Seminar6 — Peer-to-peer searches: An overview

by 金仲達 教授 清華大學資訊工程系

2003.10.28

報告撰寫者：政大資科所碩一 謝淳達 92753008

1 演講主題及摘要

簡介何謂 Peer to Peer , 這個現今網路上最熱門的應用, 並且深入介紹目前已經有的一些 P2P Projects 的實做方法, 並且分享演講者目前相關研究的內容。

2 要點整理

什麼是 Peer to Peer computing 呢? 簡單來說, 就是「如何在網路上找到我們想要找的東西」!假設網路上有我們所需要的東西, 那我們如何透過 P2P ,「快速」並且「正確」地找到我們想要找的東西? 最傳統的方法, 就是在網路上放置一台大型的 server , 把他的 IP address 公佈出來, 並且該 server 提供一個 directory , 記錄網路上所有東西的所在地, 然後任何想找東西的人就連上他, 透過該 directory 就可以找到想找的東西了。

但是上述這種傳統的作法很麻煩的一點是, 必須要有一個「中央 server」。這樣有個缺點, 會變成東西都是該 server 在管, 如果今天我要跟朋友分享一個音樂 CD, 他會說這是非法的所以不讓我上傳。還有很多其他缺點, 所以現在世界上所有的 P2P

Project 都是在研究「分散式 P2P」。做分散式的好處有很多, 可以利用網路上許多的電腦資源來達到更強大的運算功能, 而這是傳統 centralized P2P 無法做到的。現今大家最耳熟能詳的, 就是 Napster 或是 EZpeer 這類音樂分享軟體了。這兩個都是分散式 P2P , 他們都必須提供一個 centralized server, 並提供 directory 紀錄每個檔案的所在地, 想抓音樂的人就必須連上這類的 center server 透過該 directory 搜尋音樂。

但是這樣的機制有個問題就是, 因爲必須有 center server , 而且分享的東西又是非法的, 所以會被人家告。那有沒有辦法可以不要有這種 centralized server 但又可以達到相同功能的呢?

Gnutella 就是一個有 distributed directory 的 protocol 。但是沒有了 centralized directory 之後, 要怎麼搜尋檔案呢? 他的觀念是, 我今天要找一個檔案, 那我就去問我的朋友, 然後我的朋友再去問他們的朋友, 然後把問到的結果傳回給我。由於 directory 是分散的, 這樣唱片公司就沒有辦法告了。

但是像 Gnutella 這種搜尋方式就等於是要 flooding 整個網路, 所以非常耗時耗資源。那我們有沒有辦法不要像這樣四面八方的問朋友, 而是透過某種機制「有方向性」的去尋找, 這樣會比較有效率一點。Freenet 這個計畫就是爲了解決 flooding 這個問題, 這個計畫可以看作是 Gnutella 的延伸。他跟 Gnutella 的差別在於他加一個 hash key 。假設我們今天要搜尋 123.mp3 這個檔案, 那我們就透過一個 hash function 取得這個檔案的 hash key , 透過這些檔案的 hash key 來決定出「方向性」。而且透過這種機制也會讓這些資訊集中在某些 peer 管理。譬如有個 peer 他常常搜尋古典音樂, 他知道在那裡可以找到各式各樣的古典音樂, 那麼他的朋友們就知道以後要找古典音樂就找他問很快就會找的到了。漸漸的該 peer 就成爲尋找古典音樂的「專家」。

上面所講的都是「un-structure peer-to-peer」, 也就是說這些 peer 彼此之間沒有一個邏輯的關係, 所以有可能會發生找不到想找的資料的情況。因此後面就有人研究「structure peer-to-peer」。

CAN (Content Addressable Network) 這個計畫就是在實做 structure P2P 。他

透過 hash function 把檔案 hash 到 N 維空間的某個位置, 搜尋的時候, 每個 peer 只需要知道他的 N 個鄰居即可, 然後透過他的鄰居一步步走過去就可以找到想要的資料。至於資料的管理, 如果有個 peer 加入的話, 他會先透過某種機制決定他要在這 N 維空間的某處, 該處就會有個 peer 把他的資訊 copy 一些給新加入的 peer 管理。另外一個計畫 chord , 則是改進 CAN 的搜尋效率, 他並不透過他的鄰居這樣一步步走, 而是只紀錄離他 2 的一次方, 二次方, 三次方 ... 的 peer 的 ip , 透過類似 binary search 的方式讓我們能比較快找到想要的資料。這種方法的搜尋效率是 O(logN) , 比前面幾個方法都還要快上許多。

不過有個直得注意的問題是, 上面所有的 project 所說的「鄰居」只是 logical 的鄰居, 也就是說, 假設我今天在台灣的話, 我的鄰居有可能是在美國, 日本或其他很遠的地方, 所以我的鄰居有可能是在很遠的地方, 這將會造成搜尋時間的拉長。所以或許我們在 locality 的時候可以將 physical location 考慮進去, 把 實際地理上較接近的 node 擺在一起, 即可避免這樣的問題。

上述的這些 project 都是透過 hash function 讓搜尋有了「方向性」, 也讓資料管理有「地域性」, 而加速搜尋的速度。但是透過 hash function 有個很大的問題是, 任何資訊一旦經過了 hash 之後, 就破壞了原本資料所含有的許多資訊。例如, 假設我今天要搜尋所有檔名有「張惠妹」的音樂, 那麼上述的 project 都沒有辦法處理這樣的 query 。此外, 他們也沒有辦法處理「最大」「最小」或是「範圍」這類概念的 query 。而金仲達教授他們的 project 就是要解決這樣的問題。

3 評論

隨著寬頻網路的日漸普及, 以及人類對於資源分享的需求漸增,peer to peer 這種網路應用可說是越來越熱門了。從很久以前的 Napster , 到今日的 eDonkey , EZpeer 等等, 人們都希望透過這類軟體來取得更多想要的資源。然而很多人都透過這類軟體交換有版權的數位資料, 所以最近這類的軟體供應商司都面臨了官司問題。如果 distributed P2P 眞的能發展出來並且應用的話, 那麼就不怕被那些唱片業者告了。但是這當中當然牽涉到許多法律以及道德方面的問題。

但是 distributed P2P 還有一些問題。例如, 跟傳統的 centralized P2P 比起來,distributed P2P 會使得每個 node 的資源消耗量增加。因爲沒有了 centralized server directory 之後, 每個 node 都必須負責管理一部份的資料, 要決定如何尋找檔案, 還要當別人搜尋檔案時的中繼站, 而這些都會增加記憶體,CPU, 以及網路頻寬的使用量。在現今人們不願分享出資源的心態上, 這種 distributed P2P 若會使得個人電腦的速度變慢的話, 將會降低使用的意願。尤其對於一些本來資源就不是很高的 mobile devices , 例如 PDA 來說, 更是不太可能實行。也因爲資源的有限, 所以 distributed P2P 的執行效率有可能會因爲某些 devices 的低效能而造成效能低落的問題。

此外, 不知道會不會有這種情況就是, 假設台灣中部突然來個大停電, 造成中部所有 node 都突然 shutdown , 那麼會不會造成北部和南部的 peer 沒有了中部的這些 node 作爲仲介而無法互通有無? (假設這些鄰居都有根據 physical location 作適當的安排)

另外, 假設那些中部的 node 都是找尋某些類型檔案的專家的話, 這些 expert nodes 突然 shutdown , 因爲來不及將資料分享出去, 而使得這些專家「後繼無人」, 那是不是意味著我們必須再多花多餘的時間重新 training 這類的 expert 呢?