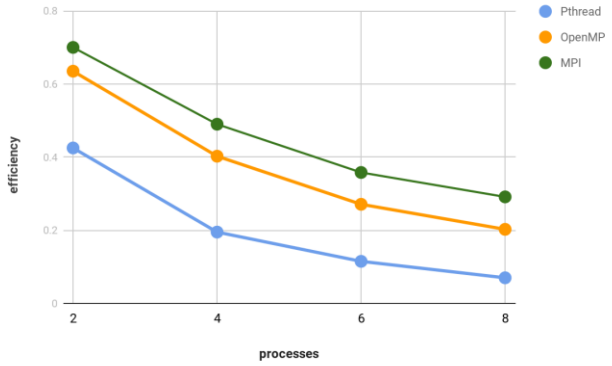
圖(四) 不同資料量與模型所需時間表格與圖

Process	Serial	Pthread	OpenMP	MPI
2	1	0.85	1.27	1.40
4	1	0.78	1.61	1.96
6	1	0.69	1.63	2.15
8	1	0.56	1.62	2.33



圖(五) 不同模型在不同數量的核心以 100 天為輸入的加速效果表格與圖

process	Pthread	OpenMP	MPI
2	0.425	0.635	0.7
4	0.195	0.403	0.49
6	0.115	0.271	0.358
8	0.07	0.203	0.29



圖(六) 不同模型在不同數量的核心以 100 天為輸入的效率表格與圖

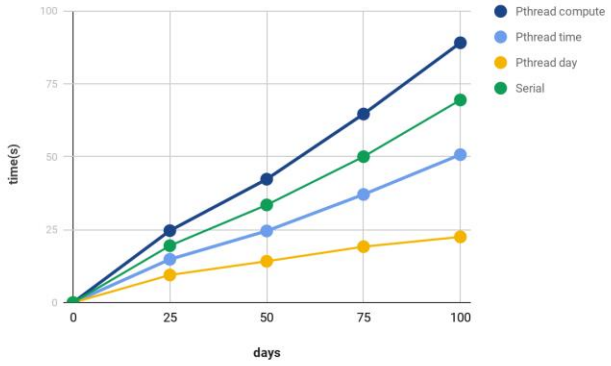
EXPERIMENTAL RESULTS

從圖(四)至圖(六)可以看出，Pthread 在不同資料量時所花費的時間皆為最多，而 MPI 的表現為最好 OpenMP 次之。我們認為 Pthread 會較 serial 原因為每個 thread 的計算量不足，導致產生新的 thread 的 overhead 大過多加一個 thread 所減少的時間，才會發生 Pthread 使用的時間比 serial 的還要多。

從圖(五)可以看出，三種模型的 scalability 並不好，加速最高為 MPI 在 8 核時為 2.33 倍。依比例增加核心的數量，加速卻沒有等比例的上升。這是因為程式加速的部分只有在第五步，其餘的都是以 serial 執行，導致加速的倍率受到限制。

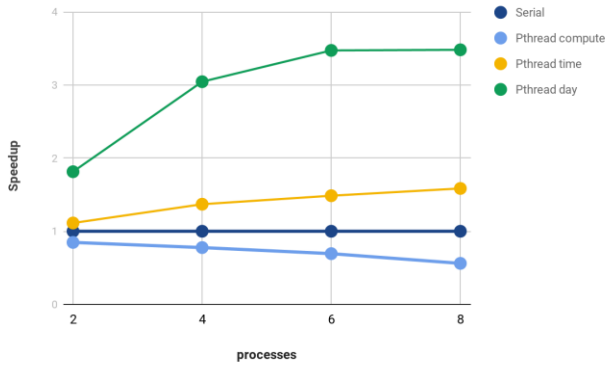
本研究平行化的部分只有在第五步，但為了驗證我們先前對於 Pthread 計算量過少的假設，將平行化的部分擴大到第四步和第五步，以及全部的步驟兩種。增加平行化部分的目的為增加每個 thread 的計算量，來驗證計算量太小的假設，實驗結果如圖(七)和圖(八)。

Days	Serial	Pthread compute	Pthread time	Pthread Day
25	19.49	24.65	14.82	9.46
50	33.48	42.29	24.55	14.16
75	50.01	64.61	37.05	19.16
100	69.44	89.03	50.69	22.51



圖(七) 不同平行化程度的 Pthread 與資料量所需時間表格與圖

Process	Serial	Pthread compute	Pthread time	Pthread Day
2	1	0.85	1.11	1.81
4	1	0.78	1.37	3.04
6	1	0.69	1.49	3.47
8	1	0.56	1.59	3.48



圖(八) 不同平行化程度的 Pthread 在不同數量的核心以 100 天為輸入的加速效果表格與圖

圖(七)為不同計算量的實驗結果，可以看出來隨著計算量提升，整體所花的時間也會更少，如 Pthread day 在資料量為 100 天時所需的時間只有 22.511 秒。從圖(八)可以看出，確實將一個 thread 計算量加大，Pthread 加速後的效果是比 serial 好的，Pthread time 為給每條 thread 多種時間的資料，加速在 8 核時為 1.59 倍，Pthread day 為將一天的資料給一條 thread，在 8 核時加速最高可達 3.48 倍，是比原先最好的 MPI 的 2.33 倍來的高許多。也可以觀察出原始的 Pthread compute 跟加大計算的 Pthread time 與 Pthread day 相比，當核心的數量增加時，Pthread compute 表現會越來越差，驗證了我們一開始的假設，增加一個 thread 的

overhead 過高，也可以觀察出確實讓每條 thread 的計算量提高會有助於整體的加速效率。

RELATED WORK

在過往的研究中，林威辰(2011)是先建構了一個虛擬交易所，藉由臺灣期貨交易所提供的歷史委託單來模擬真實世界的選擇權市場。使用了「Convexity of Option Price」和「Put-Call-Future Parity」這兩種套利策略來進行分析。並在虛擬交易所中，加入兩個競爭關係的交易者，其中一個使用 CPU，另一個使用 CUDA 來進行交易。結果發現使用 CUDA 的套利次數和獲利均能打敗 CPU。陳郁文(2016)主要是延續林威辰(2011)的研究，並增加了「Spread Strategy」的套利策略。

他們研究所使用的資料是台指月選選擇權進行回測，然而 2012/11/21，臺灣期貨交易所上市了台指周選選擇權，因此與他們不同的是，我們使用此合約來進行回測，此外，因為歷史委託單資料並不容易取得，因此我們只使用成交明細來進行回測。在平行化部分，我們分別使用了 Pthread、OpenMP 及 MPI，利用這些平行化模型來與未平行程式進行比較。

CONCLUSION

本研究將平行運算的技術應用至金融市場中，以選擇權交易的套利機會作為實作主題。為了降低尋找套利機會的時間，本研究首先測試兩種程式語言何者所需時間較短以作為研究使用，並再分別測試 array 與 vector 兩種不同的儲存方式測驗，最後選用以 C++搭配 array 為實作基礎。

本研究實作三種不同的平行化模型，包含 Pthread、OpenMP 以及 MPI，實驗結果顯示 MPI 的加速與效率表現皆為最好，在使用 8 個核心時加速可達 2.33 倍，而 Pthread 本身的 overhead 導致所需時間比一般的版本還要久，也透過加大計算量去驗證所發現的問題。透過實驗結果顯示，分散式記憶體(distributed-memory)在本研究中較分享式記憶體(shared-memory)還要適合做為尋找套利機會的平行化模型。

REFERENCE

[1]林威辰. 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, 2011

[2]陳郁文. 以 CUDA 架構實作在線套利交易機制平台. 國立交通大學資訊與科學工程研究所，碩士論文, 2016

[3]周琳. 台指選擇權套利分析.碩士論文，輔仁大學金融與國際企業學系金融碩士班, 2016

[4]林晏瑋. 以盒型價差策略探討台指選擇權市場之效率性與套利機會. 碩士論文. 國立中央大學財務金融學系碩士在職專班, 2012

[5]台灣期貨交易所, 2019. Online accessed 20-Sep-2019.

SOURCE CODE

https://github.com/wcchang1019/pp_final_project