國立政治大學資訊管理研究所

碩士學位論文

指導教授：劉文卿　博士

以大量平行運算為基礎的策略挑選與期貨模擬交易平台

研究生：戴睿 撰

中華民國一百零三年七月

摘要

策略交易係指運用可量化的分析方法(如技術分析)，完全沒有主觀判斷進行的投資行為。然而對一般投資者而言，眾多的技術指標複雜程度遠超過人腦所能負擔，且進行策略交易必須具備相當充足的金融市場相關知識與程式撰寫能力。本研究為降低投資者進入策略交易領域的門檻，實作一投資平台能同時滿足眾多的技術指標計算、在多人使用的環境，使用支撐向量機協助使用者產生投資策略，其結合雲端運算可隨機擴充的分散式資料庫管理系統(HBase) 、具高度伸縮性的應用程式管理架構(JMX) 及訊息交換中介軟體(JMS)，並使用企業級伺服器管理平台(JBoss)實現該投資平台平行運算節點的佈署與相關管理策略。

關鍵詞：平行運算、JMX、JMS、JBoss

目錄

圖目錄

表目錄

第一章　緒論

第一節 研究背景與目的

證券投資市場在台灣一直是熱門的金融投資管道。大多數投資人都希望能從其中獲得超額的報酬。曾永政(2012)認為，在台灣，期貨的流通性比股票好，波動幅度也較大，對於向下操作的限制也較寬鬆。且由於股票當沖現股數目少，當沖融資所伴隨的手續費及利息，使股票市場的進出門檻相較期貨高，加上其波動幅度小等因素，使股票獲利難超越期貨。David Aronson(2006)認為在投資市場上所有的分析方法包括主觀分析與可量化分析，但如果不能透過重複執行來驗證其效力的話，根本稱不上是知識，毫無價值可言。而運用可量化的分析方法，係純粹科學的客觀觀察與統計推論，完全沒有任何人為詮釋的空間，只有單純的買進、賣出、續抱及空手等行為，即稱為策略交易。但這些分析方法需透過電腦快速、反復的運算才能實現，因此也稱程式交易。

策略交易用電腦分析以往的市場運作規律與特徵，期望在未來有相同樣式出現時可以作為投資的依據。程式交易較之人工交易，有以下幾個無法取代的優點(Pardo Robert,2008)：可被驗證(verifiability)、可被計量(quantifiability)、具客觀性(objectivity)、具一致性(consistency)及具延展性(extensibility)。但策略交易需要使用者同時具備金融相關知識與程式設計能力，門檻極高。因此我們希望可以建立一個讓一般投資者也能輕易進行策略交易的投資平台。

眾多的技術指標計算在多人使用的環境下，對於電腦的運算能力要求極高，只有大型主機或超級電腦能滿足此需求；而能執行平行運算的超級電腦價格昂貴且大型主機的平行運算程式設計師難求(Lo, Alfred; Bloor, Chris; Choi, Y K.2000)。幸而隨著科技進步，個人電腦價格下降以及運算能力提升，使用個人工作站來處

理複雜運算工作變得可行具吸引力，所以現行分散式運算傾向使用多部由網路連結的小型電腦來完成工作。

在 IDC 的研究報告中預測 2020 年的資訊成長幅度將是 2009 年的 44 倍 (Gantz & Reinsel，2010)。在未來，資訊的成長量超乎我們所能想像，使用傳統序列的方式處理與分析海量資料變得不可行，唯分散式運算與多處理器才能改善處理速度(Boja, C; Pocovnicu, A; Batagan, L.2012)。分散式系統另一個好處是儲存空間可以動態擴充，使我們不必為了增加硬碟空間而更換新設備(A. Greenberg, J. Hamilton, D. Maltz, and P. Patel.2009)。

另一方面，隨著科技的進步，投資分析工具也產生了變革。過去使用者從電視與報紙上得知股價資訊，到了個人電腦問世與網路的普及化，出現了可以即時揭露資訊的看盤軟體；而電腦的運算能力提升，使得以電腦程式回測與分析策略的程式交易可行性提高(姜林杰祐，2009)。

本研究希望結合雲端運算可隨機擴充的分散式資料庫管理系統(HBase) 、具高度伸縮性的應用程式管理架構(JMX) 及訊息交換中介軟體(JMS)，並使用企業級伺服器管理平台(JBoss)實現平行運算，建立一個滿足大量使用者同時使用的策略產生、挑選、產生買賣訊號與即時下單的期貨模擬交易平台。

第二章文獻探討

第一節　平行運算

平行運算係指同時使用多台電腦資源(處理器)解決計算問題的過程，其設計的目的係為了要提高運算效能，能更快地解決問題並且充分利用電腦資源的一種計算方法。它最初用於解決科學領域的問題，例如天氣預測、地質探勘等需要複雜數學的計算。再來慢慢的運用到實務實驗上，例如計算汽車空氣阻力係數；最後開始有人運用到金融領域的計算。

平行運算可分為時間平行與空間平行。時間平行亦稱為流水線技術(pipeline)，為一種程序執行十多條指令重疊進行操作的平行運算，例如在 CPU 中將多個不同功能的電路單元組成一條指令處理流水線，之後將一條 X86 指令分成數個步驟後由該電路單元分別執行，如此即能實現在一個 CPU 時脈週期完成一條指令，提高 CPU 的運算速度；而目前主流的研究方向則為空間的平行運算問題。根據 Michael Flynn 的說法，空間平行運算問題可分為指令(Instruction)與資料流(Data)兩個維度，其所提出的費林分類法(Flynn's taxonomy)係平行運算中常見的指令運行分類方法(Thomas Rauber, Gudula R¨unger, 2007)。指令和資料流的組合可分成四個象限：

1.單指令單資料流(SISD)：一個處理單元取得單一的程式與儲存空間，典型的例子為 von Neumann model。

2.單指令多資料流(SIMD)：多個處理單元每個有獨立的記憶體存取空間，處理單元從資料存取處取得相同的資料及各自獨立的程序儲存區取得指令，可能係不同的指令平行處理，這種執行模式是十分嚴格的，目前現實世界中不存在此種運行方式的商用平行處理電腦。

3.多指令單資料流(MISD)：有多個處理單元，每個都具有一個獨立的資料存取空間

(共享或分散式的)，即指令同步且平行地被應用到處理不同的資料上。

4.多指令多資料流(MIMD)：多個處理單元都具有單獨的指令及程序儲存空間及資料存取空間(共享或分散式的)，每個處理單元載入獨立的指令及一筆單獨適用於該指令的資料元素，並將結果存回資料存取空間。多核心處理器或集群系統即為MIMD 模型的例子。

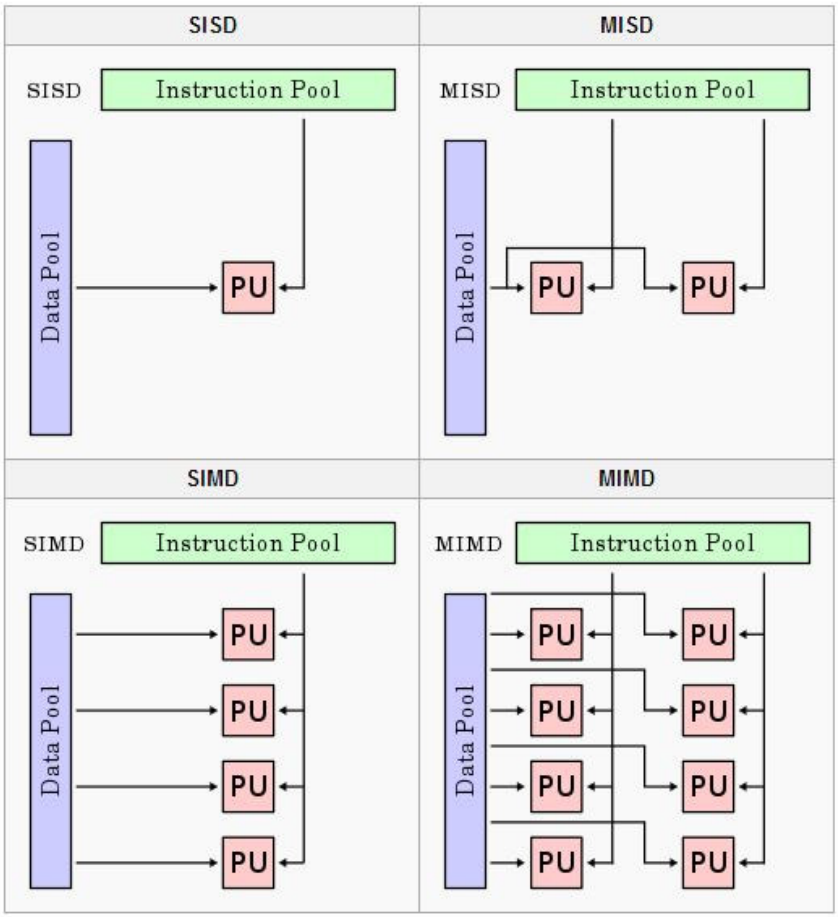

圖 1.Flynn's taxonomy

(資料來源: Wikipedia, 2013)

本研究所提出的策略挑選與期貨模擬交易平台，為求同一時間能夠快速提供給大量投資者使用，將為數眾多的投資者請求平均分散到多部主機上，同時進行股票市場多個標的交易資料的運算，係屬於平行運算中MIMD的應用。

平行運算在通訊上分成兩種方式：內存共享(shared memory)以及分散式記憶體

(distributed memory)。內存共享係指多個處理單位共用一個address space，藉由共享記憶體位置內的資料，讓不同的平行運算單元進行通訊。而分散式記憶體則讓每個處理單位有獨立的local address space，讓資料透過訊息交換的方式在不同運算單元間同步。內存共享的優點可以讓平行運算訊息傳遞速度極快、效率高，但伴隨的缺點係記憶體需要極高的同步以及互斥鎖定機制；而分散式記憶體雖然訊息傳輸速度稍慢，但訊息可以非同步傳輸並讓記憶體的配置不必限於同一個地理區域，能彈性的增減，因此本研究採用後者做為本平台之通訊方式。

第二節 HBase

HBase 係一個非關聯式(non-relational)、欄導向(column-based)、開放碼(open-source)的分散式資料庫，建構於 HDFS(Hadoop Distributed File System)。其特點是可以伸縮，適用於大量資料讀寫與隨機存取。HBase 專案起源於 2006 年，由 Chad Walters 與 Jim Kellermen 創立。概念模仿 Google Bigtable—一個分散式結構化資料儲存系統。在 HBase 中，區域(region)代表的是一個資料表的水平分割。Hmaster 負責分配區域給已經註冊的 regionserver、監控 regionserver 的運作情形、維護 regionserver 的負載平衡以及資料表結構管理等工作。regionserver 負責儲存零到多個區域並處理客戶端的讀寫操作，同時 regionserver 也會定期向 H master 回報自身之運作情形。此專案的成立目標是將數台機器集結成一虛擬主機以儲存數以億計行的資料表。

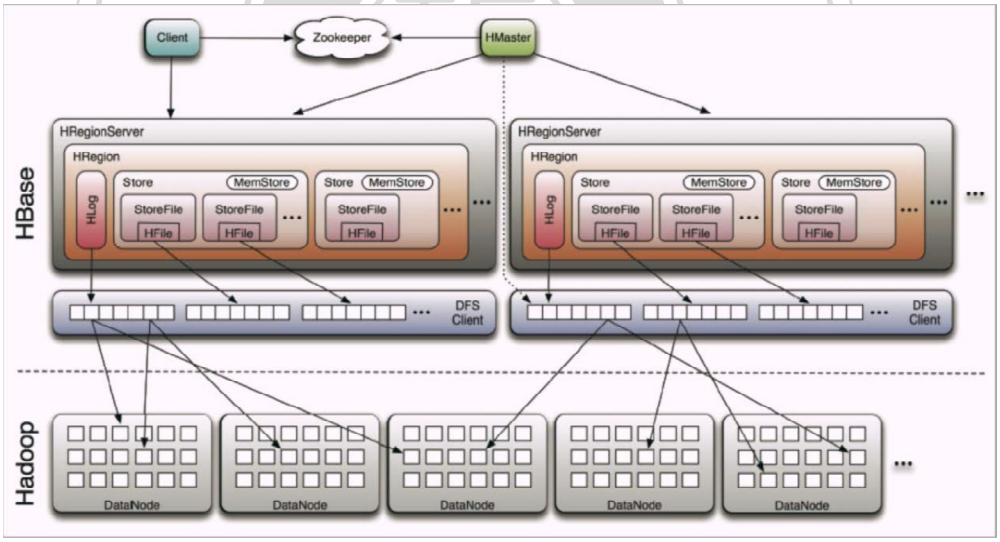

圖2.HBase架構圖

(資料來源：James Chin, Zikai Wang, 2011)

Ian Thomas Varley(2009)提到，關於關聯式與非關聯式資料庫並沒有絕對的優劣，而是多了種選擇。關聯式資料庫預先設定資料框架及嚴謹的綱要(schema)，在處理小量資料實有極高的效率；但面對處理的海量資料時，資料庫需要具備擴展性(scalability)，普遍的共識會避開採用關聯式資料庫而選擇用 key/value 的格式儲

存資料，即非關聯式資料庫。

本研究所接收的即時市場報價資訊，其內容包括時間、區域、市場、標的、時間粒度、開盤價、最高價、收盤價、最低價、成交量、成交價等。HBase 會將單個時間點上的所有資料儲存為一筆 row，平均一天產生 42 萬餘筆的資料，一年會產生近一億筆資料。面對如此龐大資料，使用傳統的關聯式資料庫需要極高的成本(George, L，2011)，且搜尋時間長，採用非關聯式資料庫才能有效處理；而 HBase 以序列化的方式儲存資料，使讀寫速度大幅減少。加上硬配置可伸縮、隨需擴充的特性，使系統未來即使面臨無法預測的使用者數量劇增，仍可以視需求擴充記憶體，輕鬆解決本系統對於大量使用者對海量資料進行讀寫的需求。

第三節 JMX

JMX(Java Management Extensions)係一個 Java 應用程式與系統資源(伺服器程式、硬體設備)的管理架構，它的出現主要係解決以下三個問題(J. Steven Perry，2002)：

1.如何讓我的資源變得可被管理的。

2.可被管理後，這些資源如何可被取得及被看見。

3.應用程式又如何取得這些資源。

JMX 主要可以分成三個層次：

1. Instrumentation level：這層定義四種(standard、dynamic、model、open)應用程式及系統資源變得可被管理(使資源變成 MBeans)的類型，及 MBeans 之間接收和發送通知(notifications)的方法。
2. Agent level：該層包含註冊及新增、刪除等管理 MBeans 的方法。負責管理 MBeans 的單元叫做 MBean server，它本身也是一個 MBean，所以也可以被管理。而一個 MBean sever 加上在它底下所註冊的 MBeans 及在任何一個 Java Virtual Machine(JVM)中運行的 agent service 所產生的集合我們稱之為 JMX agent。
3. Distributed services level：此層次包含 JMX agent 連接到管理 MBeans 的應用程式的中介軟體。該中介軟體可分為兩部分：protocol connector 及 protocol adaptor。透過 protocol adaptor，外部的應用程式(例如一個網頁)可以連接到一個或多個 JMX agent 進行 MBeans 的管理。而 protocol connector 主要是內部 JMX agent 和在其下註冊的 MBeans 連接傳遞訊息的管道。

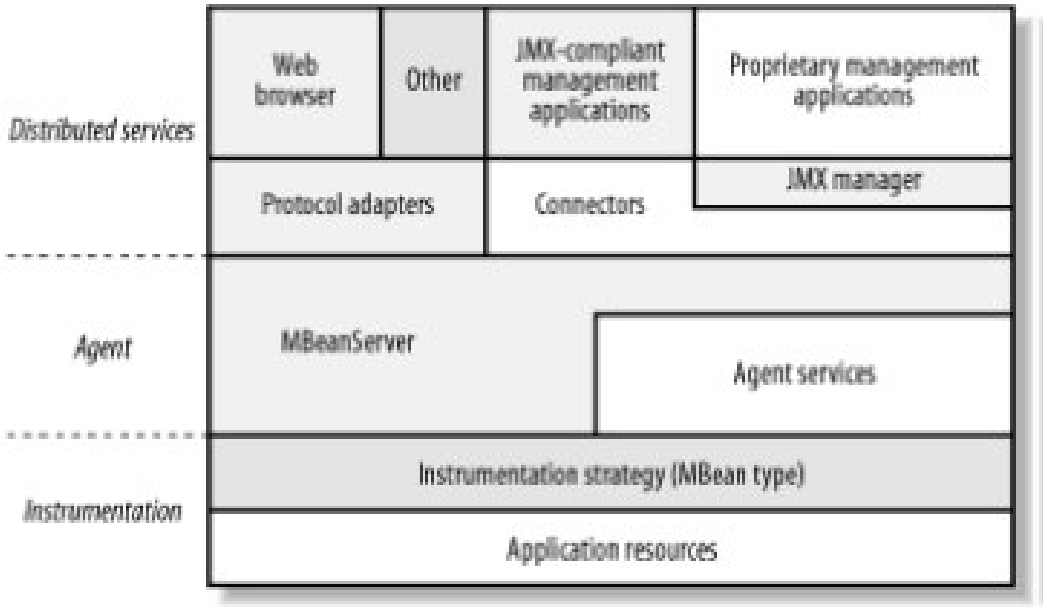

圖 3.JMX 架構圖

(資料來源: J. Steven Perry, 2002)

管理一個企業級的應用程式，早先的做法，管理者必須使用許多額外的套件，管理的環境是困難且複雜的。而JMX的管理架構讓程式開發者可以十分容易的將硬體或軟體的資源封裝成Java物件並配置在軟硬體資源分散的環境底下。(Ben G. Sullins, Mark B. Whipple，2003) 本研究實作的系統建置目標為大量使用者同時進行大量平行運算，運算的效能是極大的挑戰，使用JMX做為系統的管理架構係一個容易實現又成本較低的解決方案。

第四節 JMS

JMS(Java Message Service)為 Java 平台上訊息中介軟體的技術規範，係一個用於在兩個或多個 Clients 之間傳遞訊息的通訊介面，JMS 可讓分散式的應用程式之間做到鬆散耦合 (Loosely Coupled)、具可靠性(Reliable)、以及非同步(Asynchronous)的通訊。 JMS 訊息服務中，主要的三個角色(Mark Richards, Richard Monson-Haefel & David A. Chappell，2009)：

1. JMS Provider(MOM 中介軟體)：作為訊息交換的通道。

2. JMS Producer/Publisher：訊息生產者

3. JMS Consumer/Subscriber： 訊息消費者

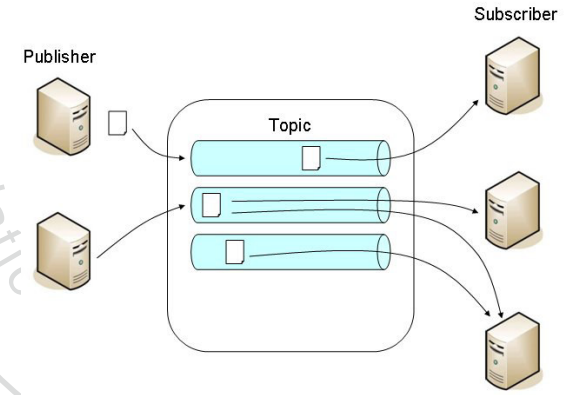

圖 4.JMS 概念圖

(資料來源: Mark Richards, Richard Monson-Haefel & David A. Chappell, 2009)

JMS依照訊息傳送模式又可分為發佈訂閱模式(publish & subscribe)與點對點模式(point to point)，訊息的通道分別稱為主題(topic)與佇列(queue)，差別在於主題只有訂閱該主題的接收端能收到訊息，而佇列則能讓所有接收端都收到。

在平行運算中，節點與結點之間聯絡、資料交換與同步的工作相當耗費時間與資源，因此我們利用JMS鬆散耦合的特性，讓節點之間不用知道彼此的存在也可

以交換訊息。非同步的功能則是讓訊息暫存在主題(Topic)中，需要的應用程序再去所訂閱的主題中取得訊息，避免產生某些應用程序等待訊息，其他程序又在等待該應用程序手中所握資源的浪費情形發生。

第五節 JBoss

JBoss為應用程式伺服器平台，目前最新版本JBoss Application Server 7係Java Web Server從複雜和單一的形式轉向更輕便、模組化和敏捷的重大變革。該版本使大型系統開發人員得以重新思考如何開發和佈署企業級Java應用。JBoss Application Server 7建構於先前的良好基礎上，並提供更出色的性能、更低的內存占用率、分散式的伺服器管理機制(JBoss，2012)。

JBoss Application Server 7新加入domain的觀念，使多台JBoss 應用程式伺服器配置可以集中於一點，統一配置與佈署。有別於以往大家所認知平行運算中的群集(cluster)。以下是JBoss AS 7中所提及的名詞：

1.Server Instance：在實際應用中，一個企業級系統可以由多個Application Server、Web Server和Database Server組成，應用程序代碼可以分布在這多個Application Server上。這些Application Servers之後我們稱之為Server Instances。

2.Cluster：多個Server Instances 同時運行並一起工作。構成Cluster的Server Instances可以運行在一台機器或多台機器上。Cluster最主要功能提供高擴展性(隨需求增減運算資源)及可用性(隱藏內部故障的伺服器)。在JBoss AS 7之前的版本以Cluster稱之，到了7之後，為了與Domain做區別改稱為Server Group。

3.Domain：Domain係一組相互關聯的Server Instances 當成一個單元來管理。讓管理者從單一的控制點配置與部署多個Server Instances。 Domain的範圍包含了應用程式及其所需要的資源與服務。一個Domain可包含0個、1個或多個Clusters。

4.Host：一台實體的機器。

5.Host Controller：負責自己主機上Server Instances的生命週期(如啟動與停止) 的程序。

6.Domain Controller：一個被指派管理整個Domain的Host Controller。確保所有Host Controller都可以獲得目前的配置資訊，協調Host Controllers 間的工作，及Host Controller底下的Server Instances遵守目前的配置及管理策略。

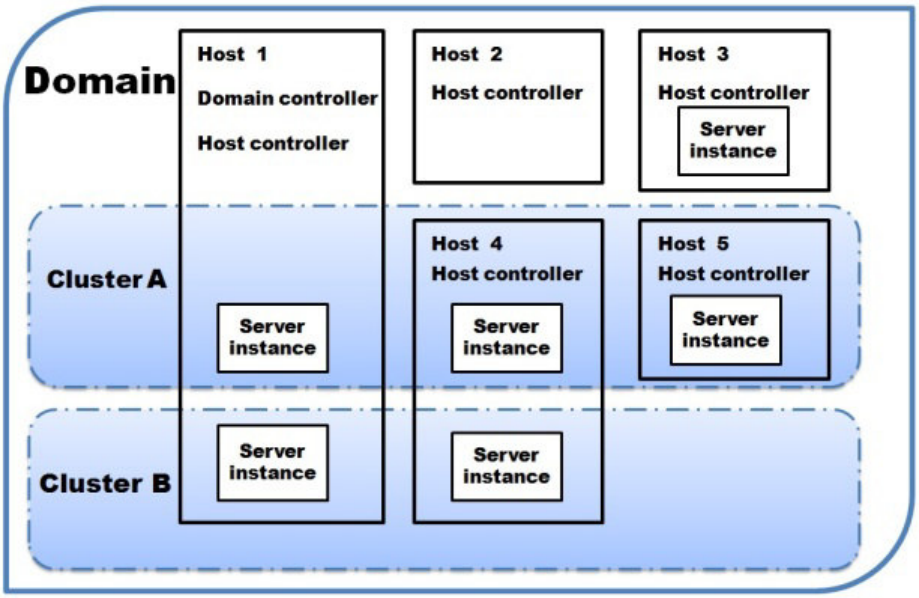

圖 5.JBoss AS 7 關係圖

(資料來源：本研究整理)

	cluster	domain
定義	多個server instances 同時運行並一起工作	一組相互關聯的server instances 當成一個單元來管理
範圍	運行在一台或多台機器上的server instances	一個admin server 及多個managed server instances，和application components及這些component所需資源與服務。
包含	一個cluster	可以是多個clusters(clustered or non-clustered server instances or hybrid)
強調	運算資源的高可用性及擴展性	跨越多台主機的集中管理政策(centralized administration and management policies)
邏輯上的觀點	從客戶端看來是單一個server instance(hides its internal servers from the clients)	在伺服器端即視為單一個server instance

表 1.Domain 與 Cluster 比較表

(資料來源：本研究整理)

當需要在新增 Serve Instance 時，可以到 JBoss 的管理介面，如下圖：

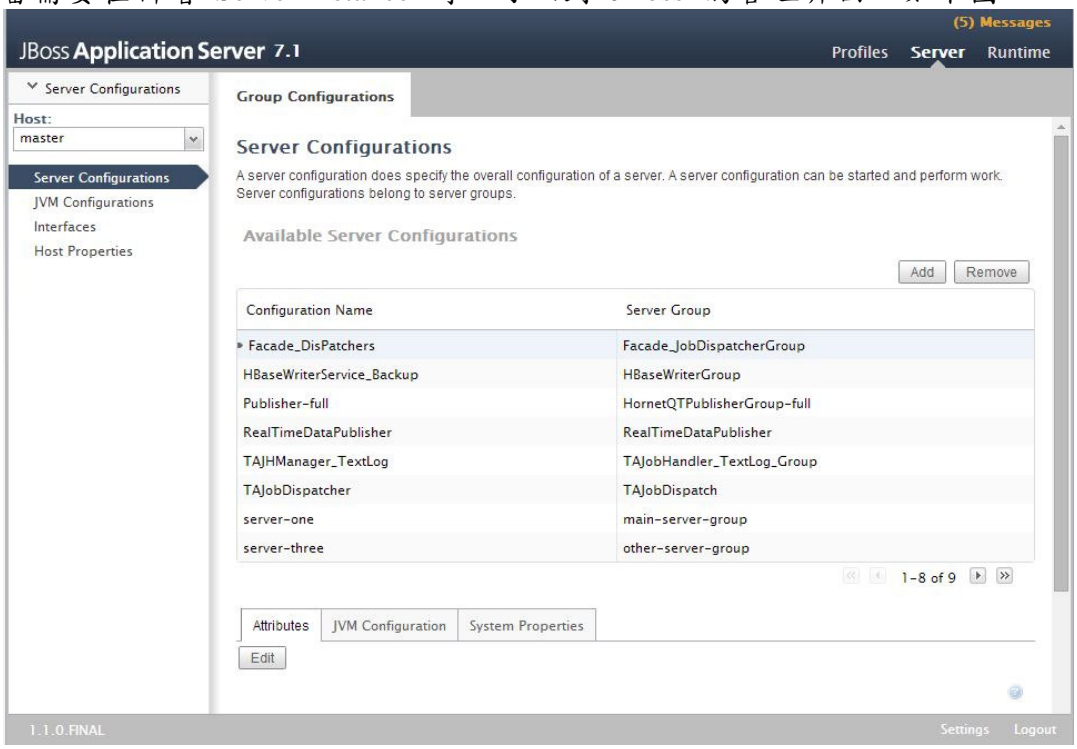

圖 6.JBoss AS 7 管理介面圖

(資料來源：本研究整理)

選擇右上角中間之 Server 選項，點選 Add 後，產生 Server Instance 建立相關資訊，使用者需要輸入該 Server Instance 名稱、其所屬之 Server Group 以及 Port 的位移值等相關資訊。

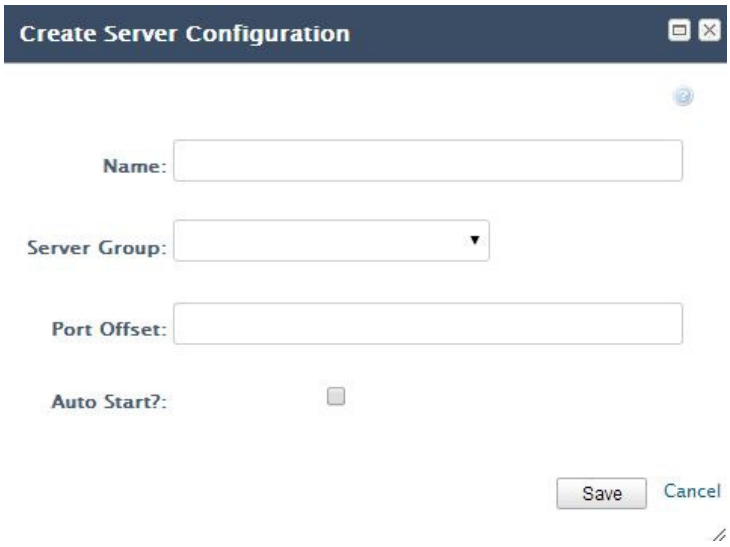

圖 7. Server instance 新增示意圖

(資料來源：本研究整理)

第六節 技術分析

技術分析(Technical Analysis)用在金融商品投資，藉由市場供需的作用來預測股價(指數)未來的走勢。技術分析者認為，在現實的金融環境下，效率市場假說(Efficient Market Hypothesis)不會成立，並認為依據過去股價(指數)變化的歷史軌跡，可以整理出某些標準型態，而未來股價(指數)的走勢必然依照這些既定型態重複出現。

關於技術分析，John Magee提出一下幾點原則(Robert D.Edwards, John Magee, W.H.C.Bassetti, 2007)：

1.股票(指數)價格是由供給和需求雙方互相作用決定。

2.供需受到多種理性和非理性影響。

3.市場小波動可忽略，因為股票價格長期而言是保持一種趨勢。

4.趨勢的變化是由供給和需求關係改變所反應。

5.歷史趨勢會一再重演，投資者可以利用過去價格變動趨勢，來預測價格未來趨勢。

6.股票(指數)價格反應市場中一切的消息，包括：基本面、消息面和心理面。

技術指標是利用統計、計量方法等數理計算模型，將市場上描述市場行為的各種公開資料，轉換成具有分析未來走勢意義的參考數值，並能產生買賣訊號，提供投資人判斷買賣股票時機的一種工具。本研究中採用了移動平均（Moving Average）、乖離率（BIAS）、指數平滑異同平均（MACD）、隨機指標（Stochastic Oscillator， KD Line）、威廉指標（WMS%R）、趨向指標（Directional Movement Index，DMI）、動量指標（Momentum Index，MTM）、震盪量指標（Oscillator，OSC）及相對強弱指標（Relative Strength Index）、中間意願指標(CR)與 KBar Patterns 等十一種常見的指標訊號來進行交易策略及投資標的組合的評估

第七節 支撐向量機(Support Vector Machine)

支撐向量機 (Support Vector Machines, SVM)是一種分類(Classification)演算法，由Vapnik及其他合作者在1992年根據統計學習理論提出的一種新的監測式學習(supervised learning)方法(William H., Teukolsky, Saul A., Vetterling, William T., Flannery, B. P. , 2007)。SVM與傳統機器學習技術的差別在於，它可處理線性與非線性問題，並且不會受到資料量大小之限制，能從有限訓練樣本得到決策規則對獨立的測試集仍能夠得到較小的誤差，而不需事先提供充足的資訊範圍。

SVM 的主要概念係針對訓練資料集，利用定義的特徵值，以數學的方式訓練函數，計算出一個最理想的超平面(hyper-plane)，透過此超平面分類測試資料判斷其準確率，當準確率超過一標準值且具意義時，即可分類新的未知資料，將所有欲分類的資料快速分類至正確的類別(胡翠峰, 2004)。

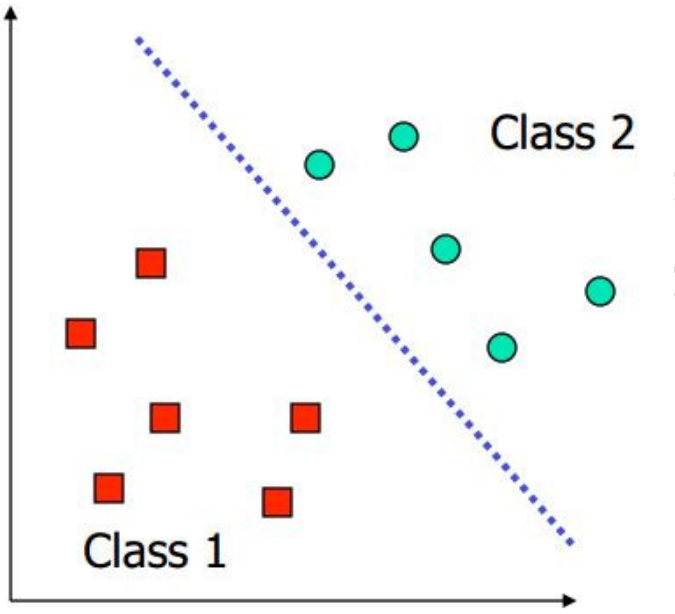

圖 8.支撐向量機示意圖

(資料來源：Martin Law, 2011)

本研究將所有 TA 視為一個集合 set(TA1、TA2、TA3…TAn)，並透過訓練得來的 model 將此集合之成員在超平面下進行分類，以決定目前的投資動作，如買進、觀望、賣出、續抱等。採用支撐向量機做為運算交易策略績效的演算法，以每日交易資料進行多次訓練及測試，產生之策略已有 97%指示投資者應做多而實際也為多頭市場之準確率，20%指示投資者做多但實際應做空之機率，測試結果顯示採用支撐向量機做為本研究策略之演算法可行。

第三章研究流程與系統架構

第一節 研究流程

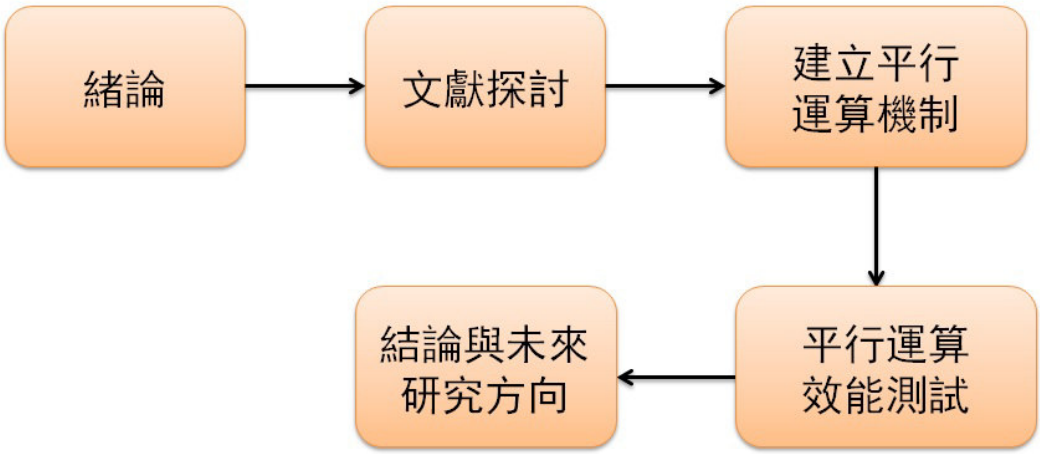

圖 9.研究流程圖

(資料來源：本研究整理)

步驟一：緒論。

說明本研究的目的與使用平行運算建置的原因。

步驟二：文獻探討。

整理建構基於平行運算、HBase 雲端運算資料庫、JMX 管理架構、JBoss 企業級應用程式伺服器平台與 JMS 訊息交換中介軟體之策略挑選與模擬交易平台所需之各項技術與理論。

步驟三：建立交易平台的平行運算機制。

基於文獻探討中所討論之各項技術與理論，設計出適宜本研究使用之模型與

架構，進而實作開發。

步驟四：平行運算效能測試。

平行計算機制完成後，觀察資源利用率、各工作之平均運算時間等，找出本研究平台交易室的運算量與所需運算資源，進一步利用 Amdahl’s law 得出本平台中之平行運算比例。

步驟五：結論與未來研究方向。

探討本研究的發現、貢獻，對於不足或缺失，提供可行的改善方案及未來進一步的研究方向。

第二節 系統架構

本研究系統平台目標為廣大使用者提供即時複雜的技術分析運算，市場範圍目前囊括台灣股票及期貨。以台股為例，六種時間粒度(一分鐘、五分鐘、三十分鐘、一天、一周、一月)，而期貨的時間粒度則為一秒鐘。台股平均每天會產生五十五萬筆 KBar 原始資料，期貨產生一萬九千多筆資料，一年共產生約一億一千五百萬筆原始資料，處理如此海量資料需要極快速計算能力，以及極大的記憶體空間才能容納與日俱增的資料量。

為了達到以上目標，我們需要能夠快速處理海量資料的資料庫管理系統，可以快速讀寫與隨需擴充；另一方面，由於無法預先知道本平台使用者的數量與運算流量，為了避免在運算流量高峰期系統對使用者服務中斷，在資料邏輯層需要有高容錯性與高可用性，且能負荷大量運算工作的應用程式伺服器，採用雲端運算與平行運算便是為了解決這兩難題。

本研究系統平台前端表示層使用 JavaScript，商業邏輯層使用 JMX(Java Management Extensions)作為管理架構，以 JMS(Java Message Service)為訊息中介軟體，並在 JBoss 企業級應用程式伺服器平台實現平行運算。資料層則使用 HBase 雲端運算分散式資料庫，HBase 則建構於 Hadoop 雲端運算平台之上。

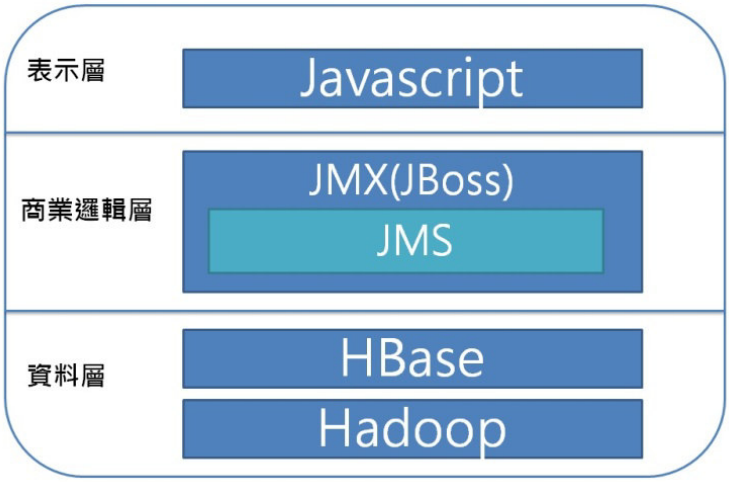

圖 10.平台系統運作架構圖

(資料來源：本研究整理)

本研究所實做系統平台可分為五大模組：分別為規劃室、交易室、下單機、資料存取室以及市場狀態室。

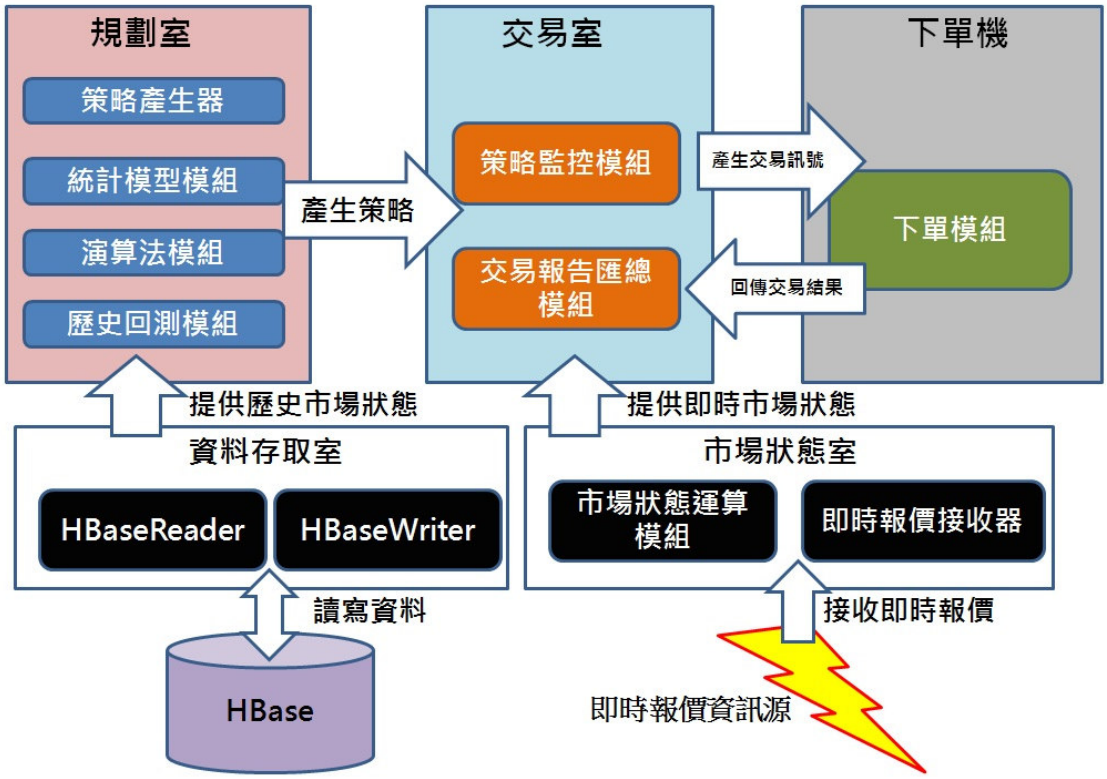

圖 11.模擬交易平台流程圖

(資料來源：本研究整理)

1.規劃室：分為策略產生器、演算法模組、統計模型模組及歷史回測模組。

(1)策略產生器

策略系由一個或多個技術分析值所組成的一組判別式。本研究整理了常用之技術分析指標與K線型態，並依其不同之特性，設計成若干不同之策略。

一個交易策略分為兩大部分－買進訊號與賣出訊號。每組策略具有一個權重

值，當買進（賣出）訊號為真之所有子策略權重值相加之後達到買進（賣出）訊號門檻值，則發生對應的買進（賣出）訊號。

適應函數所代表的是策略之依據，本研究選取了三個指標來衡量交易策略的適應程度：

$$\text{年化報酬率} = \left(\sqrt[N]{\frac{\text{期末資金}}{\text{期初資金}}} - 1\right) \times 100\%$$

$$\text{相對獲利指標} = \sqrt[N]{\left(\frac{\text{期末資金}}{\text{期初資金}}\right) / \left(\frac{\text{期末資金}}{\text{期初資金}}\right)}$$

$$\text{人性化指標} = \left(\frac{\text{平均獲利金額}}{\text{平均虧損金額}}\right) - \left(\frac{\text{虧損交易筆數}}{\text{獲利交易筆數}}\right)$$

回測後，系統將評估每組策略的適應值，適應值越大表示該組策略越有機會被保留。

若使用者滿意在策略產生器產生之交易策略的獲利水準，皆可儲存成為交易室中使用者專屬的投資策略。

(2)歷史回測模組

由於先前產生的策略可能會隨時間失效，使用者可以利用策略績效回測功能，在選定某段過去的時間區間內，進行單一標的單一策略進行回測模擬，或是多標的多策略的交叉回測模擬，以驗證策略的獲利能力。

(3)演算法模組

目前本系統使用的演算法為支撐向量機，並預留空間，當未來有更多人工智慧演算法加入本系統時，讓使用者可以自行選擇偏好的演算，產生交易策略。

(4)統計模型模組

除了演算法模組，本系統也預留未來增加其他統計模型模組的彈性空間，供系統使用者選擇。

2.交易室模組：分為策略監控模組及交易報告匯總模組。

(1)策略監控模組

利用先前在規劃室產生的交易策略，並接收市場狀態運算模組產生的即時市場狀態，對特定的投資標的產生買賣訊號，供投資者參考。

(2)交易報告彙總模組

當使用者在策略監控模組觀察到交易訊息產生後，使用者可以選擇下單，交易報告匯總模總會將使用者的下單請求送至外部下單機，並將交易結果回傳給使用者。若是進行期貨投資，交易報告彙總模組會自行下單，並回傳結果。

3.下單機

(1)下單模組

與券商介接的下單功能，供投資者在交易室觀察產生買賣訊號的特定標的並下單。下單功能的設定根據使用者投資的市場而有所不同。股票市場需使用者手動確認才能下單，而期貨市場為求效率則為自動下單。

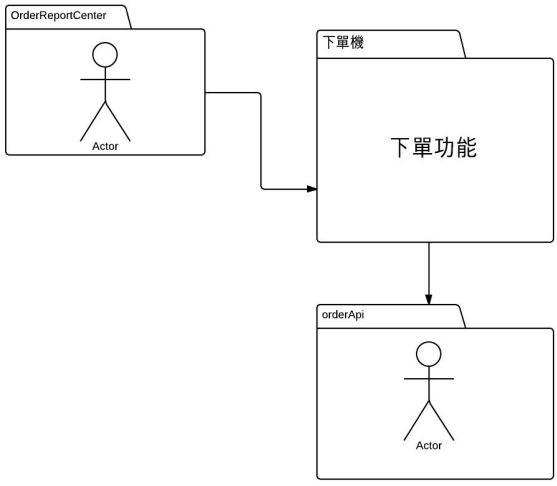

圖 12.交易室 use case diagram

(資料來源：本研究整理)

4.資料存取室：分為 HBase Reader 與 HBase Writer。

(1)HBase Reader

當規劃室需要歷史股價做為產生交易策略和歷史回測所需的輸入資料時，HBase Reader 會從 HBase 分散式雲端資料庫中讀取所需標的 KBar 資料以及經過技術指標計算的買賣規則。

(2)HBase Writer

HBase Writer 主要的功能是將資料寫進 HBase 資料庫。將使用者在規劃室產生的策略存回資料庫；另一方面，也提供市場即時狀態非同步地寫入資料庫。市場狀態運算模組將運算結果放入 JMS 通道，HBase Writer 會訂閱相關的 JMS Topics，將市場狀態資料自 JMS 移至 HBase。如此一來，市場狀態運算模組不用直接參與資料庫的輸入輸出，可以專心處理大量即時產生的市場報價。

5.市場狀態室：分為即時報價接收器以及市場狀態運算模組。

(1)即時報價接收器

即時報價接收器接收即時產生的市場報價，資料單位為 Tick，亦即每有一筆交易產生，就會產生一筆資料。市場狀態運算模組接收到來自期交所及證交所產生的市場即時報價後，轉換成不同時間粒度的 KBar 資料。以台股為例，會轉換成一分鐘、五分鐘、三十分鐘、一天、一周及一月等。期貨則是 1 秒、1 分鐘、10 分鐘。

(1) 市場狀態運算模組

市場狀態運算模組的工作是計算各個技術指標產生買賣規則。自 Java Messaging Service(JMS)訊息交換通道中接收經過即時報價接收器轉換成各個時間粒度的 KBar 資料，進行技術指標的運算，產生無數條買賣規則。這些買賣規則會

重新丟進 JMS 訊息交換通道，讓 HBase Writer 將其寫入資料庫。本研究總共使用 11 種技術指標，分別為 K 線樣式(KBar Patterns)、隨機指標(KD)、移動平均線(MA)、動量指標(MTM)、震盪指標(OSC)、相對強弱指標(RSI)、威廉指標(WMS%R)、中間意願指標(CR)、人氣指標(AR)以及平滑移動平均線(EMA)。

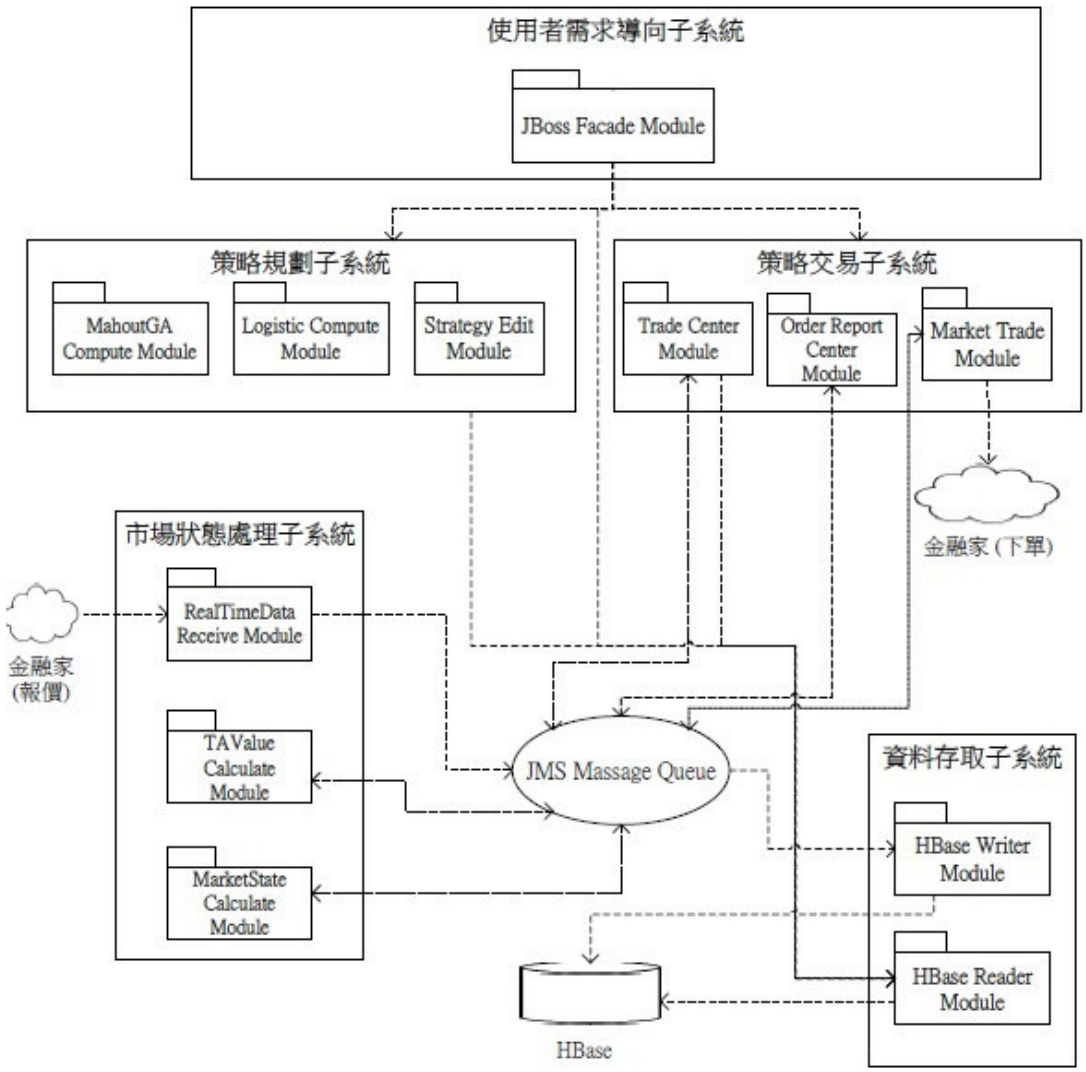

圖 13.模擬交易平台架構圖

(資料來源：本研究整理)

在系統資源配置方面，本研究用六台運算能力相同之四核心主機架設成一群集，指派其中一台主機做為平台的管理中心，即 JBoss 平台中 Domain controller 之角色。該主機為接收使用者服務請求及與資料庫讀寫之節點，其他節點則專心處理管理中心分派的運算工作，不接受客戶端的服務請求與涉及資料庫讀寫，以求提升平台運算效率，必要時管理中心亦能支援運算工作。

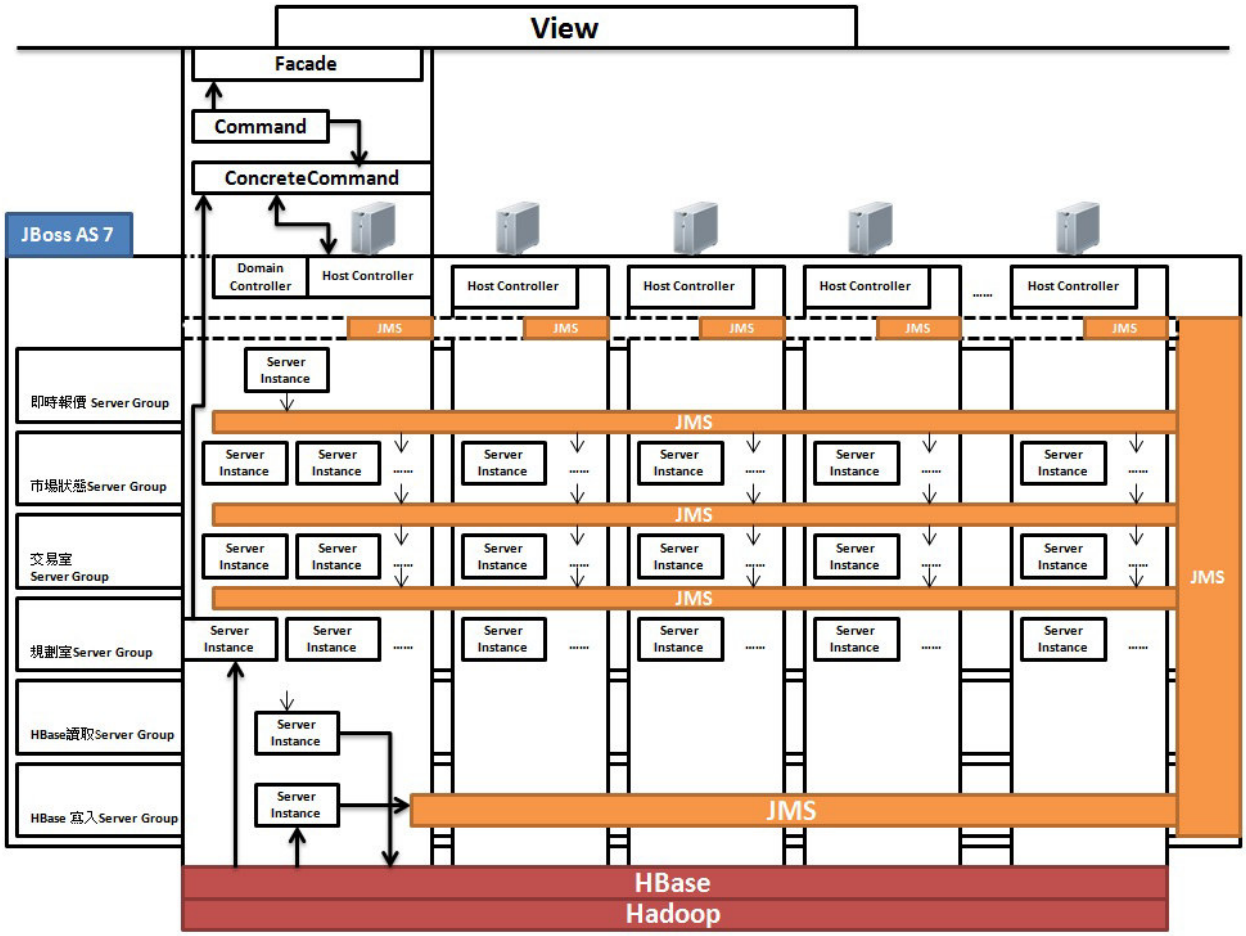

圖 14.模擬交易平台軟硬體資源配置圖

(資料來源：本研究整理)

第三節 平行運算機制

本研究將負責同類型服務的應用程式伺服器實例(Server Instance)放在同一群集(Cluster)或稱為伺服器組(Server Group)，集合所有系統功能成為一個可被管理的域(Domain)。並建立 JMS 群集，該 JMS 群集隸屬在 JBoss 的管理域之下，如此能達到跨硬體分享資訊流同時又能讓不同群集的成員保持鬆散耦合，使各應用程式伺服器僅需記憶同群集成員的資料，節省記憶體空間。而 JMS 的非同步資料傳輸特性可避免彼此間互相等待，使資源利用率提高。

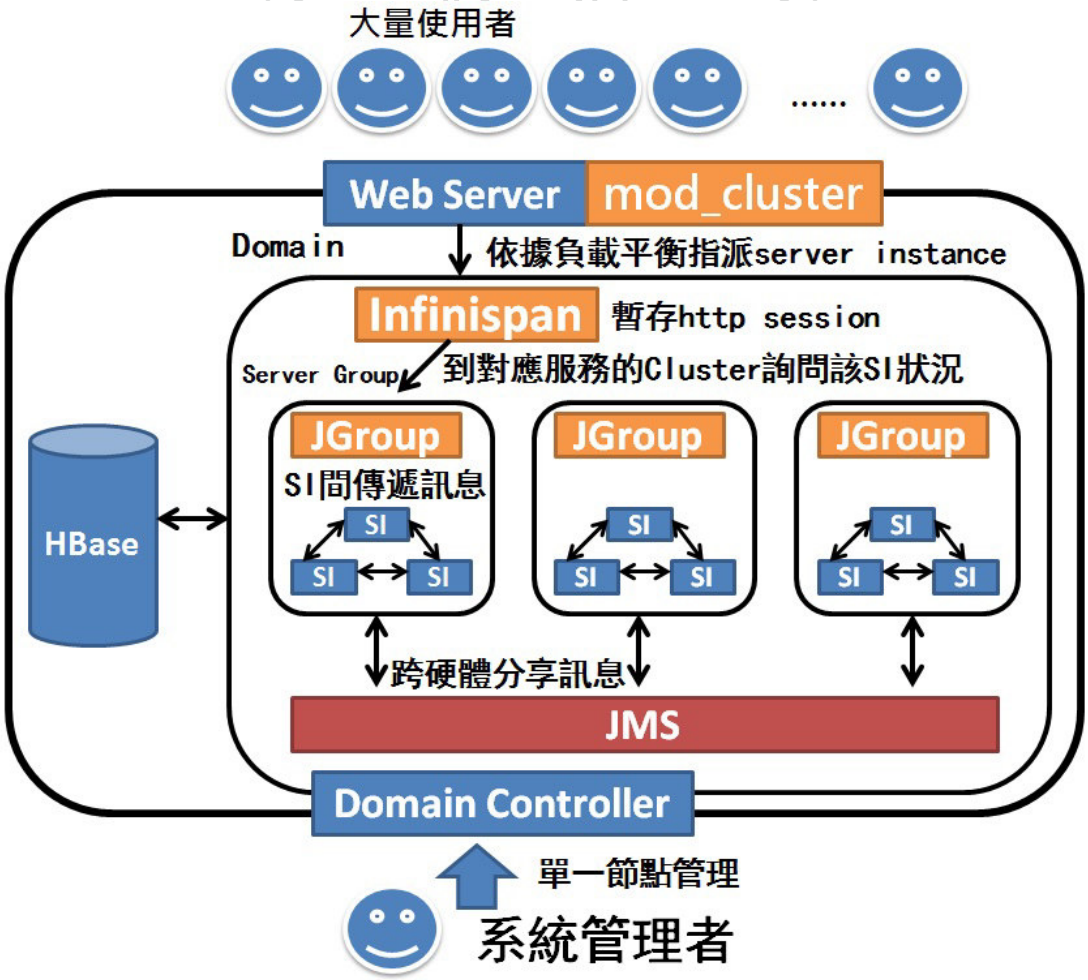

圖 15.平台管理機制圖

(資料來源：本研究整理)

並利用 JBoss 中子系統解決平行運算的三個問題：Mod_cluster 負責負載平衡(load-balancing)、Infinispan 負責系統可用性(availability)、JGroup 負責系統的擴展性(scalability)。

(1) Mod_Cluster

mod_cluster 主要功能為負載平衡。web server 使用 mod_cluster 定時 IP 多點傳送(multicasting)詢問所有應用程式伺服器的狀況，如被使用與存活情形。web server 端會依據回報情況更新名單，依該名單將使用者的請求指派給空閒的應用程式伺服器完成工作。

```
<subsystem xmlns="urn:jboss:domain:modcluster:1.0">
    <mod-cluster-config advertise-socket="modcluster">
        <dynamic-load-provider>
            <load-metric type="busyness"/>
        </dynamic-load-provider>
    </mod-cluster-config>
</subsystem>
```

圖 16.mod_cluster 配置範例圖

(資料來源：本研究整理)

(2)Infinispan

Infinispan 功能為解決系統可用性的問題。Infinispan 本身為依資料快取空間，我們將其作為 Session Container。它主要的工作自 Web Server 取得為使用者提供服務的應用程式伺服器名單後，到各群集中找該群集負責人(Group Coordinator)詢問該應用程式伺服器的情況，若該應用程式伺服器不可被使用，則 Infinispan 會請求 Web Server 重新指派另一台應用程式伺服器為使用者提供服務。

```
<subsystem xmlns="urn:jboss:domain:infinispan:1.4">
  <cache-container name="web" aliases="standard-session-cache"
   default-cache="local-web" module="org.jboss.as.clustering.web.infinispan">
      <local-cache name="local-web" batching="true">
          <file-store passivation="false" purge="false"/>
      </local-cache>
  </cache-container>
</subsystem>
```

圖 17.Infinispan 配置範例圖

(資料來源：本研究整理)

(3)JGroup

JGroup 是為了讓運算資源更具擴展性。JGroup 是一個群集內成員彼此聯絡的方式。所有成員透過 IP Multicasting 不斷向其他成員傳達自身狀態。因此一旦應用程式伺服器被使用、被釋放、新加入、被移出群集或當機時，其他成員都會知道。而每個群集都會有個群集負責人(Group Coordinator)，它本身也是一台應用程式伺服器。主要的工作除了處理在該台應用程式伺服器的運算工作外，群集負責人還要匯整在群集中所有成員的存活狀態，負責向 Session Container 傳達成員的狀態。JGroup 的傳遞訊息方式有 TCP 及 UDP，選擇 TCP 傳輸訊息傳遞較可靠，UDP 則犧牲可靠性但傳遞訊息較為快速。本系統平台需要快速傳遞即時訊息即回應結果，故選擇 UDP。

```
<subsystem xmlns="urn:jboss:domain:jgroups:1.1" default-stack="udp">
    <stack name="udp">
        <transport type="UDP" socket-binding="jgroups-udp"/>
        <protocol type="PING"/>
        <protocol type="MERGE3"/>
        <protocol type="FD_SOCK" socket-binding="jgroups-udp-fd"/>
        <protocol type="FD"/>
        <protocol type="VERIFY_SUSPECT"/>
        <protocol type="pbcast.NAKACK"/>
        <protocol type="UNICAST2"/>
        <protocol type="pbcast.STABLE"/>
        <protocol type="pbcast.GMS"/>
        <protocol type="UFC"/>
        <protocol type="MFC"/>
        <protocol type="FRAG2"/>
        <protocol type="RSVP"/>
    </stack>
</subsystem>
```

圖 18.JGroup 配置範例圖

(資料來源：本研究整理)

第四章 實驗數據分析

第一節　速度測試與分析

本章節將實際測試在即時交易環境中，交易室策略監控模組在一台機器與多台機器的測試環境中運算速度的差異。本研究採用Amdahl's law做為平行運算效能的衡量指標。

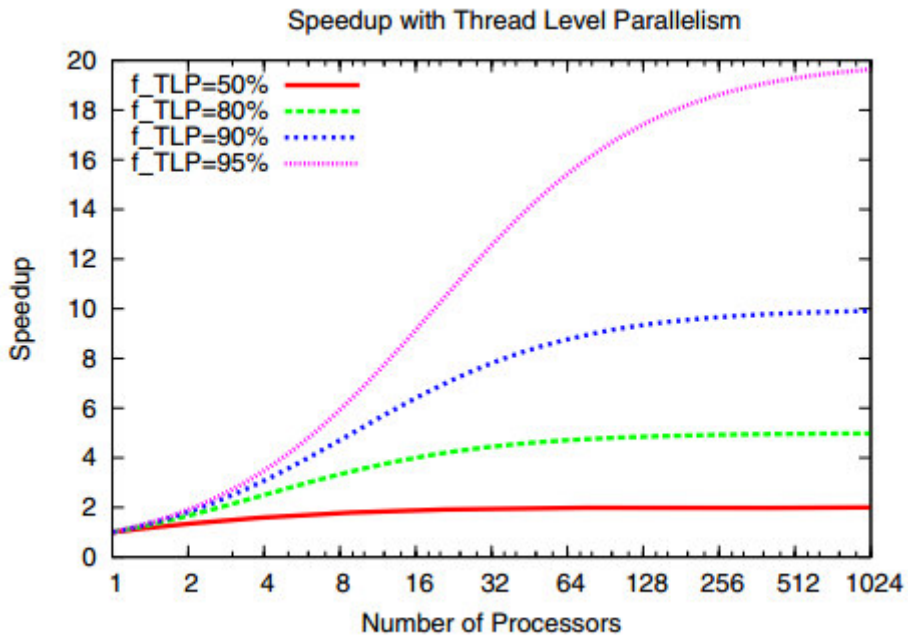

圖 19.Amdahl's law 速度與處理器數量關係圖

（資料來源：H. Shen and F. Pétrot, 2011）

Amdahl's law常用來預測在多處理器的平行運算中理論下的速度最大化，因其代表了處理器平行運算之後效率提升的能力，由於平行運算的速度受限於程式中的僅能序列化處理的片段，因此系統僅有一部分可以經由平行運算改善效率，其餘則無法。其公式為：

$$T(n) = T(1)(B + \frac{1}{n}(1 - B))$$

$n \in \mathbb{N}$ ：表示程式中平行處理的執行緒數量。

$B \in [0,1]$ ：表示程式執行中的無法平行運算的部分。程式的總和為1，B的範圍為0到1。因此，一個可以同時運行n條執行緒的平行運算，其理論的速度最大值為：

$$S(n) = \frac{T(1)}{T(n)} = \frac{T(1)}{T(1)(B + \frac{1}{n}(1 - B))} = \frac{1}{B + \frac{1}{n}(1 - B)}$$

每個程式透過平行運算加速都有其最大極限，即(1 - B)部分，即使規劃再多處理器的平行運算，也無法超越極限，本研究希望找到本平台B的部分，並得出在現實情況下，較為經濟的平行運算處理器群集規模。

本研究實驗次數為7次，測試處理器數的變量分別為單核心(1台處理器)、單點(4台處理器)、群集兩台(8台處理器)、群集三台(12台處理器)、群集四台(16台處理器)、群集五台(20台處理器)、群集六台(24台處理器)，市場資料為8種台灣期貨市場的商品，包括台灣期貨、小型臺指期、台灣50、電子期貨、金融期貨、MSCI臺指期貨、櫃買期貨、非金電期貨，資料的時間粒度為1秒。每次運算皆由11種技術指標平均組成。由於某些技術指標需要用到數根到數百根的KBar資料，或是其他技術指標的運算結果做為輸入資料，在輸入資料不全的情形下，極可能無法計算，所以必須先進行資料清理，去除掉前200筆資料，以免影響實驗數據。每種技術指標的資料隨機抽取100筆，去除極端值，計算其平均運算時間，結果如下：

運算時間/處理器數	Kbar	AR	CR	EMA	KD	MA	MTM	OSC	RSI	WMS %R	BIAS	時間（毫秒）
1	9	441.18	1035	230.22	1482.66	247.86	93.6	337.32	754.56	491.58	880.56	5994.54
4	1	3409.96	6515.54	1839.4	12154.18	1751.38	556.6	2628.92	5695.62	3991.12	6038.82	4458.154
8	1	1828.38	3412.06	1024.52	7057.14	1014.38	297.16	1413.58	3033.08	2289.78	3539.3	2490.938
12	1	1109.54	2082.32	682.9	3393.14	691.14	198.58	944.36	1866.72	1154.66	1957.12	1408.048
16	1	977.74	1507.84	591.26	2510.22	561.1	167.2	862.14	1300.3	915.26	1387.48	1078.954
20	1	657.84	1021.5	474.76	1488.72	483.72	130.2	567.04	918.44	774.14	806.4	732.276
24	1	572.04	710.32	389.22	1077.62	390.34	114.76	509.16	571.26	578.9	527.2	544.082

表2. 單點與叢集模式技術指標運算時間表

（資料來源：本研究整理）

由此表觀察可得之，在單核心模式下技術指標運算的時間，每秒的期貨市場即時報價平均需要近6000毫秒才能完成運算。其中KBar Pattern此種技術指標每次只會用到一到交易規則，因此運算時間在單點與叢集模式皆僅需一毫秒，運算八種期貨市場商品亦僅需9毫秒；而其他十種技術指標在群集模式，隨著處理器數量遞增，運算速度也有顯著的改善。下圖為各技術指標在單點與群集模式運算時間的比較圖：

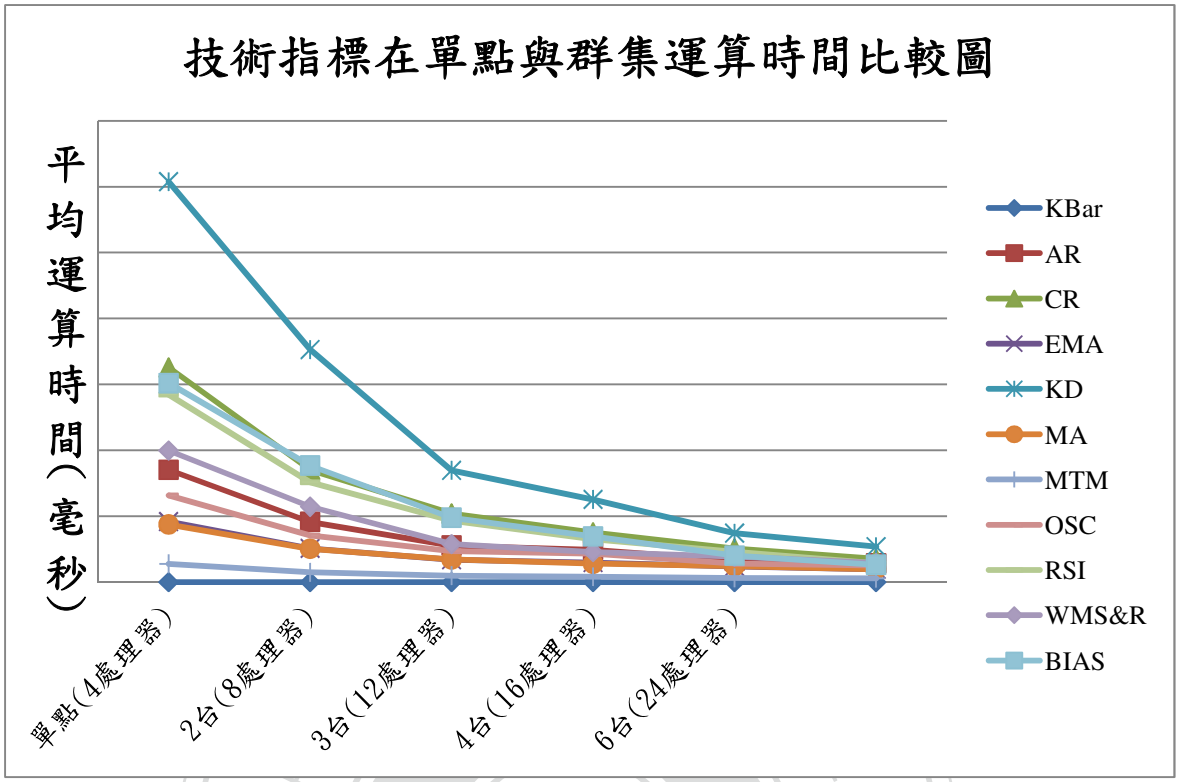

圖20.技術指標在單點與群集模式運算時間比較圖

(資料來源：本研究整理)

觀察此圖可得知，每種技術指標在處理器增加時，提升的運算速度不盡相同，如KBar Pattern因其計算特性在任何模式下，運算速度改善之情形極微；反之計算時間越長之技術指標，如KD，在群集模式下運算時間縮短幅度越顯著；扣除KBar Pattern因在群集模式中任何改善，所有本研究所採用之技術指標，在處理器數量變化下之平均運算時間， 整理為下圖：

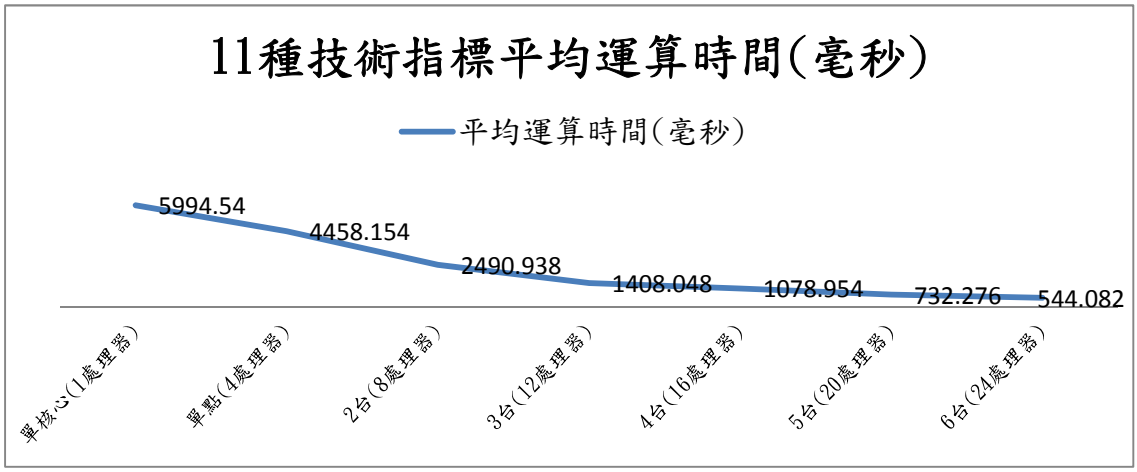

圖21.處理器數量變化下之平均運算時間

(資料來源：本研究整理)

本次測試結果原希冀能找出類似Amdahl's law所述之平均運算速度提升之理論最大值，然而囿於現實與成本考量，本研究所設計之群集僅含六台，畫出之曲線如下圖：

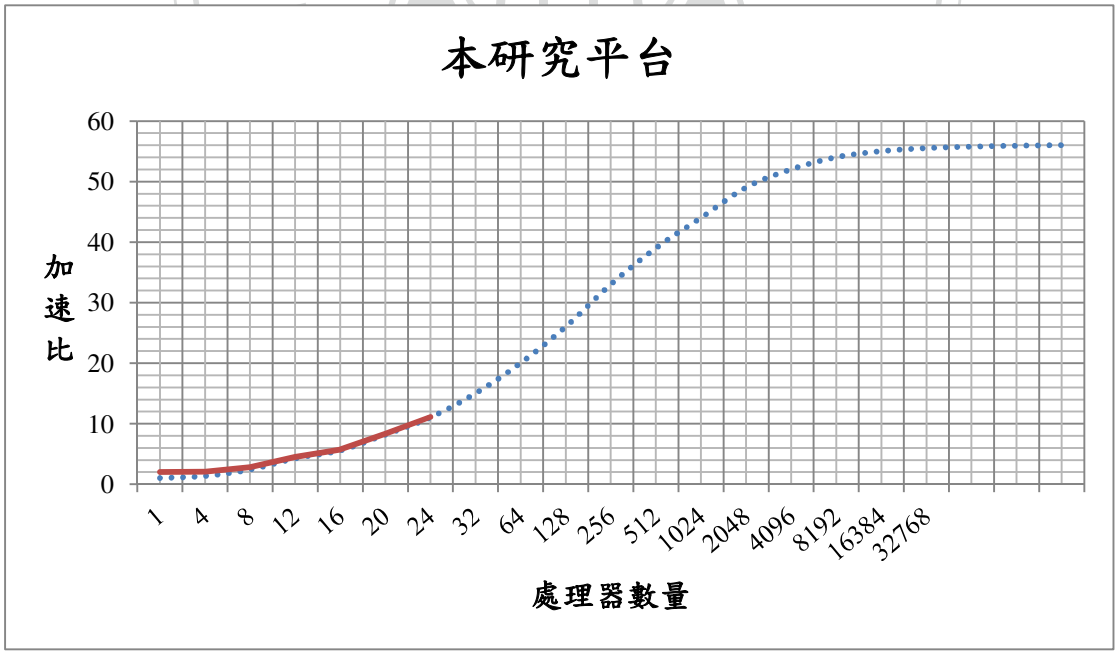

圖22.本研究平台處理器數量與加速比關係圖

(資料來源：本研究整理)

在圖14中觀察，要找到Amdahl's law所述平行運算之理論加速最大值，群集之處理器數量需1000台以上，才能觀察到趨近理論最大值之曲線；本研究在有限資源情況下，24台處理器之叢集環境中計算台灣期貨市場八種商品之市場狀態，較單台處理器得到11倍以上之加速比，足見本研究所設計之平行運算架構極為有效提升運算效能，尤其適合需要大量運算之期貨交易平台、及其他金融商品之交易平台。根據圖17之曲線，此次測試結果亦能得出，如擴大本研究設計之平行運算架構之群集規模，能更有效提升平行運算之效能。

第五章 結論

第一節 研究結論

程式交易的進入門檻很高，投資者需要具備程式設計能力與金融方面專業知識，才有能產生投資策略。而產出的投資策略也須經過不斷的歷史回測，才能保有其獲利能力。所需具備的知識、投入的時間與心力，對一般投資者都是阻力。

本研究所實作投資模擬平台，為了降低投資者使用程式交易的門檻，同時滿足大量投資者使用需求，結合HBase雲端分散式資料庫、JMX應用程式管理架構、JMS訊息交換中介軟體，並使用JBoss企業級伺服器管理平台建立平行運算的管理機制。同時運用支撐向量機協助投資者產生投資策略。本研究所得之測試數據，可得出以下幾點結論：

1. 本研究所佈署之平行運算環境，其運算能力於台灣期貨市場之計算可行(544毫秒)。
2. 本研究所設計之六台4核心主機群集，共24台處理器，因在現實與經濟考量的情況下，無法根據Amdahl's law找出本系統可平行運算之部分的理論最大值以及無法平行運算之部分佔本系統之百分比，但本系統在24台處理器群集平行運算情況下相較於單台處理器系統可加速將近11倍。
3. 依據本研究所設計之平行運算架構實踐於本研究之期貨模擬交易平台，擴大群集規模，可以得到更優良的運算效能。

第二節　未來展望

本研究未來改善的方向可以分成兩部份，第一部份為改善交易策略的績效，第二部份為改善平行運算的機制。

本研究使用支撐向量機產生策略，做為交易平台之演算法可行，但仍有20%指示投資者做多但實際應做空之機率，期待未來能改善其誤判之機率。此外期望未來策略產生的方法可囊括更多人工智慧演算法，例如：類神經網路、決策樹、粒子群聚演算法、螞蟻演算法……等，及其他的統計模型，以期能找到對未來投資市場，面對不同類型、特性的投資標的，能夠產生更準確預測的交易策略。另一方面，由於交易策略皆是由技術指標產生的，屬於技術分析，未來希望可以加入基本面分析，如企業評價，讓使用者可以更容易觀察各企業的體質，以挑選優質穩健的投資標的。

由於本系統著重在於處理即時市場報價資訊並迅速回傳給使用者，為避免記憶體讀取輸出的時間延遲，平行運算的層次在於運算工作的平行分派到不同主機上，再透過跨硬體訊息中介通道彼此交換訊息，而沒有將每個運算工作(Job)拆分成多個任務(Task)。為解決輸入輸出時間過長的問題，未來在訊息中介通道與分散式資料庫中間可以加一層in-memory Database，作為資料暫存之用。同時配合在硬體層級進行平行運算的程式語言，如Scala，以期能改善平行運算的效能。而在JBoss框架底下，server instance需要逐一在管理介面啟動，耗時且成本高，未來希望可以找到能動態開啟server instance的方法。

參考文獻

英文部分

1. George, L(2011), *HBase: The Definitive Guide.* O'Reilly Media, Inc, pp. 5–13.
4. J. Steven Perry(2002), *Java Management Extensions.* O'Reilly Media, Inc, pp. 8–10.
5. Ben G. Sullins, Mark B. Whipple(2003), *JMX in Action*. Manning Publications Co, pp. 8–11.
6. Francesco Marchioni(2011), *JBoss AS 7 Configuration*, Deployment, and Administration. Packt Publishing Ltd, pp. 117–131.
7. Thomas Rauber, Gudula R¨unger(2007), *Parallel Programm*ing:For Multicore and Cluster Systems, Springer-Verlag Berlin Heidelberg, pp. 21–23.
8. Wikipedia(2013), "*Flynn's_taxonomy"*, Retrieved from http://en.wikipedia.org/wiki/Flynn's_taxonomy.
9. JBoss(2012), "*JBoss 7.1 Admin Guide(2012).*" Retrieved from https://docs.jboss.org/author/display/AS71/Admin+Guide .
10. Mark Richards, Richard Monson-Haefel & David A. Chappell(2009), *Java Message Service*. O'Reilly Media, Inc, pp. 9–10.
11. Lo. Alfred, Bloor Chris, Choi. Y K(2000). *Internet Research*10. 2: 160-169.
12. Amza, C. (1996), *Shared memory computing on networks of workstations*, IEEE Computer.
13. J. Gantz and D. Reinsel(2010). *The Digital Universe Decade - Are You Ready?* Technical report, EMC Corporation.
14. A. Greenberg, J. Hamilton, D. Maltz, and P. Patel(2009). *TheCost of a Cloud: Research Problems in Data Center Networks*. ACM SIGCOMM CCR.
15. Pardo, Robert(2008), *The Evaluation and Optimization of Trading Strategies.* John Wiley & Sons, Inc.
16. James Chin, Zikai Wang(2011), *HBase: A Comprehensive Introduction*. Retrieved from http://cs.brown.edu/courses/cs227/archives/2011/slides/mar14-hbase.pdf
17. Aronson, David R(2006)., *Evidence-based Technical Analysis.* John Wiley & Sons Inc
18. Ian Thomas Varley,(2009), *No Relation: The Mixed Blessings of Non-Relational Databases*, Master's Thesis, University of Texas
19. William H., Teukolsky, Saul A., Vetterling, William T., Flannery, B. P. (2007), . *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press

20. Martin Law(2011), *A Simple Introduction to Support Vector Machines*, Department of Computer Science and Engineering Michigan State University
21. H. Shen and F. Pétrot,(2011), *Using Amdahl's Law for Performance Analysis of Many-Core SoC Architectures Based on Functionally Asymmetric Processors,* in Proc. ARCS, 2011, pp.38-49.
22. Robert D.Edwards, John Magee, W.H.C.Bassetti (2007), *Technical Analysis of Stock Trends(9th Edition),* Taylor & Francis Group, LLC

中文部分

1. 黃柏翰(2012)，*基於Hadoop 雲端運算架構建立策略交易與回測模擬平台*，國立政治大學資 訊管理學系碩士學位論文。
2. 楊雅菱(2011)，基於雲端環境與服務導向架構之交易策略評估平台框架，國立政治大學資 訊管理學系碩士學位論文。
3. 姜林杰祐(2009)，*程式交易：觀念,方法,技術與解決方案*，新陸書局。
4. 曾永政(2012)，*期貨程式交易SOP*，聚財資訊：p13-p19。
5. 胡翠峰(2004)，*模糊相關與支援向量學習應用在文件多重分類之研究*，義守大學資訊管理碩士論文。