以使用者側寫、文件結構及伺服器端提示為基礎之行動網際網路資訊預取

吳秀陽　李青松

國立東華大學資訊工程學系

showyang@csie.ndhu.edu.tw

摘要

礙於無線通訊技術與頻寬的限制，在行動計算環境上的資訊服務，無法達到與有線網路相同的水準，加上使用者的行動性，常使得訊號無法送達而造成斷訊，因此無法普及。預取技術利用使用者瀏覽網頁的閒置時間預先取回可能被存取的頁面，是解決上述問題之可行而有效技術。傳統的預取方法統計過去使用者的存取紀錄，利用統計的結果以預測將來的存取。這種預測方法對於系統中未曾有過存取記錄的使用者而言，效果不明顯。

我們提出一個以網頁內容爲基礎，加上使用者側寫、伺服器端特殊結構網頁分析、過去使用者集體存取行爲等機制的預取方法。對使用者存取序列作分析，利用資料分類（data classification）的技術分出存取行爲接近的使用者群組。使用者側寫經由對使用者瀏覽行爲的監測並更新，以表現使用者當時的瀏覽興趣。伺服器端將行動客戶端歸類至與其存取行爲接近的群組，再參考目前使用者瀏覽頁面產生暗示。一般頁面從客戶端所屬的群組中搜尋，特殊結構的網頁則計算其存取傾向，得到此群組將來可能存取頁面，並作爲暗示傳給客戶端。行動客戶端收到暗示後計算出最符合目前興趣的頁面，而後預取之。實作與實驗結果，預取命中率可以達到 30%以上。如果使用者存取記錄可以包含因爲瀏覽器的快取（cache）機制所隱藏的部分存取行爲，則預取的命中率可再提升。

KEYWORD: 行動計算、預取、伺服器端暗示、使用者側寫、瀏覽行為、資料分類

1. 簡介

隨著網路科技與硬體的快速進步，無線通訊從實驗階段逐漸落實到可提供現實應用的網路環境[2][3][4]，提供使用者不受時間與空間的限制，隨時隨地擷取資訊並完成資料交換的能力。「行動計算」即是在無線通訊的環境中，提供使用者存取或處理資訊的能力；亦即使用者帶著無線通訊設備，透過無線通訊網路媒介滿足傳輸資訊、處理資料目的，達到遍在計算（ubiquitous computing）的境界。

行動計算環境雖然提供了便利的通訊機制，然而無線網路因訊號易受干擾，或 Hand off 動作不夠平順等情況，常有斷訊（disconnection）發生。具體的解決方法有預取（prefetch）[7]、複製（replication）[5]。

預取的構想爲預測將來會存取的資料，在系統閒置的時間做預先存取的動作，並將預取的資料放到本地端的 cache。當用戶端真正的需要此份資料時，可以直接存取已存在 cache 的資料，不必再向伺服器端做 requests。因此，成功的預取將減少用戶端等待時間，也降低伺服器端與網路的負載。另外，利用系統閒置時間做預取，也增進系統的 throughput。

過去預取研究，多以統計使用者存取記錄的方式，在客戶端或伺服器端記錄存取的序列，透過資料壓縮理論與機率上的考量，以舊的存取序列預測使用者未來的存取行爲。這類方法在客戶端需有紀錄使用者存取的機制，而且通常紀錄一段時間後，才開始產生預測命中的效果。系統遇到新的使用者時，必須重新紀錄存取序列，再次作訓練（training）的動作。我們認爲網站上的內容，決定使用者大部分瀏覽的行爲，特殊結構的頁面也可能影響使用者行爲。舉例而言，瀏覽網站時通常會被和興趣相關的關鍵字（keywords）吸引，而點選 LateX2HTML 類型的網頁時，也常點選下一頁（next）的超連結，此即網頁結構影響使用者行爲的例子。另一方面，內容較爲專一的網站，使用者的存取行爲有相似性[1]，因此集體的行爲可用以預測新使用者的行爲。實際從網站的存取日誌中取出最常來訪的使用者，並分

析存取序列相似性後分出數個群組，觀察後發現每個群組確有其特定的瀏覽傾向。在有了這樣的觀察結果後，如果將已來訪使用者歸類爲其所屬群組，並採用其群組的使用者集體存取行爲預測，應能得到不錯之預取命中率。而面對新的使用者時，則試著將其歸類到某群組或某幾個群組，以這些群組的使用者集體存取行爲，以期解決新使用者加入的問題。

以上述構想爲出發點，在不更動現存超本文傳輸協定（HTTP）的情況下，我們發展一套適用於無線網路環境之預取演算法，並且加以實作評估。我們從網站上的存取日誌檔觀察使用者集體的存取行爲，接著分析網頁內容與結構是否影響使用者瀏覽行爲，並依此分析提出適合之預取演算法，接著就演算法定義出系統架構並實作之，最後用實驗驗證預取演算法的有效性。實驗的部分包括分群樣本數與預取效能的關係、預取命中率、預取的overhead、預取與否對系統反應時間的影響等。

2. 瀏覽行為觀察與分析

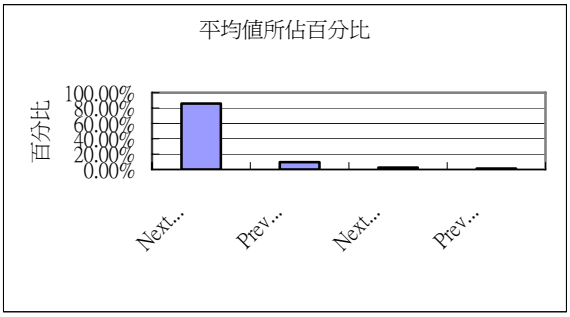
使用者的瀏覽行爲模式，Carlos R. Cunha 與 Carlos F. B. Jaccoud 的研究中將其分爲兩種[6]，一種是對於新資訊有極高興趣，因此每次存取的網頁，都是未曾存取過的。這類使用者對瀏覽過的資訊，再次瀏覽的機率較低。另一種瀏覽行爲則是遵照過去的存取行爲，對於存取過的網頁常常瀏覽。代表這樣的使用者對於已瀏覽過的網頁，有很大的機率重複瀏覽。

我們的假設是，使用者的瀏覽行爲，不單單僅是這樣的模式，經常是兩者的綜合。以下說明爲驗證假設而作的觀察與分析。

2.1 使用者瀏覽網頁的行為

Lindo Tauscher 與 Saul Greenberg 的研究[9]，觀察到使用者在重新瀏覽已瀏覽過之網頁時，有數種不同的行爲；Virgilio Almeida 和 Azer Bestavros 提出有關在 Web 上存取時產生的區域性特徵（reference locality）[1]，其研究顯示，WWW 上的存取確實具有空間與時間上的區域特性。時間區域特性表示最近存取過的頁面有可能再度被存取，我們視之爲使用者依興趣瀏覽的表現。空間區域特性即目前瀏覽頁面鄰近超連結所代表的頁面有可能被存取，是由網頁結構所造成。符合我們對於內容與特殊網頁結構影響使用者瀏覽行爲的設想。Web server 通常提供網站管理者記錄網頁被瀏覽順序的機制。這些使用者瀏覽的過程記錄於存取日誌檔（access log），包括了使用者的 IP address、存取的日期時間、在 HTTP 通訊協定中 Web server 回覆的狀態代碼（status code）、存取檔案名稱、大小等資訊。存取日誌檔也將使用者瀏覽網頁時的次數、先後順序關係表現出來。因此可以自其中觀察出使用者的存取模式（access pattern），也就是使用者瀏覽紀錄中顯著的存取現象。像是使用者存取網頁時，常採用的瀏覽路徑，在存取日誌檔中即一段存取記錄的序列。

網頁特殊結構影響使用者瀏覽行爲方面，經過對 Power Point 檔轉 HTML 頁面存取記錄的觀察，發現使用者點選「下一頁」（next page）超連結的傾向最高，如圖一。因此預取時計算存取傾向比考慮內容來的簡單有效率。LaTeX2HTML 的頁面也有如上特性。



圖一 PPT2HTML 頁面點選超連結傾向

2.2 與預取策略相關之知識與應用

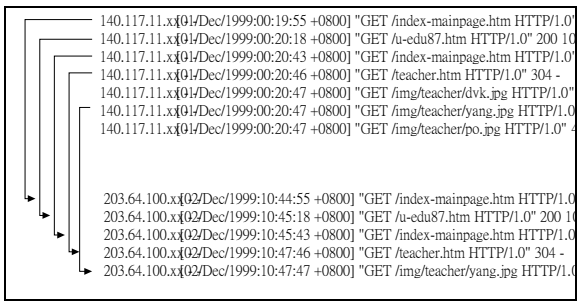
我們的預取方式，參考使用者瀏覽目的與網頁內容之相關性，以及網頁結構造成使用者存取模式改變等兩項因素爲主要考量。客戶端依使用者側寫檔（user profile）記錄使用者的興趣，以代表其瀏覽網頁的目的，在伺服器端則運用資料分類的技術，分析存取日誌檔中使用者存取模式，並將存取模式相似者歸類爲同一群集，藉此得到數個存取模式互異之群集。這些不同的群集代表不同的存取模式，隱含的意義則是各群組使用者瀏覽網頁的目的不同。另外，分類出的群集是之前所有使用者的群體存取行爲，在新的使用者對伺服器發出資料索求時，可回應以暗示。

2.2.1 以使用者存取日誌分類使用者群集

從 Web server 收集到的存取日誌檔，內含眾多使用者的瀏覽網頁過程。然而這些瀏覽次序交相參雜難以分析，因此先將每個使用者從

中分類出來，再依其存取網頁的時間予以排序。每個使用者的 access log 就其每段 session 中存取模式，與其他使用者 access log 每段 session 之存取模式做比較，以得出其差異性。存取序列的長度爲每個 session 存取次數之和，比較差異時存取序列不必相同長度，如圖二。距離的衡量說明如下：

1. 兩序列裡的存取相對位置不變，且符合此條件的存取次數大於兩序列長度平均之一半時，判定爲相同之存取模式。
2. 兩個序列長度相差過多時，距離加 1。相差過多的衡量標準是某序列長度大於另一序列長度一倍時。

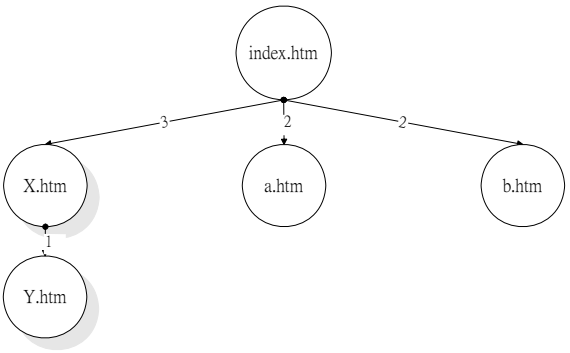


圖二　使用者存取模式之差異

從使用者在存取日誌檔中，兩兩作其差異性衡量的結果，可得到每對使用者之間的距離，這樣的關係應用矩陣表示，我們稱之爲**差異度矩陣**。在過去的研究中提出不少從差異度矩陣區分群集的演算法，我們採用的是 Single Link Algorithm。這個演算法的運作是每次取距離最接近的兩個點作爲新的點，直到最後剩下一個點（群集）爲止。至此便可以分類出存取模式相似的使用者群集。

2.2.2 伺服器端暗示

被區分出的各個使用者群集代表不同的瀏覽模式。我們將每個群組的存取模式萃取成特殊資料結構，以利於產生伺服器端暗示。使用者存取網頁產生的存取序列，一般以首頁的索引頁（index.htm）爲起點。接著由索引頁繼續存取其內含的超連結。因此，將索引頁視爲樹的根節點（root node），延伸出的超連結爲其子節點（child node），形成存取模式樹。從存取模式樹上某個節點 X 到其子節點 Y 的路徑（path），即爲瀏覽網頁 X 後再瀏覽網頁 Y 的存取模式，節點與節點間之連線代表存取的次數，方向代表存取先後順序，如圖三。



圖三　存取模式樹

根據存取模式樹，伺服器端從使用者目前存取頁面判斷其所屬使用者群集，得到將來可能被存取的頁面，我們稱之爲候選頁面（candidate pages）。

2.2.3 客戶端使用者側寫

客戶端的使用者側寫，表現使用者興趣並評估候選頁面預取與否，紀錄的資訊包括：

1. 使用者的瀏覽興趣。
2. 使用者最近瀏覽網頁的大致內容。

第一項資訊採用網頁之文件向量（document vector），以 16 個領域表示使用者的瀏覽興趣。文件向量定義爲 16 個維度的向量。每個維度代表某個領域，其權重（weight）越大，代表網頁內容與該領域的關聯越強。這些領域的分類參考自 altavista 與 openfind 等搜尋網站[10][14]。詳細的定義如下：

(人物團體, 電腦網路, 日常生活, 醫藥保健, 藝術人文, 國家政治, 音樂戲劇, 地理自然, 影視娛樂, 資訊媒體, 休閒運動, 教育研究, 工商經濟, 社會文化, 知識學術, 區域, 世界)

第二項資訊從保留最近瀏覽網頁內容的關鍵字組中獲得。我們在 Web 伺服器上的設定作了修改，使用 Java Servlet 代理回應客戶端的資料索求，並事先在伺服器端對每份網頁作建構文件向量與萃取關鍵字的動作，因此在服務使用者的資料索求時，可以將被索求網頁的文件向量與關鍵字組附加在其中，而客戶端利用 IBM WBI 開發工具提供之環境與 API 撰寫之程式，將回應網頁中的附加資訊予以記錄，再將此資訊於網頁中移除後顯示於瀏覽器。被記錄的文件向量與關鍵字組則作爲動態更新的使用者側寫的資料來源。客戶端在系統不忙碌時，將使用者目前瀏覽頁面與側寫送至伺服

器端，伺服器端將使用者歸類並產生暗示回應給客戶端，一旦伺服器端的暗示被送到客戶端時，客戶端利用使用者側寫檔的資訊以評估候選頁面預取與否。

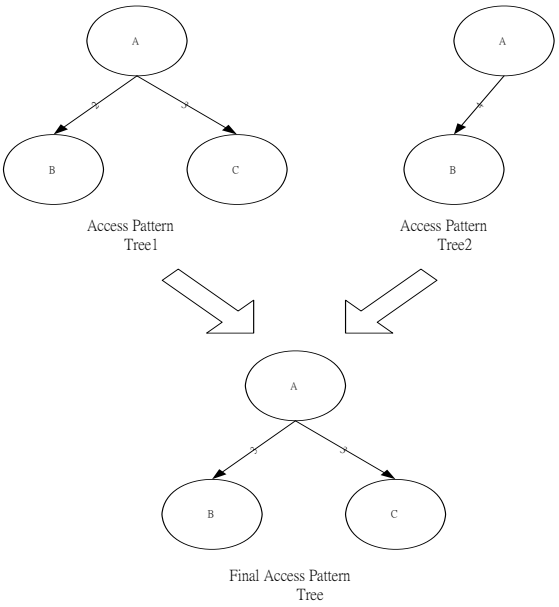
將使用者側寫格式如此定義的考量，在於減少伺服器端傳輸網頁內容之描述時的資料量，以適應網路頻寬不足。而在客戶端只需要作簡單運算，就可以更新使用者目前瀏覽興趣與正在瀏覽網頁之大致內容，減少行動客戶端的計算。

3. 預取演算法

伺服器端以分群的結果爲依據，產生與該群組相關的存取模式樹。一旦使用者發出資料索求，伺服器端程式將使用者歸類爲某群組，並參考存取模式樹中目前網頁最可能接著被存取者，搜尋出符合的預取候選頁面（candidate pages），然後將候選頁面傳回給客戶端，作爲選擇預取頁面參考資料。

3.1. 存取模式樹產生演算法

我們自每個群組中產生存取模式樹，記錄使用者存取序列，以得到該群組使用者的群體存取行爲。具體而言，每個群組中的使用者之存取序列對應其存取模式樹，針對單一使用者產生的存取樹，節點間存取次數爲相對應存取序列出現次數。產生存取模式樹之演算法。在取得存取日誌檔的內容時，採用 window size 爲 3 的方式逐行讀取，即每次讀取鄰近三行的存取紀錄。以行 1 之頁面爲樹的根節點，如果行 1 頁面裡的超連結包含行 2 代表之頁面，將行 2 頁面設爲行 1 的子節點。同理，行 1 頁面裡超連結包含行 3 頁面時，行 3 頁面設爲行 1 的子節點。第三種情形則是行 2 頁面的超連結包含行 3 頁面時，行 2 頁面設爲行 1 的子節點。每次設定時，將根節點與子節點的存取次數更新。爲了求得群體行爲，我們將個別使用者之存取樹做合併（merge）的動作。這樣的動作將兩樹間相同的存取模式保留並更新存取次數，不同的存取模式則加入合併後之存取樹，值得注意的是，合併動作未必會產生單一存取樹，兩樹間完全沒有存取模式相同者即無法作合併，無法合併者，保留原樹，因此產生存取模式樹群（forest）。全部合併後的存取模式樹群，即爲群體存取行爲。圖四表示個別存取樹做合併的動作。



圖四　存取模式樹之合併

3.2 內容導向預取演算法

伺服器端預先將每個網頁內容以文件向量與數個關鍵字表示之，在使用者發出資料索求時，將網頁與其文件向量與關鍵字組回應予客戶端。客戶端則以當時網頁的文件向量與關鍵字組更新側寫。在此定義側寫格式爲伺服器端傳送之文件向量與關鍵字，側寫更新的方法，以文件向量而言，採用算術平均，對每個維度作更新，其計算式爲：

$$\left(x \times n + x'\right)/\left(n+1\right)$$

算式中的 x 爲某一維度當時的權重（weight），x'爲當時文件向量的權重，（n+1）爲更新的次數。關鍵字組則取出現頻率最高的數個關鍵字組。因此客戶端之使用者側寫，隨著瀏覽的內容不斷更新。就使用者側寫而言，文件向量表示使用者大致的興趣，而關鍵字組則爲相關興趣細部表現。

伺服器端暗示之產生，依照客戶端所傳之目前瀏覽頁面，從存取模式樹群中搜尋出與目前瀏覽頁面相同者，再從其子節點找出將來最有可能被存取的候選頁面，搜尋子節點依照給定之搜尋深度爲 threshold，大於搜尋深度的頁面不予加入暗示。而客戶端收到候選頁面的暗示後，將使用者側寫與候選頁面比較，與使用者目前之瀏覽興趣越相關越好。由此觀念出發，我們提出利益函數（profit function）以計算出最與使用者興趣配合的頁面，其計算式

爲：

profit = -(keyword match count) + distance between vectors.

這個利益函數求出之值，越小表示越適合預取。考慮行動環境裡頻寬限制，預取的頁面以固定 threshold 限制之。

內容導向演算法分爲伺服器端與客戶端，敘述如下：

1. 客戶端：
   - 對伺服器端發出資料索求。
   - 將目前瀏覽頁面之訊息以 HTTP POST method 傳給伺服器端。
   - 等待伺服器端暗示傳回。
   - 依照利益函數計算出最有利之候選頁面
   - 對最有利之候選頁面預取。

2. 伺服器端：
   - HTTP GET method 的資料索求傳回索求之頁面。
   - HTTP POST method 的資料索求依據使用者目前瀏覽頁面搜尋出候選頁面，傳回暗示予客戶端。

3.3 結構導向預取演算法

伺服器端暗示的產生，當發現使用者在存取結構導向的頁面（目前是 Powerpoint 檔轉換成 HTML 者）時，從內容導向轉而產生這類頁面的預取暗示。結構導向頁面的暗示產生方式，我們採取計算其傾向的方式計算將來最有可能被存取者。說明傾向的計算以前，先定義相關名詞：

- $NumOfNext_i$：表示在存取日誌檔中，使用者從目前頁面點選「Next」的次數。
- $NumOfPrevious_i$：表示在存取日誌檔中，使用者從目前頁面點選「Previous」的次數。

傾向的計算方式爲：

1. $tendency = NumOfNext_i / NumOf\,\mathrm{Pr}evious_i$ if $NumOf\,\mathrm{Pr}evious_i \neq 0$
2. $tendency = NumOf\,\mathrm{Pr}evious_i / NumOfNext_i$ if $NumOfNext_i \neq 0$
3. $tendency = 100$ if $NumOf\,\mathrm{Pr}evious_i = 0$
4. $tendency = -100$ if $NumOfNext_i = 0$

從上式可以看出當使用者點選「Next」的傾向最高爲 100，點選「Previous」的傾向最高值爲 -100，值的正負號代表下一頁或前一頁的傾向。伺服器端依照計算出之傾向，得出接下來可能存取的候選頁面，傳回給客戶端作爲暗示。客戶端由暗示直接預取候選頁面。

結構導向演算法同樣分爲伺服器端與客戶端，分述如下：

1. 客戶端：
   - 對伺服器端發出資料索求。
   - 將目前瀏覽頁面之訊息以 HTTP POST method 傳給伺服器端。
   - 等待伺服器端暗示傳回。
   - 對最有利之候選頁面預取。
2. 伺服器端：
   - HTTP GET method 的資料索求傳回索求之頁面。
   - HTTP POST method 的資料索求依據使用者目前瀏覽頁面計算其傾向，得出最有可能被存取者，傳回暗示予客戶端。

3.4 線上使用者歸類演算法

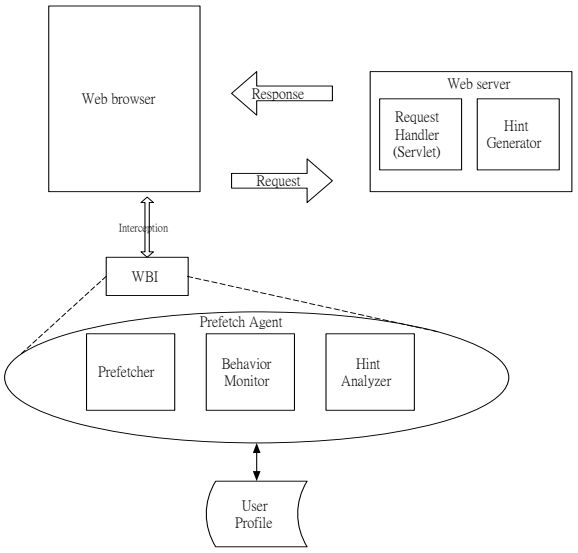
新的使用者因瀏覽興趣不同，可能產生與存取日誌檔相異之存取序列。然而在對來訪頻率最高的使用者們經由分群方式得出的數個群組，並且將這些群組的存取序列轉換成存取模式樹後，即可表現使用者的集體行爲，而非單一使用者行爲，因此嘗試將新使用者存取模式歸類至某一或某些群組，藉群體行爲預測新的使用者行爲。在稍後的章節中我們將以實驗證實我們的設想。

線上使用者歸類的演算法，需要將系統裡客戶端傳送訊息加入預取失誤次數，一旦大於最大預取失誤時，改變使用者的群組，藉這樣的方式調適使用者所在群組，茲說明如下：

1. 如果客戶端預取失誤未達最大失誤次數，且尚未歸類者，任意挑選某一群組對應之存取模式樹群，尋找其中包含目前使用者瀏覽之網頁，依此節點往下搜尋固定深度，找出可能被存取的其他網頁，產生暗示並回應予客戶端。
2. 如果客戶端預取失誤未達最大失誤次數，且已經被歸類者，從該群組繼續搜尋相對應之網頁，並產生暗示予客戶端。
3. 如果客戶端預取失誤已達最大失誤次數，表示使用者的存取模式不與此群組的使用者相同，則將其歸類至其他群組。
4. 重複上述三步驟。

4.系統架構

我們將以上概念實作成系統，由於 Web 伺服器端與客戶端爲分散式系統，因此系統採分散式架構。在不更改超本文傳輸協定（HTTP）的前提下，伺服器端處理資料索求與產生伺服器端暗示；客戶端則發出一般的資料索求，以及更新使用者的側寫，並發出預取的資料索求。圖五爲行動預取系統的架構：



圖五 預取系統架構

實作的預取系統在新的使用者發出資料索求時，以群體行爲作爲預取的依據，適度彌補舊有預取方式需要長時間統計資訊的缺點。系統傳送的控制訊息與最佳預取候選計算都力求精簡，有助運用於行動計算環境。而針對內容與特殊結構作預取，則增加系統彈性。

5.系統實作與效能評估

本章節敘述行動預取系統的實作，並且其是否合乎行動計算環境的需求作實驗並評估之。預取系統應該具備預取命中率高、節省使用者等待下載時間、不過度浪費網路頻寬等特性，評估時可以就這些特性作探討。

5.1 實驗環境與平台

系統在Windows平台下以Java語言與Java Developer ToolKit1.2.2 開發環境[15]發展預取程式。爲了能夠偵測瀏覽器的資料索求，使用 IBM 的 WBI 環境與 API[16]開發程式以攔截瀏覽器的資料索求。伺服器端方面，利用 Java Servlet[13]在 Apache Web 伺服器[11]端處理資料索求，客戶端則以 Java 程式語言搭配 WBI Developer Kit，撰寫程式以接收瀏覽器的資料索求與伺服器給瀏覽器的回應，並且更新使用者的側寫。

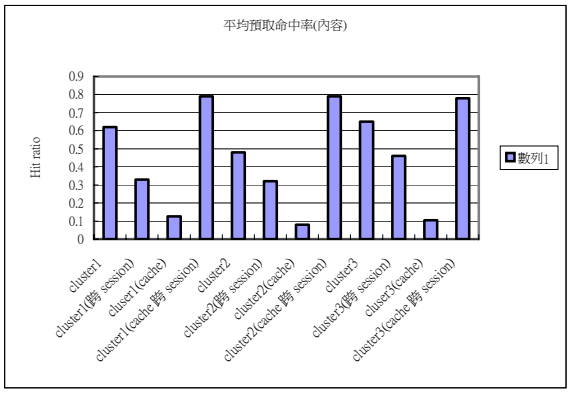
5.2 效能評估

系統的效能評估，分別對結構導向預取與內容導向預取做評估。結構導向預取採用真實存取日誌檔的前 2/3 做傾向計算，後 1/3 做真實的存取以測試之;內容導向演算法同樣採用存取日誌檔的前 2/3 建造存取模式樹，後 1/3 做真實的存取以測試之。

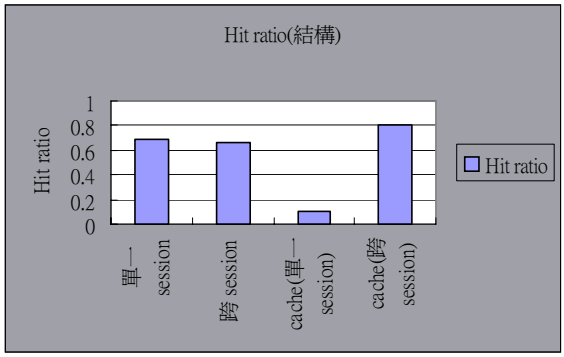
預取命中率

預取的命中率評估方面，從東華大學資工所網頁中取出一個月的存取日誌檔，分析出瀏覽最頻繁的前 20 位使用者，依照存取模式的相似度產生距離矩陣，依距離矩陣與 single link clustering 的分群方法將此 20 位使用者分群，最後分成 3 個群組。針對這 3 個群組，將存取日誌的前 2/3 作爲伺服器端建立存取模式樹的依據，後 1/3 作爲使用者點選網頁的模擬。計算各群組中使用者的平均預取命中率（就 HTML 而言）。實驗中預取時將瀏覽器的快取關閉，客戶端的預取空間未做替換（replacement）的動作。針對每個群組中每位使用者的一個存取 session，測量其平均預取命中率。圖六顯示預取命中率可以達到 4 成以上。特別值得注意的是，群組 2 的使用者每次點選不同網頁的傾向較高，且與群組 1 的存取模式有小部分雷同，造成歸類演算法歸類失準，因此導致較低的預取命中率；群組 3 的使用者的存取模式則爲點選曾瀏覽的頁面，預取命中率較高；接著採用相同的樣本，但是跨 session 存取，而非只取一段 session 測量，結果造成預取率降低。推究其原因，是建立存取模式樹時考慮 session 的緣故，造成跨 session 的行爲較不易被預測。快取方面，快取中的資料比較採用 once per session 的機制，可以看出單一 session 中的存取多只一次，因此快取命中很低。在跨 session 的情況，命中率提高很多，且傳輸資料量節省較多。

另外我們也以真實的存取日誌對結構導向的預取做命中率評估。從真實日誌中取出 10 人爲實驗的樣本，計算其平均預取命中率，從選擇單一 session 與跨 session 兩方面做比較。就命中率而言，快取在存取檔案不同時無法發揮效用。



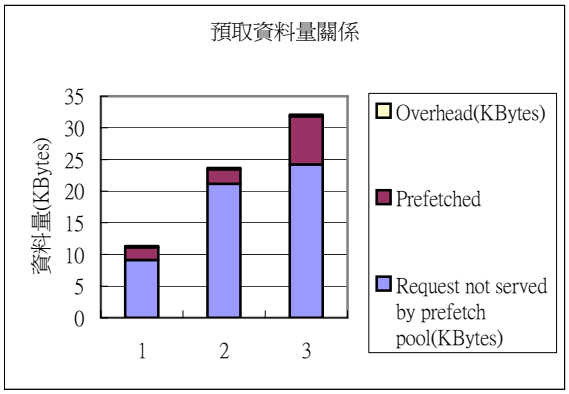
圖六 平均預取命中率（內容導向）



圖七 結構預取命中率

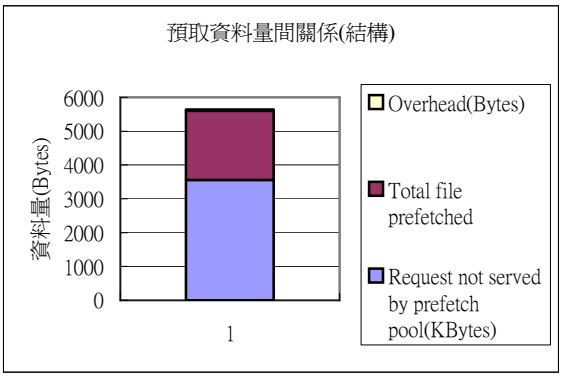
預取資料量之關係

在三組使用者中，計算每次存取平均資料量，觀察各類資料數量上關係。由圖 5-6 可以觀察出，預取系統爲增高命中率而增加的 Hint 傳輸與其他控制訊息佔系統運作時整體資料量中非常小的部分。跨 session 的預取命中率約 30%，所以有較大量的資料無法從 local 得到。如圖 5-7。



圖八 平均預取資料量比較（內容導向）

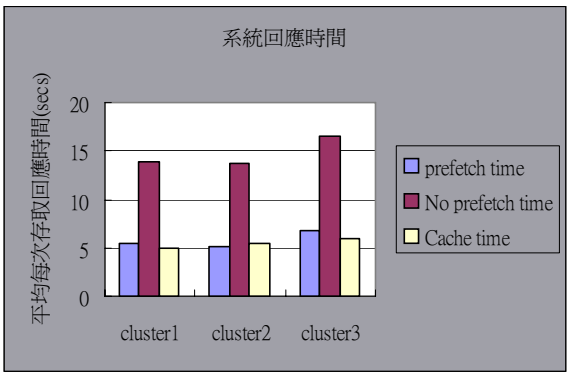
結構導向演算法則有預取部分在比例上較高的現象，因爲此類網頁的預取命中率較高，觀察後可以得知（圖九）。



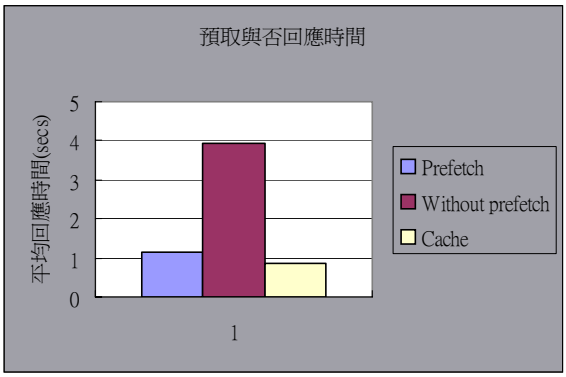
圖九 平均預取資料量比較（結構導向）

預取與否和系統回應時間

本實驗以跨 session 存取爲對象，計算預取與否和單獨採用快取的系統回應時間（圖十、圖十一）。預取顯然比不採用預取節省更多的時間，無論採用何種預取方法皆然。快取的回應時間較預取互有上下，從傳輸資料量的節省也可觀察出來。



圖十 系統回應時間比較（內容導向）



圖十一 平均預取時間比較（結構導向）

5. 結論與展望

我們將研究時的觀察與設想，轉換成實際的行動預取系統，期望能夠針對行動環境裡頻寬與計算能力的限制，作全面性考量，找出合適的演算法並予以實作，最後達到減少使用者等待時間，支援斷訊操作，兼且不增加過高網路流量。

在本架構中，利用伺服器端有餘裕的計算能力，先行分析每個網頁的關鍵字組與其文件向量，在客戶端發出資料索求時，能夠附加這樣的訊息於其上，供客戶端更新其使用者側寫。而客戶端傳送目前瀏覽網頁與預取失敗次數等訊息給伺服器端，伺服器端依此訊息搜尋相對應之存取模式樹，接下來最可能存取的頁面之搜尋結果當作暗示，並回傳給客戶端，客戶端計算暗示與目前使用者瀏覽興趣的相似度，取出最有利的預取頁並預取之。

REFERENCES

[1] Virgilio Almeida, Azer Bestavros, Mark Crovella and Adriana de Oliveira, Characterizing Reference Locality in the WWW, In Proceedings of IEEE PDIS'96: The International Conference in Parallrl and Distributed Information Systems, Miami Beach, Florida, USA, December 1996.

[2] M. Berndtsson, B. Lings, Logical Events and ECA Rules. HS-IDA-TR-95-004, Department of Computer Science, University of Skovde, 1995.

[3] S. Chakravarthy, V. Krishnaprasad, Anatomy of a Composite Event Detector, UF-CIS-TR-93-039, University of Florida, FL. 1993

[4] S. Chakravarthy, V. Krishnaprasad, ECA Rule Integration into an OODBMS：Architecture and Implementation, UF-CIS-TR-94-023，University of Florida, FL. 1994

[5] Y.T.Chang, An Active Data Replication Scheme for Mobile Information Management, Department of Computer Science and Information Engineering NDHU, July 1998.

[6] Carlos R. Cunha and Carlos F. B. Jaccoud, Determining WWW User's Next Access and Its Application to Pre-fetching, Symposium on Computers and Communication'97, Alexandria, Egypt, 1-3 July 1997

[7] S.T. Hsu, "A Preferenced-Oriented Approach to Intranet Prefetching" Department of Computer Science and Information Engineering, NDHU, July 1998

[8] Tomasz Imielinski and B. R. Badrinath, Mobile wireless computing: challenges in data management; Commun. ACM 37, 10 , Oct. 1994, Pages 18 – 28

[9] Lindo Tauscher and Saul Greenberg, How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems, International Journal of Human Computer Studies, Special issue on World Wide Web Usability, 47(1), p.97-138, Academic Press, 1997.

[10] AltaVista Company, http://www.altavista.com

[11] Apache HTTP server project, http://www.apache.org/httpd.html

[12] Goto Software, http://www.webearly.com/us/description.htm.

[13] JAVA™ SERVLET API The Power Behind the Server, http://java.sun.com/products/servlet/index.html

[14] Openfind Information Technology INC., http://www.openfind.com.tw

[15] The Source for Java™ Technology, http://java.sun.com/

[16] WBI Development Kit for Java, http://www.alphaworks.ibm.com/tech/wbidk