國立交通大學

資訊科學與工程研究所

碩士論文

CGDG 桌機格網之應用層框架一般化與資源分配管理

The Study and Design of the Generic Application Framework and Resource Allocation Management for the Desktop Grid CGDG

研 究 生：鄒忻芸

指導教授：吳毅成　教授

中 華 民 國　九 十 九　年 八 月

CGDG 桌機格網之應用層框架一般化與資源分配管理

The Study and Design of the Generic Application Framework and Resource Allocation Management for the Desktop Grid CGDG

研 究 生：鄒忻芸　　Student：Hsin-Yun Tsou

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

August 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年八月

CGDG 桌機格網之應用層框架一般化與資源分配管理

研究生：鄒忻芸　　指導教授：吳毅成

國立交通大學 資訊科學與工程研究所

摘要

桌機格網為一項重要的技術。概念上，為利用網路，將閒置的電腦資源組成一虛擬的超級電腦。然而應用上，鮮少被利用在電腦對局應用問題。在 2009 年時，由交大吳毅成教授研發團隊提出一套利用桌機格網(Desktop Grid)解決許多過去未解的電腦對局應用問題(Computer Games)的方法，如:建構六子棋開局庫，在本論文稱該系統為 Computer Game Desktop Grid (CGDG)。

由於這套桌機格網是由交大吳毅成教授研發團隊剛研發不久，尚須增加許多功能，因此本論文的目的為致力於改進應用層框架一般化與資源分配管理的部分。應用層框架一般化是指將原本僅適用於六子棋應用的桌機格網系統，擴展於大部分的電腦對局應用問題;而資源分配管理則是在多個組織與使用者加入時，提出一套分配資源與管理的方法。

The Study and Design of the Generic Application Framework and Resource Allocation Management for the Desktop Grid CGDG

Student: Hsin-Yun ,Tsou Advisor: I-Chen,Wu

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract

Desktop grid is an important technology. However there are no desktop grid systems specific for computer games. So our laboratory developed a desktop grid system for computer games in 2009. We used this desktop grid system to solve the computer games problems successfully, like the Connect6 openings. Then we named this system as Computer Game Desktop Grid (CGDG) in this paper. Since this system was built one year ago, it still needed many new functionalities.

This paper focuses on improving the functions about the generic application framework and resource allocation management for this desktop grid CGDG.

誌謝

這篇論文得以完成，首先要感謝我的指導教授吳毅成教授，沒有老師的建議及指導，絕對無法順利的完成論文。也感謝鈊象電子在研究上的經費支助。

感謝口試委員許舜欽教授、陳志昌教授，及徐讚昇研究員所提出寶貴的意見指教，讓我發現自己論文不足的地方並加以改進。

在此次系統的實作過程中，很感謝陳靖平，幫助我順利解決了很多困難和複雜的步驟，以及韓尚餘在實驗室機器管理調配上的大力支援和幫忙。

感謝實驗室的同學，家茵與彥成，在我情緒低落與徬徨時給予我心靈的支持，以及所有 97 級的各位同學：彥成，家茵，俊嶧，郁雯，柏廷，大家一起水深火熱，共同奮鬥的這段革命情感。謝謝實驗室的各位學弟們，在口試前的張羅，讓我得以心無旁騖的準備口試。

最後僅以這篇論文，獻給所有在我求學路程上曾幫助過我、關心我的人，還有我摯愛的家人，謝謝他們在我最困難的時候，為我的關懷祈禱和加油照顧。謝謝你們。

民國九十九年八月　於　新竹交大工程三館 CYC-511 實驗室

目錄

圖表目錄

第一章、緒論

桌機格網(Desktop Grid)[12]為一項重要的技術，主要被利用在科學計算、生物科技…等大量平行化計算的應用領域，但卻鮮少有人將其應用在電腦對局應用問題上，因此在 2009 年時，由本交大吳毅成教授研發團隊提出一套適用於電腦對局應用的桌機格網系統，在本篇論文稱該系統為 Computer Game Desktop Grid (CGDG)。在本章節中的章節 1.1 將會簡介桌機格網，章節 1.2 之中則會介紹本篇論文的研究動機與目的，章節 1.3 為論文大綱說明。

1.1. 桌機格網介紹

近幾十年以來，科學家不斷的在努力著想要解決大量計算和大量資料存取的需求。在早期，是利用超級電腦(Super Computer)來執行大量的計算，但是超級電腦的價格昂貴且系統複雜難以維護，因此，為了解決這樣的問題，桌機格網[12]可便誕生了。

桌機格網看作是分散式計算(Distributed Computing)的延伸，它透過網路收集處於閒置狀態的電腦計算資源或儲存空間，任何想分享資源(Resource)的電腦都可以自由的加入以及離開且資源動態性比較高，彼此配備差異大。而它在經濟和便利性上的優勢，使得它逐漸取代超級電腦，成為現在運用在大量計算上的主要系統。因其計算資源都是志願提供，亦可視為志願者，桌機格網計算也被稱做是志願者計算(Volunteer Computing)。

桌機格網的專案最早起源於 1996 年，名為 GIMPS (Great Internet Mersene Prime Search)，為運用在數學上，幫助尋找 Mersenne 質數的專案，此後，在 1997 到 1998 年間，Bayanihan、 Popcorn、 Superweb 與 Charlotte…等以 Java 為基礎的桌機格網開始相繼出現。到 1999 年，運行在 BOINC[2][11]上的 SETI@home（Search for ExtraTerrestrial Intelligence at Home，搜尋外星智慧）的專案被提出

後，吸引了數以萬計的工作端加入，桌機格網開始廣為人知。一直到現今，許多公司開始將他們商業模組的概念加入桌機格網中。

1.2. 研究動機與目的

在 2009 年時，交大吳毅成教授研發團隊提出一套利用桌機格網解決許多過去未解的電腦對局應用問題(Computer Games)的方法[17]，如:建構六子棋開局庫…等，在本篇論文稱該系統為 Computer Game Desktop Grid (CGDG)，而由於這套桌機格網是由實驗室剛研發不久，尚須增加許多功能，因此本論文的目的為致力於改進應用層框架一般化與資源分配管理的兩大部分。

應用層框架一般化，是指針對此桌機格網系統的應用層部分，提出一個一般化的框架，讓大部分的電腦對局應用問題可快速的加入該系統，如實驗室的數獨[16][19]、三角殺棋[20]、六子棋[8][14]…等應用。

在數獨方面[16][19]，過去，我們所找到最小提示數的盤面為 17，而在實驗室的應用中，我們希望可以證明最小提示數為 16 或 17，採取的作法為利用 54 億的盤面中，尋找是否存在一個提示數為 16 的初盤而得到證明，又因其 54 億個盤面彼此獨立的特性，該工作具有大量平行獨立計算的特性，若在單一電腦運算約需花費 2417 年。

三角殺棋方面[20],則是希望可以全解九層三角殺棋,不論是在遊戲規則中，拿到最後一顆子的人贏或者拿到最後一顆子的人輸的情形,皆可全解其必勝與必敗，其所使用的演算法可參考[20]，而由該演算法可知，該工作具有傳輸量相當大以及須耗費大量記憶體的特性。

六子棋方面，起因為六子棋是個相當新的遊戲，於 2005 年發表，相較於象棋[13]、圍棋[15]，這個遊戲的開局譜相對少很多，因此在 2010 年時，由交大吳毅成教授研發團隊發展了一套六子棋驗證系統[4]，裡面所採用的演算法為工作層級證明數演算法(Job-Level Proof Number Search)，簡稱 JL-PNS；此後我們利用

這套系統開始建構六子棋開局庫，而由該演算法[1]可知，該工作具有高度動態優先權以及低延遲需求的特性。

資源分配管理則是為了因應將來的群體計畫，在該計畫執行之後，將會有多個組織與使用者加入此系統，不同組織之間該如何去分配資源以及管理使用者，將是該計畫將要面臨的問題，而本論文即是針對這樣的情況下，提出一套分配資源與管理的方法。

1.3. 論文大綱說明

本篇論文第二章是研究背景的部分，在研究背景中，會介紹電腦對局應用問題以及桌機格網相關技術的背景資料，電腦對局應用問題方面，包括數獨、三角殺棋、以及六子棋，此三種應用，桌機格網方面，則包括桌機格網簡介、桌機格網與格網的比較，以及目前相關的系統及應用，第三章則會說明系統實作上的細節，包括應用層框架一般化與資源分配管理兩大部分的設計與實作，第四章則會提出結論以及未來的工作與展望。

第二章、研究背景

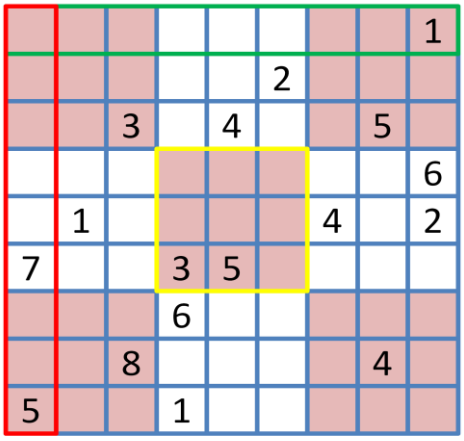
在本章節中，會介紹電腦對局應用問題以及桌機格網相關的背景資料，在 2.1 章節中將先針對目前實驗室應用在桌機格網上頭的數獨、三角殺棋、六子棋，此三種電對局應用問題進行規則、想解決的問題、以及平行化計算的特性做介紹。2.2 章節將針對桌機格網作簡介並比較與傳統格網不同的地方，並說明目前 BOINC 與 XtremWeb 兩大桌機格網系統不適用於電腦對局應用問題的原因，與我們研發的桌機格網系統面對這樣的問題時，解法為何。

2.1. 電腦對局應用問題

本章節將針對目前實驗室應用在桌機格網上的數獨、三角殺棋、六子棋，此三種電腦對局應用問題進行規則、想解決的問題、以及平行化計算的特性做介紹。

2.1.1 數獨應用問題

數獨(sudoku)[16]來自於日文，是由 18 世紀瑞士數學家歐拉所發明，其規則為在每個行列和 3*3 的方塊內填入數字 1 到 9，且每個數字不可重複，即算解掉該盤面，而對於一個合法盤面來說，只會產生一個終盤，具有唯一解的特性，在初盤中會預先填入若干數字，其他空格則留白，玩家可依照初盤中的數字分布狀況，邏輯推敲出剩下的空格裡是什麼數字而在初盤的盤面上掀開的數字，我們稱之為提示數，圖表 1 即為簡易的數獨初盤，圖中的綠框為列、紅框為行。黃框為 3*3 的小方塊。

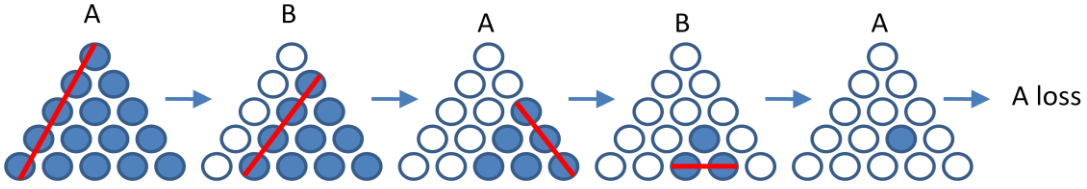


圖表 1 數獨行列方塊說明圖

在目前，我們找到最小提示數的初盤為 17，但尚無證明是否存在提示數為 16 的初盤，因此我們希望經由尋找 5,472,730,538 個終盤中，是否存在一個提示數為 16 的初盤而證明最小提示數為 16 或 17，若將該問題放在單一電腦運算，約需花費 2417 年的時間。因此我們希望可以透過桌機格網的技術協助解決該問題，而因 5,472,730,538 個終盤皆為獨立的盤面，因此在計算時，每個盤面可平行獨立運算，如此假設我們有 2500 台電腦，我們則可在一年內解決該問題[19]。

2.1.2 三角殺棋應用問題

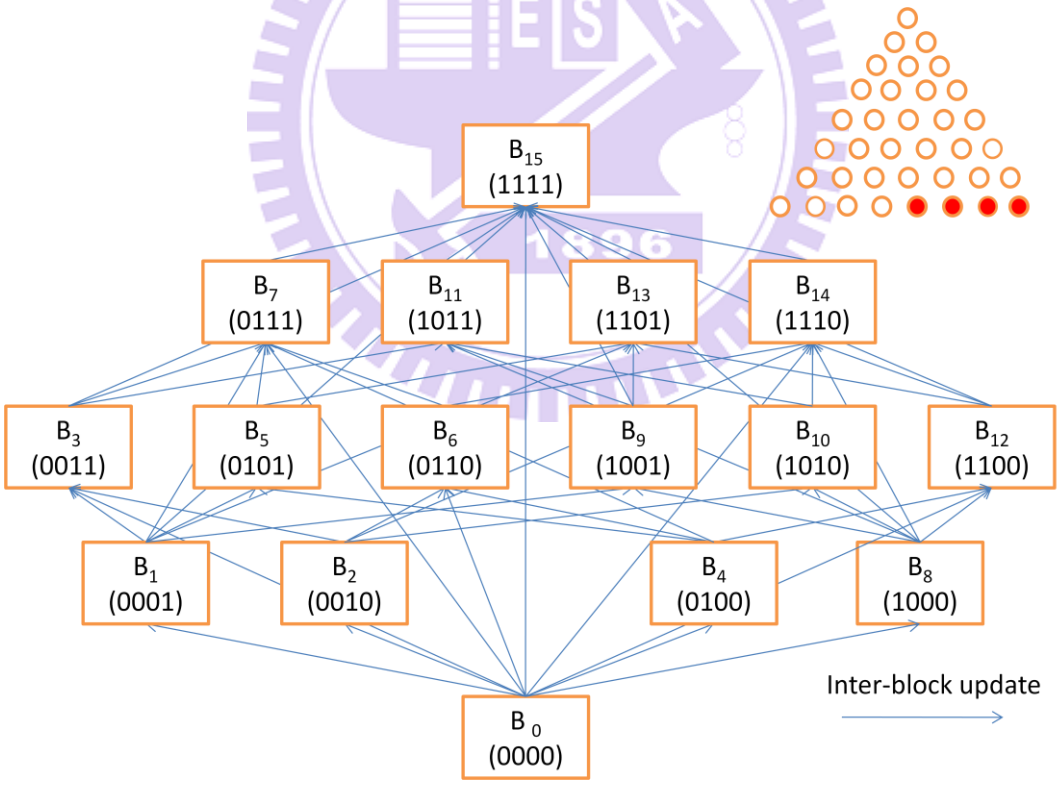
三角殺棋起源自中國，因當時的人們利用小石頭的堆砌而發展出的一套遊戲，規則上，若以5層的三角殺棋為例為玩家輪流取子，每次取的子必須在一條線上，最多可取 5 顆子，取到最後一顆子的人贏或輸，圖表 2 將以取到最後一顆子的人輸為舉例。



圖表 2 三角殺棋玩法舉例

在面對全解九層三角殺棋的問題時[20]，第一個關心的議題是: 此九層三角殺棋是必勝還是必敗，第二個關心的是：若必勝, 則怎麼勝？若必敗, 則對方怎麼勝？因此我們針對這方面的問題，希望能將九層三角殺棋不論規則是拿到最後一顆子的人輸或是贏，其全滿盤面結果都"全解"出來，而在此同時也將只要是九層三角殺棋以內所有盤面的必勝必敗結果都算出來

因九層三角殺棋有 45 顆子，若要將全部盤面的必勝必敗結果皆存下來，會有 2 的 45 次方種組合，必須耗費 4 Tera 的記憶體，是相當可觀的數字。而面對這樣的問題時，我們採取先將 4 Tera 切為每個 512MB 大小的區塊，並利用一種稱為回朔分析法的演算法，將所有盤面的勝敗結果皆算出，圖表 3 將以八層的三角殺棋為例，說明不同區塊之間勝敗結果更新的關係圖，其相依性的關係是由交大吳毅成教授研發團隊 2010 年所提出的演算法所決定。

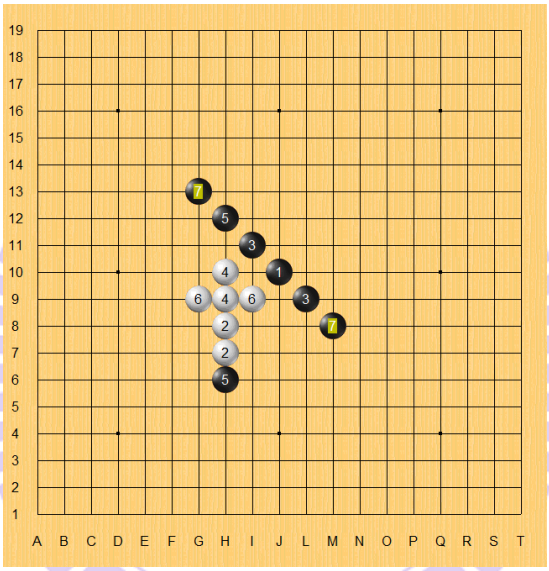


圖表 3 三角殺棋區塊結果更新關係圖[20]

由圖表 3 可知總共有 49 個連結，若為九層的三角殺棋，則可高達 48 萬條連結，其區塊間的結果傳輸量相當大。

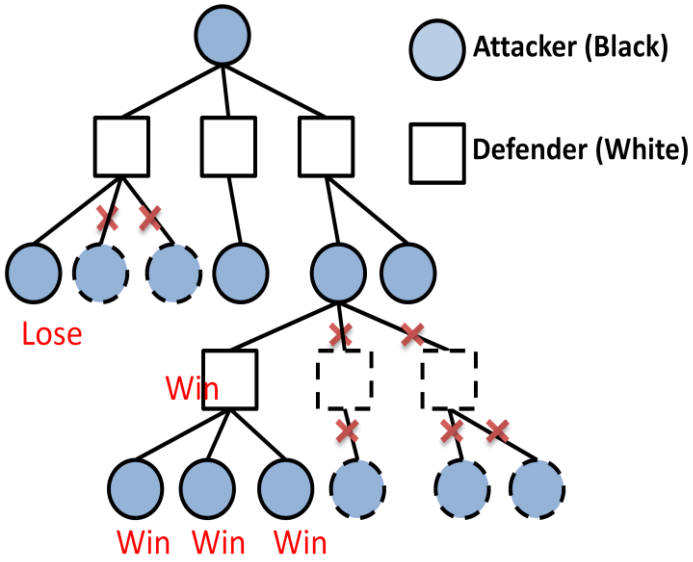
2.1.3 六子棋應用問題

該遊戲由交通大學吳毅成教授所發明，並於 2005 年 9 月發表於第十一屆國際電腦賽局發展研討會，現已成為國際奧林匹亞電腦賽局競賽項目。遊戲的特性為:規則簡單、變化複雜、遊戲公平。在規則上，一開始先由黑方下一顆子，之後黑白雙方輪流下兩顆子，先將六顆子連成一條線的人贏。圖表 4 為一個簡單的六子棋範例。



圖表 4 六子棋規則圖示(此例子為黑勝)

因六子棋在 2005 年才被提出，是個很新的遊戲，目前缺乏開局庫。面對這樣的問題，於是交大吳毅成教授研發團隊發展了一套建構在六子棋編輯器上的六子棋驗證系統希望能利用程式來幫忙大量的建構開局庫，而所使用的演算法為一種稱之為 Job-Level Proof Number Search (JL-PNS)的 PNS 演算法。但該演算法具有高度動態優先權的特性。如圖表 5 的例子，上頭標明的勝敗皆以黑的角度來看，若黑是必敗，白方必會選擇走這步，其餘分支即可不必計算，若黑方找出必勝路徑，則只要走這步即可，其餘分支也可不必計算。



圖表 5 JL-PNS 簡單說明圖

而為了盡快將這棵遊戲樹的結果證明出來，具有低延遲需求的特性，如此也可節省建構開局庫的時間。

2.2. 桌機格網

以下將介紹何為桌機格網及其與傳統格網的不同點，並說明目前的 XtremWeb 與 BOINC 這兩套著名的桌機格網系統不適用於電腦對局應用問題的原因，而由交大吳毅成教授研發團隊所開發的桌機格網之特殊性為何[2] [11]。

2.2.1 桌機格網簡介

桌機格網起源於 1996 年，又可被稱為志願者計算(Volunteer Computing)，因其工作端皆為志願提供電腦資源，亦可看做志願者。而桌機格網可看視為是分散式計算(Distributed Computing)的延伸，它透過網路收集閒置的桌機資源，任何想分享資源(Resource)的電腦都可以自由的加入以及離開且具有彼此配備差異、資源動態性比較高等特性。而它在經濟和便利性上的優勢，使得它逐漸取代超級電腦，成為現在運用在大量計算上的主要系統[12]。

2.2.2 桌機格網與格網(Grid)的比較

在桌機格網被提出以前，先有格網的概念，格網最早被提及是在 1995 年，名為 I-WAY 的實驗中。經由這個實驗，北美的 17 部高階主機，透過高速網路在很短的時間內相連接，彼此可以很容易的互相存取資源，形成最早的電腦格網。而這種資源分享的概念與電力格網(Electric Power Grid)很相似，之後格網(Grid)一詞便出現了。桌機格網與格網有許多的不同點，在此依照資源(resource)、連接性(connection)、異質性(heterogeneity)、可信度(trust)等，我們繪製成表格做比較，如下表[6]：

	桌機格網	**格網**
資源 (Resource)	個人桌機	超級電腦、叢集(cluster)、資料庫、儲存裝置
連線 (Connection)	低頻寬且大部份機器位於防火牆之後	具有高頻寬連線
異質性 (Heterogeneity)	高異質性	低異質性
可信度 (Trust)	也許會有惡意的工作端，必須再驗證結果的正確性。	可以高度信賴這邊提共的資源。
可靠度 (Reliability)	可靠度很低	有較高的可靠度
專一服務性 (Dedication)	工作端的變動很大，可能隨時離線不提供服務	有較高的專一服務性
管理性 (Manageability)	工作端分散各地，管理不易	有集中管理的機制
應用模式 (Application Model)	屬於高生產量(High-throughput)導向	屬高效能(High-performance)導向
排程 (Scheduling)	不具有本地排程者，可能有一集中(centralize)的排程者，也可能一組分散式(distributed) 排程者	一般屬於階層式的排程，有分為本地(local) 排程以及 meta 排程

圖表 6 桌機格網與格網比較表

- **資源(Resource) :**

由於近年來個人電腦越來越便宜，桌機格網的資源主要來自各地的個人電腦所組成的，遠不同於格網由超級電腦所組成，以達到相同大量運算的目的；然而，桌機格網相較於格網，也有著各種不同的挑戰，比如，它屬於高度變動的環境，因為參與的工作端隨時都有可能離開而中止計算，另它屬於高異質性的機器所組成。

資源管理系統在兩種格網中扮演重要的角色，影響整體格網效能甚大，所以，一般在於資源選擇(Resource selection) 機制做各種調整以改善整體的效能，主要分為三大類：

資源優先性(Resource Prioritization)：此方法會依工作端回傳的 clock rate 等資訊，將各工作端排序以得各優先順序，然後，有新工作將會優先分配到表現優良的工作端以改善效能。

資源排除性(Resource Exclusion)：此方法會依工作端回傳的資訊或者各種預測計算，將可能已有工作執行的工作端不再分配到工作。

工作重複分配(Task Replication)：由於擔心某個工作在某些機器不預期地拉長執行時間，所以，在於資源充足的條件，可以將相同工作分配到多台的工作端以避免該問題。

- **連線(Connection) :**

格網的主機通常在同一組織內有相當數量的機器，所以，它們之間具有高頻寬的網路連線，比如光纖網路，所以，可以進行大量交換資料的運算，比如 MPI 的運算；桌機格網的工作端則大部份處於一般 ADSL 的連線環境下，甚至處於防火牆及 NAT 內，所以，桌機格網受限於低頻寬連線的環境，工作端之間也少有做交換資料的行為。桌機格網的工作端間的連線如果可以善於利用，伺服器端的負荷將可以降低很多，比如可以用 P2P 的方式，將伺服器的檔案藉由工作端相互傳送以減少伺服器端的負荷。

- **異質性(Heterogeneity) :**

桌機格網都會面臨到不同工作端有不同的資源性質，比如不同作業系統、記憶體大小、CPU 能力等，伺服器就必需提供資源發掘(Resource Discovery) 的機制，BOINC 及 Condor 都有提供類似的功能，可以依工作對於資源需求而做適當的分配。

另一種針對異質性的作法，是將虛擬機器(Virtual machines) 應用在格網計算上，以減少工作端的異質環境條件，比如 VMPlant 及 Entropia Virtual Machine 分別應用在格網以及桌機格網，格網的開發人員只要面對同質性的環境即可，可以提高相關程式的產量。

- **可信度(Trust) :**

由於可能有惡意的工作端機器故意回傳錯誤計算結果的可能，一般會一些機制來防止此種情形，這種機制稱之為惡意容錯技術(Sabotage-tolerance technique)，可以分為三類：

重複及表決(Replication and voting)：又稱為 double check 或 majority voting，主要的方法即每個工作會分配給二個不相關的工作端，並比較兩個回傳的結束是否一致，或不一致的話，會再分配給第三個工作端，多數表決回傳結果哪一個為正確結果。

取樣(Sampling technique)：重複及表決的方法會造成資源重複計算的問題，取樣的方式即會針對新加入的工作端做測試的機制，可能事前提供一些有正確結果的工作分配給這些工作端，觀察這些新加入的工作端是否為惡意機器。

查核點確認(checkpoint-based verification)：每個工作端每隔一段時間將當下工作的狀態儲存下，產生 hash 值並傳回伺服器；伺服器則會不定期取出某一時間點的查核點加以計算到下一個時間點的查核點，伺服器任會要求工作端的也從該時間的查核點，計算相同的工作並回傳伺服器，伺服器會依其回傳結果是否相同來判斷是否為惡意工作端。

- **可靠度(Reliability) :**

可靠度的問題不論什麼系統都會遭遇到的問題，不外乎硬體錯誤(比如機器當掉、網路斷掉等)、軟體錯誤(memory leak, numerical exception 等)以及其他情形(使用者重新開機、機器 CPU 負荷過重等)。這分為兩方面，一是監測錯誤(failure detection)的機制，另一個為錯誤恢復(failure recovery)的機制。監測錯誤機制最簡單的方法即是每個工作端機器會定期回傳訊息(又稱為 heartbeat)，若有錯誤時，即不回傳，在伺服器端有一接收程式來判斷是工作端是否有錯誤發生；恢復錯誤的機制有三種，分別是：重試(Retry)：即重跑該工作，看是否仍有錯誤;重複(Replication)：在於機器當做的情形，要在不同工作端進行重新執行;查核點(Checkpoint)：若系統有做查核點備份時，即可搬遷到其他台，並從最新的查核點再重新執行。

- **專一服務性(Dedication) :**

由於桌機格網是屬於個人志願加入的模式，所以，這些工作端是隨時可以中斷的，這也是它的特性之一。有相關研究是將具有專一服務性的叢集(Cluster) 和不具專一服務性的桌機格網加以整合，讓格網系統更具有拓充性，比如 Sztaki Desktop Grid，如何將工作分配這兩種不同的系統，將是其重點。

- **管理性(Manageability) :**

當桌機格網規模大到上千台時，再加上其屬於工作端，並不能取得其管理者權限來進行操作，所以，如何管理規模大到上千台，將是這個部份的重點。

- **應用模式(Application Model) :**

相對於格網屬於要求最短時間完成一個工作，稱之為高效能(High Performance)導向，早期叢集運算及格網運算，都屬於此類。桌機格網則不見得要最短時間內執行完畢，而是要求一個期間內，有著最大量的利用性，稱之為高生產量(High Throughput)導向。

2.2.3 相關系統及應用

這裡介紹目前桌機格網的系統及一些應用實例，首先將先介紹目前兩大桌機格網系統:BOINC 與 Xtremweb，分別在 2.2.3.1 與 2.2.3.2 兩章節中做介紹，在 2.2.3.1 章節中將介紹 BOINC，2.2.3.2 章節則介紹 XtremWeb，並在 2.2.3.3 章節說明 BOIN 與 XtremWeb 這兩套桌機格網系統不適用於電腦對局應用問題的原因，而後在 2.3.4 章節中介紹由交大吳毅成教授研發團隊所開發的桌機格網其架構與特殊性，而此桌機格網系統在本篇論文中稱該系統為 Computer Game Desktop Grid (CGDG)。

2.2.3.1 BOINC 介紹

BOINC[2][6] (Berkeley Open Infrastructure for Network Computing)是一個有名的桌機格網運算的中介軟體，科學家依此計劃可以很容易地建立公開資源運算的專案，用以處理各種科學上的運算，最有名的計劃包括 SETI@Home、Background Pi 計畫（利用多餘計算能量來計算π）等。這些計畫其實包括其他生物科技、地球科學、數學、物理及天文等應用問題。

BOINC 一般分為伺服器及工作端兩部分，伺服器部份包括儲存工作、分配工作、傳檔、排程等功能；工作端部份則定期用 pull mode 的方式向伺服器取得要執行的工作，執行運算該工作，並回傳結果到伺服器，工作端若有參與多項專案時，工作端也會進行簡單的本地排程功能。當電腦閒置時，會開啟螢幕保護程式，並且開始計算工作。如果工作端想使用電腦時，螢幕保護程式會立刻退出，所以並不會影響到工作端使用電腦，透過這種方式來充分利用電腦待機時的運算資源。

SETI@Home 是一個著名利用 BOINC 這套分散式計算系統的專案之一，透過網路連結在一起的電腦資源，研究人員嘗試著透過分析『無線電波 SETI』

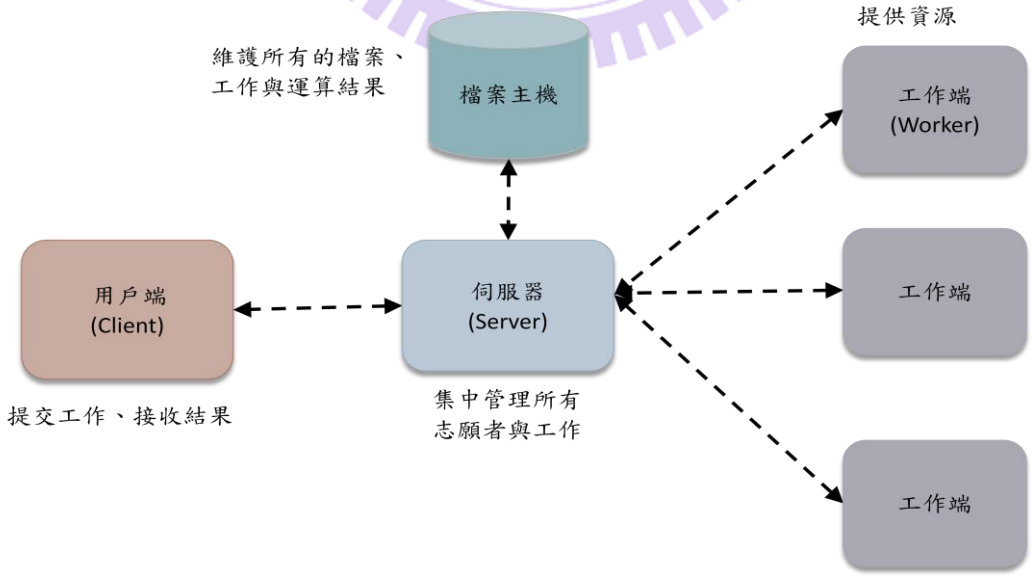
(Radio SETI)尋找是否有外星文明的存在。

2.2.3.2 XtremWeb 介紹

XtremWeb [6][9][21]是一個用 Java 撰寫的中介軟體，用於大規模的分散運算，和 BOINC 類似，工作端的機器會用 pull mode 方式向伺服器取得工作運算，而伺服器分配到工作端的排程方式屬於 FIFO (First In First Out) 的方式。此系統主要應用於物理計算這種大量平行計算的應用。此系統包含使用者、工作端、伺服器三個部分，工作端會定期回報仍在運作的狀態給伺服器，若伺服器在某個期間內都沒有得到該工作端回報目前運作的情形，則伺服器則會視該工作端已不存在，並會將該工作端未做完的工作重新分配到其他台工作端。

2.2.3.3 BOINC 與 XtremWeb 不適用於電腦對局應用的原因

圖表 7 為一般 BOINC 與 XtremWeb 的概略架構圖。

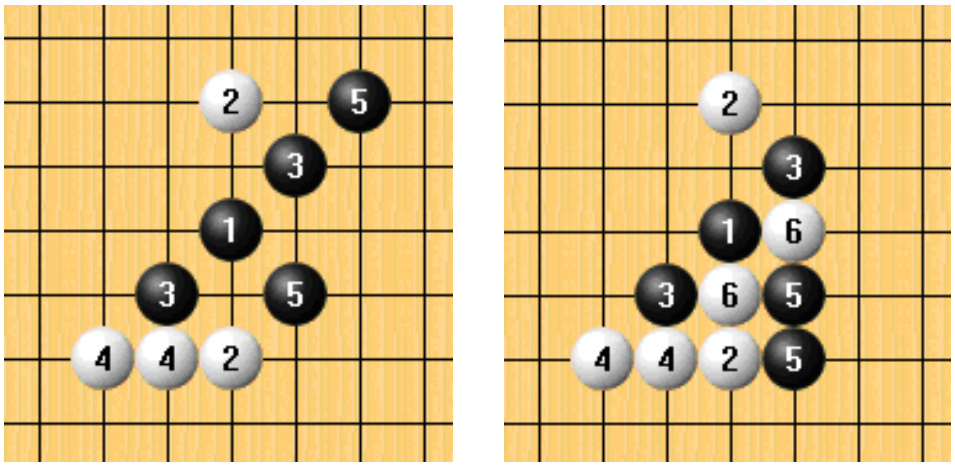


圖表 7 目前桌機格網系統（如 BOINC, XtremeWeb）之架構

由圖表 7 可知其架構包括用戶端、伺服器、工作端，以及檔案主機等部分，他主要包含用戶端(Client)、工作端(Worker)、伺服器(Server)等三個部分，用戶端提供應用程式提交工作(Job)給伺服器，並且查詢結果，工作端提供資源可看成我們一般的桌機，用來幫忙協助運算，當其處於閒置狀態的時候捐獻出電腦的資源。伺服器則集中管理所有工作端與工作，可以透過檔案主機去協助維護所有被使用的檔案以及儲存工作跟運算結果[12][17]。

由於這兩套系統主要應用於物理、醫學、生物科技等這類不需要即時回傳結果且工作之間是彼此獨立的應用，因此當工作端將工作下載下來時就會與伺服器斷線，且因彼此工作獨立的關係，對於每個工作可以給予一個固定的工作優先權，等到工作執行完畢再將結果回傳到檔案主機，使用者再透過網頁的方式去瀏覽運算的結果即可，然而，在這樣的模式下，當我們想要將電腦對局應用問題應用在此兩套系統時我們發現無法應用在其上，例如在六子棋的應用上，當我們發現一個優勢較高的走法時，我們希望其相關的工作可以擁有較高的優先權，如此可以幫助我們盡快的解掉這個問題，若一開始就給予一個固定的優先權，則將會浪費許多無意義的運算。

例如我們想要證明，圖表 8 中之黑在第三手的時候是必勝。若白方第四手下在圖表 8 左圖中位置，則黑在左下圖的第五手棋是必勝，若黑的第五手下在右下圖的位置，白在第六手的時候可擋住黑方的攻擊。若左下圖中的黑的第五手下法有更高的優先權，則我們可以在更早的時間結束此開局的分析，反之，則會算更久。因此，在這樣的情況下，若是給予工作固定的優先權，將無法適用於這類的電腦對局應用問題。而當我們發現一個必勝的走法時，其餘的運算即可取消，如是在工作端將工作下載後就馬上斷線的情況下，將會浪費許多不要的運算，且在等待結果回傳的同時，也浪費了許多等待的時間。

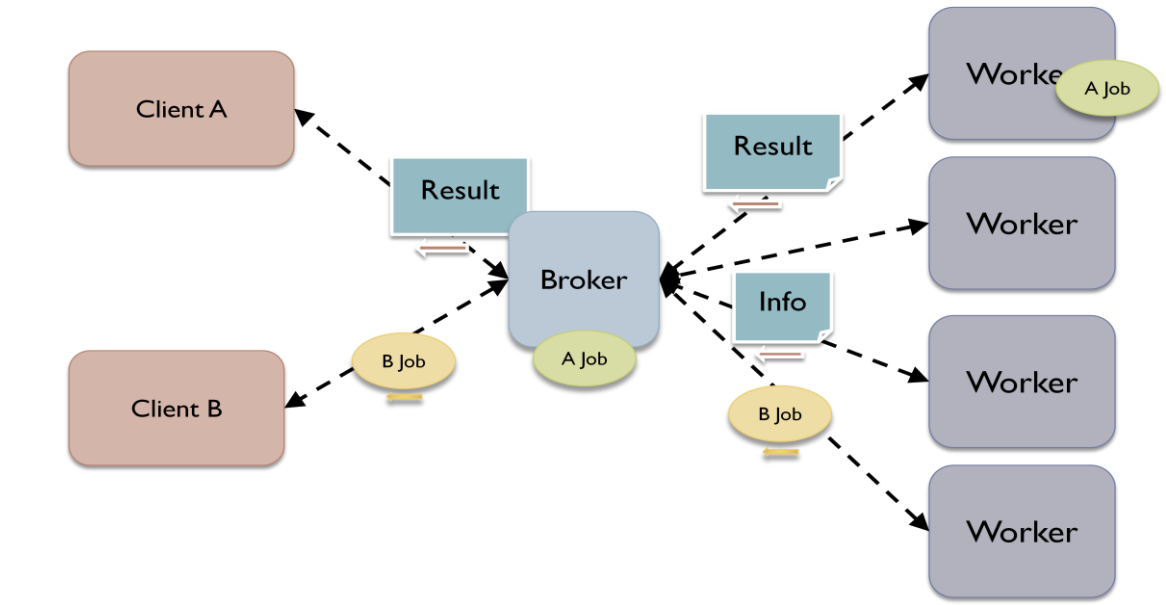


圖表 8 高度動態工作優先權(Job Priority)問題說明圖

2.2.3.4 CGDG 介紹

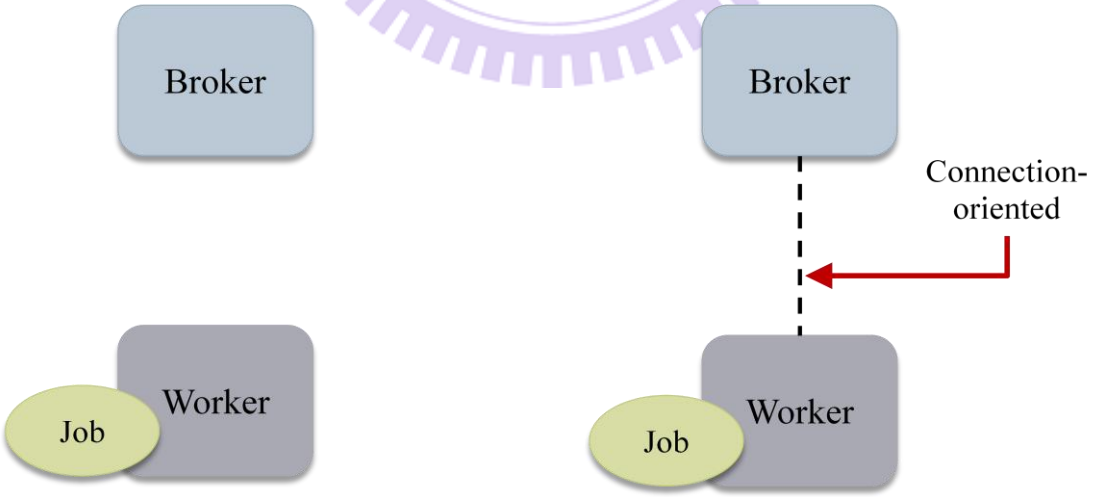
面對上述 BOINC 與 XtremWeb 不適用於電腦對局應用問題的原因，我們提出了一套桌機格網系統的做法，該系統全名為 Computer Game Desktop Grid，簡稱 CGDG，之後將針對系統的架構以及系統的特殊性做說明[17]。

在系統的架構方面，CGDG 為一個三階層式架構的系統，如圖表 9，有用戶端(Client)、協商者(Broker)與工作端(Worker)三個部分，用戶端產生工作並接受結果，協商者負責協調一切工作，管理協商所有的客戶、工作與工作端並負責轉送資訊。工作端則幫忙做工作的運算。整個系統運作流程主要為一開始協商者必須先運作起來，之後會一直聆聽客戶端與工作端的連線，而對於每一個客戶端，協商者只會保留一份工作在協商者的工作序列中，當一有工作端開始處於閒置狀態的時候，協商者會將佇列中的工作分派給工作端去幫忙作運算，當工作端將工作執行完畢以後，會馬上將結果傳送給協商者並要求一份新的工作去幫忙作運算，而協商者在接收到工作執行的結果以後，會馬上將結果轉送給客戶端，並向客戶端要求一個新的工作來做，在系統運作的期間，工作端會定期的向協商者回報工作端目前的運作情形以及工作端的系統資訊，客戶端可透過管理介面觀察系統目前運作的狀態。



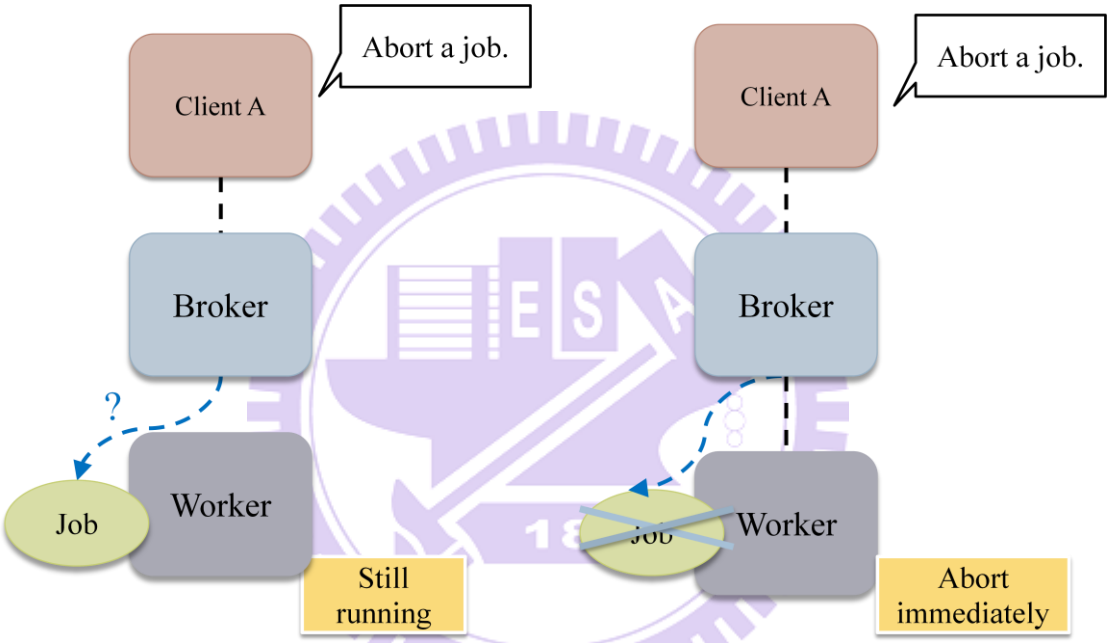
圖表 9 CGDG 系統架構圖

在系統特殊性方面，該系統主要是針對電腦對局應用上的特殊需求，在傳統的桌機格網無法滿足的情況下提出一套解法，在傳統的桌機格網上，因其主要為物理、生醫等不需要即時回傳結果的應用，其連線方式多為圖表 10 左的方式，即工作端將工作帶回家做時，與協商者即斷線，但對於電腦對局應用上，我們會希望結果能即時的回傳，如此才可減少多餘的時間浪費，為了達到此目的，首先，我們採用的策略是 TCP 的通訊協定，而非 HTTP 連線來達到此需求。



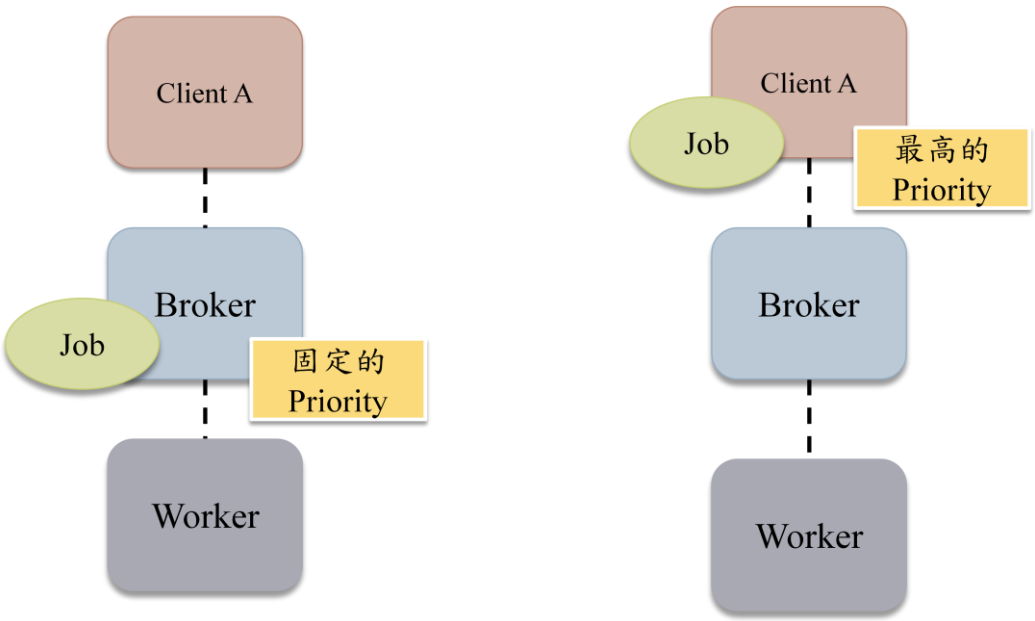
圖表 10 傳統桌機格網作法(左圖)及 CGDG(右圖)之連線方式

而在傳統的桌機格網作法上，如下圖左因工作端下載完工作後即斷線，用戶端則無法將正在執行中的工作中止，但在電腦對局的應用上，如果我們已經找到一個必勝走法，則其餘的計算則可不用再作運算，但因已經斷線的原故，所以無法將目前執行中的工作中止，針對這樣的問題，在 CGDG 系統中，我們提供中止工作的功能，藉由連線採用 TCP 的方式，我們能夠得知目前工作端的運作資訊，工作在工作端的計算過程中，能夠直接對工作端下達中止工作命令，如此可避免不必要的運算。



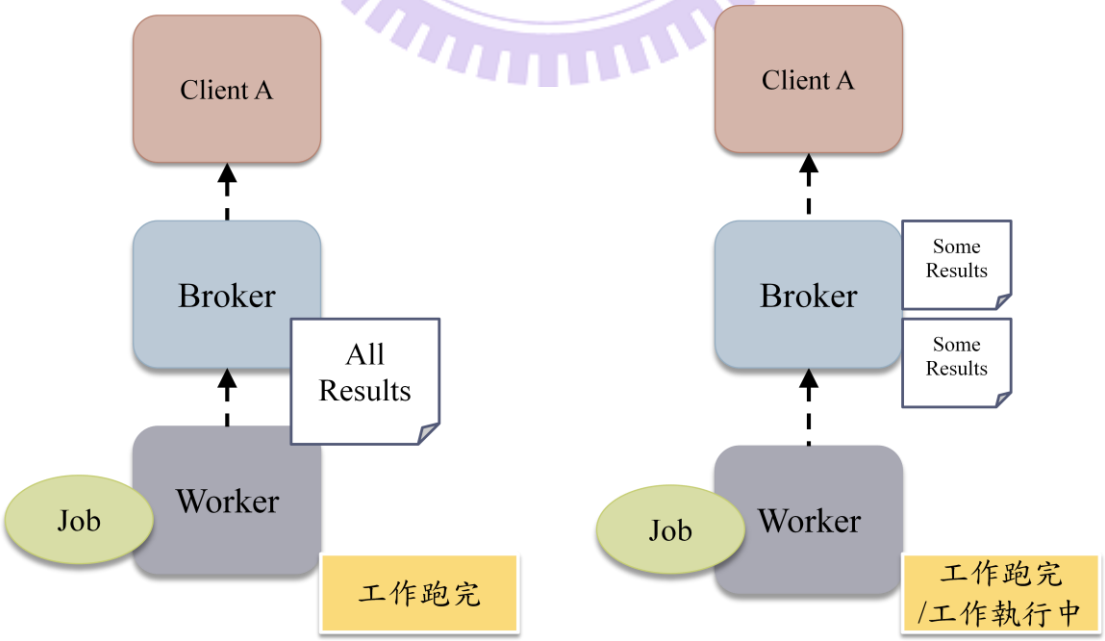
圖表 11 傳統桌機格網作法(左圖)及 CGDG(右圖)之中止工作

除此之外，為了適用於電腦對局應用這類高度動態優先權的應用，我們採取的作法並非如傳統桌機格網上，一開始即將工作的優先權設定好一個固定的值，全部丟到協商者上去控管，而是將選擇要丟哪一個工作的決定權寫在客戶端，客戶端會經由該應用的演算法結果選擇一個當前需優先處理的工作，給予最高的工作優先權，而後藉由協商者將該工作丟到遠方的工作端執行，如圖表 12。



圖表 12 傳統桌機格網作法(左圖)及 CGDG(右圖)之工作優先權設定

而對於像物理科學那種彼此工作事完全獨立的應用，即使所有的結果一次回傳，也不會影響結果，但是像電腦對局這類的應用，前面運算出來的結果會影響到後面的運算是否須要繼續做，因此像傳統桌機格網將等所有的結果運算完畢才一次將結果回傳，是不適用的，於是在我們的 CGDG 系統中，採取的是串流(Streaming)的方式，在執行的過程中就將結果回傳，如此可加速遊戲樹的結果證明，並節省建構開局庫的時間，如圖表 13。



圖表 13 傳統桌機格網作法(左圖)及 CGDG(右圖)之結果回傳方式

第三章、設計與實作

在這章節中將針對交大吳毅成教授研發團隊所開發的 Computer Game Desktop Grid (CGDG)之應用層框架一般化以及資源分配管理的設計與實作部分做說明，在 3.1 章節中會介紹應用層框架一般化的設計與實作部分，在 3.2 章節中則會介紹資源分配管理的目的與實作方法。

3.1 應用層框架一般化

在這邊我們針對應用層框架一般化部分的設計與實作部分做說明，在 3.1.1 章節中先介紹框架的設計概念，其所設計的三個模組:應用層、網格核心模組、網格圖形使用者介面模組，所提供的功能為何，在 3.1.2 章節中，則會針對應用層框架一般化的實作部分做說明，將深入描述應用層、網格核心模組、網格圖形使用者介面模組此三個模組的實作內容。

3.1.1 設計

在早期，這套 CGDG 桌機格網剛發展之時，是為了解決建構六子棋開局庫的問題而誕生的，但到群體計劃執行之後將會有許多不同的電腦對局應用問題想加入此系統，因此若能將應用層框架一般化，則可讓其他的電腦對局應用問題可以快速的加入此系統。

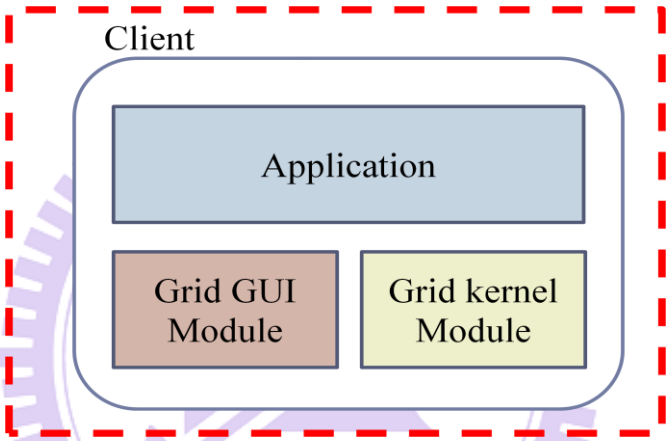
在設計概念上，我們將原本僅適用於六子棋應用問題的客戶端，模組化為應用層(Application)、網格核心模組(Grid kernel Module)、網格圖形使用者介面模組(Grid GUI Module)三個部分(圖表 14)。

在應用層方面，即為數獨、三角殺棋、六子棋等電腦對局應用問題演算法核心的部分，此處會產生應用工作，之後將該工作丟給網格核心模組去控管與並針

對網格應用模組回傳的應用工作結果做相應的處理。

在網格核心模組方面，該部分會負責所有與協商者(Broker)連線相關的事情，包括處理底層的標準輸入輸出(I/O)、解碼協商者傳回的訊息並呼叫相應的處理函式，與編碼應用層或網格圖形使用者介面模組產生的指令將之傳往協商者。

在網格圖形使用者介面模組方面，主要目的為提供有用之網格圖形使用者介面框架，使網格圖形使用者介面設計更容易(一般化)。如: 工作執行時的相關設定(包括要使用機器的數量、性能與使用者登入介面...等)、監控目前工作執行的狀態...等。



圖表 14 應用層一般化模組圖

3.1.2 實作方式

在這章節中，應用層框架一般化較詳細的執行流程，包括應用層、網格核心模組、網格圖形使用者介面模組三個部分，以下將針對這三個模組進行介紹。

- **應用層:**

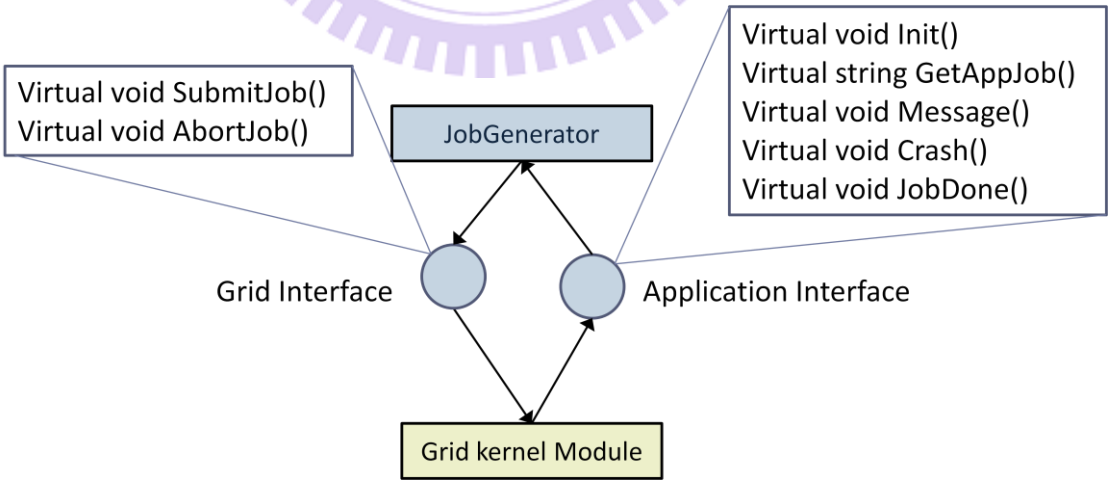
在應用層這部分，主要為電腦對局應用問題的演算法核心，所以我們將其切分為工作產生器(JobGenerator)、網格接口(Grid Interface)，和應用層接口(Application Interface)三個執行模組，整個應用層的執行模組圖可參照圖表 15。

在工作產生器部分，即為電腦對局應用問題的演算法核心，在系統運作的期

間，會產生新的應用工作，包括當工作被丟到遠端電腦時，人工智慧須(AI)要用到的參數、執行檔名稱...等，並依照先前執行完畢的工作結果產生相應的工作處理。

在網格接口部分，扮演著工作產生器與網格核心模組之間的溝通橋樑，讓工作產生器可以藉由這個接口裡的 SubmitJob()函式將產生的新應用工作轉往網格核心模組去統一控管，工作產生器也可以透過這個接口裡的 AbortJob()告知網格核心模組將工作取消。

在應用層接口部分，與網格接口一樣扮演著工作產生器與網格核心模組之間的溝通橋樑，但不同的是，網格接口為工作產生器將訊息傳往網格核心模組的接口，應用層接口則是網格核心模組將訊息傳往工作產生器的接口，網格核心模組可透過呼叫 Init()對工作產生進行初始化，若協商者向用戶端要求一個應用工作時，網格核心模組發現沒有等待的應用工作時，可透過 GetAppJob()取得一個應用工作來執行，當一有應用工作的結果回傳時，網格核心模組可透過 Message()傳送工作執行實的結果訊息，若有工作執行失敗時，網格核心模組會呼叫 Crash()告知工作產生器執行失敗的訊息，若工作執行完畢時，網格核心模組會透過 JobDone()通知工作產生器工作已執行完畢。



圖表 15 Application 執行模組圖

- **網格核心模組:**

在網格核心模組這部分，主要負責所有與協商者連線相關的事情，包括編碼器(Encoder)、標準輸入輸出(I/O)、解碼器(Decoder)、系統接口(System Interface)、指令處理者(Command Handlers)這幾個部分。

在編碼器部分，負責將用戶端產生的指令轉為可與協商者溝通的訊息格式，而在我們系統中，都是以 XML 的格式溝通，在 XML 內包括使用者識別碼、工作識別碼、工作端識別碼、應用程式的名稱、版本編號，以及要執行的工作參數等資訊。標準輸入輸出部分，則是負責用戶端與協商者之間的訊息接受與傳送。解碼器部分，則是會將協商者傳往用戶端的 XML 格式的訊息轉成用戶端可讀的訊息。

在系統接口部分則是針對協商者傳往用戶端的指令做一個訊息的接口，對於每一個傳往用戶端的指令都會有一個對應的指令處理函式，而這些指令處裡函式定義會寫在指令處理者裡面，傳往用戶端的指令包括使用者初始設定(InitUser)、通知使用者可以丟工作(NextJob)、工作已指派出去(AssignDone)、工作已被執行(Running)、工作執行中的回傳的結果(Message)、工作執行完畢(Finished)、工作執行失敗(JobCrashed)、某工作端離開此系統(WorkerDisconnected)、傳送工作端的相關系統資訊(Info)。而每個指令與處理函式的對應方式可表示成下表 16。

指令	處理函式
InitUser	Virtual void InitUser()
NextJob	Virtual void NextJob()
AssignDone	Virtual void AssignDone ()
Running	Virtual void Running ()
JobCrashed	Virtual void JobCrashed ()
WorkerDisconnected	Virtual void WorkerDisconnected ()
Info	Virtual void Info ()

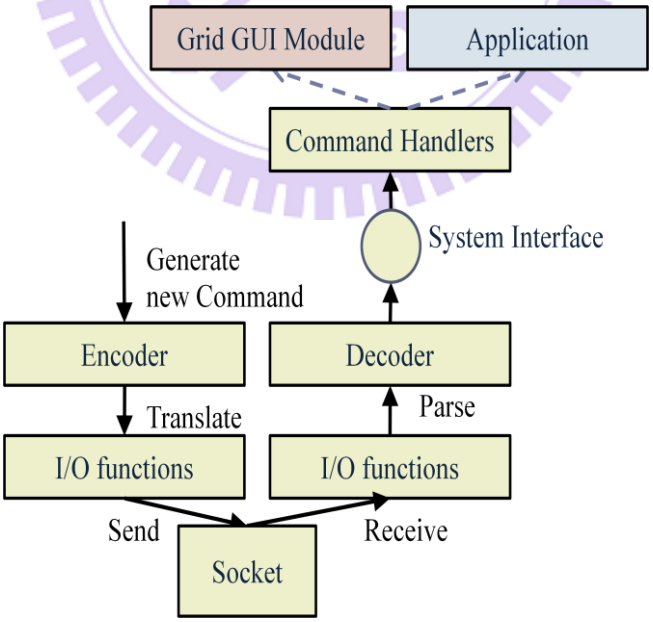
圖表 16 訊息與處理函式對應表

整個格網核心模組圖表 17，若是以六子棋為例，說明整個格網核心模組的流程，在發送指令時會有下列步驟：

1. 六子棋產生送出盤面的指令 。
2. 編碼器將所產生的指令加入使用者識別碼、工作識別碼、工作端識別碼、應用程式的名稱、版本編號，以及要執行的工作參數等資訊，並將格式轉換成與協商者溝通用的 XML 格式。
3. 透過標準輸出將放在 Send Buffer 內容從 Socket 傳送出去。

若是在接收到盤面的運算結果回傳時會有下列處理步驟：

1. 從 Socket 接收到訊息之後，放到 Receive Buffer 中等待處理。
2. 解碼器透過解碼回傳的訊息，轉成用戶端可讀的格式。
3. 系統接口將解碼器轉換出來的指令(Message)對應到相對應的處理函式(Message())。
4. 處理函式會將接受到的盤面計算結果告知應用層，應用層則可做相對應的結果處理。

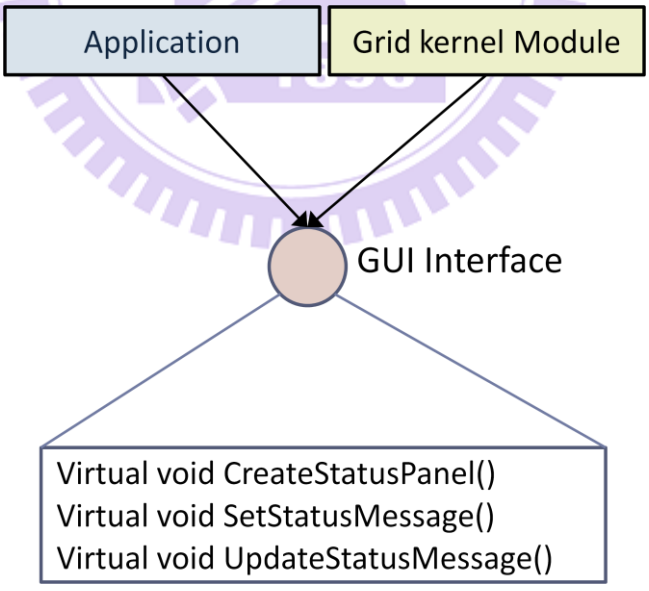


圖表 17 網格核心模組

- **網格圖形使用者介面模組:**

這部分主要實作一個圖形使用者介面(GUI)的框架，如監看目前工作執行狀態的圖形使用者介面(可監看工作內容、工作狀態、工作跑的時間)、使用者登入圖形使用者介面(包括帳號、密碼、優先權、備註等的設定)、工作執行相關初始設定圖形使用者介面(可設定應用程式名稱與版本及想使用的工作端數量)、監看工作端系統資訊圖形使用者介面(包括網路位址、中央處理器速度、記憶體大小等的資訊)，以及過濾器圖形使用者介面(可設定要使用的工作端為哪幾台)等部分。

不論是應用層或網格核心模組皆可透過圖形使用者介面接口(GUI Interface)對圖形使用者介面進行操作與設定。可透過 CreateStatusPanel()來產生新的圖形使用者介面，並透過 SetStatusMessage()來設定每個圖形使用者介面欄位要顯示的訊息，一有訊息更新時可透過呼叫 UpdateStatusMessage()來更新圖形使用者介面欄位的訊息。整個網格圖形使用者介面模組可表示為圖表 18。



圖表 18 網格圖形使用者介面模組圖

3.2 資源分配管理

在此章節將會描述資源分配管理的目的、策略，以及想解決的問題。在此討論的使用環境為當群體計畫開始執行之後有多個組織與使用者加入時該如何分配資源及管理。在 3.2.1 章節中先介紹資源分配管理的目的和所使用的情境為何，在 3.2.2 章節中，則會介紹資源分配管理的實作方法為何。

3.2.1 目的

在這套系統使用情境裡主要的角色有兩類，一個是使用者，一個是資源的提供者。以下將針對這兩個角色進行角色說明及加入該系統的動機作說明。

使用者在這套系統中即是產生工作的人，使用者會使用遠端的機器執行計算，以群體計畫的環境來說，就是進行人工智慧的運算，如:三角殺棋、六子棋、數獨…等。

資源提供者則是會加入這套系統中，並貢獻電腦的計算資源，資源提供者可分為兩大類:組織及其他，組織即是一般我們所熟知的學校單或研究機構，若不屬於上述的兩單位的資源提供者則會被歸屬於其他類別。

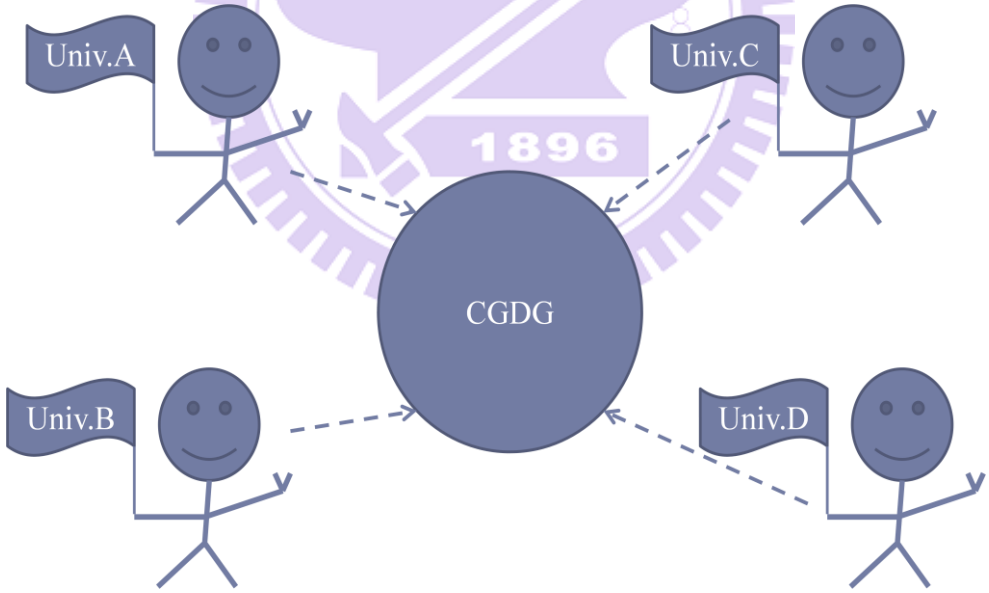
但此兩類不同的資源提供者在加入該系統的動機上卻有所不同，組織的動機主要為貢獻資源，與其他組織分享組織內的機器，但當自己組織內的人要使用機器時，會希望自己組織內的機器，自己可以優先使用，而這部分，也是我們在採取資源分配管理策略時必須考慮的地方。相對於組織的資源提供者，其它的資源提供者只是抱著純貢獻的心態加入了這個系統，他們的圖的是在參與的同時可以獲得表揚與參與感，如:當我們解出數獨最小提示數為 16 或 17 時，可以列出有幫忙運算的人的名字，以此表揚它們。

而我們在做資源分配管理的首要目的，當然就是分配資源與管理。而根據上述的使用情境，我們採取的分配管理策略目標為希望能讓提供越多計算資源的組

織，可以使用較多的計算資源，且使用者在使用所屬組織計算資源時，相較於其他組織的使用者具有較高優先權，而因是提供電腦對局應用問題方面的應用，我們希望可以提供在實驗、展示、或比賽時使用這套系統的使用者具有佔用機器不被他人使用的權限，以電腦對局應用問題跑實驗數據來說，該權限顯得很重要，因在跑實驗數據時會希望不被外界干擾以求數據準確。

3.2.2 方法

在這個情境下，因會有多個組織的不同使用者加入這套 Computer Game Desktop Grid (CGDG)桌機格網系統中,所以我們必須先分辨使用者的所屬組織，如圖表 19 所示，當 Univ.A、Univ.B、Univ.C、Univ.D 的使用者連入時，能分辨每個使用者的身分，而對於每個使用者在登入時都須先輸入帳號、密碼。



圖表 19 使用者所屬組織示意圖

而在這樣的情境下，我們首先必須建立一個帳戶管理機制，為每一個使用者註冊一個帳戶，而每個帳戶皆包含使用者所屬組織(如交大)、使用者管理權限、使用者使用權限等資訊。

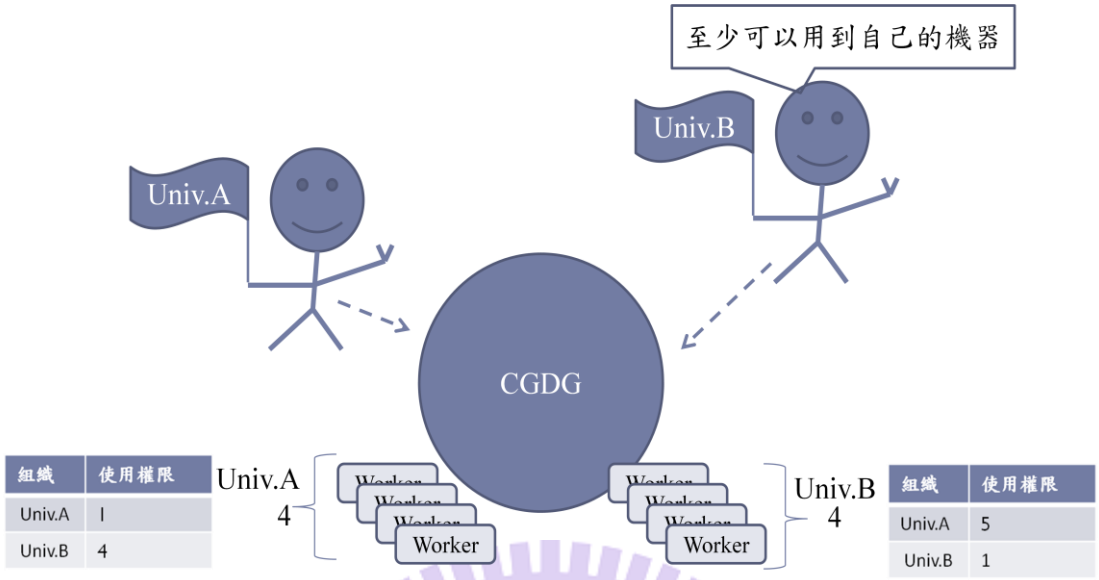
上述所提到的管理權限，是指我們會先建立所謂的帳戶管理者，其中包含系統管理者(System Administrator)、組織管理者(Organization Administrator)兩種角色，系統管理者可管理整個系統的使用者帳戶，不論是一般使用者或組織管理者都接受其帳戶管理，而組織管理者可建立組織內的擁有機器烈表，以利將來在做資源分配管理時，可對資源捐贈者進行所屬組織的分群，並設定不同組織的使用者在組織內的使用權限設定。並管理組織內的帳戶。

而在使用權限方面，具有實驗、展示、比賽用權限及一般權限，兩種不同的權限設定。具有實驗、展示、比賽用權限的使用者可以佔用機器，不被他人使用，是為了方便用戶端的使用者在跑實驗數據時，可確保所跑環境不被他人干擾，直到數據跑完時，才將機器給別人使用，在整個系統的等級最高。 而具有一般權限的使用者優先權越高等級越高，同等級的則依使用者的機器佔有率，佔有率較少的等級較高，在等級方面，區分為 1 到 7，7 個等級，數字越小的等級越高。

但有時，使用者未必想發送該最高權限的工作， 因此，我們提供了可設定較低優先權工作的功能，在 Session 優先權與設定的使用權限之間具有最高優先權即為使用者的使用權限這樣的關係。如使用權限為 1 的使用者，可發送 1 到 7 優先權的工作，而使用權限為 2 的使用者，則可發送 2 到 7 優先權的工作，其餘則依此類推。

在做完上述的設定後，在組織間的 Session 仍存在問題，如:某組織內的使用者明明有捐機器卻搶不到自己所屬組織內的資源，因此，我們採取的策略為先依照組織管理者設定的組織內的機器列表，將機器依照組織分群，若機器無法對應到任一組織，則歸類為其它。

而後，我們將組織管理者設定的使用者在該組織內的使用權限納入排程的考量，針對不同的使用者會有不同的優先權，如此，使用者在所屬組織內的機器具有較高優先權，即可搶到機器了，如圖表 20 所示。



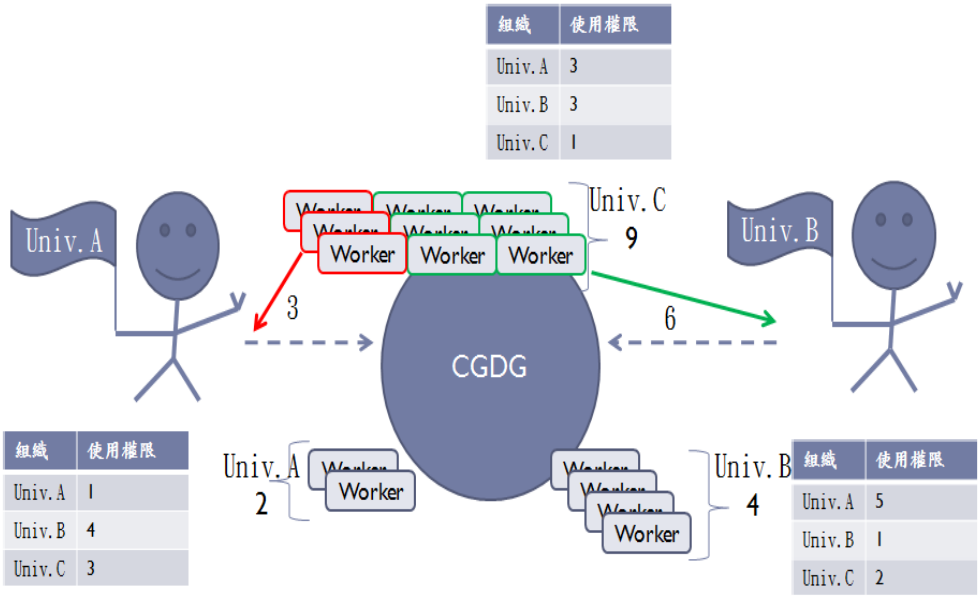
圖表 20 使用權限與排程關係示意圖

另一情境則是，若 Univ.A 的使用者送出優先權為 2 的的工作，在 Univ.A 跟 Univ.B 的機器執行時該如何看待他，我們選擇的策略是在使用 Univ.A 的資源時，工作的優先權變為 2，而在使用 Univ.B 的資源時，工作的優先權為 5。

另一個問題則是，當 Univ.A 跟 Univ.B 的使用者皆想使用 Univ.C 的機器時該如何分配。我們採取的策略有兩種，說明如下:

- **方法一:**

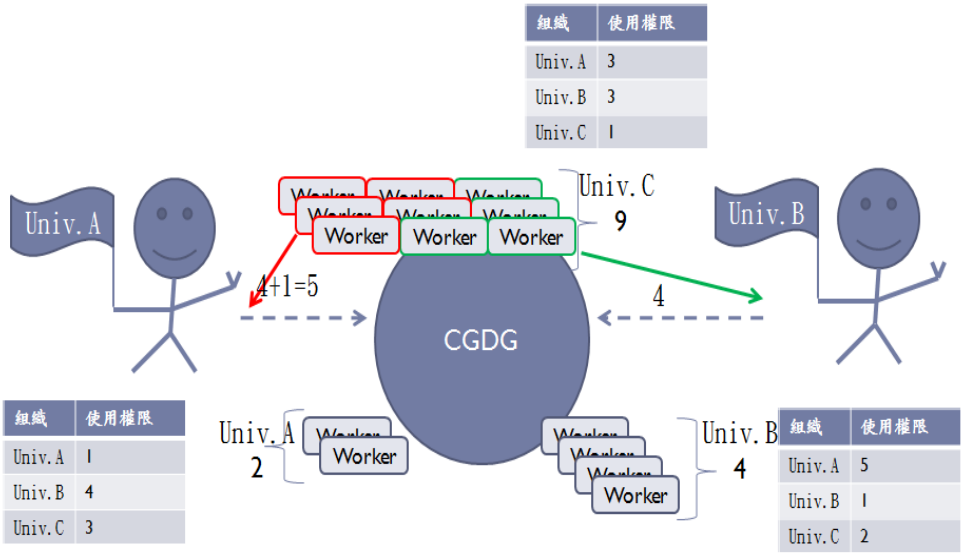
依照組織捐贈資源的比例來做分配，而一個組織的組織捐贈量會將所捐機器的機器等級列入考慮，而目前主要是用來做人工智慧運算，所以將中央處理器捐贈的總量列入優先考慮。舉例來說:假設 Univ.A 捐了 2 台機器，Univ.B 捐了 4 台機器，Univ.C 則捐了 9 台，Univ.A 與 Univ.B 捐的機器中央處理器等級皆為 100，Univ.C 的捐的機器中央處理器為 200，那 Univ.A 與 Univ.B 捐贈資源的比例為 1 比 2，因此在 Univ.C 機器組於閒置狀態時，Univ.A 可從 Univ.C 分到的資源為 3 台，Univ.B 則可分到 6 台 Univ.C 的資源。如圖表 21 所示。



圖表 21 資源分配方法一

- **方法二:**

在這方法之下我們採取的是最簡單的資源平均分配的方式，舉例來說:假設 Univ.A 捐了 2 台機器，Univ.B 捐了 4 台機器，Univ.C 則捐了 9 台，則當 Univ.C 機器組於閒置狀態時,Univ.A 與 Univ.B 分別先得到 Univ.C 的 4 台機器 剩下的 1 個資源就先搶先贏(假設 Univ.A 先連進來，則 Univ.A 搶到)。



圖表 22 資源分配方法二

在上述兩種方法中，我們可知在第一種作法下，可以達到鼓勵大家捐贈資源的效果，而第二種作法則較簡單。雖然第一種作法對於第二種作法顯得相對困難，但目前採取的策略為仍方法一，因我們希望能透過這樣的分配模式鼓勵大家捐贈資源，如此也可協助盡快的將電腦對局應用問題解決。

第四章、結論與未來展望

在此章節將會針對本篇論文做一個結論，並描述未來的發展方向。在 4.1 章節中將會針對本篇論文之應用層框架一般化與資源分配管理的部分做一個總結，在 4.2 章節中，則會描述此 Computer Game Desktop Grid (CGDG)桌機格網系統的未來研究方向。

4.1 結論

在本篇論文之中提出了一套應用層框架一般化與資源分配管理的做法，協助改良目前管理系統上功能的不足。

在應用層框架一般化方面，我們將原本用戶端的部分，切割為層用層(Application)、網格核心模組(Grid kernel Module)、網格圖形使用者介面模組(Grid GUI Module)三個部分，透過這樣模組化的動作，一般的電腦對局應用問題使用者，除了須實作該電腦對局應用問題的演算法核心部分之外，其餘的部分皆可使用此框架來快速的加入 CGDG 桌機格網系統中，如本實驗室研發的數獨、三角殺棋、六子棋等電腦對局應用問題皆以這樣的方式加入此系統。

而在資源分配管理方面，透過這樣的策略，可在群體計畫執行期間，管理加入此系統的使用者帳戶、協助分配加入此系統之組織間的資源，並達到鼓勵組織多捐贈資源的效果。

4.2 未來展望

在未來展望方面，會繼續擴充這套系統，試著將目前系統與現有的桌機格網系統做整合，如:BOINC，或是利用現有的套件強化目前系統在功能上不足的地方，且由於目前系統並無對工作端上的資料進行備份，將會遺失一些已經運算過

的計算結果，只能再重新運算此工作，造成資源上的浪費，因此未來將強化資料儲存以及資料搬移的部分，而由於目前系統皆是採取 TCP 連線的方式，在網路的負載量相對較高，未來希望能將原本單一協商者(Broker)的架構改良為階層式的架構。

參考文獻

[1] Allis, L.V., Meulen, M. van der, and Herik, H. J. van den, Proof-number search, Artificial Intelligence, Vol. 66(1), pp. 91-124, 1994.

[2] BOINC, available at http://boinc.berkeley.edu/.

[3] Calder, B., Chien, A., Wang, J., and Yang, D., "The Entropia Virtual Machine for Desktop Grids", CSE technical report CS2003-0773, October 28, 2003, Dept. of Computer Science and Engineering, Univ. California, San Diego.

[4] Chen, Ching-Ping, Wu, I-Chen, and Chan, Yi-Chih, "ConnectLib – A Connect6 Editor", available at http://www.connect6.org/Connect6Lib_Manual.htm, 2009

[5] Chinese Association for Artificial Intelligence, Chinese Computer Games Contest, available at http://www.aigames.cn/.

[6] Choi, S. and Kim, H. and Byun, E. and Hwang, C., A taxonomy of desktop grid systems focusing on scheduling. Department of Computer Science and Engeering, Korea University, Tech. Rep. KU-CSE-2006-1120-01, 2006

[7] Condor, available at http://www.cs.wisc.edu/condor/

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

[9] Fedak, G., Germain, C., Neri, V., and Cappello, F. Xtremweb: A generic global computing system. In Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2001): Workshop on Global Computing on Personal Devices, IEEE CS Press, Brisbane, Australia, 582-587, 2001.

[10] Kondo, D. and Chien, A.A. and Casanova, H., Resource management for rapid application turnaround on enterprise desktop grids. Proceedings of the 2004 ACM/IEEE conference on Supercomputing, 2004

[11] SETI@home. available at http://setiathome.ssl.berkeley.edu.

[12] SungJin Choi, Rajkumar Buyya, A Taxonomy of Desktop Grids and its Mapping to State-of-the-Art Systems, 2008

[13] WIKIPEDIA-Chinese Chess, available at http://en.wikipedia.org/wiki/Chinese_Chess

[14] WIKIPEDIA-Connect6, available at http://zh.wikipedia.org/wiki/Connect6

[15] WIKIPEDIA-GO, available at http://zh.wikipedia.org/wiki/Go

[16] WIKIPEDIA- Sudoku, available at http://zh.wikipedia.org/zh-tw/Sudoku

[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.

[18] Wu, I-C., Huang, D.-Y., and Chang, H.-C. "Connect6", ICGA Journal, Vol. 28, No. 4, pp. 234-241, December 2005.

[19] Wu, I.C., Lin, H.H., Solving the Minimum Sudoku Problem,2010.

[20] Wu, I.C., Shan, Y.C., Lin, H.H., Kao K.Y., Solving 9 Layer Triangular Nim,2010.

[21] XtremWeb, available at http://www.xtremweb.net/