淡江大學電機工程學系

專題實驗報告

Big Data 實作

暨分散式 Google Search 優化系統

指導教授　：　莊博任 教授

學　生　：　涂耘昇

陳相儒

中華民國一百零二年九月

目錄

第一章、摘要

本專題名為：Human's Valuable Search Results，以下簡稱 HVSR 系統，主要是透過蒐集使用者瀏覽網頁的行為模式，進行雲端運算分析，來改變 Google 現有的連結排序，給使用者新的連結排序結果。試圖縮短使用者在搜尋時，所耗費大量的時間。

HVSR 透過 Chrome 的 plug-in 來蒐集使用者搜尋時，點選連結後瀏覽的行為資訊(關鍵字、連結、瀏覽時間、分頁切換次數等...)，並把這些資訊上傳到 HVSR 的 Server 儲存，匯集大量來自不同使用者的行為資訊。當使用者在關鍵字搜尋時，就可以透過這些匯集而來的大量資訊，分析出此關鍵字與連結之間的相對價值，最後按照價值高低依序排列在圖塊上，呈現出最符合大眾需求的結果。

HVSR 後端分別使用分散式與非分散式兩套系統，進而分析比較，在 Big data 的情況，使用分散式與非分散式之間的差異性。分散式系統是使用 Hadoop 來實現，包含 HDFS 存取資料與 MapReduce 運算;非分散式系統則採用 MySQL 資料庫直接透過 Server 端運算。

HVSR 這個技術是分析使用者提供的行為模式數據而綜合得出的結果，所以，我們能明白一件事，當使用者人數越龐大，使用者行為的模式就會越齊全，那麼該分析結果就會越精準，這是合乎情理的；不論使用者是要搜尋專業資訊或是景點美食，只要使用這套軟體，就能立刻讓使用者找到"最佳解答"。且 HVSR 可以透過眾多使用者搜尋的關鍵字，分析當

今產業發展的趨勢概況與未來發展的方向。

第二章、前言

2.1、研究動機

常常在搜尋上花上大量時間而感到困擾或是常常在連結中點進去後才發現網頁內容無法解決自己的需求，最終解答卻在第三頁、第五頁之後出現，這會造成使用者的不便，而且極度耗費時間，常常經歷諸如此類的事情，就能合理判斷出排序高的連結並不一定會是問題的解答，這也常常造成使用者在搜尋上的困擾，促使使用者耗費極大的時間在搜尋結果上，是非常沒有效率的。

在專業問題上，研究學術論文、大學研究報告、碩博士論文、發表專利文章、或是研究方面的問題(工具、平台、環境數據等...)，往往搜尋到的東西都是排山倒海而來的數據與資訊，使用者要從中尋找出能夠解決問題點的作法、流程、參數，甚至還得在不熟悉的語言中探究其有效作法，大量的資訊在腦袋中彙集與整理，往往會花費許多的精力與時間，拉長了研究的完成時間，為了讓在專業領域的使用者族群能快速的完成其研究與實驗，因此減輕查找大量資訊以及縮短搜尋時間，不僅是使用者所期盼抑是我們所努力之目標。

雖然說網路的資源是共享的而且浩瀚無垠，方便了許多使用者獲取所需的資訊，但是誰不希望能更快速搜尋到正確解答呢？如果能在按下 GOOGLE SEARCH 的那瞬間，額外提供使用者關鍵字的"最佳答案"之連結，就能縮短搜尋時間又能提高解答的正確性。因此我們希望透過 HVSR

這套 Chrome Plug-in 的開發，能夠解決使用者在搜尋關鍵字的時候，可以更快速更有效率的得到解答，以改善在搜尋時的困擾。

2.2、研究問題

從 Google 搜尋引擎對搜尋結果排名的方式，我們將會發現，Google 不外乎是利用內文匹配系統，以及 PageRank 來產生連結的排名，或是利用 SEO 專員來提升排名。但是這都只是系統的技術使然，或專業人員為網站所精心設計的最佳呈現方式，以提升 Google 搜尋結果的排名高低，沒有辦法從使用者的瀏覽時間、下關鍵字、操作行為等(以下統稱行為模式)去調整搜尋結果的順序，這些資訊是 web sever 無法取得的，所以需要 plug-in 來協助，而目前看起來也沒有 plug-in 能做到嘗試取得類似的資料。因此我們要透過 plug-in 的協助，取得使用者瀏覽網頁的行為模式，並從這些訊息中找出彼此之間的關聯，最後分析出真正眾多使用者所關注的內容，給予他們一個新的搜尋結果排序，並讓使用者更容易找到他們所需要的連結內容，且提升搜尋的效率。

第三章、系統架構

3.1、概念

本專題所建立的架構為圖 3.0.1 所示，此圖為基本架構圖。本專題的架構可分成如圖 3.0.1 所示的三個區域，主要是根據不同層的功能來區分，因此每一次各自獨立。

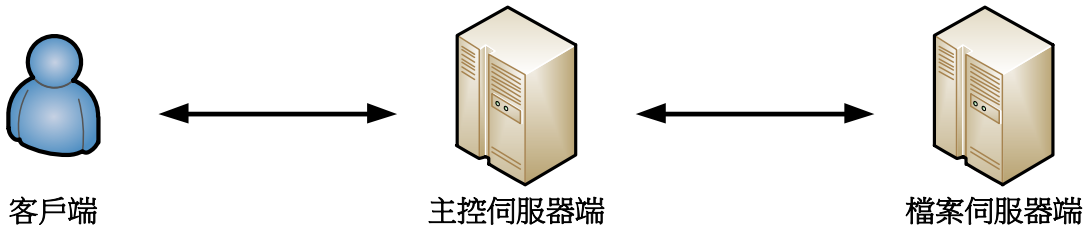


圖 3.0.1、系統基本架構圖

客戶端只需要透過瀏覽器查詢關鍵字，就會透過瀏覽器的外掛程式連線至主控伺服器端即可服務使用者。當使用者輸入於查詢的關鍵字後，主控伺服器端會收到傳自瀏覽器上外掛程式傳過來的資訊，並做出應對動作。同時透過外掛程式記錄客戶端進入連結後的瀏覽行為資訊，並回傳至主控伺服器端，且將這些資訊存入檔案伺服器端。使用的瀏覽器為 Chrome，並搭配 Chrome extension 外掛程式，可以透過 Chrome extension 的設計開發，蒐集使用者瀏覽行為資訊與呈現排序結果給使用者點選，以及與主控伺服器端溝通。

主控伺服器端服務了所有連線進來的使用者，不同使用者有著不同的服務需求，主控伺服器也溝通了檔案伺服器端與客戶端的重要橋梁，歸納不同的服務請求，並回應請求，以及下指令至檔案伺服器的功能與運算功能。

檔案伺服器端分別使用了兩套系統 MySQL 資料庫與 Hadoop 平台，使用兩套系統的用意是因為可以藉由非分散式系統與分散式系統的差異，比較出運算、存取速度，並透過比較分析出效能較高的系統。

因為分成的三個區域，當發生錯誤時可以根據不同的錯誤取向，很迅速得找出出錯誤點做修正。或是有新的運算單元要加入時，可以很快速的延伸擴充。

3.2、主要細節架構

本專題之系統設計取向主要是根據不同使用者輸入不同的關鍵字產生不同的行程來服務。如下圖 3.1.1 所示，為單一使用者流程圖。

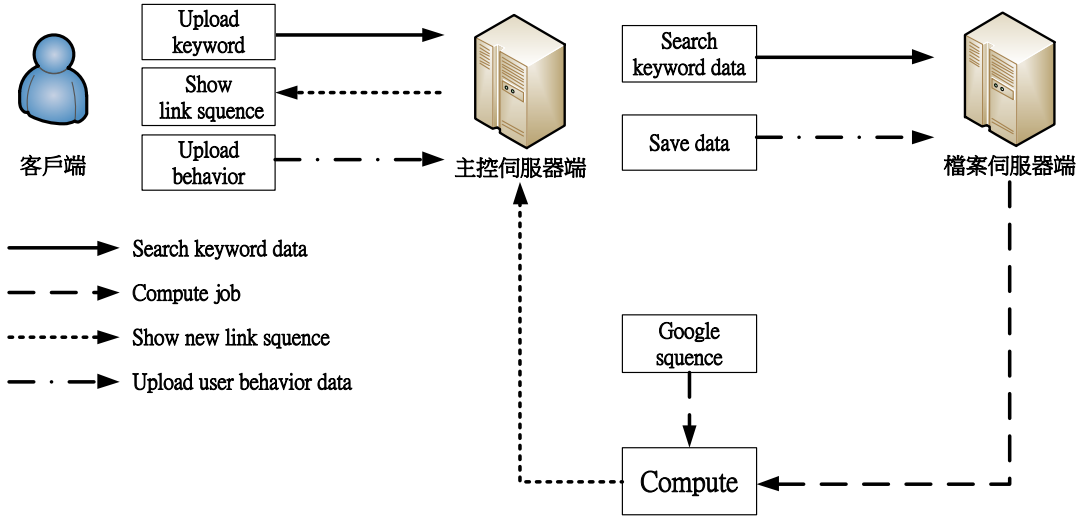


圖 3.1.1、主要細節架構

根據不同使用者輸入不同關鍵字來執行處理，如圖使用者輸入某關鍵字，會將關鍵字透過本專題在瀏覽器(Chrome)設計的外掛程式(Chrome extension)上傳至主控伺服端。當一有主控伺服端一接收到使用者的連線請求，便將關鍵字讀入並下指令至檔案伺服端讀取關鍵字的資料，如圖 Search keyword data 路徑。

圖 Computer job 路徑，是將從檔案伺服端讀取的關鍵字資料與 Google 原有排序合併做運算，產生出新的排序結果。

產生出新的排序結果後會透過主控伺服端回傳自客戶端的外掛程式，如圖 Show new link squence 路徑，並透過外掛程式呈現在搜尋頁面中，提供給使用者新的連結排序，並且可以點選進入連結中。

使用者搜尋關鍵字並點選進入連結後，客戶端的外掛程式會蒐集使用者在該連結的瀏覽行為直到關閉連結後，上傳至主控伺服端並下指令將瀏覽行為資訊存至檔案伺服端，如圖 Upload user behavior data 路徑。

3.3、Client 架構

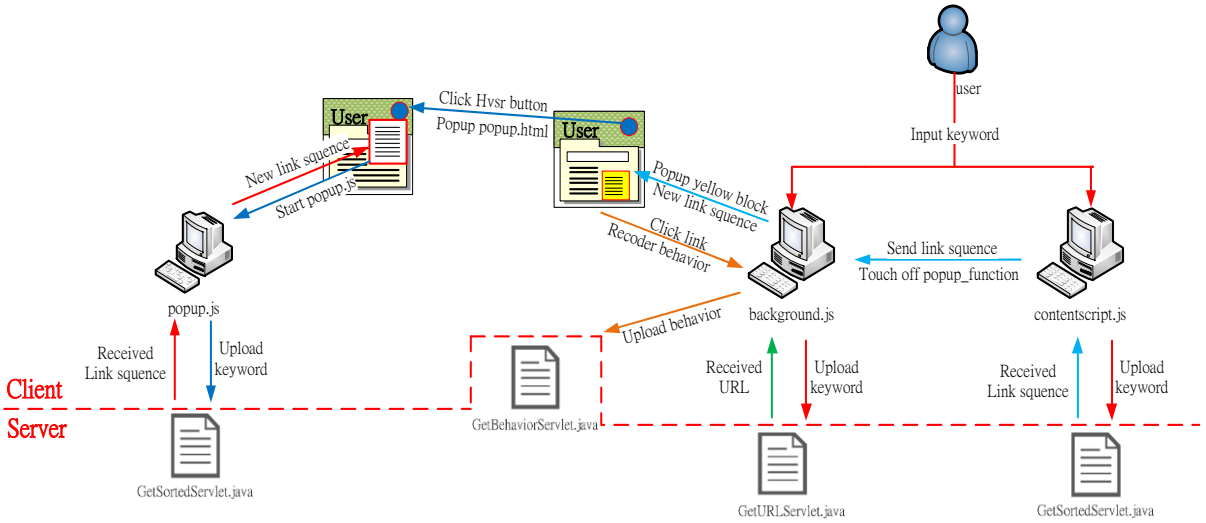


圖 3.2.1、Client 架構

使用者在 google 輸入 keyword 後，HVSR 開始同步執行 contentscript.js 與 background.js。

contentscript.js 工作是把使用者輸入 keyword 的 url 傳給 server (GetSortedServlet.java)，撈取新的連結排序並傳給 background.js 觸發 popup_function，彈跳出黃色圖塊給予使用者新的連結排序可以點選進入網站。

background.js 將使用者輸入的 keyword 傳給 server(GetUrlServlet.java)，撈取 keyword 的每一個連結 url。並等待，contentscript.js 觸發，跳出黃色圖塊，給予使用者點選新的連結排序。當使用者點選連結時，比對 server 撈取的連結 url，判斷是否為 keyword 對應的連結。是，開始記錄使用者在該連結的行為資訊(ex:tab 切換次數、實際瀏覽時間、網頁存在時間…等)，當使用者跳出連結，記錄結束並回傳給 server(GetBehaviorServlet.java)。

當使用者按下 HVSR 的 button 時，會跳出 popup.htlm 頁面並執行 popup.js，popup.js 會把使用者輸入 keyword 的 url 傳給 server(GetSortedServlet.java)，撈取新的連結排序呈現在 popup.html 上，給予使用者點選進入網站。

3.4、Server 架構(一般系統)

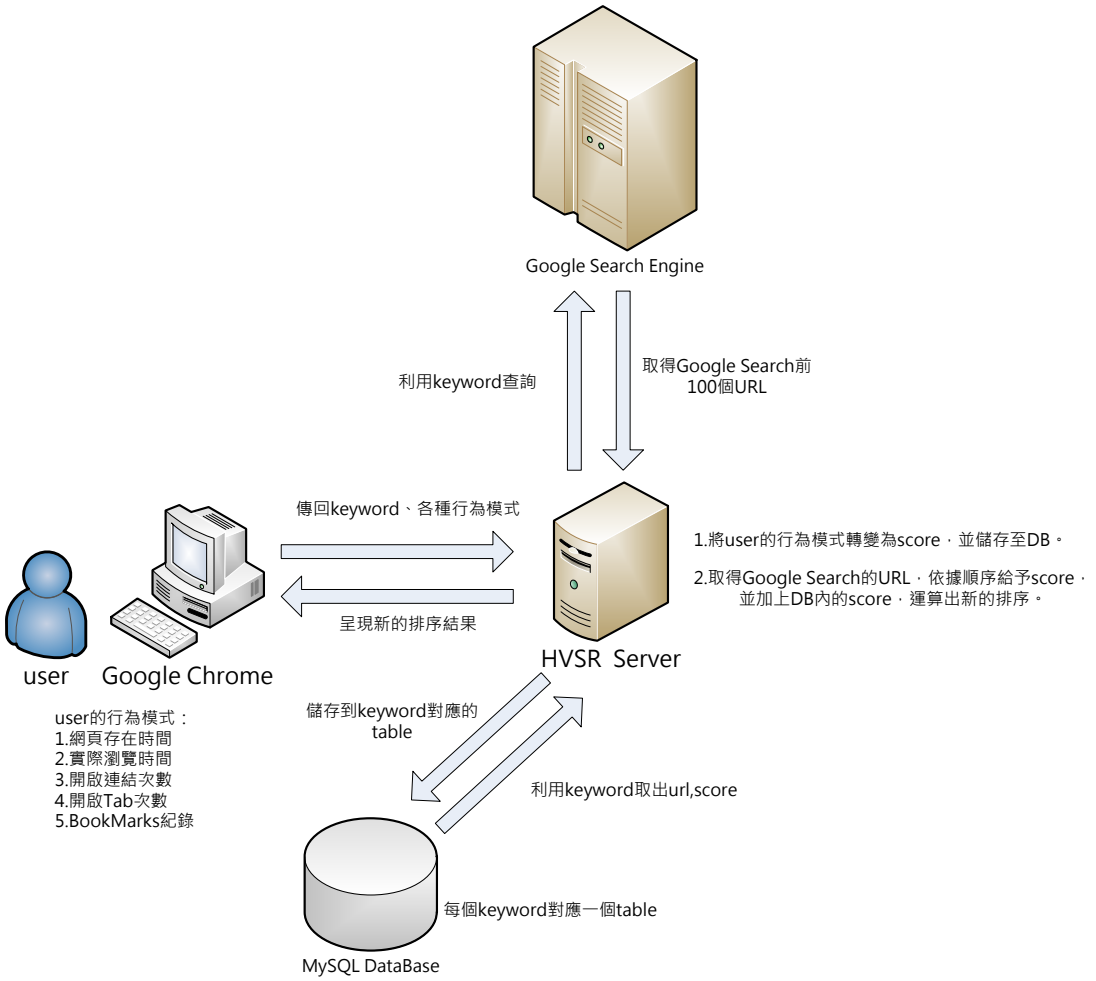


圖 3.3.1、Server 一般系統架構

一般系統採用 Tomcat server 搭配 MySQL database。當使用者使用 chrome 搜尋資料時，回傳資訊關鍵字，server 接收後，進一步去抓取 google 本身的排序以及資料庫內的資訊，並搭配演算法結算出新的排序回傳給 client。

使用者搜尋資料的同時，client 端程式會回傳相關的行為模式，

server 收到後存入 database。

3.5、Server 架構(分散式系統)

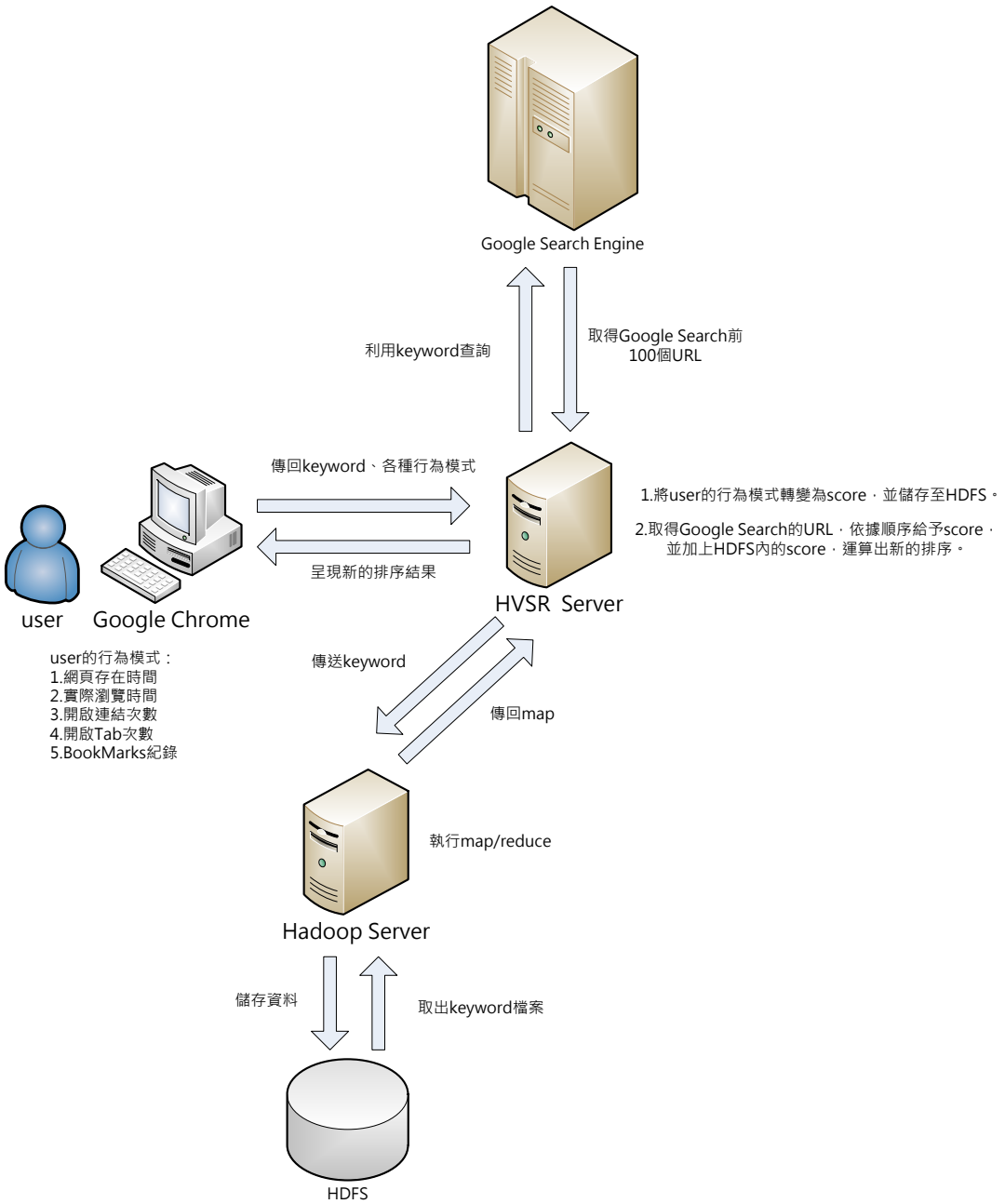


圖 3.3.2、Server 分散式系統架構

與一般系統差別在於將 server 的運算工作移至 Hadoop server 做分散式運算，並且將資料儲存在 HDFS 檔案系統。運算時將 HDFS 內容取出，做 map/reduce 叢集運算以提升效率。

3.6、資料庫設計

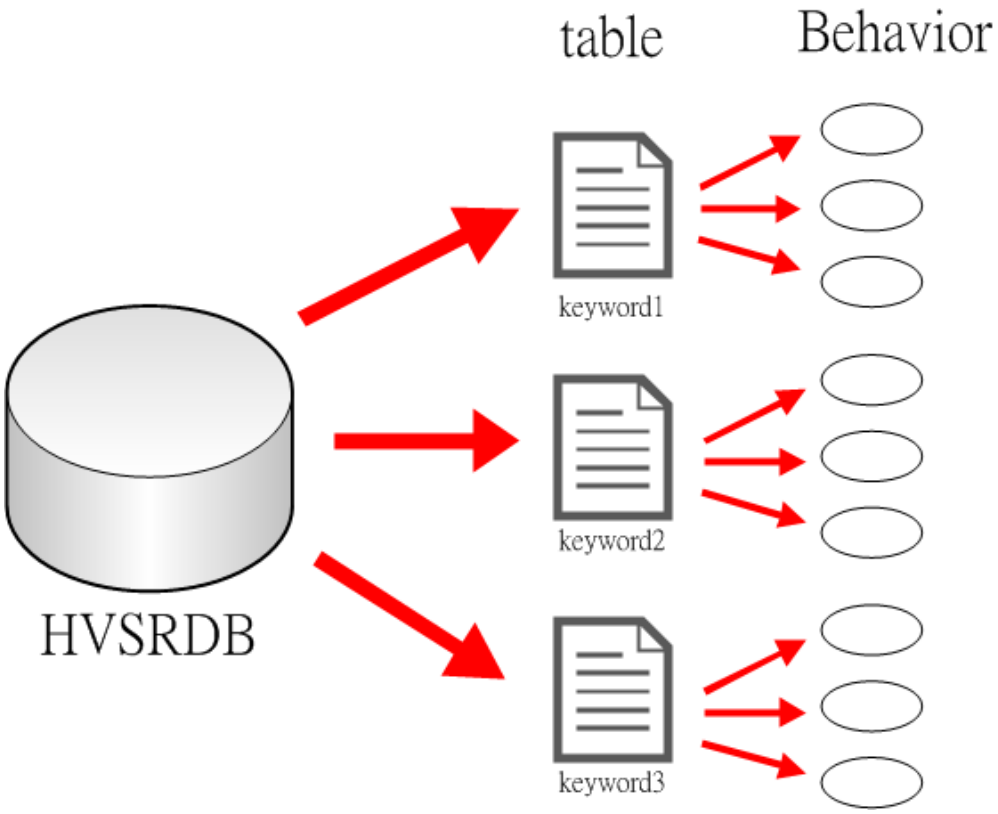


圖 3.4、資料庫設計

HVSRDB 將不同的關鍵字設定成 table，而每個 table 都存放著數個 Behavior 的物件。當 client 送來 Behavior 物件時，會存入其對應的 table 中。如此一來，將各個使用者搜尋過的記錄用有條理的方法儲存起來，也讓未來 server 從資料庫拿資料時更有效率、更快速。

3.6.1、DAO 設計

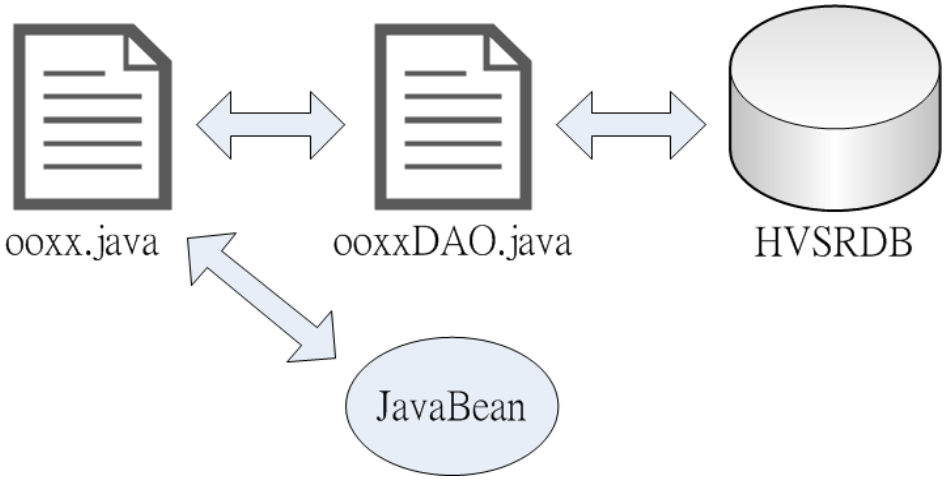


圖 3.5、DAO 設計

DAO 設計的目的是要讓，程式執行部分與儲存資料部分做個區別，各司其職。換句話說，主程式只要跟 DAO.java 和 JavaBean 溝通即可，剩下的資料庫讀取寫入則由 DAO 和 MySQL 協調。如此一來，提升程式的閱讀性以及維護性。

3.7、程式架構

3.7.1、架構圖

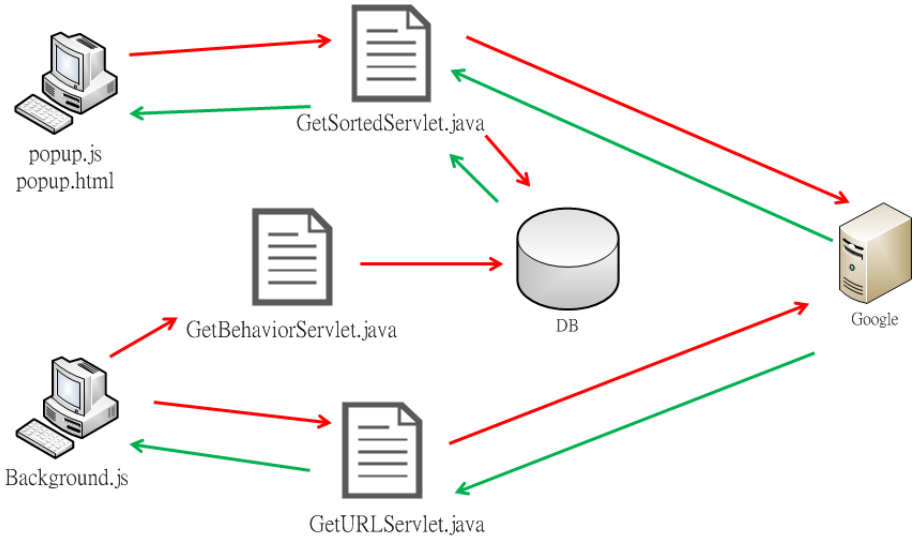


圖 3.6.1、程式架構

Client 程式為 google chrome extension，server 程式則是在 Tomcat 執行。

3.7.2、提供 client 蒐集資訊

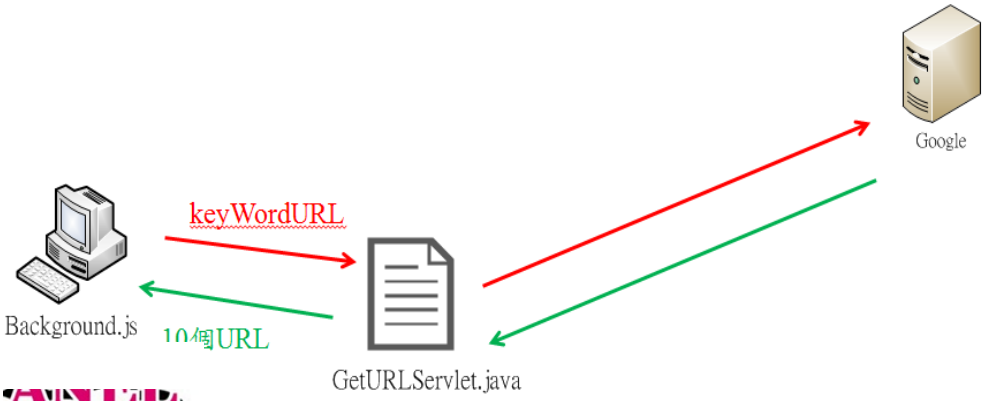


圖 3.6.2、蒐集資訊架構

每當使用者上網輸入關鍵字時，回傳該關鍵字，接著取得 google chrome 的查詢結果，再回傳 client，讓 client 知道使用者瀏覽那些網址時需要記錄其行為模式。

3.7.3、收集使用者行為模式

圖 3.6.3、收集行為模式架構

當使用者搜尋某特定網頁，再關閉該網頁的同時，client 程式會回傳行為模式，server 接收後，儲存置資料庫。

3.7.4、運算結果並呈現

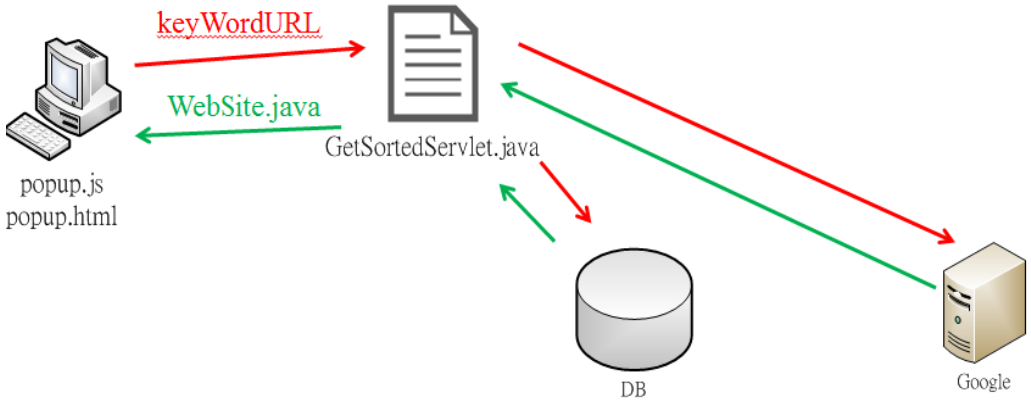


圖 3.6.4、結果呈現架構

當使用者開啟 google 搜尋，輸入關鍵字的同時，回傳關鍵字給 server，server 利用關鍵字從資料庫取出相關資料，並抓取 google 的搜尋結果，經過 HVSR 演算法後，呈現出新的排序。

第四章、程式說明

4.1、Chrome 收集使用者行為資訊流程介紹

4.1.1、background.js 流程

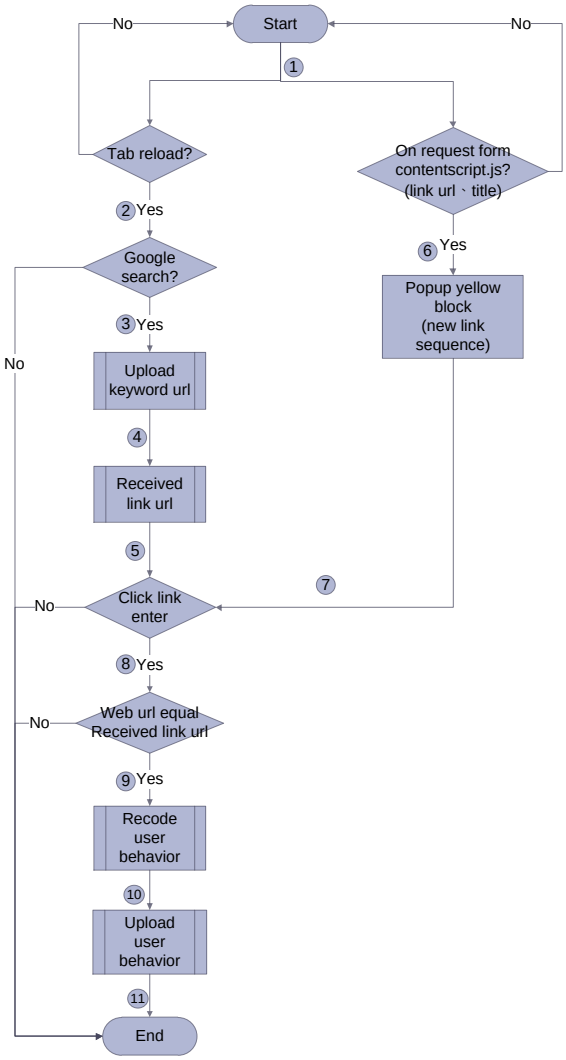


圖4.1.1、background.js流程圖

background.js流程圖說明:

1. 開始執行時，有2個觸發事件等著被執行

    1.1觸發事件:網頁更新

    1.2觸發事件:contentscript.js請求，並回傳新的排序連結

2. 當1.1事件被觸發後，判斷是否在google的搜尋頁面搜尋關鍵字?

    是，前往 3；否跳出(end)

3. 確認是在google的搜尋頁面後，上傳搜尋時的網址(keyword url)

4. 上傳網址到server後，接收server回傳，搜尋關鍵字結果的連結網址

    (link url)

5. 得到連結網址網址後，等待使用者點選進入連結

6. 當1.2事件被觸發後，瀏覽器右下角跳出黃色方塊，內容為新的排序連結，提供使用者點選並進入頁面

7. 跳出黃色方塊後，等待使用者點選進入連結

8. 使用者點選進入連結後，比對網址是否為搜尋關鍵字結果的連結網址

(link url)?

是，前往 9；否，跳出(end)

9. 比對網址正確後，開始記錄使用者在該連結網頁的瀏覽行為

10.紀錄完成使用者行為後，並將使用者行為紀錄上傳至server

11.上傳完成後end，等待下一次reload

4.1.2、收集使用者行為資訊(recodeUserBehavior)流程

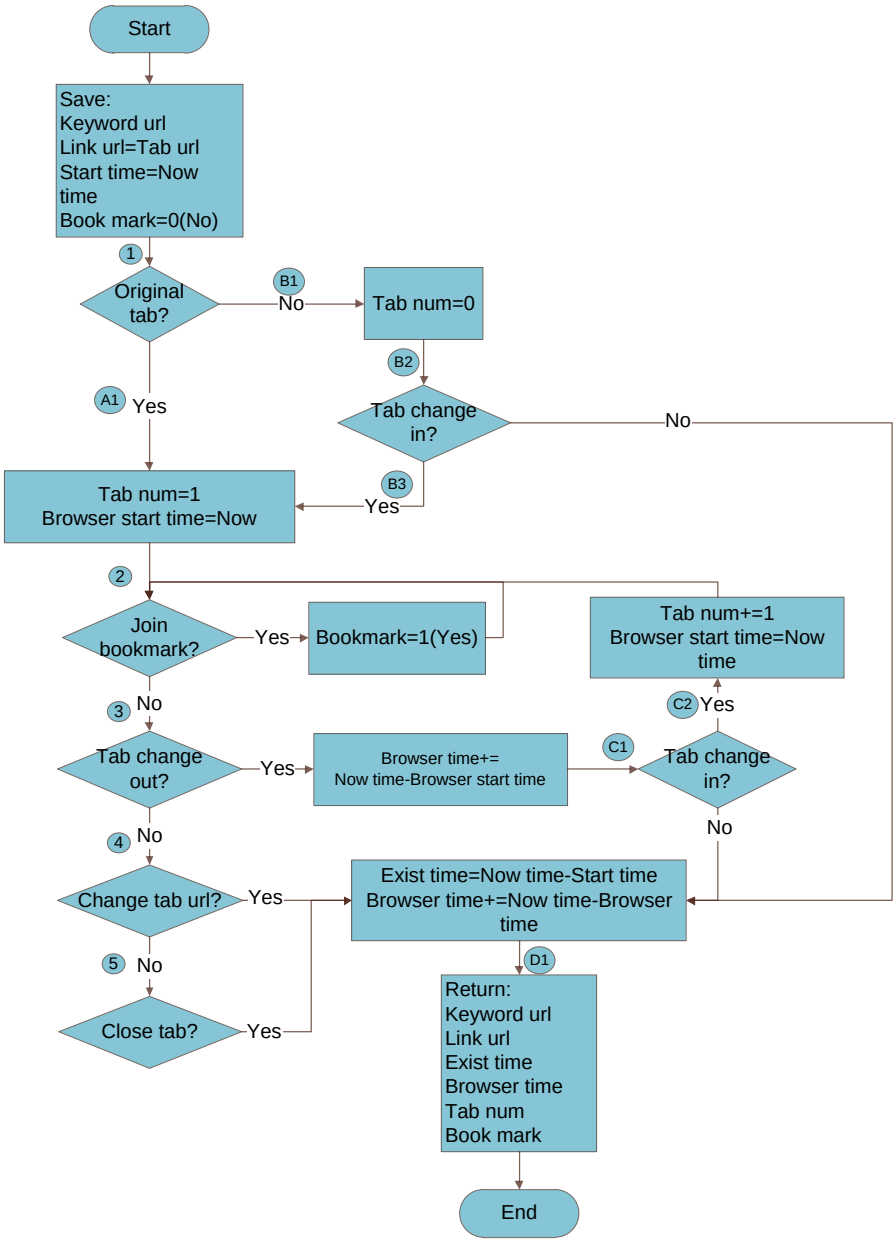


圖 4.1.2、收集使用者行為資訊流程圖

收集使用者行為資訊(recodeUserBehavior)流程圖說明:

1.判斷是原tab進入連結頁面，還是新增tab連結

原tab前往A1；新增tab前往B1

A1. 開始記錄:tab數量=1；瀏覽起始時間=現在時間

B1. 預設tab數量=0

B2. 判斷是否切換tab到此連結?

是，前往B3;

否，紀錄: 網頁存在時間=當下時間-網頁起始時間、瀏覽時間=現在時間-瀏覽起始時間，並累加瀏覽時間

B3. 開始記錄:tab數量=1；瀏覽起始時間=現在時間

2. 事件觸發:是否加入書籤? 是，bookmark=1(yes)，回到2

3. 事件觸發:是否跳出當下tab?

是，瀏覽時間=現在時間-瀏覽起始時間，並累加瀏覽時間

C1. 判斷是否跳回tab?

是，前往C2，累加tab數量、瀏覽起始時間=現在時間，回到2

否，紀錄:網頁存在時間=當下時間-網頁起始時間、瀏覽時間=現在

時間-瀏覽起始時間，並累加瀏覽時間

4. 事件觸發:是否改變當下tab的url?

是，紀錄:網頁存在時間=當下時間-網頁起始時間、瀏覽時間=現在時間-瀏覽起始時間，並累加瀏覽時間

5. 事件觸發:是否關閉當下tab?

是，紀錄:網頁存在時間=當下時間-網頁起始時間、瀏覽時間=現在時間-瀏覽起始時間，並累加瀏覽時間

D1. 回傳使用者資訊:關鍵字url、該連結url、網頁存在時間、瀏覽時間、tab切換次數、是否加入書籤

4.1.3、contentscript.js 流程

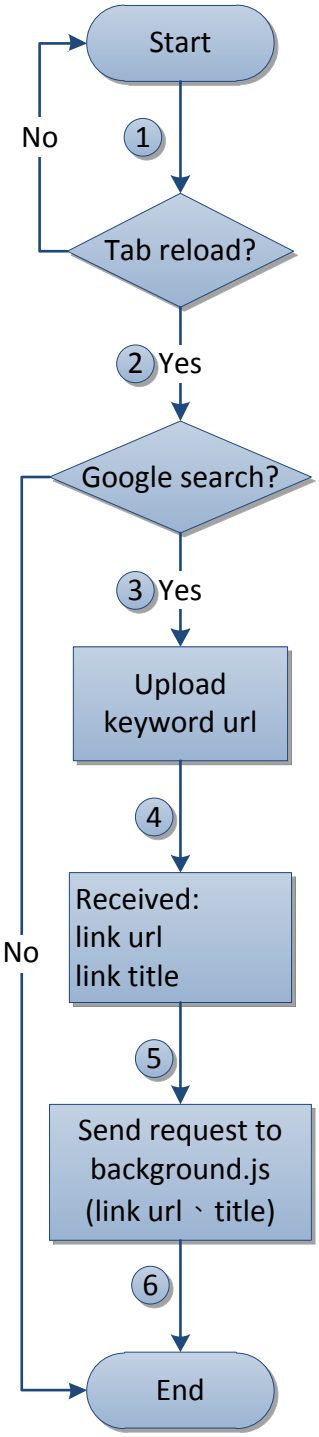


圖4.1.3、contentscript.js流程圖

contentscript.js流程圖說明:

1. 一開始先判斷是否更新網頁? 是前往2；否等待更新

2. 判斷是否在google的搜尋頁面搜尋關鍵字?

   是前往 3；否跳出(end)

3. 確認是在google的搜尋頁面後，上傳搜尋時的網址(keyword url)

4. 上傳網址到server後，接收server回傳，搜尋關鍵字結果的

   連結網址(link url)、連結標題(link title)，並有新的排序結果

5. 得到新的排序結果後，連同排序結果打包發出請求觸發至

   background.js並傳送

6. 發出請求觸發並傳送完成後end，等待下一次reload

4.1.4、popup.js 流程

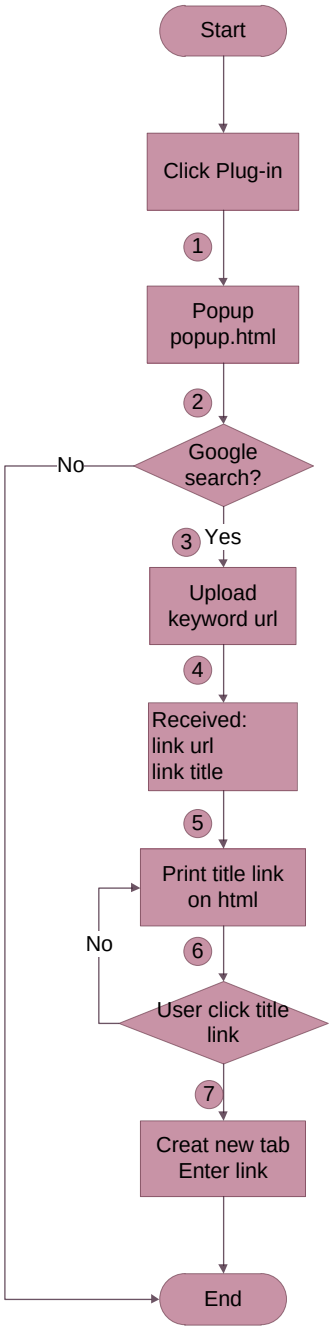


圖5.1.4、popup.js流程圖

popup.js流程圖說明:

1. 當使用者按下Plug-in按鈕後，跳出popup.html視窗

2. 判斷是否在google的搜尋頁面搜尋關鍵字?

   是前往 3；否跳出(end)

3. 確認是在google的搜尋頁面後，上傳搜尋時的網址(keyword url)

4. 上傳網址到server後，接收server回傳，搜尋關鍵字結果的

   連結網址(link url)、連結標題(link title)，並有新的排序結果

5. 得到新的排序結果後，嵌入popup.html中，內容為新的排序連結，

   提供使用者點選

6. 等待使用者點選popup.html的連結

7. 當使用者點選popup.html後，新增頁面進入該連結網址

8. 進入網址後end

4.2、前端 UI

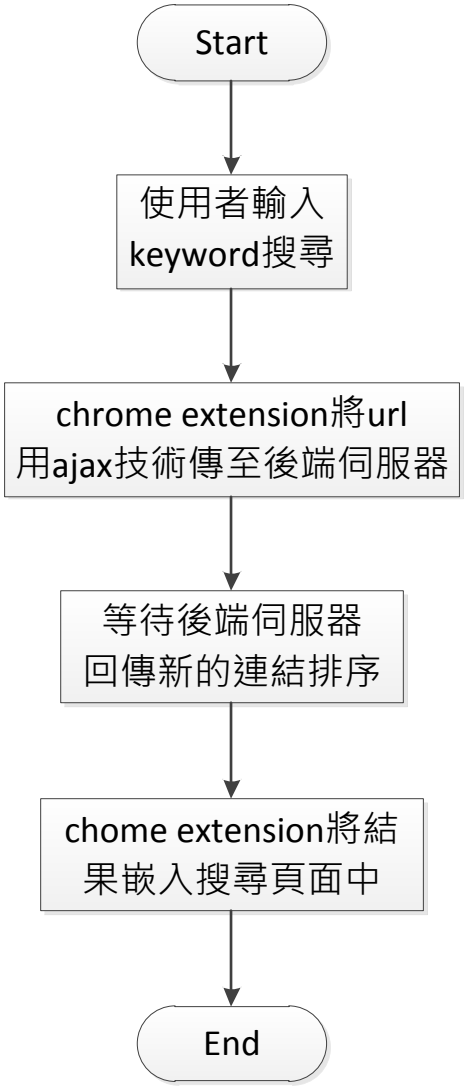


圖5.2.1、前端UI流程圖

當使用者在google搜尋關鍵字時，透過本專題開發的chrome extension將搜尋關鍵字的url使用ajax技術傳送至伺服器後端，並等待接收新的連結

排序。當chrome extension接收到從後端伺服器回傳的新的連結排序後，會透過chrome extension的Content Script技術，將結果嵌入在搜尋頁面中，並提供使用者點選連結進入。

4.3、Server 處理流程

4.3.1、收集資訊(GetURLServlet.java)流程

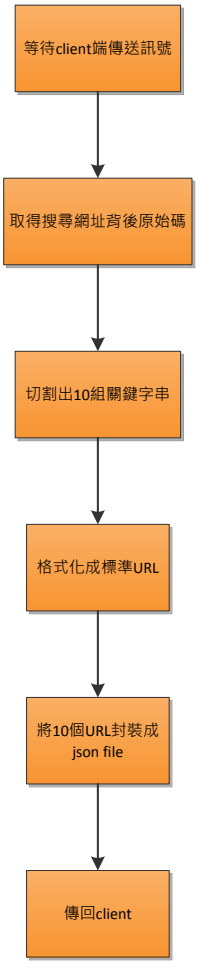


圖 4.3.1、GetURLServlet 流程圖

GetURLServlet.java 主要功能是，告訴 client 端的瀏覽器，當使用者瀏覽哪些網頁需要記錄其行為模式。一開始 Tomcat server 啟動時，等待 client 傳來搜尋結果的網址，取得後先進行進行解碼的動作，瀏覽器預設的編碼是 UTF-8，因此收到資料的同時需要解碼才能還原成最初的樣貌。圖 4.3.1

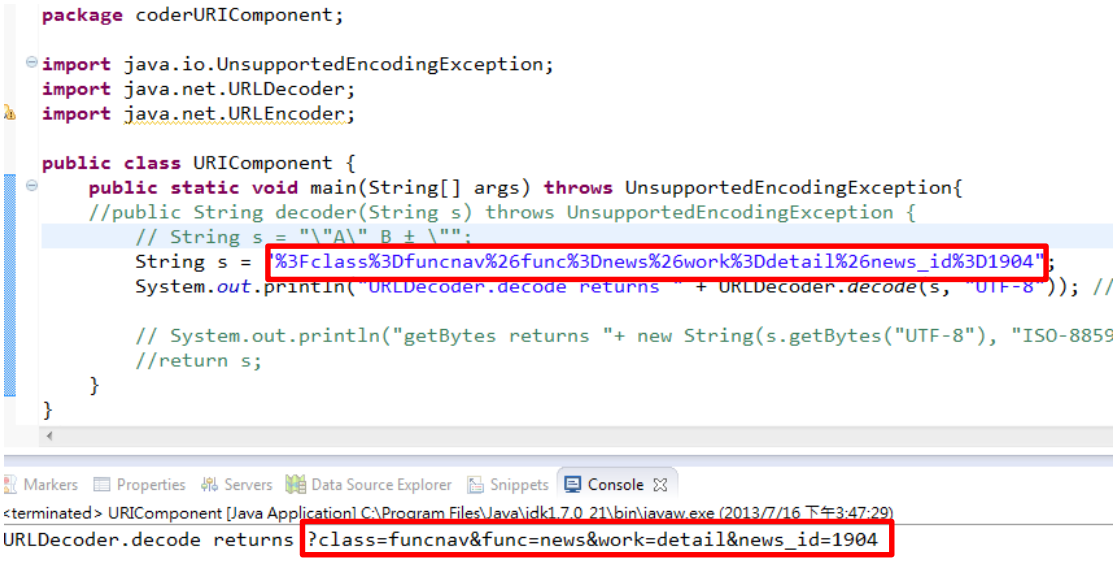


圖 4.3.1、將 URL 解碼

同樣是 google search，在上方的網址列輸入關鍵字，如圖 4.3.2，或是在一般搜尋列網址，如圖 4.3.3，呈現的網址不盡相同，如圖 4.3.4 與圖 4.3.5。為了統一網址的規格，執行，將它轉換成 https://www.google.com/search?q=ooxx…的形式，以利於未來使用。

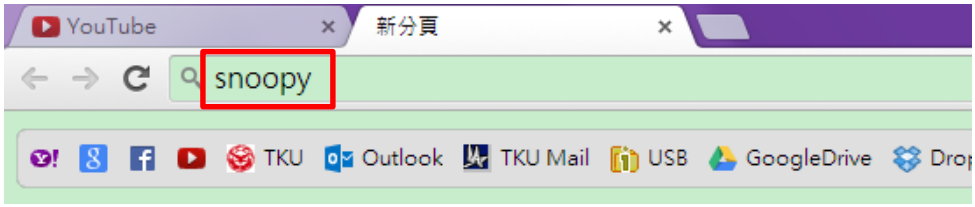


圖 4.3.2、在網址列輸入查詢字

圖 4.3.3、在 google 首頁輸入查詢字

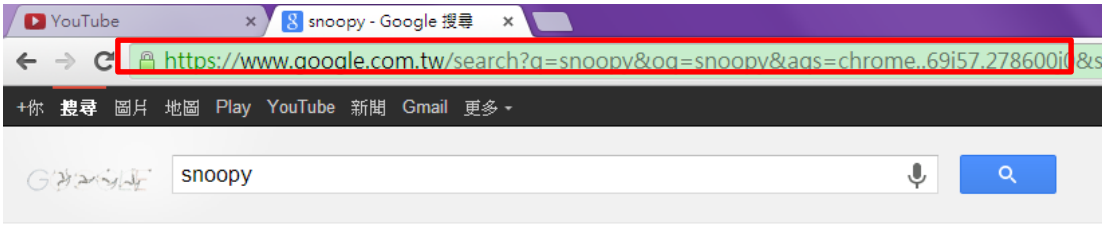


圖 4.3.4、網址列

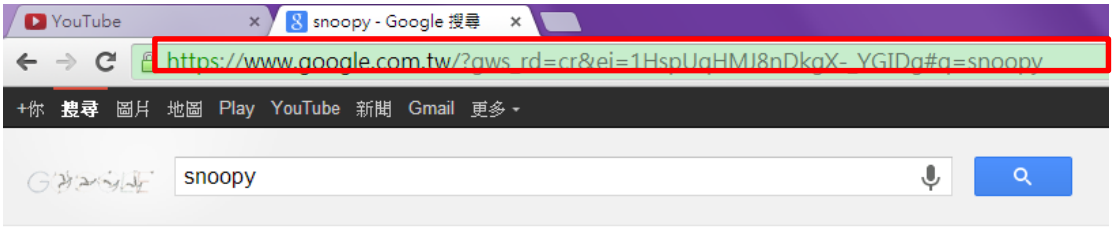


圖 4.3.5、網址列

取得標準的搜尋網址之後，進一步利用搜尋網址背後的原始碼，如圖 4.3.6，取得 10 組關鍵的 URL，如圖 4.3.7

圖 4.3.6、檢視背後原始碼

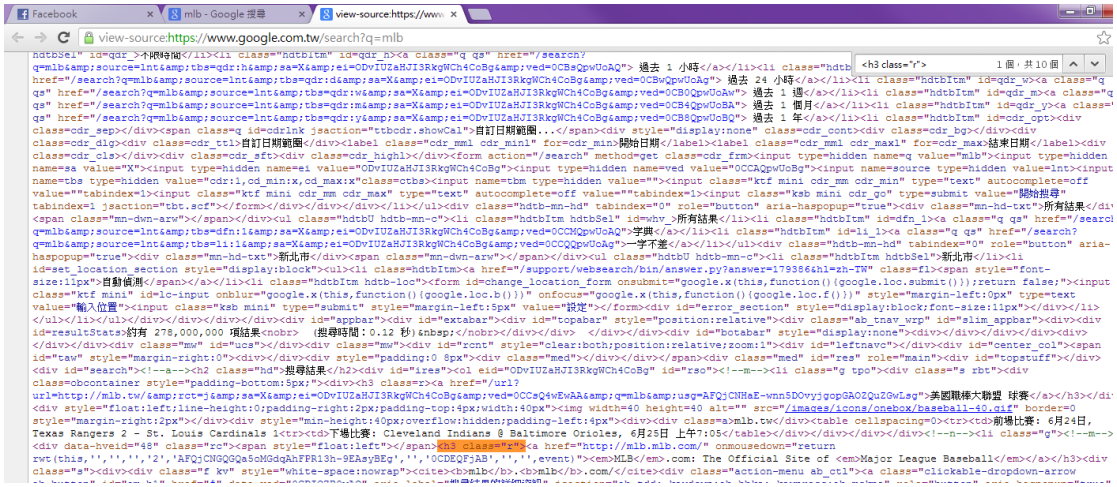


圖 4.3.7、網頁原始碼

在網頁原始碼裡頭，利用特殊關鍵字<h3 class=”r”>取得數個字串，預設會抓到 10 組字串，再從字串中擷取 URL。此外，修改網址的參數可以得到不同數量的結果，如圖，在網址後面加入 num=100，如圖 4.3.8，查詢的結果出現 100 個，如圖 4.3.9。

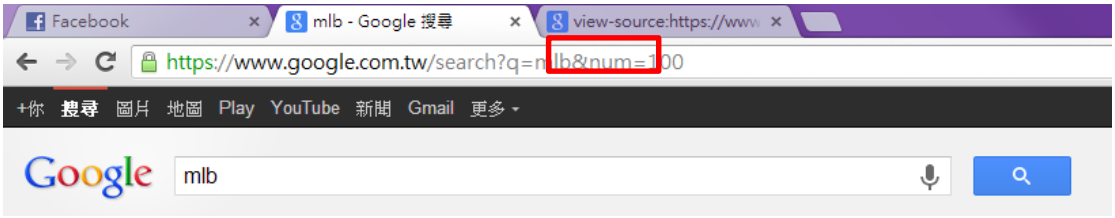


圖 4.3.8、加入參數 num = 100

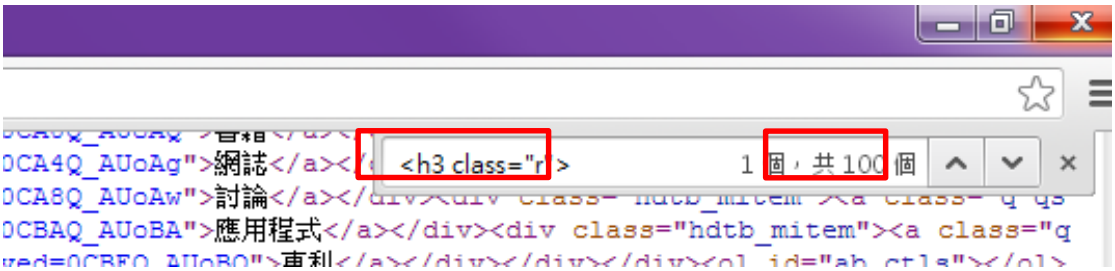


圖 4.3.9、查詢結果 100 個

jsoup API 是用來處理網頁背後原始碼的各種方法，設定特殊關鍵字得到一個 Iterator，裡頭包含數個字串，URL 包含在字串之中，如圖 4.3.10。

```
<a href="/url?q=http://www.sportslogos.net/teams/list_by_league/4/Major_League_Baseball/N/logos
<a href="/url?q=http://www.flashscore.com/baseball/usa/mlb/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYD
<a href="/url?q=http://www.detroitnews.com/section/sports0104&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9n
<a href="/url?q=http://www.mlb-picks.com/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;ved=0CNYCE
<a href="/url?q=http://www.bucerius.whu.edu/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;ved=0CN
<a href="/url?q=http://www.tiqiq.com/mlb/mlb-tickets&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp
<a href="/url?q=http://www.csnphilly.com/baseball-philadelphia-phillies/tonight-mlb-rangers-eye
<a href="/url?q=https://www.theshownation.com/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;ved=0
<a href="/url?q=http://www.foxdeportes.com/mlb/scores&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&am
<a href="/url?q=http://www.comcast.net/sports/mlb/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;v
<a href="/url?q=http://www.baseballnation.com/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;ved=0
<a href="/url?q=http://www.sportsclubstats.com/MLB.html&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ8
<a href="/url?q=http://www.teamrankings.com/mlb/&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;ved
<a href="/url?q=http://www.footballfanatics.com/MLB&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp
<a href="/url?q=http://www.topix.com/mlb&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&amp;ved=0CPwCEE
<a href="/url?q=http://www.2ksports.com/games/mlb2k11&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ&am
<a href="/url?q=http://www.oddschecker.com/baseball/mlb&amp;sa=U&amp;ei=Cf3IUfW0JsXtkgX9nYDYAQ8
<a href="/url?q=http://www.crawfishboxes.com/2013/6/23/4457466/mlb-scores-cubs-14-astros-6&amp
```

圖 4.3.10、取得多組字串

接著，將字串分析，從中取出正確的 URL，呈現結果如圖 4.3.11。

```
<terminated> GetURL [Java Application] C:\Program Files\Java\jdk1.7.0_21\bin\javaw.exe (2013/6/25 上午11:52:10)
http://ball0000001.pixnet.net/ --> 31
http://beanies1com9020.webs.com/ --> 37
http://download.sofun.tw/mlb-online/ --> 23
http://tw.sports.yahoo.com/mlb/ --> 98
http://56ei.com/ --> 30
http://cheapmlbjerseyspurchase.webs.com/ --> 8
http://www.az-sportsnet.com/sports/%3FbID%3Dmlb --> 76
http://us.playstation.com/games/mlb-10-the-show-ps3.html --> 43
http://www.mlbtraderumors.com/ --> 52
http://mlb.ftv.com.tw/ --> 96
https://www.facebook.com/mlb --> 82
http://885168.tw/ --> 45
http://www.thescore.com/mlb --> 44
http://aero0001.pixnet.net/blog/category/1836600 --> 29
http://www.tvbs.com.tw/news/news_list.asp%3Fno%3Dokinawa2920130624093726%26%26dd%3D2013/6/24%2520%2
http://www.ximen.com.tw/store_315 --> 3
https://play.google.com/store/apps/details%3Fid%3Dcom.bamnetworks.mobile.android.gameday.atbat -->
http://www.fengyuncai.com/asp/mlb.asp --> 84
http://foxsports.pixnet.net/blog/post/142236219-mlb-%25E5%2585%25AD%25E6%259C%2588%25E8%25BD%2589%2
http://www.ptt.cc/bbs/MLB/ --> 81
http://www.sbnation.com/mlb/2013/6/22/4455990/biogenesis-suspensions-mlb-ped-ryan-braun-alex-rodrig
http://www.baseballnation.com/ --> 47
http://hchao23.pixnet.net/ --> 24
http://www.gogosports.com.tw/index_mlb.html --> 78
http://www.vegasinsider.com/mlb/scoreboard/ --> 25
```

圖 4.3.11、從字串解析出 URL

將 URL 打包成 json 格式傳回 client，client 端程式會將 json file 解開並比對使用者瀏覽的網頁是否為這些網頁。若有則開始記錄其行為模式。

4.3.2、GetBehaviorServlet.java 流程

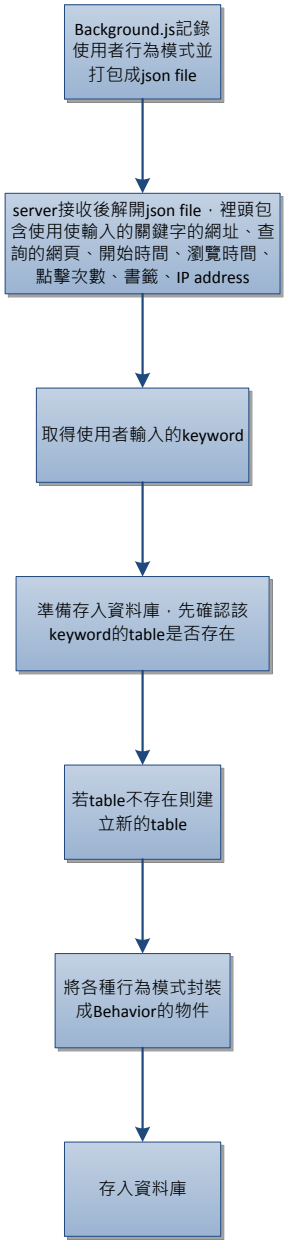
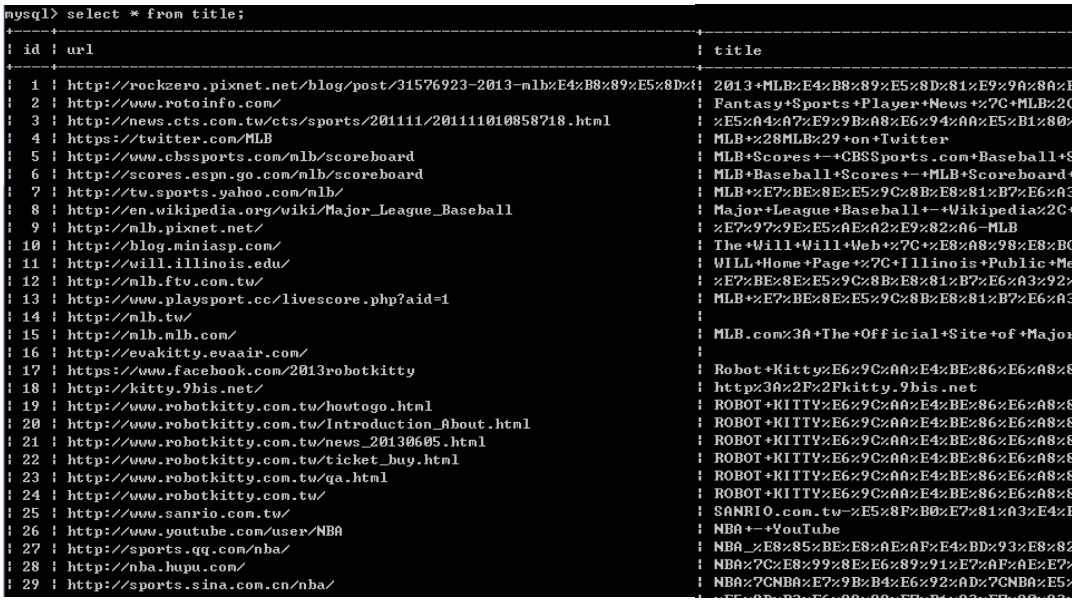


圖 4.3.2、GetURLServlet 流程圖

client 端程式的 background.js 會記錄使用者行為模式，包含使用使輸入的關鍵字的網址、查詢的網頁、開始時間、瀏覽時間、點擊次數、書籤、IP address，包裝成 json file 傳送給 server。

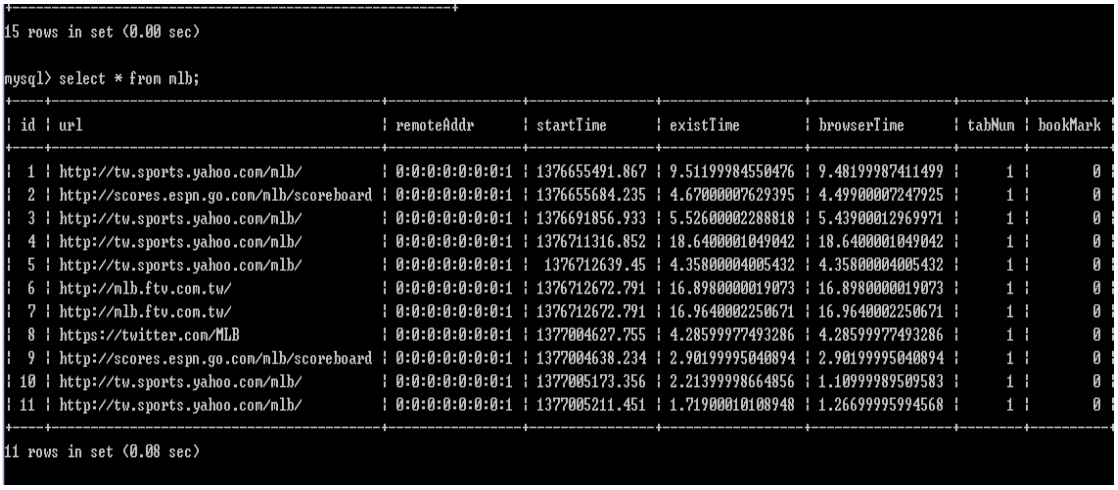
Server 收到資料後，解析關鍵字網址的字串中取出關鍵字，並進一步向 HVSRDB 確認，是否存在該關鍵字的 table，若不存在則產生一個新的 table，以便未來存入資料，table 如圖 4.3.12。



```
mysql> select * from title;
+----+-------------------------------------------------------------------+-------------------------------------------------------------------+
| id | url                                                               | title                                                             |
+----+-------------------------------------------------------------------+-------------------------------------------------------------------+
|  1 | http://rockzero.pixnet.net/blog/post/31576923-2013-mlb%E4%B8%89%E5%8D%{ | 2013+MLB%E4%B8%89%E5%8D%81%E9%9A%8A%E
|  2 | http://www.rotoinfo.com/                                          | Fantasy+Sports+Player+News+%7C+MLB%2C
|  3 | http://news.cts.com.tw/cts/sports/201111/201111010858718.html     | %E5%A4%A7%E9%9B%A8%E6%94%AA%E5%B1%80%
|  4 | https://twitter.com/MLB                                           | MLB+%28MLB%29+on+Twitter
|  5 | http://www.cbssports.com/mlb/scoreboard                           | MLB+Scores+-+CBSSports.com+Baseball+S
|  6 | http://scores.espn.go.com/mlb/scoreboard                          | MLB+Baseball+Scores+-+MLB+Scoreboard+
|  7 | http://tw.sports.yahoo.com/mlb/                                   | MLB+%E7%BE%8E%E5%9C%8B%E8%81%B7%E6%A3
|  8 | http://en.wikipedia.org/wiki/Major_League_Baseball                | Major+League+Baseball+-+Wikipedia%2C+
|  9 | http://mlb.pixnet.net/                                            | %E7%97%9E%E5%AE%A2%E9%82%A6-MLB
| 10 | http://blog.miniasp.com/                                          | The+Will+Will+Web+%7C+%E8%A8%98%E8%B0
| 11 | http://will.illinois.edu/                                         | WILL+Home+Page+%7C+Illinois+Public+Me
| 12 | http://mlb.ftv.com.tw/                                            | %E7%BE%8E%E5%9C%8B%E8%81%B7%E6%A3%92%
| 13 | http://www.playsport.cc/livescore.php?aid=1                       | MLB+%E7%BE%8E%E5%9C%8B%E8%81%B7%E6%A3
| 14 | http://mlb.tw/                                                    |
| 15 | http://mlb.mlb.com/                                               | MLB.com%3A+The+Official+Site+of+Major
| 16 | http://evakitty.evaair.com/                                       |
| 17 | https://www.facebook.com/2013robotkitty                           | Robot+Kitty%E6%9C%AA%E4%BE%86%E6%A8%8
| 18 | http://kitty.9bis.net/                                            | http%3A%2F%2Fkitty.9bis.net
| 19 | http://www.robotkitty.com.tw/howtogo.html                         | ROBOT+KITTY%E6%9C%AA%E4%BE%86%E6%A8%8
| 20 | http://www.robotkitty.com.tw/Introduction_About.html              | ROBOT+KITTY%E6%9C%AA%E4%BE%86%E6%A8%8
| 21 | http://www.robotkitty.com.tw/news_20130605.html                   | ROBOT+KITTY%E6%9C%AA%E4%BE%86%E6%A8%8
| 22 | http://www.robotkitty.com.tw/ticket_buy.html                      | ROBOT+KITTY%E6%9C%AA%E4%BE%86%E6%A8%8
| 23 | http://www.robotkitty.com.tw/qa.html                              | ROBOT+KITTY%E6%9C%AA%E4%BE%86%E6%A8%8
| 24 | http://www.robotkitty.com.tw/                                     | ROBOT+KITTY%E6%9C%AA%E4%BE%86%E6%A8%8
| 25 | http://www.sanrio.com.tw/                                         | SANRIO.com.tw-%E5%8F%B0%E7%81%A3%E4%B
| 26 | http://www.youtube.com/user/NBA                                   | NBA+-+YouTube
| 27 | http://sports.qq.com/nba/                                         | NBA_%E8%85%BE%E8%AE%AF%E4%BD%93%E8%82
| 28 | http://nba.hupu.com/                                              | NBA%7C%E8%99%8E%E6%89%91%E7%AF%AE%E7%
| 29 | http://sports.sina.com.cn/nba/                                    | NBA%7CNBA%E7%9B%B4%E6%92%AD%7CNBA%E5%
```

圖 4.3.12、每個 URL 對應一個 title

接著將各種行為模式的資料包裝成 Behavior.java 的物件，存入該 table 之中。如此一來，完成資料蒐集的功能，隨著時間的累積，會有越來越多筆使用者的資訊，往後需要使用時，可從資料庫拿出來運算，table 內容如圖 4.3.13。



```
15 rows in set (0.00 sec)

mysql> select * from mlb;
+----+-----------------------------------------+---------------+---------------+-----------------+-----------------+--------+----------+
| id | url                                     | remoteAddr    | startTime     | existTime       | browserTime     | tabNum | bookMark |
+----+-----------------------------------------+---------------+---------------+-----------------+-----------------+--------+----------+
|  1 | http://tw.sports.yahoo.com/mlb/         | 0:0:0:0:0:0:0:1 | 1376655491.867 | 9.51199984550476 | 9.48199987411499 |      1 |        0 |
|  2 | http://scores.espn.go.com/mlb/scoreboard | 0:0:0:0:0:0:0:1 | 1376655684.235 | 4.67000007629395 | 4.49900007247925 |      1 |        0 |
|  3 | http://tw.sports.yahoo.com/mlb/         | 0:0:0:0:0:0:0:1 | 1376691856.933 | 5.52600002288818 | 5.43900012969971 |      1 |        0 |
|  4 | http://tw.sports.yahoo.com/mlb/         | 0:0:0:0:0:0:0:1 | 1376711316.852 | 18.6400001049042 | 18.6400001049042 |      1 |        0 |
|  5 | http://tw.sports.yahoo.com/mlb/         | 0:0:0:0:0:0:0:1 |  1376712639.45 | 4.35800004005432 | 4.35800004005432 |      1 |        0 |
|  6 | http://mlb.ftv.com.tw/                  | 0:0:0:0:0:0:0:1 | 1376712672.791 | 16.8980000019073 | 16.8980000019073 |      1 |        0 |
|  7 | http://mlb.ftv.com.tw/                  | 0:0:0:0:0:0:0:1 | 1376712672.791 | 16.9640002250671 | 16.9640002250671 |      1 |        0 |
|  8 | https://twitter.com/MLB                 | 0:0:0:0:0:0:0:1 | 1377004627.755 | 4.28599977493286 | 4.28599977493286 |      1 |        0 |
|  9 | http://scores.espn.go.com/mlb/scoreboard | 0:0:0:0:0:0:0:1 | 1377004638.234 | 2.90199995040894 | 2.90199995040894 |      1 |        0 |
| 10 | http://tw.sports.yahoo.com/mlb/         | 0:0:0:0:0:0:0:1 | 1377005173.356 | 2.21399998664856 | 1.10999989509583 |      1 |        0 |
| 11 | http://tw.sports.yahoo.com/mlb/         | 0:0:0:0:0:0:0:1 | 1377005211.451 | 1.71900010108948 | 1.26699995994568 |      1 |        0 |
+----+-----------------------------------------+---------------+---------------+-----------------+-----------------+--------+----------+
11 rows in set (0.08 sec)
```

圖 4.3.13、使用者行為模式

GetBehavior.java 扮演著非常重要的角色，記錄所有使用者的資料，長久下來，資料量會愈來愈龐大，也就愈接近我們所要探討 big data 的主題，並利用分散式系統在大量資料中快速運算出結果，勢必是未來要研究的主軸。此外，有了多筆資料，準確度也會相對地提升，可用性會更高。

4.4、運算結果

運算的部分有兩種方法，其一是用單一的 Tomcat server 運算，其二是利用分散式系統，多台電腦運算以提升效率。

4.4.1、一般系統(GetSortedServlet.java)

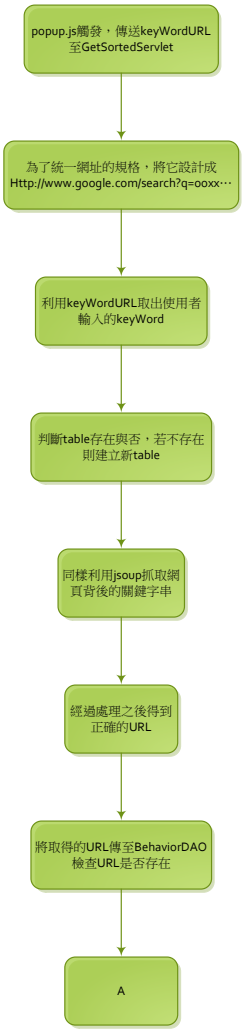


圖 4.3.14、GetSortedServlet 流程一

GetSortedServlet.java 是程式運算的核心，首先 client 端的 popup.js 會偵測使用者瀏覽的網頁，當使用者輸入查詢關鍵字時，popup.js 會傳回網址。Server 收到網址後，會統一網址的規格，將它設計成 http://www.google.com/search?q=ooxx…，並解析出使用者輸入的關鍵字，即為 ooxx。取出關鍵字之後，開始檢查 HVSRDB 內是否有有該關鍵字的 table，若 table 不存在，則產生新的 table 準備儲存資料。

建立完 table 之後，開始進行與 GetURLServlet.java 一樣的方法，取得 google 的排序，同時，給予分數的排序由高至低，即為 google score。加入 google score 的目的是為了讓整體排序更加穩定，避免讓初期資料庫的內容影響 google 原先的排序。隨著資料庫內容增長，漸漸地減少 google 排序的比重，逐漸依賴 HVSRDB 的資料。

若是 table 存在，代表該關鍵字的資料存在於資料庫，可以將資料拿出來運算，反之，table 不存在則參考 google 原先的排序為主。

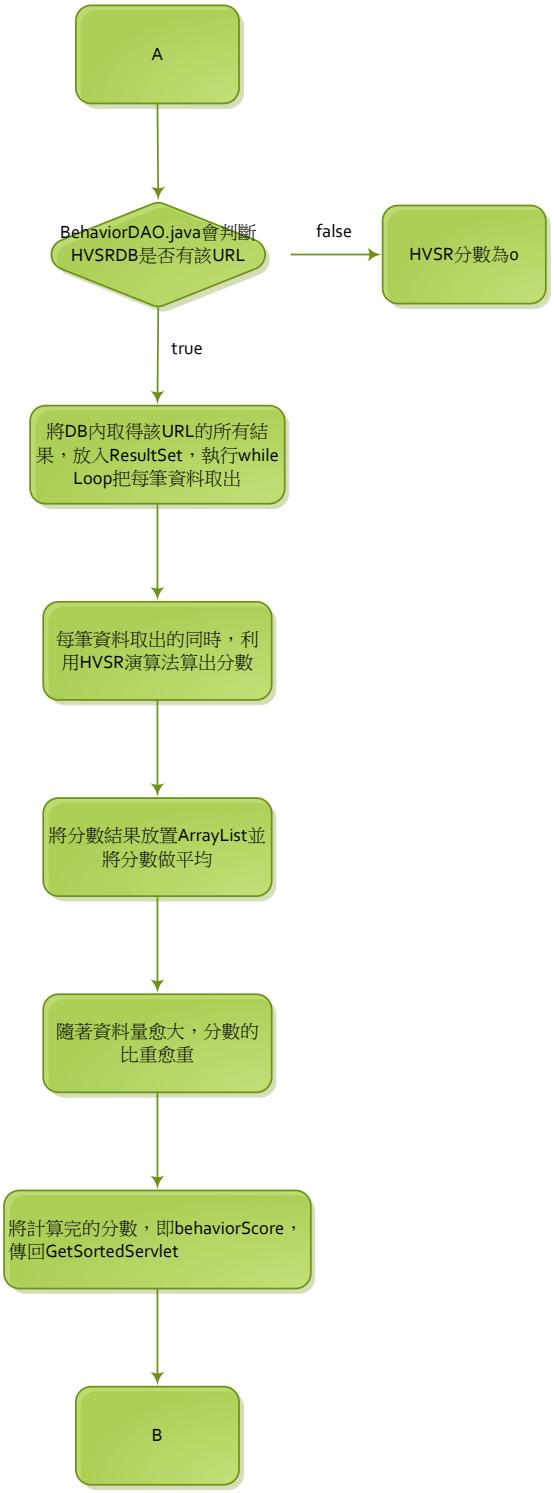


圖 4.3.15、GetSortedServlet 流程二

若是 HVSRDB 有關鍵字的 table，取出裡面的資料準備運算分數，即為 HVSR score，反之若是沒資料，HVSR score 為零。

將關鍵字的 table 資料取出，取出結果會是一個 ResultSet，再利用 while Loop 把 ResultSet 的內容逐一取出，取出的每筆資料裡頭都包含著 URL、瀏覽時間、點擊次數、書籤紀錄。接著，利用 HVSR 演算法算出每筆資料的分數，最後把相同 URL 的資料分數加總即為 HVSR score。

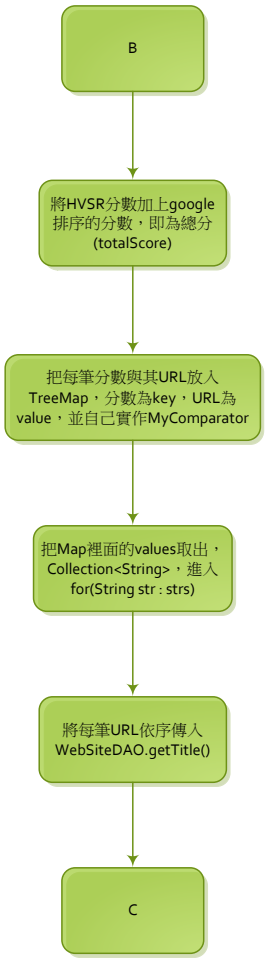


圖 4.3.16、GetSortedServlet 流程三

得到 HVSR score 之後，加上原先 google 原先排序分數，得到總分 total score。最後將每個 URL 及 total score 放入 TreeMap 中，並自行撰寫排序法 MyComparator，將分數由高至低排序，如圖 4.3.17。

```
100.0 , http://mlb.tw/
99.0 , http://mlb.mlb.com/
98.01420312499006 , http://tw.sports.yahoo.com/mlb/
97.00560879652147 , http://scores.espn.go.com/mlb/scoreboard
96.00880555477407 , http://www.playsport.cc/livescore.php?aid=1
95.03464236110449 , http://mlb.ftv.com.tw/
94.01315625011921 , http://www.cbssports.com/mlb/scoreboard
93.0 , http://mlb.tw/
92.00662499997351 , http://mlb.pixnet.net/
91.02249652726783 , http://en.wikipedia.org/wiki/Major_League_Baseball
90.01169791569312 , https://twitter.com/MLB
```

圖 4.3.17、新的排序結果

既然已經有了排序，接著就是要將每個網頁的 title 抓出來，即為一般所看的標題，如圖 4.3.18。

圖 4.3.18、新的排序結果

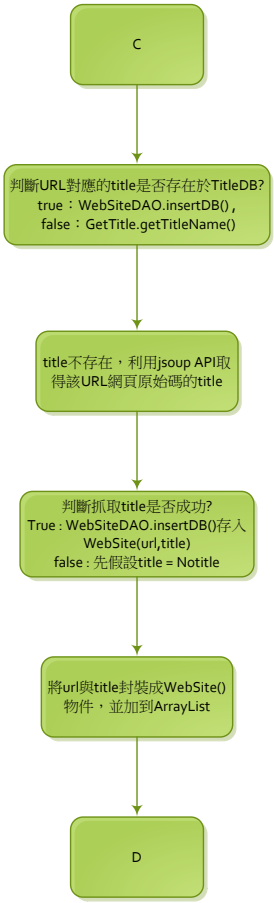


圖 4.3.19、GetSortedServlet 流程四

抓取 title 的方法，首先先從資料庫的 TitleDB 尋找 title 是否存在，若存在就直接拿來使用，若不存在則使用 jsoup API 去抓取某個網址<title>與</title>中間的內容。

倘若正確的取得了 title，將其存入資料庫，以便下一次使用且提升速度。但在抓取的過程中，往往會因為許多情況，如：Connect time out、MySQLNonTransientConnectionException…等狀況。導致 title 無法正確地取得，這時，先假設 title 為 Notitle，未來還有機會抓取 title 時，再嘗試

取得 title。

完成之後，將 url 與 title 封裝成 WebSite 物件，並加入 ArrayList，正因 ArrayList 是個依照順序的 List，所以放入的排序仍舊照著順序，不會因此而錯亂。

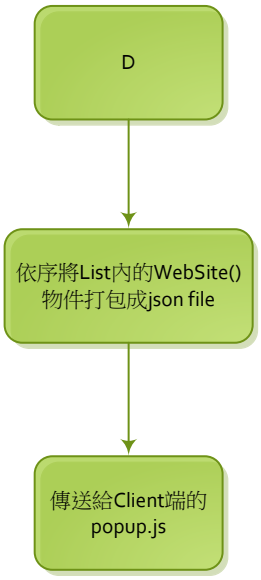


圖 4.3.20、GetSortedServlet 流程五

最後，將 ArrayList 內的 WebSite 物件打包成 json file，準備傳送給 client 端的 popup.js，呈現給使用者。

4.4.2、分散式系統(GetSortedByHadoopServlet.java)

使用分散式系統運算是為了提升效率，試想，若是有成千上萬的使用者使用本系統，光是單一的 server 運算是明顯不足的。所以運算的部分改由 GetSortedByHadoopServlet 執行。

程式當中，與 GetSortedServlet.java 大同小異，差別只在運算 HVSRDB 內的資料交給 Hadoop 分散式系統運算，以下為 Hadoop 運算的流程。

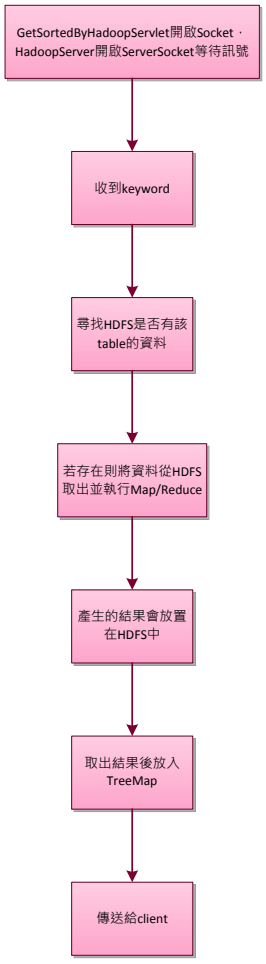


圖 4.3.21、GetSortedByHadoopServlet 流程

首先，GetSortedByHadoopServlet.java 收到 client 端的訊號，經過一連串的處理後，傳送該關鍵字給 HadoopServer.java，接著再去尋找 HDFS 內是否有該關鍵字的資料，如圖 4.3.22。

```
1  www.mlb.tw 10.8 1 0
2  www.yahoo.com 15.7 2 1
3  www.ftv.com 20.6 3 0
4  www.mlb.tw 13.5 1 1
5  www.yahoo.com 25.4 3 0
6  www.ftv.com 27.3 3 0
7  www.ftv.com 20.9 5 1
8  www.mlb.tw 83.5 4 1
9  www.yahoo.com 23.5 2 1
```

圖 4.3.22、使用者行為模式在 HDFS 中

HDFS 是 Hadoop 的檔案系統，具有容錯的功能，並且能夠使用 Map/Reduce 處理資料。目前的資料都是以字串的形式呈現。Map 的做法是，將每筆的資料逐列取出，取出的資料再用空格做為切割，切割出來的小部份即為 URL、瀏覽時間、點擊次數、書籤紀錄，與先前的結果相同。進一步利用這些算出 HVSR score，map 完畢後的結果會是許多組 URL 對應一個 score。Reduce 會將相同 URL 部分放入一個 Iterator，再取出每筆資料將各個數值做加總，所得的結果如圖 4.3.23。

```
1 www.ftv.com 0.1689351851851852
2 www.mlb.tw  0.5524305555555555
3 www.yahoo.com   0.3310185185185185
```

圖 4.3.23、map / reduce 結果

接下來的做法與 GetSortedServlet.java 一樣，加上 google score，成為 total score，最後將 URL 與 total score 結果包裝成 TreeMap，並實作 MyComparator，再包裝成 json file 傳給 client。

map / reduce流程

在HDFS內，儲存的資料格式如圖4.3.24：

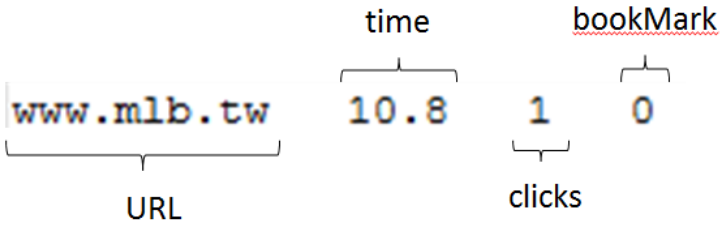


圖4.3.24、單筆資料型態in HDFS

資料格式分為URL、瀏覽時間、點擊次數、書籤紀錄，存在於HDFS之中。map開始時，會解析每筆資料，接著依序用空格分開，得到各個值，

進一步利用HVSR演算法，以時間、點擊次數、書籤紀錄為參數，算出一個分數，即為HVSR score。假設使用者輸入於google查詢mlb，系統會將URL和HVSR score寫入/input/mlb.txt中，如圖4.3.25：

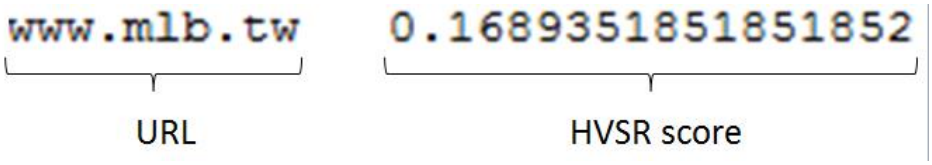


圖4.3.25、map結果

map完成之後，輪到reduce執行，reduce會找出相同的URL，並將其HVSR score全部加起來，示意圖4.3.26。最後，每個不同的URL都對應著一個分數，reduce結果會產生在/output/keyword/part-r-00000內，如圖4.3.27。

```
www.mlb.tw    0.5198464599713549
www.mlb.tw    0.1689351851851852
www.mlb.tw    0.6598765322148984
www.mlb.tw    0.9649871164166988
www.mlb.tw    0.9876626974169552
```

圖4.3.26、reduce過程

```
www.mlb.tw    3.301307991205093
```

圖4.3.27、reduce結果

第五章、HVSR 使用手冊

5.1、HVSR-Extension 安裝

使用手冊 HVSR-Extension 安裝步驟使用第二種方法，自行至瀏覽器擴充功能新增。

1.前往擴充功能頁面:

開啟 Chrome 瀏覽器並在網址列輸入"chrome://extensions"，即可進入擴充功能頁面，如圖 5.1.1 所示。

圖 5.1.1、Chrome 擴充功能頁面。

2.載入 HVSR 資料夾:

如圖 5.1.2 所示，點選"載入未封裝擴充功能"，並選擇 HVSR 資料夾目錄，按下 "確定"即可。

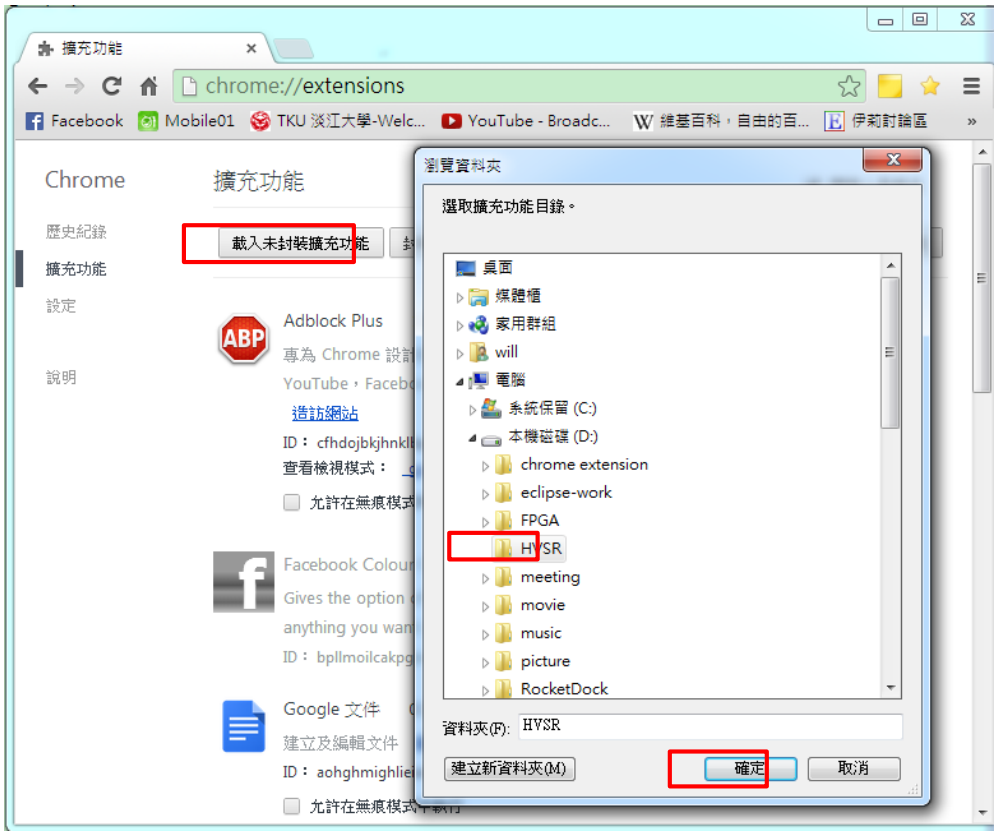


圖 5.1.2、載入 HVSR 資料夾。

3. HVSR 載入結果:

HVSR 成功載入後，結果會出現在擴充頁面中，並且在網址列右方會有 HVSR 的 button 圖示，如圖 5.1.3 所示。

圖 5.1.3、HVSR 載入結果。

5.2、Google 查詢關鍵字

輸入欲查詢的關鍵字(ex:tku)並搜尋，右下角即會出現 HVSR 黃色方塊如圖 5.2.1，內容為新的連結排序可以點選進入。

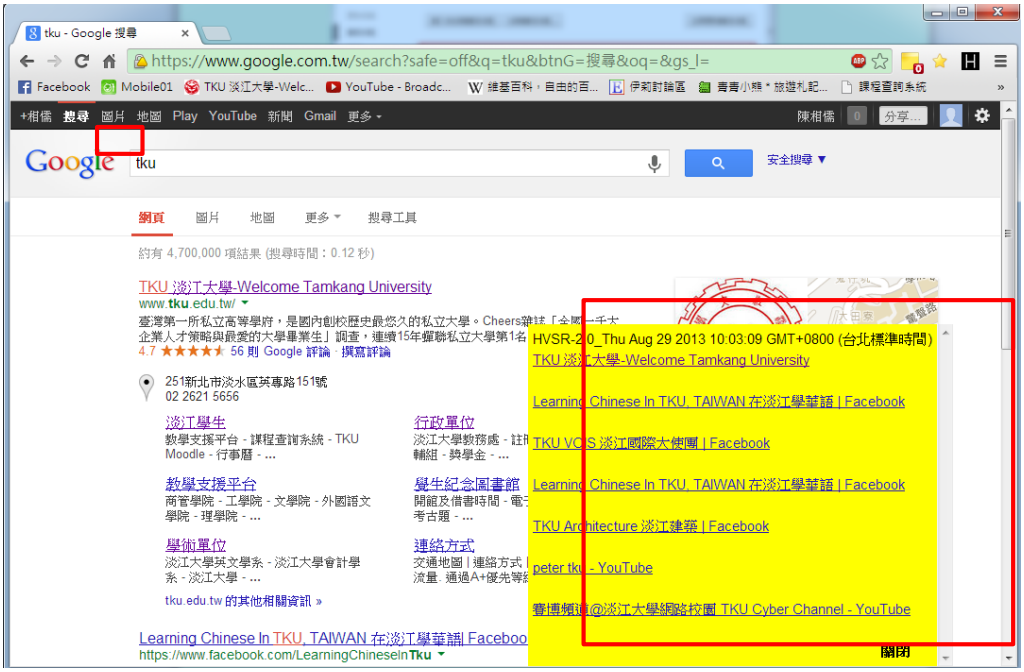


圖 5.2.1、HVSR 查詢關鍵字結果

5.3、關閉 HVSR 排序黃色圖塊

點選黃色圖塊右下角"關閉"按鈕如圖 5.3.1，即可關閉黃色圖塊。

圖 5.3.1、HVSR 黃色圖塊關閉按鈕

5.4、點選 HVSR button 圖示

點選右上角 HVSR button 圖示，即會跳出新的排序連結圖塊如圖 5.4.1，與黃色圖塊用途相同。當使用者關閉黃色圖塊後，欲再使用 HVSR 的連結排序，即可點選 HVSR button 獲得連結排序。

圖 5.4.1、HVSR button 圖示與排序連結圖塊。

5.5、HVSR-Extension 移除

1.前往擴充功能頁面:

開啟 Chrome 瀏覽器並在網址列輸入"chrome://extensions"，即可進入擴充功能頁面，如圖 5.5.1 所示。

圖 6.5.1、Chrome 擴充功能頁面。

2.移除 HVSR-Extension:

點選 HVSR 右方垃圾桶圖示 如圖 5.5.2，即可移除 HVSR。

圖 5.5.2、HVSR 垃圾桶圖示。

第六章、效率與評估結論

本專題使用 hadoop 分散式系統運算，接下來要探討節點數與資料量的關係。因此，準備了 1000 萬、2000 萬、4000 萬、8000 萬、16000 萬筆資料，另外，搭配 1、2、4、8 個節點。欲算出 map/reduce 所需要的時間。經過實驗之後，結果如下表 6.1 與圖 6.2。

	1000 萬筆	2000 萬筆	4000 萬筆	8000 萬筆	16000 萬筆
1 node	58.852	107.879	191.965	365.366	607.598
2 nodes	43.962	74.833	125.923	204.180	323.702
4 nodes	40.901	60.182	77.168	138.255	249.023
8 nodes	32.972	44.650	68.622	83.193	128.936

表 6.1、節點數與資料量實驗結果(單位：秒)

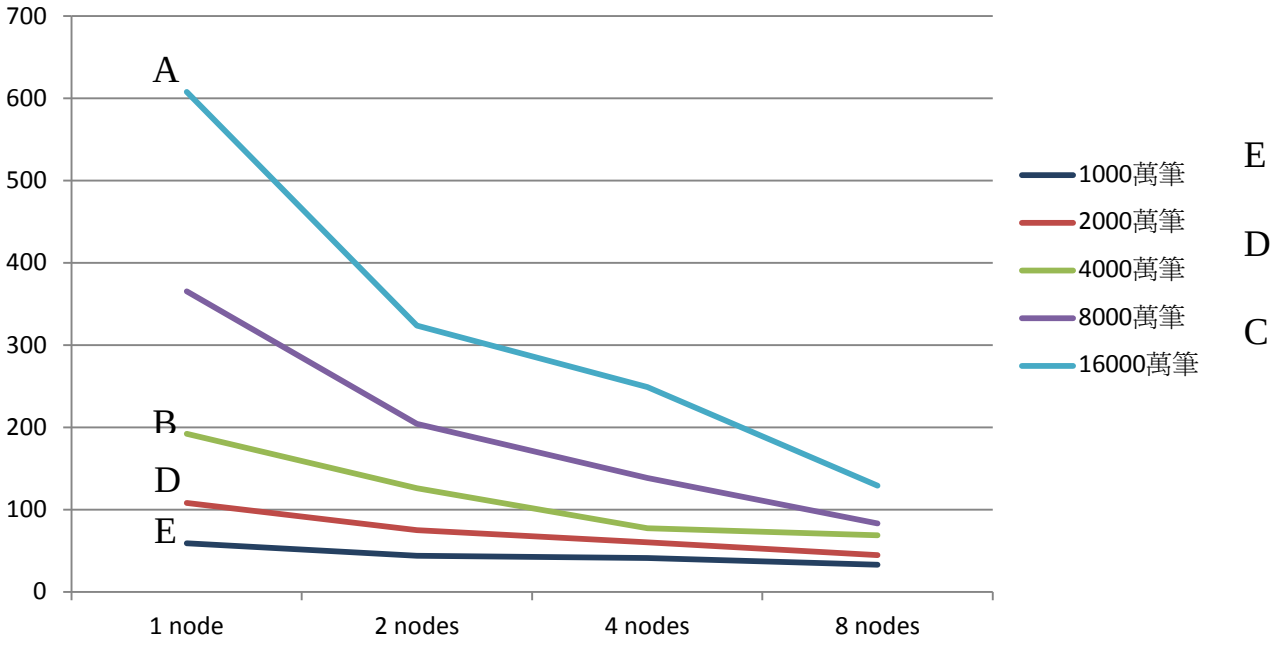


圖 6.2、map / reduce 所需時間關係圖(單位：秒)

由實驗結果得知：在資料量愈大的情況下，節點的多寡對於程式執行的時間影響顯著。當節點數非常大時，資料量呈倍數成長，但執行時間增加的幅度並不會太大，這就是使用分散式系統的優點。如今是個 big data 的時代，若是要解決大量資料運算，勢必要使用分散式系統，多節點的處理才能增加效率。

第七章、未來展望

針對這次專題實作，由於這次專題主要為實驗且與經費上的限制，因此 HVSR 無法上架於 Google 的 Chrome Extension 商店供使用者下載，只能透過自行掛載擴充方式使用，相對來說使用的人是比較少的，因而無法透過大量使用者取得大筆資料，因此在最後的模擬分析只能自行使用亂數產生大筆資料，來實作分析分散式系統的效能。在未來有機會的話，我們會將 HVSR 上架，並從眾多使用者中取得大量的瀏覽行為模式資訊，且從大筆的資料中實際分析在分散式系統下的效能評估。

起初在規畫 Hadoop 架構時，由於缺乏經驗並沒有將 Hbase 規劃在其中，因此在分散式系統運作時並沒有辦法發揮到最大效能。之後我們除了要將 Hbase 規劃進去，還要找尋相關文獻看看是否有更好架構流程，能夠提高分散式系統的效能。

在這次專題實作，我們證明了可以透過瀏覽器的掛載程式來蒐集使用者在瀏覽網頁的行為模式資訊，這些資訊我們相信是非常有價值的，但是我們並非是專業的資料探勘者，也沒有受過資料探勘相關課程的訓練，因此我們只能從現今所學的相關知識內容，試著去分析出各種行為模式資訊中的關聯，並連結出一個結果為新的連結排序。希望在未來我們在專業知識的能力補足時，能夠從這些行為模式資訊中找出更讓人驚豔更有價值的成果。

除了上述幾點外，其實 HVSR 仍有許多地方需要改進。然而還有許多的缺失是由於不足的學術論文基礎，也因為我們在一開始所尋找的論文並不夠深也不夠廣，造成很多地方缺少學術力量的支持。然而我們相信 HVSR 會是一種新的資料探勘方式，希望在未來可以加入更多的學術論文做為支柱，使得 HVSR 有更加成功的發展。

參考資料

[1] 維基百科－Big data

http://zh.wikipedia.org/wiki/%E5%A4%A7%E6%95%B8%E6%93%9A

[2] 趨勢科技 2013 Big Data 騰雲駕霧創意程式大賽

http://contest.trendmicro.com/2013/tw/#

[3] 維基百科－分散式系統

http://zh.wikipedia.org/wiki/%E5%88%86%E6%95%A3%E5%BC%8F%E6%AA%94%E6%A1%88%E7%B3%BB%E7%B5%B1

[4] Eclipse 介紹

http://www.cc.ntu.edu.tw/chinese/epaper/0020/20120320_2003.html

[5] 維基百科－Tomcat

http://zh.wikipedia.org/wiki/Apache_Tomcat

[6] 安裝 tomcat

http://youthhng.pixnet.net/blog/post/35658541-%5Bjsp%5D-jsp%E7%92%B0%E5%A2%83%E5%AE%89%E8%A3%9D---eclipse-%2B-tomcat

[7] windoop 下載

https://code.google.com/p/windoop/

[8] Google chrome Extension 官方網站

http://developer.chrome.com/extensions/getstarted.html

[9] 國網中心－Hadoop 簡介、安裝與範例實作

http://trac.nchc.org.tw/cloud/wiki/NCHCCloudCourse090914

[10] Java7 API

http://docs.oracle.com/javase/7/docs/api/