以網路快取伺服器爲基礎的檔案參考文件搜尋演算法

游象甫
國立中央大學資訊工程研究所分散式系統實驗室
國立中央大學電算中心

郭廖軒
國立中央大學資訊工程研究所分散式系統實驗室

曾黎明
國立中央大學資訊工程研究所分散式系統實驗室

摘要

檔案下載(downloading archive)是 Internet 上相當有用的服務，許多使用者透過網路下載共享軟體，驅動程式及其他工具程式。我們之前的研究專注於以網路代理伺服器提供檔案下載服務，實驗結果顯示許多使用者注意到這項代理伺服器的功能，同時此種策略成功的提高快取伺服器的檔案再利用率。然而一些使用者建議我們能夠進一步提供檔案的參考文件來幫助他們使用檔案。參考文件內容可能包含檔案版本、作者及使用說明等資訊。透過索引說明文件，使用者可以關鍵字的方式來搜尋檔案，比傳統以檔名方式來的方便。因此我們提出兩種方法從代理伺服器的快取找尋檔案的參考文件：iterative 和 backtracking 方式。第一種方法是讀取所有快取內的 HTML 物件找尋參考文件；第二種方式以回溯使用者存取檔案的順序找尋文件。Iterative 方法幾乎可以找到所有在網路快取伺服器中的參考文件，而 backtracking 方法則可以較低的成本找到參考文件。我們並且以 14 位使用者來評估提出的方法，每個使用者被要求下 10 個查詢，每個查詢最多傳回 10 個結果。實驗顯示在 iterative 方式中，第一筆就傳回正確結果的比例爲 0.72；在 1,088 筆回傳結果中 62%是正確的。此外，我們發現 backtracking 方式比期望的結果好很多，第一筆就傳回正確結果的比例爲 0.71；在 1,055 筆回傳結果中 62%是正確的，它回傳正確結果的比例與 iterative 方式相當，但其總共讀取的 HTML 物件卻只有 iterative 方式的 1/26。

關鍵字: FTP, WWW, 網路快取伺服器, archive service, description search, ProxyFTP

1 緒論

下載檔案是 Internet 上相當有用的服務之一，它幫助使用者分享資源及交換訊息。通常這些檔案(archive)具有下列特徵：

- 這些檔案可能是應用程式、驅動程式或工具軟體等。
- 爲了降低檔案大小及易於安裝，通常這些檔案被壓縮過。所以使用者不能直接讀取這些檔案。
- 一般透過檔案名稱無法幫助使用者瞭解這些檔案，使用者需要說明文件的幫助，而這些文件包含了版本、作者及使用說明等資訊。
- 這些檔案大都是以".exe" ,".zip" ,".gz" ,"tgz" ,"com" ,".z "作爲副檔名。

我們之前的研究專注於以網路代理伺服器(Proxy cache server)來提供檔案下載服務，並且發展了一個系統稱爲 ProxyFTP [11]，它滿足了使用者對於檔案下載的需求。圖 1 表示 ProxyFTP 的運作方式，ProxyFTP 分別扮演網路代理伺服器及 FTP [7] 伺服器。圖中粗的實線表示 WWW 與 FTP 的使用者將 ProxyFTP 視爲一個普通的代理伺服器。但在他們使用 ProxyFTP 時，ProxyFTP 會從本身的快取收集檔案並且產生檔案列表 ls-lR.gz。然後如圖中虛線所示，Archie 會從 ProxyFTP 和其它 FTP 伺服器蒐集檔案列表。因此 Archie 會有 WWW 與 FTP 的檔案資訊。如細線所示，當 archive consumers 需要檔案時，他們可以透過 Archie 找到檔案的位置，然後從各個 FTP，Web 或是 ProxyFTP 下載。

ProxyFTP 有幾個重要的優點：第一，因爲檔案蒐集是自動的，所需的管理成本相當低。此外，ProxyFTP 不需要另外的網路頻寬來 mirror 檔案，對於無法同時提供足夠資源給網路代理伺服器和 FTP 伺服器的機構，這樣的優點非常重要。另外，自動搜尋檔案的方式也可以幫助商業的檔案站台(Archive sites)[1][6]以更快且成本更低的方式蒐集檔案。

第二、ProxyFTP 降低近端使用者的等待時間；當使用者在 ProxyFTP 找到他們所需的檔案

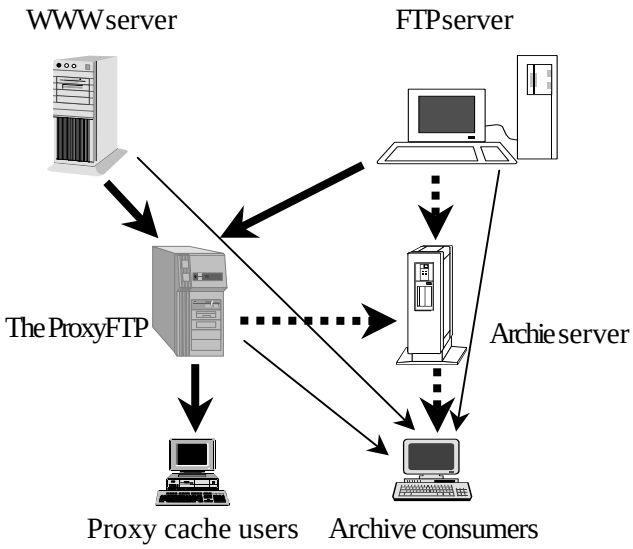

圖 1: ProxyFTP 的服務架構

時，可以直接從 ProxyFTP 下載，而不需連到遠端的站台。第三、它改進了網路快取伺服器的檔案再利用率。一般而言，WWW 物件透過 Uniform Resource Locators(URLs) [4] 來存取，但由於 URLs 相當冗長，使用者通常必須透過檢視網頁來找到檔案。因此，檔案的被使用是靠著網頁是否有檔案的連結，如果網頁被更改或刪除的話，則網路代理伺服器中快取的檔案將難以再被使用。我們提出的策略允許使用者直接存取快取的的檔案而不需透過網頁，因此提昇快取檔案的再利用率。

然而，一些 ProxyFTP 的使用者建議系統能夠進一步提供檔案的參考文件或網頁(referring documents or referring pages)來幫助檔案的使用，這些文件通常包含版本、作者、使用說明及下載位置等資訊。藉由索引參考文件，使用者可以方便的利用檔名或描述的方式來找尋他們所需的檔案。在這裡我們提出兩種方法從網路代理伺服器中找檔案的參考文件：iterative 和 backtracking 方式。第一種方法是讀取所有在快取內的 HTML 物件中找尋參考文件；第二種方式以回溯使用者存取檔案的順序找尋文件。此外，兩個方法都會產生檔案與其參考文件的關連(associations)。

最後我們測試提出的方法，建立參考文件的索引，並且以 14 位測試者評估文件的正確性。每個使用者被要求下 10 個查詢，每個查詢最多傳回 10 個結果。實驗顯示在 iterative 方式中，第一筆就傳回正確結果的比例爲 0.72；在 1,088 筆回傳結果中 0.62 是正確的。而 backtracking 方式第一筆就傳回正確結果的比例爲 0.71；在 1,055 筆回傳結果中 0.62 是正確的，其回傳正確結果的比例與 iterative 方式相當，但其總共讀取的 HTML 物件卻只有 iterative 方式的 1/26。

本文的其他部分如下：第二節介紹提出的搜尋檔案和其參考文件的方法；第三節是效率測試；第四節介紹相關研究；最後是我們的結論。

2 檔案及其說明文件搜尋

首先定義一些術語。一個快取物件包含兩個部份：URL 及本體(body)。URL 包含通訊協定、主機、目錄和檔名。本體則是指物件內容。HTML 物件指的是副檔名爲“.html”, “.htm”, “.dhtml”, 或“.txt”的物件，而對 HTML 物件的存取稱爲 HTML 存取(HTML access)；而檔案物件(archive object)則指副檔名是“.exe”, “.zip”, “.gz”, “.tgz”, “.com”, “.rar”, “.tar”, “.deb”, “.slp”, “.rpm”, 或 “.z”的物件，而對檔案物件的存取稱爲檔案存取(archive access)。爲了簡化問題，在本文中只有 HTML 物件爲可能的參考文件；另外一些提供檔案下載服務的 WWW 站台 (如：”www.download.com”)，我們稱之爲 authority hosts。

2.1 The Iterative Scheme

我們的目的是建立檔案與其參考文件的關連(associations)，一旦這種關連被建立，找到檔案就可以找到其參考文件；反之，找到參考文件也可以跟著找到檔案。Iterative 方法從快取中尋找參考文件的程序如圖 2 所示，假設 *A* 是個 sequence，記錄了所有使用者對網路代理伺服器的存取，每筆存取包含三個欄位：存取時間，客戶端位置及存取的 URL；而我們要找的關連每筆包含兩個欄位：參考文件的 URL 及檔案的 URL。

根據存取的 URL，Step2 選出 HTML access。然而符合下列狀況的 HTML access 將不會被挑出：第一，它的 URL 檔案名稱不像是檔案的說明文件，例如 ”error.htm”；第二，它的提供者不像是參考文件的來源，例如，"www.cnn.com" and “stock.kimo.com.tw” 分別提供新聞和股市資訊。

Step3 選出來自 authority hosts 的 HTML access 或是其 URL 路徑或檔名包含了 “software” 或 “download”等關鍵字的 HTML access。如果 Step3 成立，則 *digAuthorityAssociation* 這個函式會

```
# The iterative scheme
(1)   Foreach a in A
(2)     If isHTMLAccess(a.accessURL) and not irrelevantAccess(a.accessURL)
(3)       If isAuthorityAccess(a.accessURL)
(4)         merge(AuthorityAssociation, digAuthorityAssociation(a.accessURL))
(5)       Else
(6)         merge(CandidateAssociation, digAssociation(a.accessURL))
(7)       End if
(8)     End if
(9)   End foreach
(10)  merge(CandidateAssociation, AuthorityAssociation)
(11)  removeNoiseAssociation(ReferenceAssociation, CandidateAssociation)
```

圖 2: The iterative scheme

根據 HTML access 的 URL 取得這個 HTML 的本體，並且去分析其內容來得到檔案的下載連結。一旦檔案下載連結被找到，這個函式會爲兩者建立一筆關連；如果沒有檔案連結被找到，這個函式還是會產生一筆關連，但檔案的 URL 欄位值是 null，這是因爲這些 HTML 網頁還是可能包含有用的資訊，雖然他們沒有檔案的下載連結。又因爲一個參考文件可能與很多檔案有關連，因此 *digAuthorityAssociation* 回傳所有找到的關連，同時把這些關連合併到 *AuthorityAssociation*。

如果 Step3 不成立，就像 *digAuthorityAssociation* 一樣，*AuthorityAssociation* 會讀取 HTML 物件，然後分析其中是否含有檔案

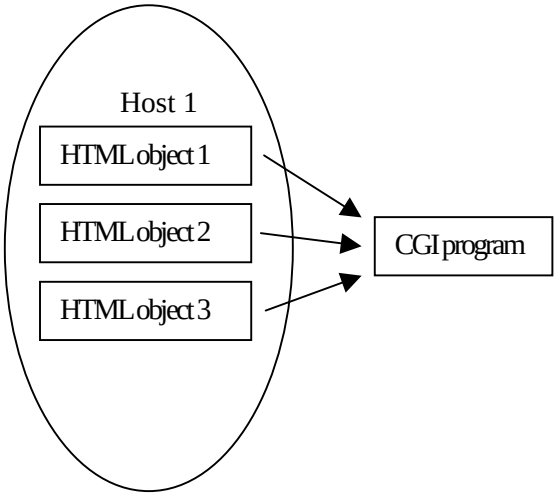

圖 3: 同一網站內 CGI 程式被多個網頁引用示意圖

的下載連結。如果有的話，會建立檔案與文件的關連。然後所有找到的關連會被合併到 *CandidateAssociation*；如果沒有發現檔案連結，則不會產生任何關連，也就是這個 HTML 網頁不會被選爲參考文件。

在迴圈結束之後，Step10 合併 *AuthorityAssociation* 到 *CandidateAssociation*。最後一個 Step 是負責從 *CandidateAssociation* 移除雜訊關連(noise associations)，然後產生最後的關連列表 *ReferenceAssociation*。本文中我們僅討論由 Common Gateway Interface (CGI)所產生的雜訊關連。許多商用網站使用 CGI 程式(如：adclick.exe 或 adserver.exe) 在網頁裡加上自己的廣告，這些程式會出現在 HTML 物件中，而且也有檔案副檔名，如".exe" , ".com"。如此會使得 Step 4 和 Step 6 建立錯誤的關連，影響方法的精準度。

我們觀察到一個現象可以找出 CGI 程式：爲了增加廣告的效益，WWW 管理者通常把所有的網頁都加上廣告的 CGI 程式，這會造成一個特定的檔案同時被在同一主機內的好幾個網頁所連結，如圖 3。而另外一個造成同樣現象的可能原因是有一些商業網站提供免費網頁空間，因此在網站中有多個使用者爲同一個檔案撰寫網頁。但在一般的狀況下，後者引用到同一檔案次數比前者小很多，因此可以檔案被同一網站的網頁的參考次數來判定是否爲 CGI 程式。

圖 4 爲去除雜訊關連的演算法。在 Part I 中我們計算檔案被參考的次數。Step5 檢查兩個狀況：第一，檢查關連的檔案欄位是否是 null，因爲有一些關連是由 iterative 方法的 Step4 所產生，可能沒有與任何檔案相關；第二、檢查檔案的副檔名

```
(1)   Sub removeNoiseAssociation(R, C)
# Part I
(2)       Foreach c in C
(3)           host = getURLhost(c.HTMLURL)
(4)           extension = getExtension(c.ArchiveURL)
(5)           If c.archiveURL != null and (extension == "exe" or extension == "com")
(6)             REFNUM[host][c.archiveURL] = REFNUM[host][c.archiveURL] + 1
(7)           End if
(8)       End foreach
(9)       CGIprogram = getCGIprogram(Threshold, REFNUM)
# Part II
(10)      Foreach c in C
(11)          If not defined CGIprogram[c.archiveURL]
(12)            push(R, c)
(13)          End if
(14)      End foreach
(15)  End sub
```

圖 4: 移除含 CGI 程式關連的演算法

```
# The backtracking scheme
(1)   Foreach a in A
(2)     If isHTMLAccess(a.accessURL) and not irrelevantAccess(a.accessURL)
(3)       If isAuthorityAccess(a.accessURL)
(4)         merge(AuthorityAssociation, digAuthorityAssociation(a.accessURL))
(5)       Else
(6)         push(HTMLAccessWindow, a)
(7)       End if
(8)     End if
(9)     discardAgedAccess(HTMLAccessWindow, a.accessTime, BacktrackingTime)
(10)    If isArchiveAccess(a.accessURL)
(11)      Foreach q in HTMLAccessWindow
(12)        If q.clientHost = a.clientHost and neverCheckedBefore(q)
(13)          merge(CandidateAssociation, digAssociation(a.accessURL))
(14)          discardObject(HTMLAccessWindow, q)
(15)        End if
(16)      End foreach
(17)    End if
(18)  End foreach
(19)  merge(CandidateAssociation, AuthorityAssociation)
(20)  removeNoiseAssociation(ReferenceAssociation, CandidateAssociation)
```

圖 5: The backtracking scheme

是否爲".com" ,".exe"。Step6 計算檔案被同一個網站的網頁引用的次數，這個次數被存放在 *REFNUM* 的關連矩陣裡。Step9 根據 Threshold 的值，從 *REFNUM* 陣列裡挑出可能的 CGI 的程式然後存放於 *CGIPprogram* 裡。Part II 則根據 *CGIprogram* 將所有的雜訊關連移除。

最後，透過參考文件與檔案的關連，我們可以敘述的方式來搜尋檔案。一開始利用網站檢索引擎(indexer)建立參考文件的索引，當檢索引擎根據使用者的查詢找到與其有關的參考文件時，系統會根據這文件與檔案的關連回傳所有有關的檔案，然後結束搜尋。

2.2 The Backtracking Scheme

一個使用者的行爲：先瀏覽網頁再下載檔案，被用來更有效率地發現參考文件。在一般的情況下，使用者很少直接打入 URL 來下載檔案，因爲 URL 太長而且不好記。因而，使用者會先看過網頁後，再按下他們要的檔案連結。這樣的現象表示一個檔案的參考文件，有很大的機會就是使用者下載檔案前所存取的那個網頁。基於這樣的理由，我們回溯(backtracking)使用者在快取伺服器的物件存取順序來找參考文件。

Backtracking 的方法列在圖 5。如 iterative 方法，假設 *A* 是個 sequence；同樣地，每筆關連包含兩個欄位：參考文件的 URL 及檔案的 URL。Step2, 3 and 4 與 iterative 的方法相同。然而，當存取 *a* 不滿足 Step3 時，這筆存取會被存放到 *HTMLAcessWindow*，如 Step6 所示。Step9 根據最後一個存取的存取時間(*a.accessTime*)和最大的回溯時間(*BacktrackingTime*)丟棄 *HTMLAcessWindow* 中過期的存取。這步驟有兩個意義：第一、隨著回溯時間的增加，找到包含檔案連結的文件機會愈低；第二、由於緩衝區大小的限制，不可能保留所有的 HTML 存取記錄。

當發現一筆檔案存取時(Step10)，這個程式將會檢查 *HTMLAccessWindow* 是否有與這檔案存取有相同客戶端來源的 HTML 存取，而且這個 HTML 存取是第一次被檢查。如果有 HTML 存取符合條件，Step13 會讀取這個 HTML 物件，並且檢查其內容是否含有檔案連結，然後合併所有發現的關連到 *CandidateAssociation*。迴圈結束後，Step19 將 *AuthorityAssociation* 合併到 *CandidateAssociation*。最後移除 CGI 後，產生 *ReferenceAssociation*。

2.3 挑選關連

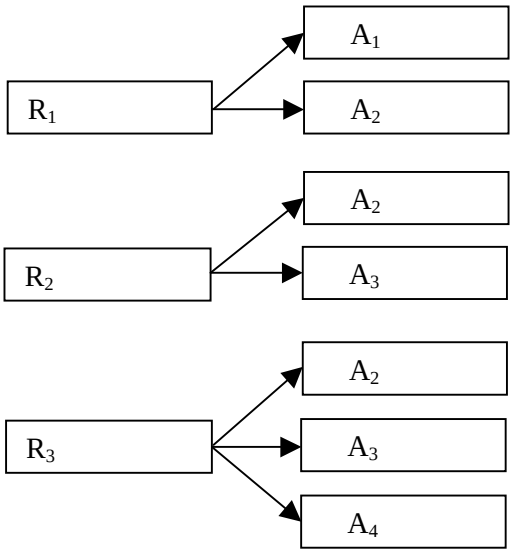

圖 6: 根據 archive 被關連參考的次數來排序

如前面提到，一旦檢索引擎根據使用者的查詢找到與其有關的參考文件時，我們可以根據參考文件與檔案的關連回傳有關的檔案，然後結束搜尋。但是，一個參考文件可能與很多個檔案相關，那麼哪一個檔案才是正確的？如果傳回全部的檔案，很有可能摻雜不相干的檔案，如此會導致回傳精準度的降低。我們提出兩個方法來處理這種一對多的關係：首先，最靠近使用者輸入的關鍵字的檔案連結，應該有最大可能是正確的。因此最直接的方法，就是分析整個網頁的內容來找到符合的檔案，但是這樣會增加查詢的反應時間。

另外一種方法是根據不同的參考網頁來挑選檔案，它包含四個步驟：

1. 計算在檔案與文件關連中每一個檔案被參考文件引用的次數。
2. 根據被引用的次數把他們分成幾個群組。
3. 每個群組中依照他們的參考文件所含的關鍵字多寡來排序。
4. 依照被引用的次數及關鍵字的數量來挑選檔案與文件關連，愈高的愈優先。

圖 6 表示出我們的方法。假設檢索引擎回傳 7 個關連：$<R_1, A_1>$, $<R_1, A_2>$, $<R_2, A_2>$, $<R_2, A_3>$, $<R_3, A_2>$, $<R_3, A_3>$,$<R_3, A_4>$，R_i 和 A_i 分別代表參考文件與檔案。我們假設 R_1 有最多關鍵字，R_2 次之，R_3 最少。然後我們計算每個檔案被參考文件引用的次數，A_1 及 A_4 都只被引用一次，A_2 被引用三次，A_3 兩次。然後 A_1 及 A_4 被分在同一個群組，A_2 及 A_3 在不同的兩組。最後因爲 A_2 被引用的次數最多，所以 A_2 (*i.e.* $<R_1, A_2>$, $<R_2, A_2>$, and $<R_3, A_2>$) 有關的關連被排在最前面，A_3 其次，A_1 及 A_4 在最後。再根據參考文件所含的關鍵字數量，最後的順序是$<R_1, A_2>$, $<R_2, A_2>$, $<R_3, A_2>$, $<R_2, A_3>$, $<R_3, A_3>$, $<R_1, A_1>$, 和 $<R_3, A_4>$。

2.4 演算法分析

由找參考文件的整個過程，我們發現讀取 HTML 物件(*i.e.* Step 4 和 6 在圖 2； Step 4 和 13 在圖 5)是最花時間的步驟，原因是這些步驟花很多時間在網路或 disk I/O 上，相對之下，其它的步驟在記憶體處理，所花的時間可被忽略。

因此把焦點放在讀取 HTML 物件上面。既然我們的方法利用快取伺服器，則存取 HTML 物件會有兩個可能：第一種，HTML 物件已經在快取中，可直接取出；第二種，不在快取中，必須從原本的 WWW 站台取得。在這兩種情況下，如果 HTML 物件不在記憶體中的話，均需要額外的 disk I/O。爲了簡化問題，我們僅考慮最壞的狀況：所有 HTML 物件均沒有暫存於記憶體中。假設分析一個參考文件的時間爲 T，存取 disk 的時間爲 T_D，網路傳輸時間爲 T_N，快取擊中率爲 r。因此 $T = r * T_D + (1 - r)T_N + (1 - r)T_D = T_D + (1 - r)T_N$ ，$(1\text{-}r)T_D$ 是從原本的 Web 站台將參考文件讀出所需要的 disk 時間。當快取是完美時(*i.e.* $r = 1$)，$T=T_D$。此外如果 $T_N >> T_D$ 則 $T = (1\text{-}r)T_N$。

最後我們計算兩個方法所需要的時間。在 iterative 方法中，假設有 m 個不同的 HTML 存取記錄在 A 中，則執行的時間爲 mT。在 backtracking 方法中，假設有 k 個檔案存取記錄在 A 裡，u_i 代表在每筆檔案存取記錄所對應相同客戶端的 HTML 物件數量，$1 \le i \le k$。n 代表總共需要讀取的 HTML 物件，則 $n = \sum_{i=1}^{k} u_i - R$ ，R 代表重複的 HTML 物件數目，所以 backtracking 方法所需要的時間爲 nT，因爲 $n \le m$ 它的上限爲 mT；假設只回溯一步 (*i.e.* $u_i = 1, 1 \le i \le k$)，且所有的 HTML 物件均相異 (*i.e.* $R = 0$)，則這個方法所需時間的下限爲 kT。

3 效率分析

我們利用使用者的存取記錄來分析方法的效果。所分析的存取記錄檔和快取物件主要來自國立中央大學最主要的網路快取伺服器上 proxy.ncu.edu.tw，這部伺服器架設在雙 Pentium III CPU 與 2GB RAM 的 PC 上，作業系統是 FreeBSD [5]，代理伺服器的軟體是 SQUID [9]，每天大約有 4,500 個使用者和 5,000,000 次存取。我們分析 2001 年 5 月 30 日到 6 月 12 日十四天的使用者存取記錄檔，同時蒐集這十四天的 HTML 物件。此外，我們利用測試版的 Tornando [10] 搜尋引擎來索引參考文件。

表 1: 利用 iterative scheme 可以發現 11,558 個參考文件提供者.

Rank	Host name	Referring pages
1	home.kimo.com.tw	5,618
2	Toget.pchome.com.tw	4,330
3	www.geocities.com	1,578
4	home.pchome.com.tw	1,517
5	home1.pchome.com.tw	1,068
6	members.nbci.com	1,023
7	taiwan.cnet.com	784
8	freehomepage.taconet.com.tw	753
9	www.netvigator.com.tw	733
10	members.tripodasia.com.tw	534

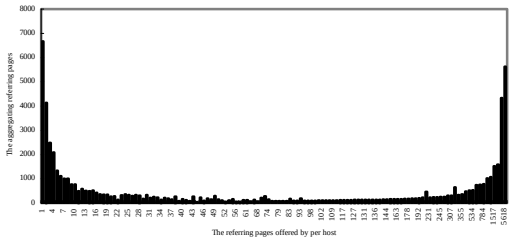

圖 7: 在 iterative scheme 中參考文件的分佈

3.1 The Iterative Scheme 效率分析

Referring Page Analysis

利用 iterative 方法，我們從 871,811 個 HTML 物件挑出 61,716 個參考文件，比例是 7.1%。如表 1 所示，這些文件分佈在 11,558 個網站上，前十大網站提供了大約 30% 的參考文件，這些網站有的是專門提供檔案下載的網站或是提供免費網頁空間的 WWW 站台。

圖 7 表示參考文件在網站的分佈，Y 軸代表在提供相同數目的參考文件網站，其所有參考文件數目總和；X 軸代表每個網站提供的參考文件數目。我們可以發現參考文件分部在大量的網站上，並沒有完全集中在少數的網站上，這種現象造成收集參考文件相當困難。

Archive Analysis

找到的檔案數目是 550,999 個，超過 80%的檔案(437,971)是以 URL 的形式存在，這使得我們可以直接告訴使用者哪裡可以下載檔案。類似地，這些檔案分佈在 24,117 個網站上，前十大網站也提供了大約 1/3 的檔案下載連結，即 133,337 個，如表 2 所示。這些網站有的是專門提供檔案下載的網站或是提供免費網頁空間的 WWW 站台。此外 437,971 個檔案分佈於 44,427 個參考文件上，每一個文件平均擁有 10 個檔案下載連結。

圖 8 表示檔案結連在網站的分佈，Y 軸代表提供相同數目的檔案連結網站，其所有檔案連結數目的總和；X 軸代表每個網站提供檔案連結的數目。同樣地，我們發現檔案連結分部在大量的網站上，並沒有完全集中在少數的網站上，這種現象也造成收集檔案連結相當困難。

表 2: 利用 iterative scheme 可以發現 24,117 個 archive 提供者

Rank	Host name	Archive URLs
1	home.pchome.com.tw	34,134
2	home.kimo.com.tw	23,651
3	www.8bn.com	17,976
4	www.geocities.com	13,607
5	Members.xoom.com	12,295
6	www.netvigator.com	7,549
7	cracks4u.com	7,498
8	www.aceroms.com	6,428
9	members.nbci.com	5,717
10	home1.pchome.com.tw	4,482

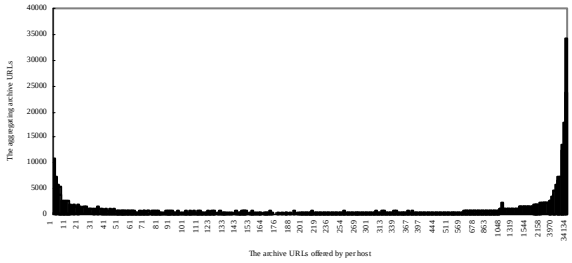

圖 8: 在 iterative scheme 中 archive URL 的分佈

Association Analysis

我們總共找到 763,816 個檔案與參考文件的關連(associations)。幾個產生關連的重要步驟其效果列在表 3，"Remove the irrelevant access log"只需要花一點 disk I/O 即可減少 8%的 HTML 存取，是相當快而且有效；"Generate the authority association"存取了 14,008 個網頁，其中 42%的網頁(5,897)包含檔案連結，其所產生的 candidate association 佔了 9%；相對的，"Generate the candidate association" 存取 788,512 個網頁，只獲得 56,189 個 candidate referring pages，比例只有 7.1%，但這步驟產生的 candidate association 佔 91%

最後一步移除含有 CGI URL 的關連，圖 9 顯示檔案連結被同一個網站引用次數的分布，大約 85%的檔案 URL 只被引用一次。本文中被引用次數在前 0.5%的檔案連結我們視爲 CGI URL。表 3 顯示在去除 CGI URL 之後 candidate association 減少 1.8%，(69,830 + 707,976 – 763,816)/(69,830 + 707,976)，但是 candidate referring page 減少 12.1%，(70,197 – 61,716)/70,197。我們進一步發現雜訊網頁主要來自於演算法的 Step 6，佔了全部的 99%，(70,197 – 61,716 – 91)/(70,197 – 61,716)，而

表 3: iterative scheme 中幾個重要步驟的效果

Step name	The HTML accesses before applying the step	The found referring pages after applying the step	The pages with archive URLs	The found associations
Remove the irrelevant accesses according to their URLs	871,811	802,520	NA	NA
Generate the authority associations	14,008	14,008*	5,897*	69,830*
Generate the candidate associations	788,512	56,189*	43,988*	707,976*
Remove the CGI URLs	70,197	61,716	44,427	763,816

* These results were not processed by the CGI programs removal.

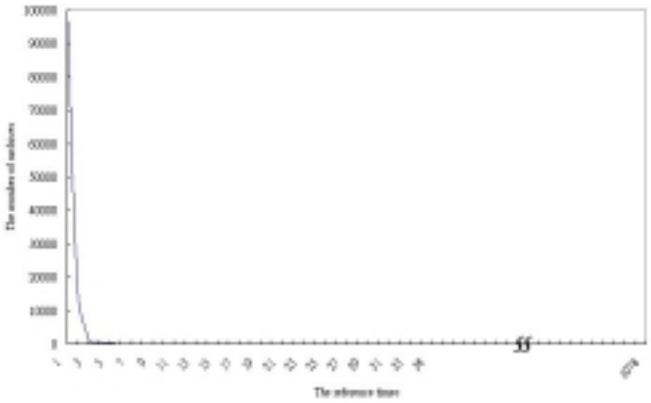

圖 9: 在 iterative scheme 中 archive URL 被參考次數的分佈

來自於 Step 4 的只佔了 1%。

Precision Analysis

我們集合了 14 位主修資訊工程的使用者來測試描述搜尋的效果。每個人查詢十次用來找所需要的檔案，而且使用者所下的描述並不需要太精準。事實上，使用者常常輸入相當模糊的關鍵字來找尋他們需要的。因爲我們希望使用者能夠在一頁的回傳結果裡找到他們所要的，因此每個查詢最多回傳十筆結果，每筆回傳結果包含檔案 URL 與其參考文件。測試者被要求檢查文件的內容及檔案的 URL 來確定回傳的結果是否正確。在 140 筆查詢中總共回傳的結果是 1,088 筆。76%回傳正確的參

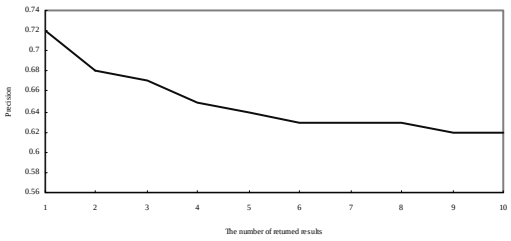

圖 10: iterative scheme 中每個查詢回傳結果數目與準確度的關係

表 4: 利用 backtracking scheme 可以發現 8,053 個參考文件提供者.

Rank	Host name	Referring pages
1	toget.pchome.com.tw	4,330
2	home.kimo.com.tw	3,529
3	home.pchome.com.tw	1,221
4	www.geocities.com	1,141
5	taiwan.cnet.com	781
6	members.nbci.com	593
7	home1.pchome.com.tw	535
8	members.tripodasia.com.tw	496
9	freehomepage.taconet.com.tw	459
10	www.netvigator.com.tw	450

考文件，其中 62%的回傳包含正確的參考文件與正確的檔案 URL；而 14%是回傳錯誤的檔案 URL，其原因是 Section 2.3 所提的方法沒有從多個關連裡正確的選擇使用者所需的檔案。另一項結果是查詢第一筆回傳結果正確的比例爲 0.72，但當回傳的筆數增加時，準確度降低，如圖 10。

3.2 The Backtracking Scheme 效率分析

Referring Page Analysis

在 backtracking 時間是 10800 秒的情況下，我們從 284,192 個 HTML 物件挑出 42,684 個參考文件，比例是 15%。如表 4 所示，這些參考文件分佈在 8,053 個網站上，前十大網站提供了大約 32%的檔案相關文件(13,535 個)，這些網站幾乎與表一所列的網站相同。

Archive Analysis

找到的檔案數目是 429,933 個，超過 80%的檔案(347,993)是以 URL 的形式存在。同樣的表 5 顯示這些檔案分佈在 19,280 個網站上，前十大網站大約提供了 1/3 的檔案下載連結，即 114,763 個，這些網站與表 2 列的網站相同。此外這些檔案散佈於 30,197 個參考文件上，每個文件平均有 11.5

個檔案下載連結。

表 5: 利用 backtracking scheme 可以發現 19,280 個 archive 提供者.

Rank	Host name	Archive URLs
1	home.pchome.com.tw	29,169
2	home.kimo.com.tw	20,425
3	www.8bn.com	16,582
4	www.geocities.com	11,132
5	members.xoom.com	9,988
6	cracks4u.com	7,498
7	www.netvigator.com.tw	6,750
8	www.aceroms.com	4,880
9	members.nbci.com	4,780
10	www.soft999.com	3,559

Association Analysis

總共找到 554,950 個檔案與參考文件的關連。幾個重要步驟的效果列在表 6，"Remove the irrelevant access log" 和 "Genera the authority association"這兩個步驟與 iterative scheme 中提到的相同，"Generate the candidate association"分析了 270,170 個網頁，然而只獲得 31,586 個 candidate referring pages，比例是 11.7%，此外這步驟找到的 candidate association 佔了 87.5%。

最後一步移除含有 CGI URL 的關連，如前面，被引用次數在前 0.5%的檔案我們視爲 CGI URL。表 7 顯示在去除 CGI URL 之後 candidate association 減少 1%，(69,830 + 490,937 – 554,950)/(69,830 + 490,937)，但 candidate referring page 減少 6.4%，(45,594 – 42,684)/45,594。我們發現雜訊網頁主要來自於演算法的 Step 13，佔了全部的 96.9%，(45,594 – 42,684 – 91)/(45,594 – 42,684)，而來自於 Step 4 的只佔了 3.1%。

表 6: backtracking scheme 中幾個重要步驟的效果

Step name	The HTML accesses before applying the step	The found referring pages after applying the step	The pages with archive URLs	The found asso-ciations
Remove the irrelevant accesses according to their URLs	871,811	802,520	NA	NA
Generate the authority associations	14,008	14,008*	5,897*	69,830*
Generate the candidate associations	270,170	31,586*	26,435*	490,937*
Remove the CGI URLs	45,594	42,684	30,197	554,950

* These results were not processed by the CGI programs removal.

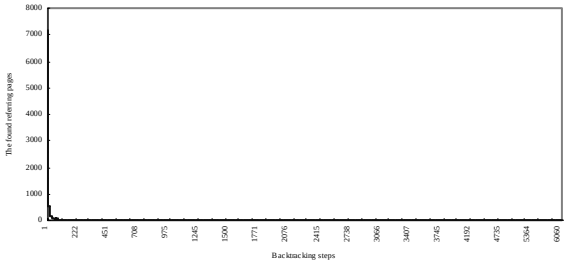

圖 11: 發現的參考文件數目與 backtracking 次數的關係

Backtracking Analysis

圖 11 顯示參考文件與 backtracking steps 的關係分佈，當 backtracking 的次數增加，這個曲線快速下降。這代表大部分的參考文件在一開始的 backtracking step 被找到。圖 12 顯示參考文件與 backtracking 時間的關係分佈，這個圖的曲線並不如前一個地快速下降，但是它仍然表示出大部分的參考文件在一開始的 backtracking time 被找到。另一項有趣的結果是這個曲線沒有平滑地收斂，上面有兩個突起的小碎波，這種情況可能與使用者的使用習慣有關。在我們的測試中，是以中央大學的使用者存取記錄作爲分析的對象，大部分的使用者是學生，所以他們使用網路的習慣可能會影響這個曲線，如果對不同的對象分析，結果也許會有些不同，然而我們認爲下降的趨勢是不會改變的。

在圖 13 中，藍色的虛線顯示 backtracking 次數與找到參考文件比例的關係，當 backtracking 次數等於 1 的時候，最高的比例爲 0.68；假設 iterative 可以找到所有的參考文件，當 backtracking 無限次時比例就等於 iterative 的比例，紅色的實線表示 backtracking 次數與 backtracking 能找到的文件佔全部文件比例的關係。兩條線交於 backtracking step=4 和 ratio=0.43，當 backtracking 無限次時比例

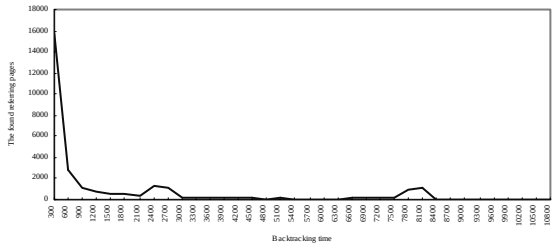

圖 12: 發現的參考文件數目與 backtracking 時間的關係

就等於 1。

在圖 14 中，藍色的虛線顯示 backtracking 時間與找到參考文件比例的關係，當 backtracking 時間爲 300 秒時，最高的比例爲 0.30；假設 iterative 可以找到所有的參考文件，當 backtracking 時間無限大時比例就等於 iterative 的比例，紅色的實線表示 backtracking 時間與 backtracking 能找到文件佔全部文件比例的關係。當 backtracking 時間無限大時比例就等於 1。

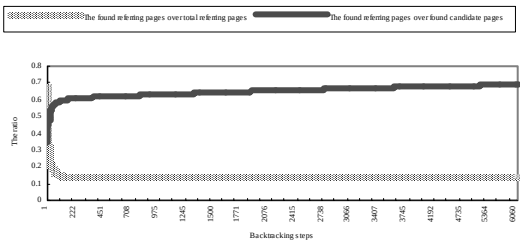

圖 13: backtracking 的次數和發現參考文件比例的關係

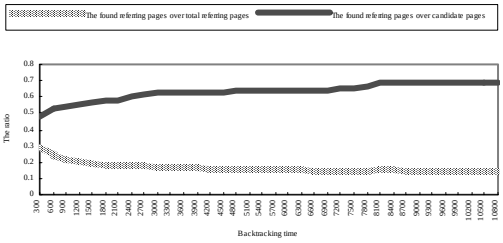

圖 14: backtracking 的時間和發現參考文件比例的關係

Precision Analysis

爲了測試 backtracking 在最糟情況下的效果，我們的 backtracking step=1 和 backtracking time=10,800 秒，其它的條件如 iterative。這個方法從 30,976 個 HTML 網頁中找到 21,051 個參考文件，找到參考網頁的比例爲 68%。在 140 筆查詢中傳回的結果有 1,055 筆。78%回傳正確的參考文件，62%回傳正確的參考文件與正確的檔案 URL。如前，16%傳回錯誤檔案 URL 是由於 Section2.3 提出的演算法沒有從多個關連裡完全正確的選到使用者所需的檔案。如圖 15 顯示，查詢第一筆回傳結果正確的比例爲 0.71，但當回傳的筆數增加時，準確度降低。

4 相關研究

以下各小節概括的說明之前關於檔案下載服務方面的研究

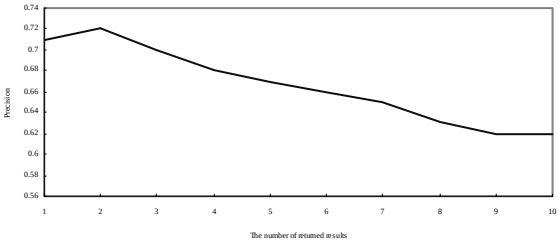

圖 15: backtracking scheme 中每個查詢回傳結果數目與準確度的關係

4.1 Archie

Archie [2][3] 是由 School of Computer Science, McGill University 開發，之後由 Bunyip Information Systems Inc. Canada 所維護。這系統被認爲是找在 FTP 伺服器裡檔案的最有效方式。使用者輸入部分或全部檔名，而 Archie 會將所有儲放該檔案的 FTP 站列出來。

表 7: ProxyFTP, CFTP and FTP servers 的比較

	ProxyFTP	CFTP	FTP servers
Archive object type (by source)	WWW/FTP	FTP	FTP
Archive list for Archie	Yes	No	Yes
Description searching	Yes	No	No
User interfaces	A public archive directory and search functions	Search functions	A public archive directory
Archive classification quality	Poor	Not available[1]	Good
Remote FTP site's directory listing	No	Yes	No
Management overhead	Little	Little	Normal

1. CFTP does not provide a public archive directory.

Archie 尋找檔案的方式是首先由管理者提供一個 FTP 站列表當成檔案列表的來源，接著 Archie 的 retrieve component 以匿名方式登入列表中的 FTP 站台，取得檔案列表 ls-lR(.gz)或藉由遞迴方式搜尋 FTP 站的目錄，自己產生檔案列表。parse component 再去除其中不必要的部份，並且進行格式轉換及壓縮。Archie 的主要缺點是無法像一般搜尋引擎一樣對 Web 站做檔案搜尋。

4.2 CFTP

Caching FTP server (CFTP) [8] 由 Mark Russell 和 Tim Hopkins 在 1998 年提出。CFTP 可以像一般 FTP 站一樣供使用者下載檔案。事實上，CFTP 提供 FTP proxy 的功能，藉由網頁介面使用者輸入 FTP 站名，然後 CFTP 會列出目的地 FTP 站的檔案和目錄。當使者選取想下載的檔案時，CFTP 會檢查檔案是否在自己的快取中，如果有就將快取中的檔案回傳給使用者，否則就連線至目的 FTP 站下載並快取檔案。表 7 列出了 ProxyFTP, CFTP 和普通 FTP 站的比較。

5 結論

檔案下載是 Internet 一個重要的服務。許多使用者透過網路下載共享軟體，驅動程式及其他工具程式。我們之前已經提出如何改善網路快取伺服器的功能來提供檔案下載服務。本文中我們提出兩種方法從網路快取伺服器中找到檔案的參考文件，來幫助檔案的使用。Iterative 方法幾乎可以找到所有在網路快取伺服器中的參考文件，而 backtracking 方法則可以較低的成本找到參考文件。最後我們索引找到的參考文件，並且以 14 位使用者測試其效率。實驗結果顯示在 iterative 方式中，查詢回傳結果第一筆就正確的比例爲 0.72，在 1,088 筆回傳結果中正確的比例爲 0.62。此外，我們發現以 backtracking 方式比我們期望的結果好很多，它第一筆就正確的比例爲 0.71，而所有回傳結果正確的比例爲 0.62，但它需要測試的 HTML 檔案卻只有 iterative 方法的 1/26。因此，若準確度不是最優先的考慮，則 backtracking 方式爲找尋參考文件的較佳解決方法。未來我們計畫改善挑選關聯的演算法，以提高搜尋的準確度。未來我們計畫改善挑選關聯的演算法，以提高搜尋的準確度。

誌謝

感謝行政院國家科學委員會支持，計畫編號 NSC 90-2213-E-008-045.

參考文獻

[1] CNET Networks, Inc., http://www.download.com

[2] P. Deutsch, “Resource discovery in an Internet environment – the Archie Approach”, *Electronic Networking: Research, Applications and Policy*, VOL. 2, NO. 1, pp. 45-51, Spring 1992.

[3] A. Emtage and P. Deutsch, “Archie – An Electronic Directory Service for the Internet”, in *Proceedings of the Winter 1992 USENIX Conference*, pp. 93-110, 1992.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1”, *RFC 2616*, June 1999.

[5] FreeBSD, http://www.freebsd.org

[6] PC Home Online, http://www.toget.com.tw

[7] J. Postel and J. Reynolds, “File Transfer Protocol”, *RFC 959*, 1985.

[8] Mark Russell and Tim Hopkins, “CFTP: a caching FTP server”, *Computer Networks*, VOL. 30, NO. 22-23, pp. 2211-2222, 1998.

[9] Squid Internet Object Cache, http://squid.nlanr.net/Squid

[10] Tornado Technologies Co., http://www.tornado.com.tw/

[11] Hsiang-Fu Yu, Yi-Ming Chen, Shih-Yong Wang, Hsin-Yi Lu and Li-Ming Tseng, “Internet Archive Service through Proxy Cache”, in *Proceedings of the 2001 International Conference on Internet Computing* (*IC’01*), VOL. 2, pp. 807, June 2001.