國立交通大學

資訊科學與工程研究所

碩士論文

棋類教學平台之雲端運算系統

Cloud Computing System for E-Learning Platform of Board Games

研究生：林彥成

指導教授：吳毅成 教授

中華民國 九十九 年九月

棋類教學平台之雲端運算系統

Cloud Computing System for E-Learning Platform of Board Games

研 究 生：林彥成　　Student：Yen-Cheng Lin

指導教授：吳毅成　　Advisor：I-Chen Wu

國立交通大學

資訊科學與工程研究所

碩士論文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

September 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年九月

棋類教學平台之雲端運算系統

研究生：林彥成　　指導教授：吳毅成 教授

國立交通大學資訊科學與工程研究所

摘要

本篇論文開發出一套適用於電腦棋類教學平台的雲端運算系統。以之前實驗室學長所開發的桌機格網系統為基礎，提出一套除了具有原本桌機格網系統就有的分散式運算功能，額外增加了資料儲存與 Map-Reduce 運算功能的系統，並將提出的系統嵌入現有的雲端系統之中，我們分別使用了 GridGain 與 Windows Azure 這兩個雲端平台來實作這個雲端系統。

Cloud Computing System for E-Learning Platform of Board Games

Student: Yen-Cheng Lin Advisor：Yi-Cheng Wu

Institute of Computer Science and Information Engineering

National Chiao Tung University

Abstract

This thesis develops a cloud computing System for an E-learning platform of board games. This system is based on the desktop grid system developed by our laboratory and adds two additional functionalities. The first functionality is to support data storage. The second is to support Map-Reduce computing. We use GridGain and Windows Azure platforms to implement the cloud computing system.

致謝

首先要感謝我的指導教授，吳毅成教授，他是一位非常認真的老師，對於學生總是盡心盡力的指導，只要我們有問題都樂於協助，並且常常提供非常關鍵有用的意見，及時指正我們，避免我們走太多冤枉路。

再來要感謝台灣微軟創新中心。微軟創新中心跟我們的合作計畫，剛好可以將我的論文應用出來。也免費提供 Windows Azure 的課程，讓我得以快速上手 Windows Azure，縮短開發時間。

另外要感謝鈊象電子股份有限公司，在我在學期間提供我豐厚的獎學金，讓我在學期間可以無後顧之憂的專心做研究。

感謝我的家人，無論我做甚麼決定都支持我，也相信我、鼓勵我。在我比較繁忙的時候，疏於跟家裡連絡也都諒解。很感謝您們都默默的付出與支持。

再來也要感謝實驗室的德中、秉宏、宏軒三位博士班學長，總是在我有問題時花時間幫我解決。感謝同學家茵，跟我一起開發系統的好夥伴，也感謝柏廷與忻芸，在我系統或是編輯器有問題時總是找你們兩位，還有駿嶧與郁雯，跟我們一起奮鬥的好同學們。最後感謝學弟尚餘與昱維，你們也幫助了我不少忙。

目錄

圖表目錄

第一章、介紹

本章節將會說明本篇論文的動機以及本篇論文的目的，接著會對雲端教學系統使用到的幾個情境分別做介紹，最後則是說明本論文的大綱。

1.1 研究動機

現今已有許多棋類應用的教學環境，例如圍棋的 LGS 編輯器[6]與六子棋官方網站[3]上面的桔棋教學系統。現有的教學環境中，大多都包含了簡單的規則說明，或是進階的技巧分析等等。但在目前多數的教學系統中卻比較少有提供棋類人工智慧運算、或是提供大量棋譜資料查詢的教學環境。

而雲端系統則是現在相當熱門的話題，它是利用大量的實體機器形成叢集，叢集中各實體機器之間利用網路來溝通、傳輸資料。我們的想法是能否利用雲端系統來加強教學系統，主要額外提供電腦人工智慧運算以及大量棋譜資料存取兩個利用雲端系統特性來加強的功能。

由於電腦棋類人工智慧的運算通常都是利用結點樹展開運算，所以當樹的結點數量很多的時候運算會變得較為複雜也會花費較多時間，但是我們如果將不同的節點分枝以平行化的分給雲端叢集中的不同機器來同時進行運算，是不是就可以相對減少運算所要花費的時間？

而大量的棋譜資料如果也可以放在雲端叢集中的不同機器之中，一來不會佔用單一一台電腦大量的儲存空間，而且在存取資料的時候也可以利用雲端 Map-Reduce 運算來平行存取資料，增加存取速度。利用雲端系統來儲存資料也兼顧了資料備份的問題。

1.2 論文目的

我們的目標是開發適用於棋類教學平台的雲端運算系統，但是我們並沒有打算要從頭自己開發一個全新雲端環境，而主要是利用現有的雲端系統(例如：GridGain[4]、Windows Azure[11])來建構我們的雲端教學環境，並且先以六子棋當作應用實例，作為往後開發其它棋類應用的範例。

1.3 情境

本論文與廖家茵[7]同學共同合作開發六子棋教學系統[12]。為了達到六子棋教學、運算、以及棋譜存取的目的，六子棋前端的教學系統提供以下三種情境：算出最佳著手、算出所有擋法、以及棋譜儲存與搜尋。以下圖片皆取自廖家茵同學之前端系統。

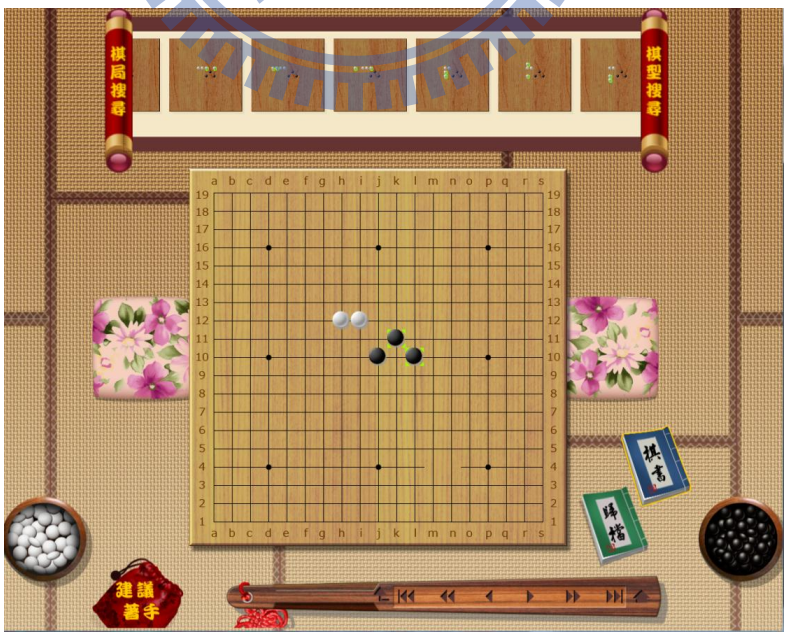
1.3.1 算出最佳著手

透過六子棋人工智慧程式計算出目前盤面上電腦認為最佳的下法。

圖表 1 算出最佳著手

1.3.2 算出所有擋法

透過六子棋人工智慧程式計算出目前盤面是否已經有一方必勝了，如果還沒有任何一方計算出必勝，則回傳所有或部份可能的擋法。



圖表 2 算出所有擋法

1.3.3 棋譜儲存與搜尋

儲存一個六子棋目前的盤面。並且由部份的棋型，或是玩家資訊來從資料庫搜尋所有符合條件的完整盤面。

圖表 3 棋譜搜尋

1.4 論文章節介紹

接下來的第二章我會介紹一些與本論文相關的背景，包括應用的棋類、之前實驗室學長已開發的桌機格網系統、以及目前的一些雲端系統。第三章我會說明我的系統如何配合前面的幾種情境來設計。第四章則是介紹如何利用現有的雲端系統來實作我的棋類教學雲端系統。最後會在第五章會下個結論，以及提出一些未來的發展工作。

第二章、背景

本章節將說明與本篇論文有關的背景介紹，包括棋類教學系統所應用的棋奕，以及目前實驗室已開發的桌機格網系統以及現今一些雲端系統的介紹。

2.1 棋奕應用

以下將說明本篇論文可應用之棋奕。

2.1.1 六子棋

本篇論文主要會以六子棋當作實作的範例，作為其他棋類要實作在本系統上之參考範例。

源起

六子棋是由五子棋變化而來。當時是交通大學吳毅成教授在與女兒下五子棋時，女兒突發奇想的問吳毅成教授說為什麼不能一次下兩顆，激發了吳教授的靈感，由此衍發出六子棋。

六子棋正式發表於 2005 年，是由國立交通大學吳毅成教授所首次發表，現在已經是國際奧林匹亞電腦對局競賽[5]的正式比賽項目中的一項。

規則

六子棋的規則相當簡單：它是由先由黑方下一子，之後雙方輪流各下兩子，直、橫、

斜任一條線先連成六子的一方就獲勝。

而最近有新的演化規則，這是由圍棋參考而來的：輪到其中一方下子時他可以選擇放棄下這一手下子，但是必須要在對方尚未放棄過下子時才可放棄，若盤面下到最後仍然是雙方和局，則有先喊放棄下這一手的那一方獲得最後的勝利。

特色

六子棋屏除了五子棋先手黑方的強烈優勢：五子棋黑方的優勢在於盤面上的子數永遠大於等於白方；而白方會有子數永遠小於等於黑方的劣勢，六子棋是無論黑白雙方下完一手後都會比對方多一顆子，而達到潛在公平的效果。而且六子棋由於一次要下兩顆子的緣故，所以變化複雜，也相對的複雜度相當高，根據 Game-Tree Space 的統計六子棋的複雜度是介於日本將棋到中國象棋之間，是排名第三複雜的棋類。六子棋的特色就是：遊戲公平、規則簡單、變化複雜。

交大六號

接著要介紹一個六子棋的人工智慧程式：NCTU6[13][14][15][16][17]，中文名稱叫做交大六號。NCTU6 是由國立交通大學資訊科學與工程研究所吳毅成教授實驗室所開發的六子棋人工智慧程式。這個程式曾經獲得兩屆國際奧林匹亞電腦對局競賽[5]，六子棋項目的冠軍。他主要提供我們的雲端運算系統兩個主要的功能，輸入一個六子棋盤面可以請求它計算：１.盤面上最佳著手２.此盤面是否已經必勝，若否則回此傳盤面所有可能的擋法。

Little Golem 遊戲網站

最後則是 Little Golem 網站[8]，他是目前一個很大的棋類線上對戰網頁平台，提供玩家在網頁上面彼此對戰各種不同的棋類，包含圍棋、五子棋等等，其中也包括六子棋。

另外這個網站除了提供玩家之間彼此對戰，也會把所有對戰過程的棋譜給記錄下來，提供給一般人下載使用，我們就是由此網站來獲得大量的棋譜資料。

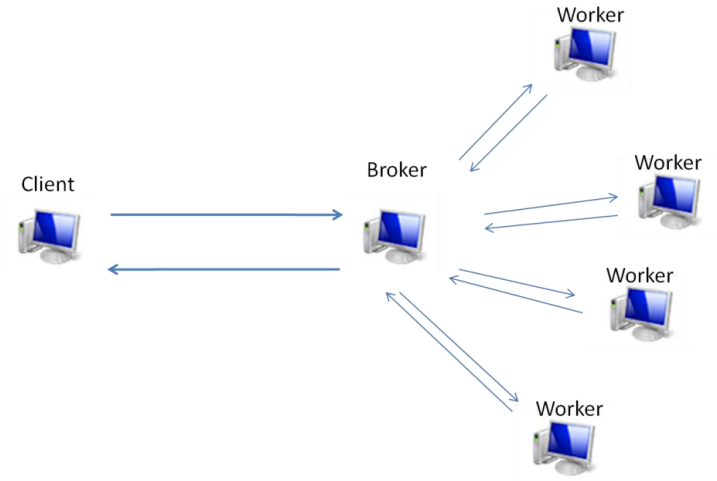
2.1.2 其它應用的棋類

其他可適用於本雲端教學系統的棋類包括：圍棋、五子棋、象棋等等。

2.2 目前已開發之桌機格網系統

這套系統是由交通大學吳毅成教授實驗室所開發的桌機格網系統，最開始是由已畢業的陳靖平[2]與詹宜智[10]學長一起開發，目前由碩士班學生鄒忻芸[10]同學維護以及增加新功能。這套系統在六子棋上主要的工作為建立六子棋的開局庫，作法簡單來說是將每個不同開局譜交給不同的電腦來運算，以求結果。另外除了六子棋之外，這套系統也提供給三角殺棋、數獨等等多種電腦對局人工智慧同時使用。

這套桌機格網系統是由以下三者所購成：Client、Broker、Worker。以下將針對各角色一一說明。



圖表 4 目前已開發之桌機格網系統架構

2.2.1 角色說明 - Client

Client 負責產生要計算的工作，並且管理所有由自己產生的工作。六子棋或三角殺棋等等應用都有自己各自不同的 Client。Client 的執行流程是會先連線到 Broker，再把要交給桌機格網系統運算的工作傳給 Broker，等到 Broker 收到工作執行完成之後再把結果傳回給 Client，Client 再處理這些結果。

2.2.2 角色說明 - Broker

Broker 是介於 Client 與 Broker 之間的溝通橋梁。他會負責接收並排序各個 Client 之間的工作，之後選定一台適合的 Worker 來執行該工作。直到 Worker 將工作結果傳回給 Broker，Broker 再把該結果傳回給原本發送此工作的 Client。同時 Broker 也會維護所有 Client 與 Worker 的資訊。

2.2.3 角色說明 - Worker

Worker 是真正執行分散工作的角色，當他收到 Broker 傳送過來要執行的工作後就會開始執行，但是回傳結果的方式主要分為兩種，一種是一個工作只會有一個回傳結果，另一種則是在執行過程中一直產生結果並隨時將結果回傳給 Broker。

Worker 執行工作的方式是執行對應該工作的執行檔，所以不管是六子棋或是三角殺棋的工作都必須要先把該工作對應的執行檔放在 Worker 端(以六子棋為例的話就是 NCTU6)，Broker 在上傳工作的時候僅需傳送給 Worker 執行檔名稱與參數。

2.3 現有雲端系統

雲端運算系統，最早是由昇陽電腦於 1983 年所提出雲端系統的概念，之後到 2006 年由亞馬遜提出第一個雲端服務，而在同年才由 Google 首次提出「雲端運算」的概念。雲端運算系統英文稱做 Could Computing System，它是利用大量的實體機器組成叢集電腦，叢集中各實體機器之間利用網路來溝通、傳輸資料。另外也是透過網路來提供一般使用者運算資源以及儲存空間。雲端系統的雲其實指的就是網際網路的那朵雲，也稱網路運算。也由於資料及運算資源都放在雲端上，所以使用者不需要知道雲端系統內部實際的組成細節，只需要知道如何使用其上的資源即可。

接下來介紹雲端運算系統一個很重要的運算方式：Map-Reduce 運算。Map-Reduce 運算最早是由 Google 所提出，用於處理非常大量的資料運算。它是利用運算工作的結合率，將一個大量的工作指定一個 Map（映射）函數與 Reduce（化簡）函數。Map 函數主要目的是將許多小的工作分別平行執行，通常會散布到不同的實體機器上來加速運段，然後在 Reduce 函數時會收集各個小工作的運算結果再進行處理，之後得到整個工作運算的結果。

以下小節將介紹幾個著名的雲端運算系統。

2.3.1 GridGain

GridGain[4]是 2005 年開始的公開原始碼專案。GridGain 是以 Java 開發的雲端平台，並僅支援使用 Java 的程式開發雲端運算系統。我們使用的版本是 2.1.1 版。

GridGain 中各機器主要利用點對點(Peer-to-peer)方式形成叢集電腦，每個 Grid 節點

(Node)都稱為 Grid instance，主要利用 IP Multicast 的方式來找到其他 Grid instance，當有一台電腦啟動 GridGain 後，GridGain 會自動去做 IP Multicast 找到其他有開啟 GridGain 之電腦，並自動連線形成叢集，不需要使用者自己設定。我們所使用的這個 GridGain 版本沒有特定的檔案系統，除了可以與現有一些分散式檔案系統合併使用外，主要專注於工作的分散運算。

以下將說明如何利用 GridGain 開發雲端程式。開發的主要方式是為每個工作實作工作內容以及兩個特殊的方法：Split(Map)與 Reduce。

- Split(Map)

  定義如何將一個工作切分成多個子工作散布到各 Grid Instance 上運算。

- Reduce

  各子工作回傳的結果如何處理得到最終結果。

  GridGain 會依程式設計者所設計的 Split, Reduce 方法來執行工作。

以下舉一個例子簡單說明 GridGain 設計一個工作的方式。假設現在工作是要計算一個數字是否為質數，以 333331 這個六位質數當作例子。在一般的做法上根據質數的定義需檢查 2-333330 之間所有的數是否存在一個數能將 333331 整除，若不存在可將 333331 整除的數則 333331 為質數。所以若是這個數字相當大則非常耗費運算的時間，我們利用 GridGain 將工作平行化，散布到不同的實體機器幫忙運算，減少運算時間。如果我們有四個可以運算的 Grid Instance，則在 Split 方法內就必須要實作如何將大工作切分成四個小工作。這邊的做法是讓不同的 Grid Instance 去負責運算不同的數字區間。讓第一個 Grid Instance 負責檢查 2-83332 之間是否有數字能將 333331 給整除，第二個 Grid Instance 則是負責 83333-166664 區間的數字，第三個則是 166665-249996 區間，最後第四個則是 249997-333330 區間，這樣就分別檢查完了所有 2-333330 之間的數字。各工作

的內容就是各 Grid Instance 將自己負責區段的數字去除 333331 看是否整除，若是則回傳 False，代表不為質數。若都沒有數字可將 333331 整除則回傳 True。而在 Reduce 方法就是去收集各區段的運算結果回傳。若是所有區段的結果都是 True 則代表 333331 是一個值數。但若運算過程中有其中一台機器回傳 False 則代表 33331 已經被整除了，也就不是值數，此時還會另外通知其他還沒運算完的 Grid Instance 不需要在運算下去了。

2.3.2 Windows Azure

在 2008 年的微軟專業開發人員大會所首次發表，Windows Azure[11]是由微軟所開發的雲端系統。可以分為公有雲或私有雲。在公有雲中，Windows Azure 所提供的雲端平台服務是將所有 Windows Azure 開發的應用程式運行在微軟機房所配置的虛擬機器裡面，若是該虛擬機器配置到的實體機器發生問題，則 Windows Azure 會自動將該虛擬機器轉移到其他台實體機器上面來運行，藉以省去使用者對於機器管理的成本，專注於應用程式的開發。

Windows Azure 主要提供兩個主要的雲端服務資源，一個是計算的資源，一個是儲存的資源，在運算資源中又分為 Web role instance 與 Worker role instance 兩種程式角色，其中的 Web role instance 主要是執行網頁應用程式(如:ASP.net, PHP)，使用者開發好網頁應用程式後將之上傳到 Windows Azure 上，Windows Azure 會給予一個網址，可以由此網址透過瀏覽器連線到該網頁應用程式。使用者也可以選擇開啟多個 Web role instance 來運行相同的網頁應用程式，並且使用相同的網址來連，來達到分流的效果。而 Worker role instance 則是執行一般的應用程式，可用 C#或其他語言撰寫，與一般應用程式開發並無太大差異，僅是將程式運行在 Windows Azure 上面。然而 Windows Azure 也提供 Web role instance 與 Worker role instance 之間的溝通，可同時使用兩者開發大型專案。只是本篇論文的系統實作僅使用到 Worker role instance。

另外一個 Windows Azure 提供的服務資源就是儲存的資源，由於前面提到提供運算資源的虛擬機器不提供資料儲存的能力，該虛擬機器的儲存空間會在程式結束之後移除。所以為了永久性的保留資料。Windows Azure 提供了資料儲存的資源，分別為 Blobs、資料表、以及佇列，首先 Blobs 就是提供一般二進位表示的檔案例如圖檔，影音檔都可以用這種儲存方式保存。第二個是資料表，它提供簡易的 Key-Value 的資料存取方試，彈是不像真正資料庫提供這麼多功能。第三個則是佇列，它並非提供真正的資料儲存能力，而是提供 Web role instance 與 Worker role instance 之間的溝通方式。但由於本系統在實作上沒有使用到 Windows Azure 提供的資料儲存資源，所以不再詳細介紹。

最後介紹的是 SQL Azure，他是以 Microsoft SQL server 為基礎設計的雲端關聯式資料庫，它的特色是將資料庫設置及資料儲存都放在雲端，也就是跟前面 Windows Azure 相同的機房之中，所以使用者不需要自行安裝資料庫，只需設定好連線即可使用。而且 SQL Azure 會自動提供異地備援資料的能力，省去許多資料庫使用的管理成本。

第三章、系統設計

此篇論文主要的目的就是在我們的棋類教學系統上面，利用雲端系統的特性提供電腦棋類人工智慧以及大量的棋譜資料。

配合前面提到的教學系統需提供的三種情境－算出最佳著手、算出所有擋法、棋譜儲存與搜尋。提供以下運算功能：

- 單一結果回傳

  Worker 工作結束僅回傳一個結果回來。
- 連續結果回傳

  Worker 工作過程中一產生部份結果就立即回傳。
- 資料儲存

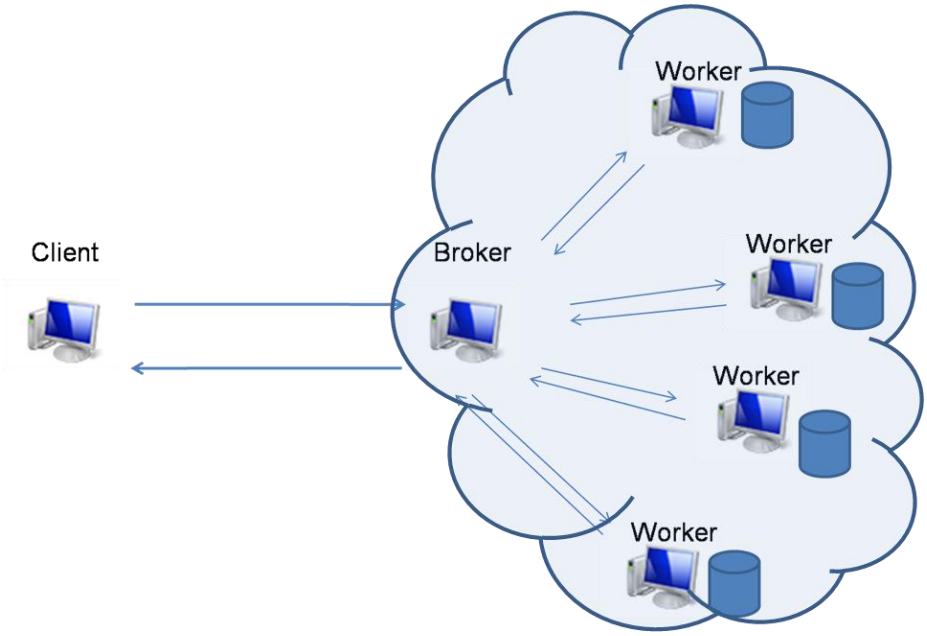
  將一筆資料儲存到雲端。
- Map-Reduce 資料搜尋

  利用雲端 Map-Reduce 運算做資料的找尋。

以下將先說明整個系統的架構，之後會對上面提到的單一結果回傳、連續結果回傳等各功能做詳細介紹。

3.1 系統架構

主要由實驗室已開發桌機格網系統來設計，將之嵌入現有雲端系統，除了提供原本桌機格網系統就有之分散運算功能外，也另外提供資料儲存以及 Map-Reduce 運算功能。



圖表 5 系統架構

3.2 各功能之設計

以下將說明在 **3.1** 節需求中提出來的運算功能，以及第一章提到的雲端教學系統的情境，兩者之間如何搭配利用。這邊將以情境為主來說明各功能如何配合。

3.2.1 算出最佳著手

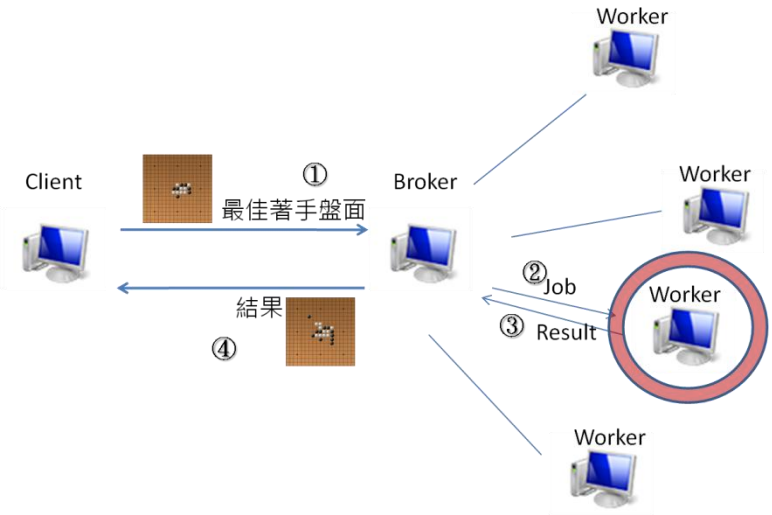
算出最佳著手是利用六子棋人工智慧程式計算出目前盤面上下一步怎麼走最好。因為每一個盤面只需要運算出一個最好的結果，所以主要利用到每個工作只會有一個結果的「單一結果回傳」功能。「單一結果回傳」這個功能也是一般桌機格網系統或是雲端系統皆有提供的最基本功能。

單一結果回傳的執行流程：

1. Client 發送一個 Job 給 Broker。
2. Broker 挑選 Worker 來執行此 Job。

3. 直到 Worker 執行完成才將結果回傳給 Broker。

4. Broker 再將結果傳回給 Client 。



圖表 6 算出最佳著手流程

3.2.2 算出所有擋法

算出所有擋法則是透過六子棋人工智慧程式先算出目前的盤面是否已經有一方已經必勝了，無論防守方怎麼擋都無法阻止攻擊方獲勝。但若是目前盤面還沒有任何一方已經被計算出必勝的話，六子棋人工智慧程式則是會回傳所有可能可以防守的方法。這些所有擋法的數量通常會非常非常的龐大，而且部份的資訊就有用。所以我們不能等到所有的擋法都計算出來之後才一次全部回傳，也就不能夠使用之前的「單一結果回傳」功能，而是要利用提供類似 Streaming 的方式，Worker 計算出了多少結果，就把那些結果回傳給 Broker 的「連續結果回傳」這個功能，這個功能也是一般市面上的桌機格網系統或是一般雲端系統中比較少見到有提供的運算功能，但是為了滿足六子棋人工智慧程式的特性我們必須要提供「連續結果回傳」。

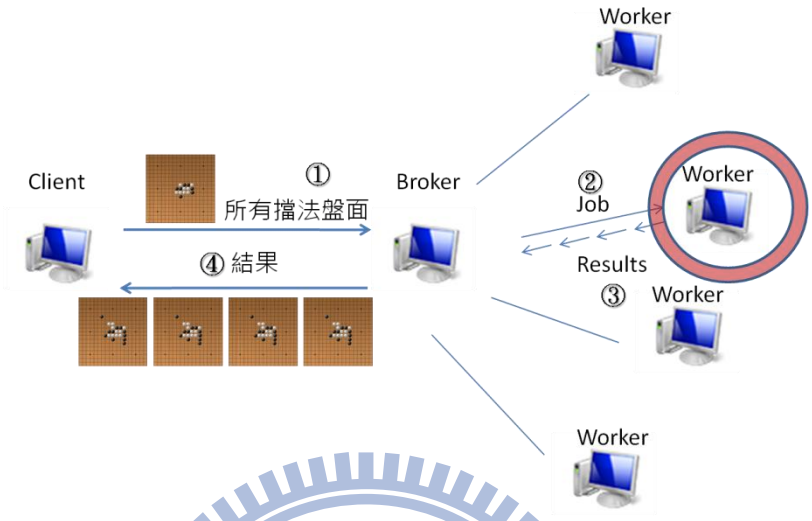
連續結果回傳的執行流程：

1. Client 發送一個 Job 給 Broker。

2. Broker 挑選 Worker 來執行此 Job。

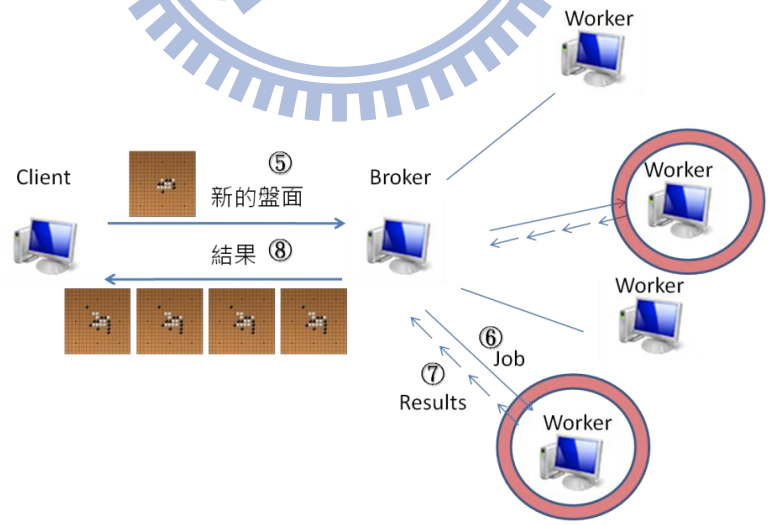
3. Worker 邊執行邊產生部分結果就回傳給 Broker。

4. Broker 也立刻將此結果回傳給 Client。

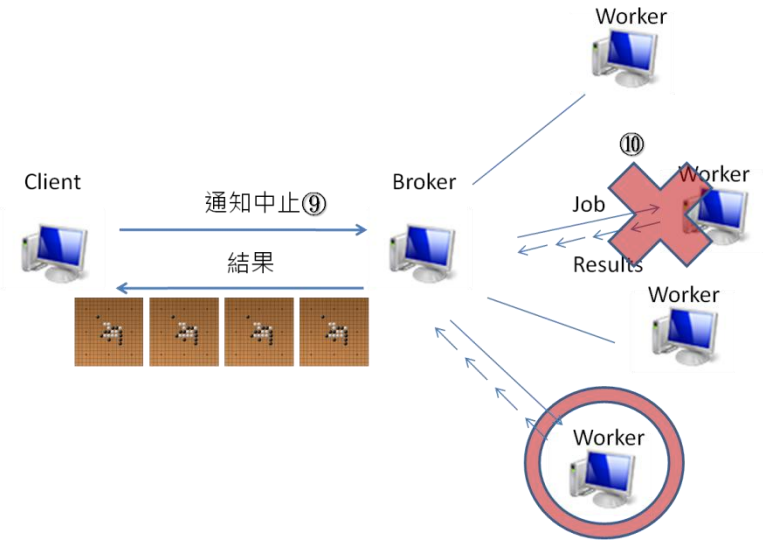


圖表 7 算出所有擋法流程-1

然而 Client 也可能從這些部份的結果去產生新的工作來執行，或是要求 Broker 中止一個正在執行中的工作，為了要做到這一點 Worker 也必須要持續的與 Broker 保持連線狀態，才能收到來自 Broker 中止一個 Job 的指令。



圖表 8 算出所有擋法-2



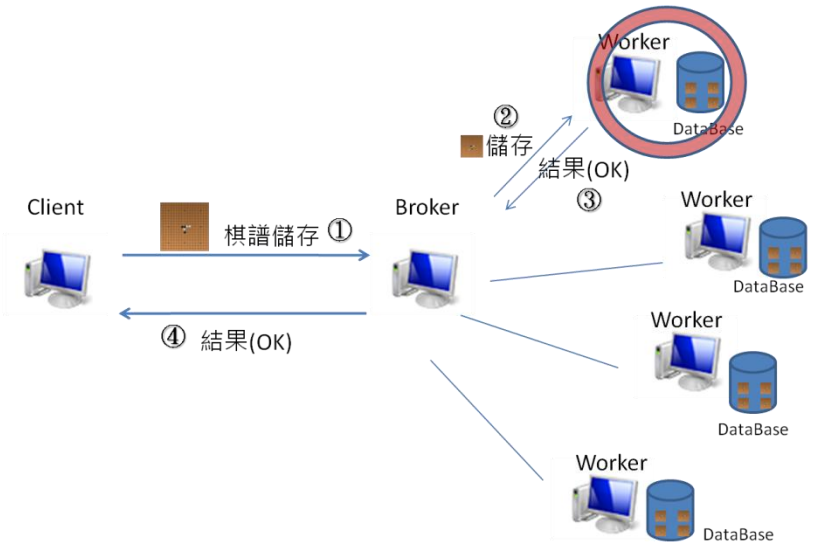
圖表 9 算出所有擋法-3

3.2.3 棋譜儲存與搜尋

首先是棋譜儲存，主要工作內容就是 Client 想要將一個盤面儲存在雲端，作法是利用「資料儲存」的功能，Client 首先會將該盤面傳給 Broker，之後由 Broker 來決定要把該盤面存在哪個 Worker 上，Client 不需要知道該盤面最後是儲存在哪個 Worker 上。

棋譜儲存的執行流程：

1. Client 發送一個要儲存的盤面給 Broker。
2. Broker 挑選 Worker 來儲存此盤面。
3. Client 不需知道該盤面儲存於哪個 Worker，僅得到來自 Broker 表示完成的訊息。



圖表 10 棋譜儲存流程

最後是棋譜搜尋的部份，棋譜搜尋是利用一個盤面、部份棋型或是 Little Golem 上的玩家資訊去雲端資料庫中搜尋一個完整的盤面。主要是要利用「Map-Reduce 資料搜尋」功能。執行流程會分為 Map 階段與 Reduce 階段，在 Map 階段 Broker 會把將此搜尋工作傳送給所有負責儲存的 Worker，所有負責儲存的 Worker 都要執行此類工作，各自搜尋本機的儲存空間有無符合的結果，之後再將搜尋結果回傳給 Broker，而 Broker 的 Reduce 階段則會接收各個 Worker 回傳的結果並排序，排序的方式我們是依照 Little Golem 網站上面雙方玩家的分數來排序，我們認為實力比較高強的玩家所下出來的盤面是比較重要也比較具代表性的。Reduce 完成之後會把排序好的所有盤面再回傳給 Client。

棋譜搜尋的執行流程：

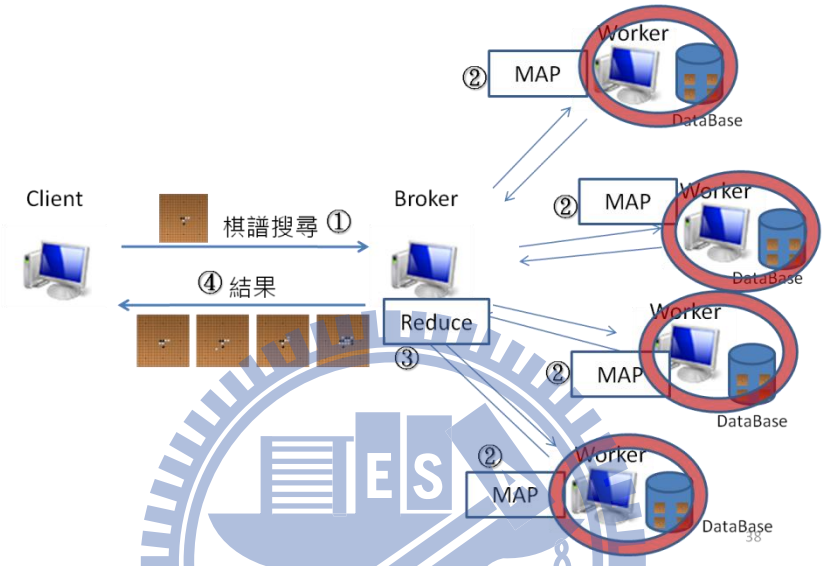
1. Client 發送一個要搜尋的盤面給 Broker。
2. Broker 上面的 Map 方法執行。
    a. Broker 將此盤面 Map 給所有負責儲存的 Worker。
    b. 由各 Worker 在各自本機的儲存空間搜尋。
3. 各 Worker 分別執行並回傳結果給 Broker。

4. Broker 上面的 Reduce 方法執行。

   a. Broker 等著接收各 Worker 的回傳結果。

   b. 蒐集完所有 Worker 的結果後進行排序。

   c. 再將排序後的結果回傳給 Client。



圖表 11 Map-Reduce 棋譜搜尋

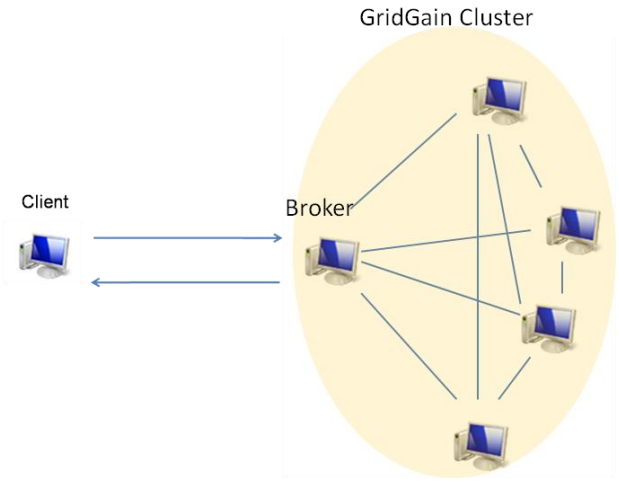
第四章、系統實作及實驗數據

本章節將說明我們如何利用現有的雲端系統來實作前面所設計的系統，主要是以 GridGain 及 Azure 來分別實作，以下將分別說明。在最後會說明兩個系統實作在做棋譜搜尋時所花時間的比較。

4.1 以 GridGain 實作

一開始將先說明以 GridGain 來實作的例子。由於 GridGain 是 Java 的平台，所以整個雲端系統都必需要利用到 Java 語言來撰寫。將 GridGain 的 jar 檔案加入我們雲端系統的專案並且繼承其中幾個關鍵類別後，就可以開始以 GridGain 來實作我們的雲端系統。

由於 GridGain 中每一個 Grid Node 都視互相為 Grid Instance，並不像我們有提供一台專門負責工作配送的 Broker 角色，所以在 GridGain 上面的實作將會設計一個特別的 Grid Instance 來當作我們的 Broker，其他沒有特別實作的 Grid Instance 則變成我們的 Worker。



圖表 12 GridGain 實作圖

首先是 Client 與 Broker 溝通的部份，因為 Client 可能是各種教學程式，或是棋譜編輯器，Client 並不限定使用 Java 來開發，也就不一定會透過 GridGain 將 Client 實作在 GridGain 的 Cluster 之中，我們目前的例子也並不是用 GridGain 來開發 Client 程式，而是將 Client 獨立在 Cluster 的外面。另外也不限定 Client 的程式語言或是使用環境等等。因此我們 Client 與 Broker 之間的溝通就是採用標準 TCP Socket 連線，因為大部分的程式都有支援提供 Socket 的連線方式，另外為了與現有的桌機格網系統配合，溝通的格式是以交換 XML 檔案的方式來溝通。

後面 GridGain Cluster 的部份，由於 GridGain 已經提供幾個簡單的雲端系統的功能，這邊我們就不需要再另外實作那些功能。包括：Broker(Grid Instance)與 Worker(Grid Instance)之間的溝通、工作如何依據開發者定義的分配方式將工作發布到 Worker 上執行、Worker 新加入或離開的簡單管理、Map-Reduce 的運算架構等等。也就是我們只需要依照前面所講的 GridGain 程式開發的方式就可以簡單設計出一個 Broker 以及後面整個 GridGain Cluster 的部份。

以下將解釋如何利用 GridGain 來實作「單一結果回傳」、「連續結果回傳」、「資料儲存」、「資料搜尋」功能之詳細說明。

4.1.1 實作單一結果回傳

以本教學系統來說，Client 在需要算出最佳著手時會利用到「單一結果回傳」的功能。Client 在要求執行此類工作時，會送出一個以 XML 格式表示的檔案。其中包含執行檔名稱、要傳入執行檔的參數、或是該執行檔的版本等等其他資訊給 Broker。其中執行檔的名稱即為 NCTU6.exe 這個由實驗室學長所開發的六子棋人工智慧程式，而傳入的參數就是告訴 NCTU6 現在要計算的是最佳著手以及給予 NCTU6 目前的盤面狀態。

Broker收到此XML檔案後會先解讀各個欄位資訊,取出重要的部份後進入GridGain的 Split 方法中。由於此類工作不需要再切割成更小的工作，所以在 Split 方法中只需選定好一台適合執行此工作的 Worker 來執行工作即可。

而 Worker 執行工作的內容就是以 Java Runtime 來執行 NCTU6.exe，並傳入對應的參數。因此所有工作要執行的執行檔都必須要預先放在 Worker 端，這也是與目前實驗室開發的桌機格網系統相配合。等到 Java Runtime 收到 NCTU6.exe 的執行結果後再把結果傳回給 Broker。

Broker 的 Reduce 方法只需收到該結果之後就立即回傳給 Client，因為不會有其他 Worker 會回傳同一個工作的結果。當 Client 拿到來自 Broker 送回的結果，就完成了利用「單一結果回傳」的算出最佳著手。

4.1.2 實作連續結果回傳

同樣以本教學系統來說，Client 在需要算出所有擋法時會利用到「連續結果回傳」的功能。此類工作執行方式上大致上與前者相同，Client 與 Broker 仍然是以 XML 溝通並且 Worker 端以 Java Runtime 執行 NCTU6。不同的是會先透過參數告知 NCTU6.exe 這次的工作是要做算出所有擋法，而執行算出所有擋法時 NCTU6.exe 會先計算此盤面是否已經有一方已經必勝，但是大部分的情況都是 NCTU6.exe 沒有算出此盤面有一方已經必勝，此時 NCTU6.exe 就會輸出所有可能的擋法，此時輸出的結果就會非常多，並且可能部份結果就有用。所以必須要提供類似Streaming的方式將結果傳回給Broker。

為了要達到單一一個工作回傳多個結果這個要求,本系統使用了 GridGain 所提供的「Session」元件來讓 Broker 與 Worker 溝通。Session 本來是讓各 Grid Instance 在分別執行工作時，考慮到可能還是會有相依性所設計的溝通管道。在工作執行前可以為此工作

宣告一個 Session，之後所有負責執行此工作的 Grid Instance 都可以將訊息寫入此 Session，並且讓每一個執行相同工作的 Grid Instance 可以讀取此 Session 中的訊息。我們的做法就是當 NCTU6.exe 算出一部份結果時，Worker 就會立刻將此部分結果寫入 Session 中。而在 Broker 端則是有 Session Listener 隨時監聽 Worker 是否有將資料寫到 Session 中。若有資料被寫入 Session 則立刻將之取出並傳回給 Client。Client 會自己將這些部份結果拼湊或是另外使用。

而在工作執行完成後其實 Worker 就不會回傳任何結果，因為結果全部都已經先透過 Session 傳回去了，執行到 Broker 的 Reduce 方法時也只會通知 Client 工作已經完成，後面不會再有更多結果。

4.1.3 實作資料儲存

Client 在需要儲存一個現有盤面的時候會利用到「資料儲存」功能。Client 一樣透過 XML 檔案將要儲存的盤面傳送給 Broker，Broker 會在 Split 方法中以盡量讓資料平均分佈的原則挑選一個 Worker 來負責儲存此盤面。這邊我們先假設所有負責儲存的 Worker 都不會離線，以方便系統實作。

而我們的棋譜資料並非直接存在 Worker 的硬碟之中，而是存放在 Worker 本地的 MySQL 資料庫當中。所以所有負責儲存的 Worker 都必須要提供 PHP + MySQL。以方便之後的棋譜搜尋。

Worker 執行工作的方式也是以 Java Runtime 執行一個執行檔。此執行檔是一隻由 PHP 寫成的棋譜儲存程式。此程式會將目標盤面存放在本地的 MySQL 資料庫當中。

然而棋譜儲存的工作並不會有特別的結果回傳，Worker 僅會回傳一個 OK 訊息給

Broker，此時 Broker 的 Reduce 方法也就不需要做任何特別的事情。可能也僅是傳送給 Client 一個 OK 的訊息。通知 Client 棋譜儲存的工作已經完成。

4.1.4 實作資料搜尋

Client 在透過一個盤面、圈選部份的棋型或是以 Little Golem 上面的玩家資訊來搜尋棋譜資料的時候，會用到「資料搜尋」的功能。

Client 透過 XML 檔案將要查詢條件送給 Broker 後，Broker 解讀各欄位之後在 Split 方法會將所有搜尋工作分配到每一個有負責儲存的 Worker 上。要求每一個負責儲存的 Worker 都要執行此搜尋工作，因為所有負責儲存的 Worker 都可能有存放符合條件的盤面。

在每個 Worker 收到此搜尋工作後，會利用 Java Runtime 執行另外一隻負責棋譜搜尋的 PHP 程式，這隻程式會依照輸入的條件到本機的 MySQL 資料庫中搜尋所有符合結果的盤面。搜尋完成之後 Worker 會將找到的所有盤面傳送給 Broker。

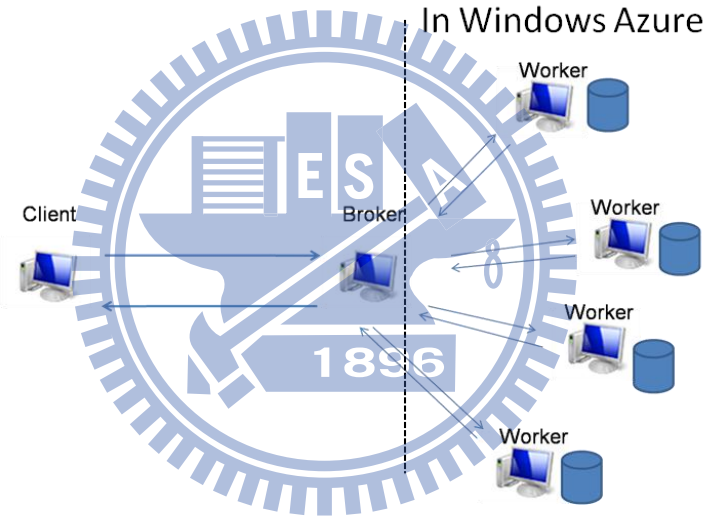
而在 Broker 的 Redce 方法中會蒐集所有 Worker 的回傳結果。等到所有回傳結果都回傳之後會進行排序，排序的方式就是依照這個盤面黑白雙方在 Little Golem 上面的玩家排名來排，分數較高的玩家所下的盤面會被我們會認為是比較重要可參考的盤面，所以會被排在比較前面。排序完之後的結果再一次傳回給 Client。以完成棋譜搜尋的工作。

4.2 以 Azure 實作

接著說明以 Windows Azure 來實作的例子。在 Azure 這邊我們主要是利用 Azure 的 Worker role instance 來開發我們的 Worker，使用的語言是 C#，原因是目前開發 Azure 的

程式語言中以C#的資源較為豐富。所有的Worker都是執行在微軟的Azure虛擬機器中。

Broker 的部份卻不使用 Azure 的 Worker role instance 來開發，主要原因是我們的系統架構都是由 Client 及 Worker 主動連線至 Broker，Client 與 Worker 需要預先知道 Broker 的 IP 位址。而所有的 Worker role 程式都是執行於 Azure 的虛擬機器中，當 Worker role instance 執行結束後虛擬機器就會關掉。重開之後 IP 位址就會重新配置，造成每次都需要更改 Client 與 Worker 設定的麻煩。因此我們將 Broker 設計在 Azure 的外面，給予 Broker 一個固定的 IP，以方便連線。另外為了方便設計，我們讓 Broker 與 Client 的連線方式與 GridGain 相同，採取 TCP socket 連線以及 XML 檔案格式。



圖表 13 以 Windows Azure 實作圖

另外還有一個地方與 GridGain 的實作不同，那就是 Broker 與 Worker 之間的溝通方式不像 GridGain 已經由 GridGain 已經處理好了，在 Azure 這邊必須要自己處理。所以我們參考之前實驗室已開發之桌機格網系統的設計方式，讓 Broker 與 Worker 之間的溝通也是用 TCP socket 連線及 XML 檔案格式。

以下將解釋如何利用 Windows Azure 來實作「單一結果回傳」、「連續結果回傳」、「資料儲存」、「資料搜尋」功能之詳細說明。

4.2.1 實作單一結果回傳及連續結果回傳

同樣是在 Client 利用六子棋人工智慧程式 NCTU6.exe 來算出最佳著手或是所有擋法時使用「單一結果回傳」以及「連續結果回傳」此二功能。另外 Client 與 Broker 之間的溝通方式也與 GridGain 類似，所以這邊都不再重複。

當 Broker 收到來自 Client 的 XML 檔案並且解析完各欄位後，由於 Azure 沒有像 GridGain 定義 Split 方法後就會自動幫你把工作傳到 Worker 端的功能，所以需要自己決定要交給哪個 Worker 執行，並且要自己把工作透過 XML 格式傳給 Worker 端。

Worker 收到來自 Broker 的工作之後，同樣是以執行 NCTU6.exe 這個執行檔的方式，執行的方式是利用 C#的 Process 來執行。所以必須預先把所有會使用到的執行檔發布到 Windows Azure 的虛擬機器中。因為 Broker 與 Worker 連線方式是採 TCP socket 連線方式，本身就已經有 Streaming 的特性，所以當 Worker 一有部份或完整結果產生，就立即將該結果透過連線傳回給 Broker。

4.2.2 實作資料儲存

這邊實作資料儲存的方式與 GridGain 的方式有相當大的差異，原因是因為 Worker 都執行於 Azure 的虛擬機器中，而當程式結束後虛擬機器也會關閉並且清掉本地的儲存空間，無法永久保存資料。當然 Azure 也有提供 Storage 的服務，包含 Blobs 與 Table 以及 Queue 等等可永久儲存資料的方式。但是因為我們的棋譜資料是存放在資料庫中，所以直接使用也是由微軟提供的 SQL Azure 雲端資料庫來當作資料儲存的空間是比較適合的。

Client 要儲存一個盤面的時候，作法上與 GridGain 差異就不大，只是 Worker 就不

是將資料儲存在本地的 MySQL 資料庫，而是將資料統一放在 SQL Azure 雲端資料庫上面。

4.2.3 實作資料搜尋

由於棋譜不是分散的儲存在各 Worker 的本地 MySQL 資料庫中，而是只存一份在 SQL Azure 的雲端資料庫中，所以就無法提供 Map-Reduce 的棋譜搜尋方式。Broker 僅會選擇一個 Worker 來執行搜尋 SQL Azure 的方式。

執行的方式一樣是透過 C#中的 Process 來執行負責搜尋 SQL Azure 的 PHP 執行檔。將結果傳回給 Broker 後再進行排序。排序後的結果再傳回給 Client。以完成棋譜搜尋的工作。

4.3 實驗數據

以下將比較 GridGain 與 Windows Azure 在棋譜搜尋上面的速度。由於 Windows Azure 那邊的實作中，資料只儲存一份在 SQL Azure，為了比較此兩系統的效能，所以 GridGain 的實作也是將棋譜資料集中在同一個 MySQL 資料庫中。

兩邊資料庫都放入 321 個遊戲盤面。每個盤面平均有 23.8 手，每一手再解析出來一共會有 7633 個盤面存在資料庫中。我們舉了六個六子棋的開局盤面來做棋譜搜尋。以下是比較表格。

盤面編號	結果數量	GridGain 搜尋時間(ms)	Azure 搜尋時間(ms)
1	25	2198	3236
2	29	511	2821
3	65	1669	3465
4	32	951	2440
5	8	760	2114
6	1	993	2159

圖表 14 GridGain 與 Winsows Azure 比較

第五章、結論與未來展望

本篇論文利用雲端系統的特性提供了棋類教學平台兩個主要的新功能：電腦人工智慧運算，以及大量棋譜資料兩個特性。讓教學系統可以讓玩家在不知道該怎麼走下一步的時候得到電腦人工智慧的最佳建議，以及獲得大量棋譜的資料可以用來比對或參考高手的棋路。

另外我們整合之前實驗室學長所開發的桌機格網系統進入雲端系統。提出了利用現有雲端系統來開發之前桌機格網系統的方法，並且額外的提供了資料儲存的功能以及 Map-Reduce 運算的能力。

我們分別運用 GridGain 以及 Windows Azure 兩個雲端的平台實作了我們的系統。說明了在不同的雲端平台中都成功的達到了我們的目的，可以讓其他以後想要做到類似事情人作為參考。

未來展望的部份，目前在 GridGain 棋譜儲存的實作方面，負責儲存的 Worker 是假設都不會離線的，但是真正的 Worker 是有可能會隨時加入或離開的，所以應該要有一個機制會自動的把儲存的資料自動備份，或是需要有幾台固定永遠都不會離開的「備援 Worker」，等機制讓自願捐贈的 Worker 可以隨時離線或加入。

另外 GridGain 在做 Map-Reduce 的時候，Map 方法會要求所有負責儲存的 Worker 都要執行搜尋的工作。但是萬一某一台機器原本就正在執行其他工作(例如建議著手)的話，搜尋的工作與其他工作之間該如何排序才能達到最好的效能。

最後就是利用 Windows Azure 實作的棋譜儲存資料都是放在 SQL Azure 上面，無法

提供像 GridGain 的 Map-Reduce 分散式搜尋。這樣存取速度就會較為緩慢，所以能否在 Azure 上提供類似 Map-Reduce 的方式來加速搜尋速度。

參考文獻

[1] Chan,Yi-Chih, Parallel Proof Number Search for Connect6, Master Thesis, National Chiao Tung University, 2009.

[2] Chen, Ching-ping, A Desktop Grid Computing System for Connect6 Application, Master Thesis, National Chiao Tung University, 2009.

[3] Connect6 Homepage, available at http://www.connect6.org/.

[4] GridGain, available at http://www.gridgain.com/.

[5] ICGA(International Computer Games Association) , available at http://ticc.uvt.nl/icga/.

[6] LGS Homepage, available at http://lgs.taiwango.net/.

[7] Liao, Chia-Yin, Multi-Touch System for E-Learning Platform of Board Games, Master Thesis, National Chiao Tung University, 2010.

[8] Little Golem, available at http://www.littlegolem.net/jsp/.

[9] Lin, P.-H., and Wu, I.-C., NCTU6 Wins Man-Machine Connect6 Championship 2009, ICGA Journal, Vol. 32(4), pp. 230–232, 2009.

[10] Tsou, Hsin-Yun, The Study and Design of the Generic Application Framework and Resource Allocation Management for the Desktop Grid CGDG, Master Thesis, National Chiao Tung University, 2010.

[11] Windows Azure, available at http://www.microsoft.com/taiwan/windowsazure/.

[12] Wu, I.-C., “Cloud-based E-learning platform for Connect6 and related board games”, Microsoft project, 2010.

[13] Wu, I.-C., and Huang, D.-Y., A New Family of k-in-a-row Games. The 11th Advances in Computer Games Conference (ACG'11), pp. 180-194, Taipei, Taiwan, 2005.

[14] Wu, I.-C., and Lin, P.-H., NCTU6-Lite Wins Connect6 Tournament, ICGA Journal, Vol. 31(4), pp. 240–243, 2008.

[15] Wu, I.-C., and Lin, P.-H., Relevance-Zone-Oriented Proof Search for Connect6, to appear in the IEEE Transactions on Computational Intelligence and AI in Games, 2010.

[16] Wu, I.-C., Huang, D.-Y., and Chang, H.-C., Connect6. ICGA Journal, Vol. 28(4), pp. 234-242, 2006.

[17] Wu, I.-C., Chen, C., Lin, P.H., Huang, K.C., Chen, L.P., Sun, D.J., Chan, Y.C., and Tsou H.Y., A Volunteer-Computing-Based Grid Environment for Connect6 Applications. The 12th IEEE International Conference on Computational Science and Engineering (CSE-09), August 29-31, Vancouver, Canadar, 2009.