應用網格建立一個高效能演化樹平行建構環境*

游坤明[1]，徐蓓芳[1]，賴威廷[1]，謝一功[1]，周嘉奕[1]，林俊淵[2]，唐傳義[3]

[1] 中華大學資訊工程學系
[2] 國立清華大學分子與細胞生物研究所
[3] 國立清華大學資訊工程學系

[1] yu@chu.edu.tw, {b9102042, b9004060, b9102004}@cc.chu.edu.tw,
jyzhou@pdlab.csie.chu.edu.tw
[2] cyulin@mx.nthu.edu.tw
[3] cytang@cs.nthu.edu.tw

摘要

以平行處理方式來計算龐大的資料運算是近年來一個非常重要的應用觀念。有許多不同的環境架構伴隨著不同的應用。網格 (Grid) 是一種建立在網際網路上的架構，網格可透過網際網路與其他網格互相分享資源，因此可以視為在使用龐大的且容易增減的資源來運算；與傳統的叢集式系統相比，傳統的叢集式系統 (Cluster) 若要增加運算能力，則必需花費比網格多的費用，因此運算能力有限。在一般所見的網格中，必須要有相同的協定、彼此認同的認證、安全性的考量以及合理的資源存取，才能讓網格在網路上互相溝通。使用網格運算我們所要處理的資料及程式，並且在合理的時間內得到正確的結果。本論文使用平行化演算法並以人類粒腺體為例，在單機、網格與叢集電腦環境中建構演化樹，並比較其效能差異。

關鍵詞：等距演化樹 ，叢集電腦計算, 網格計算, Globus Toolkit

1. 簡介

生物資訊研究領域中，科學家常常需要從演化樹的結果以了解物種間的親疏關係。從距離矩陣中建造演化樹在生物學和分類法方面是一個重要的議題，因此也產生許多不同的模型及相對應的演算法。而大部份的最佳解問題都已被証明為 NP-hard。

其中在許多不同的模型中有一個重要的模型便是假定演化的速度是一致的 [5, 17]。在這種前提下，利用距離矩陣算出的演化樹將會是一個等距演化樹(ultrametric tree)。

本論文使用一種高效能的平行化分枝界限演算法(branch-and-bound) 建立最小距離演化樹 。這個平行演算法是建立在 master-slave centralize 的架構上，並且加入了有效的負載平衡、節點與節點間通訊的策略，以解決最小權值等距演化樹建構的問題，使得時間在可容忍的範圍內完成。

近年來，對於許多以電腦輔助來求解的問題越來越多，且個人電腦的計算能力已無法滿足在合理的時間內得到結果。於是分散式的計算技術便是下一個發展的層次。本論文以人類粒腺體為例建構出演化樹，建構演化樹是一種非常複雜且耗時的計算過程，使用一般的個人電腦，將耗費大量的時間以求得結果，有時還會因資源不足造成等待許久的運作中斷，因此，要在合理的時間內得到滿意的結果，必須具有高效能的電腦，如超級電腦，但在經濟的考量下，我們可使用叢集電腦或網格來達到近似的效能。

叢集電腦可有大小不同規模，此做法的最大優點是「可擴充性」 (scalability) ：只要增加新的個人電腦，就可以提高叢集電腦的效能。在某些情況下資料是分布在不同的地區中需要互相存取，而網格是透過網路連線將好幾個在不同地區的叢集電腦串聯成的，更可以有效的利用這樣的優點來保持最新的訊息，所以在使用資源效率方面更遠勝於叢集電腦 [19]。

在網格上發展的技術為中介軟體，是用來整合網格分散的計算資源，主要角色是擔任機器間協調功能的任務。在網格的使用者和資源提供者之

* This word was supported in part by the NSC of ROC, under grant NSC-93-2213-E-216-037 and NSC-94-2213-E-216-028

間，擔任資源分配的協調工作，幫助使用者找到適合其使用的機器，並完成資料存取的交易 [19]。其中一個重要的組成要素，就是後設資料。

網格的優點之一，是有效率的使用閒置中的電腦，若是再長時間運算比較下，網格可以更有效率的使用資源。使用平行處理的環境，像是叢集計算或網格計算，必須用平行化的演算法以及使用平行化的溝通工具，例如 MPI，以幫助程式在該平臺上順利運作。

目前我們已成功的在網格的環境上執行平行化演算法，並且建構出演化樹，從網格與叢集電腦的實驗數據可看出，網格擁有與叢集電腦相似的效能。在本論文中，比較使用單機、叢集電腦及網格三種環境下的效能，在實驗結果中可顯示出，單機運算能力遠不如叢集電腦及網格；叢集電腦與網格之間的比較，若在相同節點數計算下，兩種環境效能是差不多的。

2. 背景

2.1 等距演化樹

在建立演化樹上有許多模型，其中一種為等距演化樹。等距演化樹為假設各物種的演化速率一致 [5, 13]，而等距演化樹的特性為有共同的父節點，物種存在葉節點而且在邊上有權重值的一個二元樹，在每個內節點的子樹中有同樣的路徑長到每一個葉節點上 [4]。對於一個 n * n 的距離矩陣 M 來說，定義最小的等距演化樹指的是兩兩葉節點的邊上權重總合為最小的。因為等距演化樹可以很容易的轉換為二元樹且不需要改變葉節點的距離 [13]，所以，等距演化樹是一個非常適合給電腦計算的模型。

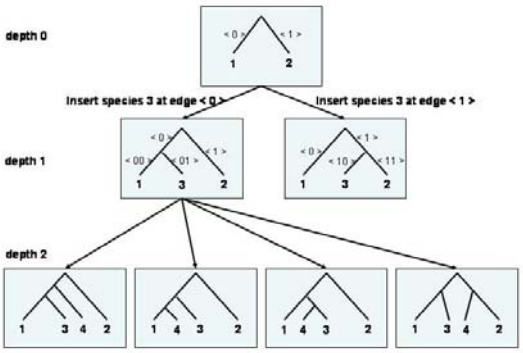

圖1. 建立分支界限樹 (BBT) [3]

如圖 1，我們可知，等距演化樹的數目 A(n)，隨著 n 的增加，演化樹的數量也快速的增加。有一些有關等距演化樹的研究先前已被提出 [6, 7, 15]。由於這些問題往往是不易解的，所以這些研究大都是基於 heuristic 演算法。舉例來說，像 UPGMA(Unweighted Pair Group Method with Arithmetic mean) [17]就是一個很常被用來建立等距演化樹的演算法。

在本論文中，我們使用 Exact Algorithms for Constructing Minimum branch-and-bound's from Distance Matrices [4]的演算法為基礎，並將之平行化。在上述方法中，使用分支界限法的策略作為找尋最小距離演化樹的方法。為了求得最小距離演化樹我們會將所有可能的樹型都找出並一一求值，但隨著物種數的增加，等距演化樹 A(n)的增加是非常快的，例如：$A(20) > 10^{21}$ ，$A(25) > 10^{29}$ ，$A(30) > 10^{37}$ ，於是上述方法中使用了分支界限法的策略來避免完全的搜尋。在本論文中，使用有效率平行化的分支界限演算法建立最小距離演化樹，在我們提出的方法中，是一個主從且集中式的平行化架構，並在此架構中加上了 loading-balancing, bounded 和 communication strategies 等機制，以增加程式的效率。

2.2 叢集計算

叢集計算(cluster computing)在隨著目前的科技下，處理器和周邊設備的普及，我們可以用低成本連接出高效能的叢集計算機。叢集計算機是以高速網路連接個人電腦或工作站而成的，可提供高效能的計算能力而且降低原來達到此效能的成本。在運作上，既然是由許多台電腦連接的，所以普通的應用程式也無法在上面發揮作用，必須設計適合在平行及分散式環境中的演算法，而且同時配合像是 MPI 這種專門用來做平行溝通的軟體，來設計應用程式。

現今在電腦和網路普及下，幾乎是可以看成所有電腦都與網際網路相連，如果把叢集電腦更廣義的角度來看，每台電腦就好像被網際網路連接的大型區網，全球就是一個大型的叢集電腦，但是事實並非如此，因為無法做到資源互相分享、計算互相分擔，所以為了達到更廣義的資源活化運算，於是網格計算的理念被提出。

2.3 網格計算

網格計算(Grid Computing)可讓分散於各地的虛擬組織，協調彼此的資源分享，同時滿足大量運算的需求。而集合分散的運算資源之外，網格計算能夠經由網路管理組織內任何一個可使用的運算資源，進而降低伺服器的閒置時間。

網格計算可以解決在同一時間內使用網路上很多資源去解決一個問題或者當一個問題需要大量處理器計算或是需要存取大量分佈不同地方的資料。耳熟能詳的例子像是 SETI (Search For Extraterrestrial Intelligence)@home 它讓上千人的電腦在閒置時的處理器中去幫助計算資料。而且這些電腦都是獨立性工作，指的是說無論有些工作需

花較長的時間，或者沒有回傳資料，都沒有關係，因為有此狀況時，它會在暫停一段時間後，自動把工作分派給其他電腦做處理。

2.4 Globus Toolkit

Globus [8, 14, 20]對訊息安全、資源管理、訊息服務、數據蒐集管理以及應用開發環境等網格關鍵理論和技術進行廣泛的研究，並且開發出可以在多種平台上執行的 GlobusToolkit，用來幫助規劃和建造大型網格試驗和應用平台，開發大型網格系統可以執行的應用程式。Globus Toolkit 同時提供了好幾種語言模式給程式設計師選擇，就類似像物件導向的方式。程式開發者更可以由 Globus Toolkit 中所提供的服務任意選取最符合需求的工具去與現存的軟體作整合。例如: GRAM 提供資源管理的協定、MDS 提供資訊服務的協定、GridFTP 提供了資料傳輸的協定…等，這些全部都有使用 GSI 安全協定在他們的連接層 [20]。

Service	Name	功能
Resource managrment	GRAM	資源分配與工作管理
Communication	Nexus	單一或多重溝通服務
Security	GSI	認證與聯繫上的安全服務
Information	MDS	分散式存取和狀態的資訊
Health and status	HBM	監測系統零件健康狀況
Remote data access	GASS	遠程存取資料經由連續及平行的連繫裝置
Executable management	GEM	結構、讀取技術與狀態執行管理
Information	GRIS	查詢計算資源現有的設定、能力及狀態
GridFTP	GridFTP	提供高效能、安全，以及健全的資料傳輸機制

表 1. GlobusToolkit 所提供的服務

2.5 MPICH-G2

MPI 是訊息傳送介面(Message Passing Interface)用來撰寫 message-passing programs 和可以廣泛的使用於平行運算的一種基礎 API。在網格應用程式上 message-passing 的優點是它提供比通訊協定 TCP/IP sockets 更高層的介面，讓我們可以直接使用通訊結果而不必知道中間是如何溝通。Globus 服務已被用來發展成 Grid-enable MPI 以 MPICH library 為基礎，Nexus 為通訊基礎，GRAM 服務為資源分配和 GSI 來做安全認證。

MPICH-G2 是 Grid-enable 以 MPI v.1.1 為基礎在網格上的實作。它使用了 Globus Toolkik(像是資源分配、安全性)的服務。MPICH-G2 准許以連接不同平台的機器來執行 MPI 的程式。MPICH-G2 會自動作資料轉換當在兩個不同平台時的傳輸和自動的選擇 TCP 以提供多重協定通訊的訊息給網路上機器及傳出有 MPI 提供的訊息給區域內機器。

2.6 UniGrid

網格計算的目的是用來整合大型網路環境下的各種資源。UniGrid 是連結國內七所大學及國家高速網路中心之電腦網格系統，建置一「國家計算網格實驗平台」，以協助推廣網格計算的觀念到各產學領域。UniGrid 將著重在使用網格計算領域最常用之程式集及工具套件 Globus。並且有提供隨時每個節點的 CPU、RAM 狀況監看。

Globus[8]提供了網格中使用的協定，可以讓使用者充分利用分散於各處的資源中建出網格計算的架構。

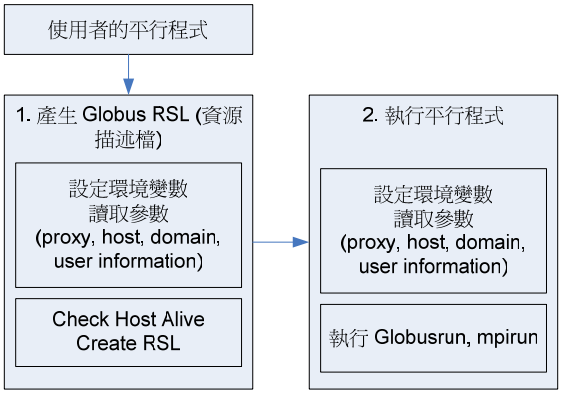

圖 2. UniGrid 上的程式流程

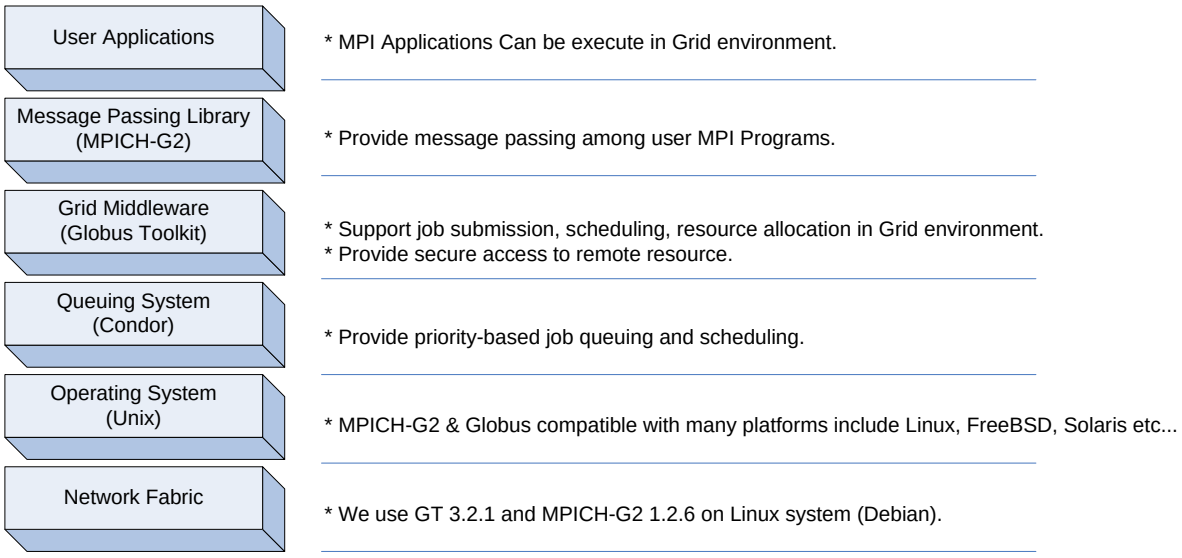

圖 3 .UniGrid 的架構圖[2]

3. 系統架構

單機上的程式處理方式與分散式系統的處理方式不同，所以當平台由單機發展為分散式系統時，若程式想要發揮平行處理的效能，就必須改變原來的程式演算法，在程式中加入訊息傳送的程式觀念。

3.1 單機演算法

在建構演化樹的問題上，一般用的是 UPGMA 這一類的啟發式演算法，所得到的解並不是最佳解。[4] 中提出了利用 branch-and-bound 來建構最佳解演化樹。雖然 branch-and-bound 的解空間會非常大，但中型的演化樹在生物學家的實際用上仍然非常有實用價值。

在 [4] 中所提出的演算法中，首先，執行 UPGMM 得到一個起始解的 upper bound (UB)，接著開始建立 branch-and-bound tree (BBT) 如果建立時 lower bound (LB) 大於目前的 UB 時就刪除此節點，選擇下一個位置繼續建立，當計算到 UB 比目前的 UB 低時就更新。直到所有物種都建立完畢，最後，權值最小的樹即是我們所要求的解。其演算法如下：

Algorithm BBU
Input: An *n* £*n* distance matrix *M*.
Output: The minimum ultrametric tree for *M*.

Step 1: Relabel the species such that *(*1, 2….. *n)* is a **maxmin** permutation.
Step 2: Create the root *v* of the BBT such that *v* represents the only topology with leaves 1 and 2.
Step 3: Run UPGMM to find a feasible solution and store its weight in *UB*.
Step 4:
while there is a node in BBT **do**
  Delete all nodes *v* from BBT if *LB*(*v*) ¸ *UB* or all the children of *v* have been deleted.
  Select a node *s* in BBT, whose children has not been generated.
  Generate the children of *s* by using the branching rule.
  If a better solution is obtained, then update *UB*.
End while

3.2 平行化分支界限演算法

雖然利用 branch-and-bound 的技巧可以利用 bound 值來避免將每個可能做搜尋，但是隨著物種數目的增加，所需的計算時間也成指數成長。所以，我們便利用平行計算的方法來加速演化樹的建構。

考慮在平行計算的環境上的特性，所以針對資料結構和演算法做了些改變和增加。在資料結構上，為了減少節點與節點之間的溝通，因此所定義的資料結構包含了每個內結點的左子節點、右子節點、父節點，與子結點的路徑。在演算法上，為了更能發揮平行處理的環境，必須讓每個節點的計算量平衡，故必須加上如 Global Pools、Local Pools、等機制讓節點與節點間可以達到動態的負載平

衡，並且我們為了減低不必要的計算，在算出一個比原來標準用的上限還低時，就會一直把資訊傳給全部的節點以達到提升計算效率。而平行化架構採用的是主從式架構，起始化時分配的資料與計算過程中所需動態分配的資料都是由 master 來做分配。

在負載平衡的問題上，一般來說可以分為靜態與動態的負載平衡 [3]。靜態的負載平衡指的是在資料的分配只在程式一開始的期間做分配，而程式執行期間不做任何的資料牽移；相對的動態負載平衡指的是會依需求而在節點間搬動及牽移資料。動態負載平衡可以分為集中式的 (centralized) 與分散式的 (decentralized) [3]。集中式的負載平衡是由一台管理主機 (管理節點) 來做調控，每個節點藉由把資料送至管理節點後再由管理節點來決定資料要如何分配。相對的分散式負載平衡則是由節點彼此間互相溝通後再彼此間一同決定的機制。一般來說，集中式的架構能夠有更好的負載平衡，因為管理節點可以知道所有節點的狀態並決定一個更好的分配，但在一個大型的平行系統下，集中式的負載平衡會因為管理節點的瓶頸而效能不佳。

Input: A n * n distance matrix M
Output: The minimum ultrametric trees

Step 1: Master computing node re-label the species such that feasible maxmin permutation.
Step 2: Master computing node creates the root of the BBT.
Step 3: Master computing node run UPGMA and using the result as the initial UB (upper bound).
Step 4: Master computing node branches the BBT until the branched BBT reach 2 times of total nodes in the computing environment.
Step 5: Master computing node broadcasts the global UB and send the sorted matrix the nodes cyclically.

Step 6:
while number of UTs in LP (Local Pools) > 0 **or** number of UTs in GP (Global Pools) > 0 **do**
    if number of UTs in LP = 0 **then**
        if number of UTs in GP <> 0 **then**
            receive UTs from GP
        end if
    end if
    v = get the tree for branch using DFS
    if LowerBound(*v*) > UB **then**
        continue
    end if
    insert next species to *v* and branch it
    if *v* branched completed **then**
        if Cost(*v*) < UB **then**
            update the GUB (Global Upper Bound) to every nodes
            add the *v* to results set
        end if
    end if
    if number of UTs in GP = 0 **then**
        send the last UT in sorted LP to GP
    end if
end while

Step 7: Gather all solutions from each node and output it.

4. 實驗結果

4.1 實驗環境及結果

在實驗的環境中，我們使用了單機、以及叢集電腦與網格的系統。單機及叢集電腦的系統如表 2。網格實驗環境使用的是 UniGrid 系統。

在實驗數據中，我們挑選人類粒腺體做為實驗數據，並以物種數目 12、14、16、18、20、22 一一執行，每一物種數目有 10 組測試資料。我們從 10 組資料中分別取中位數、平均數、最差情況來做實驗結果比較，以期消除資料相依所產生執行時間的差異。

表 2. 實驗環境

單機	
處理器數目	1
環境	中華大學平行分散實驗室
硬體設備	AMD 2000+、2GB DDR RAM
叢集電腦	
處理器數目	16
環境	中華大學平行分散實驗室
硬體設備	AMD 2000+、1GB DDR RAM
網格	
處理器數目	12
環境	國家網格計算實驗平台
硬體設備	AMD 1.3G、2GB DDR RAM
處理器數目	4
環境	東海大學高效能計算實驗室
硬體設備	AMD MP 2000+ ‘2、512MB DDR RAM’2

如表 3 及圖 4 分別為執行時間中位數的結果，我們可以發現，當物種數目增加時，計算時間也相對增加，而在圖中也可以了解，不論是叢集電腦或者是網格系統，都能夠有效的降低執行時間。

表 3. 中位數時間比較表

物種數目	單機	叢集電腦	網格
12	0.113313	0.146947	0.130881
14	2.936615	0.889956	1.180245

16	36.1053	29.2515	11.6873
18	1003.268	324.5231	332.807
20	138.684	157.6269	99.2526
22	9873.82	5911.42	2625.637

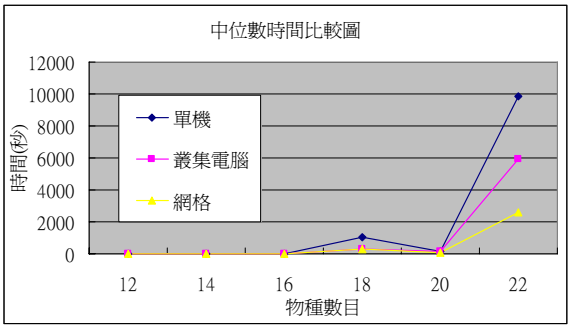

圖 4. 中位數時間比較圖

表 4 以及圖 5 為平均計算時間；表 5 以及圖 6 為最差計時間，兩個相比較，我們可以了解、平均計算時間可能被最差計算時間所影響，因為建構演化樹的問題有資料相依的情形，所以我們會選擇中位數計算時間做為我們主要的比較依據。而從圖中也可以觀察到，計算時間隨著數種數目的成長有相當快速的增加。

表 4. 平均數時間比較表

物種數目	單機	叢集電腦	網格
12	0.344878	0.166051	0.236581
14	41.99103	7.537349	7.390265
16	390.8962	207.8525	58.34718
18	2598.467	983.583	1031.805
20	1114.028	4705.797	249.0695
22	9873.82	5911.42	2625.637

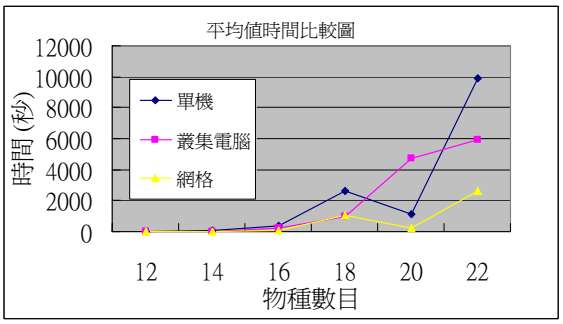

圖 5. 平均數時間比較圖

表 5. 最差狀況時間比較表

物種數目	單機	叢集電腦	網格
12	0.785476	0.395494	0.927229
14	303.738	40.4603	35.1285
16	1387.6	911.781	203.611
18	9339.67	4327.96	4606.76
20	5009.17	25064.1	1028.86
22	9873.82	5911.42	5068.7

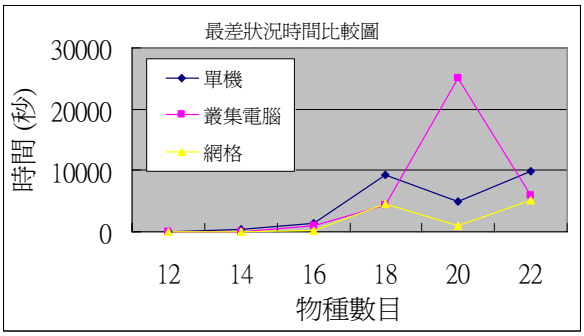

圖 6. 最差狀況時間比較圖

4.2 結果討論

我們從數據中取出中位數、平均數、最差情況來做比較。由圖表明顯看出隨著物種數目增加、計算相同物種時，單機效能最差，叢集電腦次之，網格效能最佳。正常情況下，叢集電腦的效能應比網格好，因為使用內部溝通為高速網路的叢集電腦，其效能遠高於使用網際網路溝通的網格。但是實驗結果與理論不符，這是因為實驗中所使用的叢集電腦設備較網格所使用的電腦設備差。

最初使用單機運算樣本以繪出演化樹，雖然成功建出演化樹，但是花費時間非常驚人，且運算樣本的大小有限，因此進度緩慢、效率不佳，之後採用叢集電腦。叢集電腦環境為 16 顆處理器，因此效率提高許多。但因為考慮到經濟成本以及為了應付更巨大的計算，我們考慮了更有效率的平行處理環境，網格。

所以我們開始把平行化建立演化樹的程式以網格平台來做實驗。我們以 10 組物種數目 20 的資料去實驗，實驗結果見表 6 和圖 7。實驗結果發現網格計算效能，如果在相同的節點數目，計算效能似乎較叢集電腦差了一點，但是如果網格使用 24 節點，則效能遠超過叢集電腦 16 節點。

而現今已有許多網格平台的建立，像是 UniGrid 就是聯合國內七所大學以及國家高速網路中心的叢集實驗室所成的網格實驗平台，我們可以使用更多的資源去執行程式，並且透過網格運算的技術，他會到網路中尋找閒置的電腦，並將工作依據適當的比例分配，送到這些電腦上執行，然後將結果送回，這樣做可以更有效率。

而且我們考慮了未來萬一資料是放置在世界各處或者是隨時都會更動的，那叢集電腦就顯得不

適合，且叢集電腦資源有限，如果遇到一個龐大的問題也可能需要計算很久的時間，所以即使資料分布在世界各地也可以輕鬆的應付並保持資料的最新狀況。

表 6. 叢集電腦與網格使用不同節點數比較

	叢集電腦 16 節點	網格 16 節點	網格 24 節點
1	1629	1652.96	885.517
2	10273	10691.6	6838.49
3	750	764.104	750.752
4	561	566.25	458.426
5	249	258.197	256.616
6	199	199.96	148.454
7	54	57.693	83.55
8	126	128.596	110.922

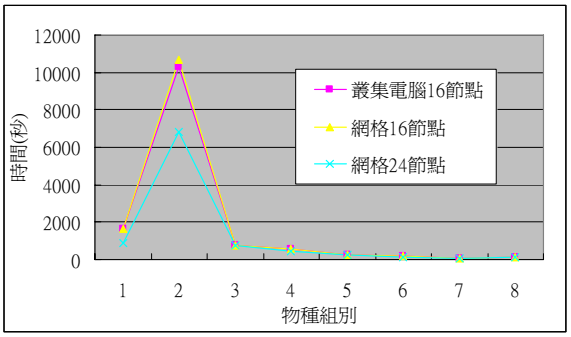

圖 7. 叢集電腦與網格使用不同節點數比較圖

5. 結論

實驗中測試的網格所使用的處理器 16 顆，與叢集電腦相比，數據中發現，網格與叢集電腦使用處理器個數相同，網格並無任何優勢，網格效能較叢集電腦差，因為叢集電腦內部溝通速度遠大於網格間所使用的網際網路溝通。未來我們可以考慮建立更有效率的網格溝通機制，相信可以大幅改善網格的效能。

我們的目標將是不只侷限於計算人類粒腺體的資料，推廣至以網格來運算蛋白質的樣本，甚至其他的資料在正規化之後皆可用網格來運算以得到結果。

目前是用網格來運算人類粒腺體的樣本，雖然花費的時間非常的多，因為剛開始時需要找出在網格上的最佳效率，但是，未來中，我們將更有效率的平行演算法及網格 API，目標在使用方面，只要將要處理的資料整理成我們目前資料輸入的形式，就可以得到我們要求的數據，所以，未來可能運用此種方法來運算類似的龐大資料，像蛋白質等等的樣本，應該跟目前的運作方式相同，在網格上運算即可得到結果。

參考文獻：

[1] 全球網格(World Wide Grid)發展趨勢

[2] 格網計算平台架設實例簡介格網計算平架設實例簡介 Introduction to Constructing Introduction to Constructing Computational Computational Grid Grid
王順泰

[3] Barry Wilkinson, Michael Allen, “Parallel Programming”, *P.H.*

[4] B.Y. Wu, K.M. Chao, C.Y. Tang, “Approximation and Exact Algorithms for Constructing Minimum Ultrametric Tree from Distance Matrices,” Journal of Combinatorial Optimization 3, pp. 199-211

[5] D. Gusfield, “Algorithms on Strings, Trees, and Sequences, computer science and computational biology,” Cambridge University Press, 1997

[6] E. Dahlhaus, “Fast parallel recognition of ultrametrics and tree metrics,” *SIAM Journal on Discrete Mathematics*, 6(4):523-532, 1993

[7] H.J. Bandelt, “Recognition of tree metrics,” *SiAM Journal on Discrete Mathematics.*, 3(1):1-6, 1990

[8] The Globus Project: A Status Report. I. Foster, C. Kesselman. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998.

[9] The Anatomy of the Grid: Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. *International J. Supercomputer Applications*, 15(3), 2001.

[10] The Nexus Approach to Integrating Multithreading and Communication. I. Foster, C. Kesselman, S. Tuecke, J. *Journal of Parallel and Distributed Computing*, 37:70--82, 1996.

[11] A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. I. Foster, N. Karonis. *Proc. 1998 SC Conference*, November, 1998.

[12] A Secure Communications Infrastructure for High-Performance Distributed Computing. I. Foster, N. Karonis, C. Kesselman, G. Koenig, S. Tuecke. *6th IEEE Symp. on High-Performance Distributed Computing*, pp. 125-136, 1997.

[13] M.D. Hendy and D. Penny, “Branch-and-bound algorithms to determine minimal evolutionary trees,” Mathematical Biosciences, 59:277-290, 1982.

[14] M. Frach, S. Kannan, and T. Warnow, “A robust model for finding optimal evolutionary trees,” *Algorithmica*, 13:155-179, 1995.

[15] M. Krivanek, “The complexity of ultrametric partitions on graphs,” *Information Processing Letter*, 27(5):265-270, 1988.

[16] Chuan Yi Tang, Solomon K.C. Wu, "Chee Kane Chang, “A scalable Fully Distributed

Parallel Branch & Bound Algorithm on PVM cluster”

[17] W.H. Li and D. Graur, “Foundomentals of Molecular Evolution,” *Sinauer Associates*, 1991.

[18] Yuji Shinano, “Kenichi Harada and Ryuichi Hirabayashi,” *Control Schemes in a Generalized Utility for Parallel Branch-and-Bound Algorithms*, *Parallel Processing Symposium*, 1997. Proceedings., 11th International , 1-5 Apr 1997, Page(s): 621 -627

[19] GridCafé (http://www2.twgrid.org/gridcafe)

[20] The Globus Project (http://www.globus.org/)