東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

以 ELK Stack 環境實作國道電子收費系統開放資料視覺化及車流分析

An Implementation of ETC Open Data Visualization and Traffic Analysis Using ELK Stack Environment

研究生: 楊舜文

中華民國一〇七年一月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 楊 舜 文 所提之論文

以 ELK Stack 環境實作國道電子收費系統

開放資料視覺化及車流分析

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 許慶賢 簽章

委 員 [illegible]

賴冠州

詹毓偉

指 導 教 授 楊朝棟 簽章

中華民國 107 年 1 月 17 日

摘要

在政府積極推動下，政府資料開放平台的提供多樣化的開放性資料，其中最廣泛應用的就是和生活息息相關的氣象資訊及空氣品質資訊以及公車資訊，除此之外國道高速電子收費系統資訊再去識別化後，也公開在此平台供大家應用。國道收費系統每日都有上百萬輛車次的交易紀錄，傳統的樞紐分析表已經無法有效分析呈現。本論文提出的架構主要應用 ELK Stack 環境架構的分析系統，對高速公路電子收費系統的開放性資料，進行即時分析與統計，透過這個系統的視覺化圖表，可以快速瞭解當前車速情況，及進行交流道車流量道統計與分析。分析系統使用 Linux Shell script 取得開放資料，經過資料清洗，Logstash 對資料類別進行過濾及分類，依不同的分類輸出至 Elasticsearch 資料庫。最後由 Kibana 設定資料搜尋條件及呈現圖表樣式。此系統解決傳統樞紐分析表的筆數限制，並在五億筆資料中搜尋及運算回應時間約 0.3 秒。另外在簡易的查詢測試當中也發現 Elasticsearch 資料庫比沒有建立 index 的 mariaDB 快 1 倍以上。

關鍵字：巨量資料、開放資料、國道電子收費系統、視覺化、分析平台。

Abstract

In the government to actively promote the Open Government Data. The Government Data open platform to provide a variety of open data. The most widely used and life is closely related to the weather information and air quality information. After "De-identification" information in Traffic Data Collection System(TDCS) , also open for everyone in this platform to use. TDCS data has millions of vehicle trips per day. The traditional analysis table has been unable to effectively and quickly present analysis and resolution. This thesis mainly uses ELK Stack three open source software combination of analysis system. Real-time analysis and statistics on the open data of the TDCS . Through the visualization of the system chart. You can quickly understand the current speed situation and analysis of traffic flow and departure traffic statistics. This analysis system using Linux Shell Script get open data, and reads the cleaned data via Logstash. Combine Logstash to filter the data category. Export to the Elasticsearch database and index it. Finally, Kibana shows the results of the analysis. This system solves the limit of the number of traditional pivot analysis tables. Searches and calculates the request duration time by about 0.3 second in 500 million data. Also in the simple query test also found Elasticsearch database than non-index MariaDB more than doubled.

Keyword：Big data, Open data, ELK Stack, ETC, TDCS, Visualization.

誌謝辭

在時隔 11 年後終於完成了年少求學時的夢想，順利取得學位，圓了當兵前未能取得學位的遺憾。學習方向雖然由電機工程跨領域到資訊工程，但在求學過程卻非常順遂。這要非常感謝我的指導老師楊朝棟教授，楊老師在我求學過程當中，指引我的研究方向，給予我許多寶貴的意見。更教導我許多從未接觸過雲端計算及大數據技術，透過實驗及專案實作磨煉技術，讓我獲益良多。對於已經在工作我來說，能兼顧學業、工作及家庭本來就很不容易，謝謝楊老師教導知識及經驗，讓我能學以致用，將這些學到的技術應用在工作上，讓我也能獲得贏在起跑點的機會。

在此也要感謝中華大學資訊工程學系許慶賢教授、東海大學資訊工程學系呂芳懌教授、國立臺中教育大學資訊工程學系賴冠教授教授及靜宜大學資訊管理學系詹毓偉助理教授在口試過程中，給予學生諸多指導及許多修改建議，讓我的論文內容能夠更臻於完善。

在求學過程中，認識了許多好朋友及好同學。雖然年齡上有很大的距離，但彼此的經驗及心得分享卻是沒有距離的。在此特別感謝遠在印尼的陳彩進同學，不論技術或態度上，從他身上學習到很多，也謝謝實驗室的同學們，希望相聚相識的緣分不會隨著畢業而消散。

最後要深深地感謝的是在背後默默支持我、一直陪伴在身邊的老婆林玉雯及爸媽，謝謝妳在我深夜做實驗時，泡杯熱茶讓我提提神。在我日不暇給時給予我時間與空間，您們的支持是我完成這個目標最大動力。謝謝!

Table of Contents

List of Figures

List of Tables

Chapter 1

簡介

1.1 研究動機

在雲端運算及巨量資料蓬勃發展的時代，雲端運算與服務是指可以隨時、隨地、依需求、使用任何裝置情況之服務，而巨量資料的定義談的不僅僅是資料量 (Volume)，還包含了時效性 (Velocity)、多樣性 (Variety) 及真實性 (Veracity)。[1] 運用於巨量資料的技術與框架也非常成熟，常見的有 Hadoop ecosystem [2] 與 Spark ecosystem [3] 這些巨量資料處理工具，不只提供分散式儲存空間更提供高效能處理技術，使得巨量資料的應用更為便利。

在巨量資料 4Vs 定義下，國內最大的公開巨量資料及開放資料就是「政府資料開放平台」[4]，這個平台的目的是為了在政府資源有限下，能夠善用民間無限之創意，整合運用政府各部門的開放資料，推動政府資料開放以及加值應用，經由政府各部門的資料開放，可促使跨機關的資料流通，提升施政效能，滿足民眾需求，例如空氣品質指數 AQI [5]、氣象觀測資料、公車即時資訊等開放資料，都被程式開發人員應用在各種平台的 APP。

對於開車的用路人而言最關心就是路況了，往往一個事故或著意外，都會導致既定的行程延誤，尤其在國道高速公路上當你遇到塞車時，更是只能耐心的等待車流舒緩，自 2006 年開始啟用電子收費已經超過十年，從一開始的計次

收費，偵測器安裝於收費站上，到 2014 年的於交流道前後設置偵測器門架依里程收費，並拆除收費站，這些數據的更是呈現倍數的成長，在 2015 年時交通部高公局更將資料去識別化後開放。

高公局所提供的國道高速公路電子收費-交通資料蒐集支援系統 (Traffic Data Collection System，TDCS) [6] 開放資料，在 2015 年 9 月舉辦第一屆 TDCS 資料創意競賽，因為釋出含個人資訊的行車路徑原始資料 M01A，雖然已經刪除車牌資料，但資料內因為含有車輛識別碼、各車量用路起訖時間與地點，在車流較少的時段，仍有可能配合其他資訊來識別出特定車輛的行車路徑，因受到爭議所以不對外開放。但高公局也連續舉辦了三屆 TDCS 資料創意競賽，從第一屆主題的自由創意，第二屆的時間預測，到第三屆的資料視覺化應用 [7]，可以知道高公局對於資料的視覺化呈現也相當重視。

1.2 論文目標與貢獻

本論文運用 ELK Stack 也就是 Elasticsearch、Logstash、Kibana(ELK Stack) 這三個開源軟體組合而成的分析系統，對 TDCS 開放資料進行即時分析與統計，依 TDCS 更新頻率提供即時及批次作業資料呈現，即時處理經過清洗後的數據，快速呈現車速資訊並利用偵測站座標顯示於地圖，並且關鍵字篩選後的結果自動繪製圖表，以一種簡單直覺的方式，讓需要的人了解資訊，更進一步透過分析後的結果來規劃及疏導壅塞的交通。

此系統架構更有別於過往應用於 syslog 收集與處理，不但可同時處理多種不同格式的開放性資料的收集與處理，更可以透過資料類別進行資料篩選過濾。目前高公局 1968 APP 以文字描述位置及車速級距來顯示各區段車速，本系統在圖表呈現上，可透過車種的篩選更容易掌握各車種行駛速度，更可以選定特定時間區段了解歷史車況，更可透過關鍵字來過濾及即時呈現圖表。在實驗當中我們透過分析結果發現以分析台中交流道為例，清楚了解各時段車流狀況及得到上班時段車輛都來自於附近縣市，更發現部分台中到新竹科學園區有固定的車流在往返。

1.3 論文架構

本論文主要利用 ELK Stack 環境建置一個分析系統，運用視覺化套件將國道電子收費系統的資料進行分析與呈現，其呈現結果可作為決策人員決策時參考依據。第一章簡述相關研究過程與結果，再於視覺化呈現的結果從中延伸出本論文的研究目標及方向；第二章介紹相關研究背景及其各項套件技術概念、本文中將使用之運用工具進行說明簡述；第三章節中將描述論文中使用的系統架構概觀與測試機器的資源配置規格；第四章實驗環境配置與實驗數據處理與結果呈現；最後於，第五章將對於本論文之實驗過程、結果，經彙整後作出最終結論，並且從中延伸出未來的研究方向。

Chapter 2

研究背景與相關研究

2.1 研究背景

2.1.1 巨量資料

大數據或稱為巨量資料 (Big Data) 就是是超過傳統數據庫系統處理能力的數據 [8]，大數據是一個抽象的概念。除了大量的數據外，還有一些其他的特徵決定了它與“海量數據”或“非常大的數據”的區別，大數據是一個龐大而且復雜的數據集合，使用一般的數據庫管理工具或傳統數據處理應用程序難以處理。這些挑戰包括資料的擷取，管理，儲存，搜尋，共享，傳輸，分析和視覺化。Big Data 4V 模型談的就是資料量 (Volume)、時效性 (Velocity)、多樣性 (Variety)、可疑性 (Veracity) 如 Figure 2.1，這 4V 就是巨量資料的四的特點: 大、快、雜、疑，而對大數據本質的觀察分析便從大型數據集中發現大量隱藏的價值，這些數據集是多樣的，複雜的，規模巨大的。而 4V 定義如下：[9] [10]。

- Volume：從不同來源產生的所有類型的數據量，並繼續擴大。收集大量數據的好處包括通過數據分析創建隱藏的信息和模式。如線上交易、網路搜尋等都會不斷產生資料。

- Variety：通過感測器，智慧型手機或社群網絡等收集的不同類型的數據，以結構化或非結構化格式收集的數據類型包括影片，圖片，文字，聲音和數據日誌，這些都很難以傳統關聯式的固定資料欄位架構來解決。[11]

- Velocity：數據傳輸頻率及處理效率。數據內容在不斷變化，為了收集及完善完整的數據，有時會導入以前存檔的數據或存放很久的數據，以及來自多個來源的即時數據等，例如用於市場預測，決策分析，那處理的時效如果太長就失去了預測的意義了，所以處理的時效對 Big Data 來說也是非常關鍵的。[12]

- Veracity：資料的可疑性是指我們如何從這麼大量的資料進行分析，如果資料的真實性本身就有問題，那分析出來的結果肯定也是有問題的，所以對於資料進行過濾，以去除異常或者偽造的資料。

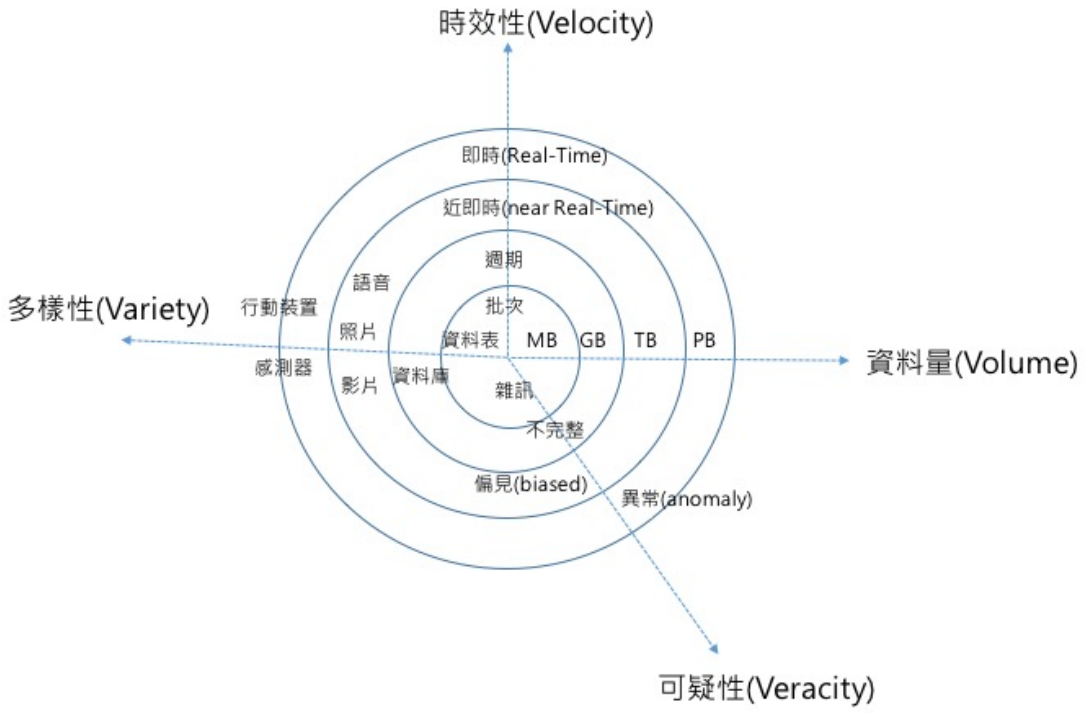


FIGURE 2.1: 巨量資料的 4V 定義

2.1.2 日誌分析

作業系統和應用服務輯，都在不停的產生日誌資料。過去，日誌資料基本都存在硬碟或磁帶上，只能用來做事件異常的事後分析與稽核，在巨量資料工具功能功能成熟後，大家漸漸習慣將收集日誌存放到 Hadoop HDFS 中，然後每天定時運行 MapReduce 任務呈現統計報表，但是面對即時性的需求，例如新上線程式的過去幾分鐘在是否有發生異常，特定時段是不是有發生異常，相對這些即時性的需求，傳統的排程作業報表，相對起來比較無法符合現在即時又多變的分析需求。[13]

透過即時取得 log 的方式，可以快速瞭解系統運作現狀，以往的 log 分析有幾個困難點：1、無法即時查看特定時間的 log。2、log 內容非結構化數據，不易判讀。3、使用 grep/awk 等工具過濾 log，只能到特定的設備單機執行。使用 ELK Stack 架構的分析平台相較之前出現問題要去尋找過往 Log 再一一排查的方式，提高效率、省時以及便捷許多。

2.1.3 ELK Stack

ELK Stack [14] 是 Elasticsearch、Logstash、Kibana 這三個開源軟體的搭配如 Figure 2.2，各取頭一個英文字母的組合。在即時資料搜尋、檢索、分析和呈現場合，三者通常是相互搭配使用，而且目前都屬於 Elasic.co 公司所維護的開源軟體，故有此簡稱。ELK Stack 需要收集 Log 的服務伺服器上部署 Logstash，利用 Logstash agent 監控並過濾收集服務產生的 Log，將過濾後的內容發送到全文檢索搜尋服務 Elasticsearch，可以使用 ElasticSearch 進行自訂搜索通過 Kibana 來結合自訂搜索進行頁面展示。

ELK Stack 具有下列幾個優點 [15]：

- 處理及分析方式簡單靈活。
- 配置簡易上手，Elasticsearch 是採用 JSON 介面，Logstash 是採用 Ruby DSL 設計，都是目前最常見的配置語法設計。

- 檢索性能高效，每次查詢都是即時運算，基本上可以達到資料查詢的快速響應。
- 橫向擴展，Elasticsearch 可以叢集方式橫向擴展其儲存空間，並進達到資料複寫功能。
- 視覺化操作簡單，在 Kibana 界面上，只需要點擊滑鼠，就可以完成搜索、聚合功能，並產生淺顯易懂的儀表板。

FIGURE 2.2: ELK Stack 架構圖

Elasticsearch 分散式搜尋系統，提供搜集、分析、儲存數據等功能，具備 REST [16] 和 JAVA API 等傳輸架構提供高效率搜索功能，建構於 Apache Lucene 搜尋引擎庫上，也在 Apache 許可條款下以開放源始碼方式發佈，應用於雲端計算中，可達到即時搜索、穩定、可靠快速與安裝使用方便。

Elasticsearch 是一個用 Java 編寫的開源全文搜索引擎，它具有分佈式配置的能力。Elasticsearch 服務器易於安裝 [17]，預設的設定值就足夠獨立使用而不需要調整，但大多數用戶最終都希望調整一些參數。運行 Elasticsearch 服務的機器稱為節點，兩個或多個節點可以形成 Elasticsearch 叢集。要設置 Elasticsearch 叢集，需要在配置文件中設置的唯一值是叢集的名稱，Elasticsearch 會自動發現網絡上的節點並將其綁定到一個叢集中。[18]

Logstash 是一個用來蒐集、過濾、分析日誌及數據的開源工具，提供了各式各樣的日誌收集及各樣輸入、過濾及輸出的外掛模組 (Plugin)，任何類型的日誌幾乎都有支援 [21]，隨著版本的更新支援的檔案格式也更多樣化，並且能夠以多種方式輸出數據。Logstash 接收不同類型的日誌，例如系統日誌，Web 服務器日誌，錯誤日誌和應用程序日誌。這些通常分佈在使用不同格式的不同

系統中。在儲存到分析數據資料庫之前，Logstash 可幫助用戶將數據解析成一種通用格式，如 Figure 2.3。此外，Logstash 提供了一種通過提供自己定義邏輯來解析日誌。[19]。Logstash 服務會依設定值，過濾日誌的雜訊並搜尋出符合的條件。當找到符合的條件時，則啟動輸出將訊息送到另一個 Logstash agent 或輸出到資料庫 [20]。

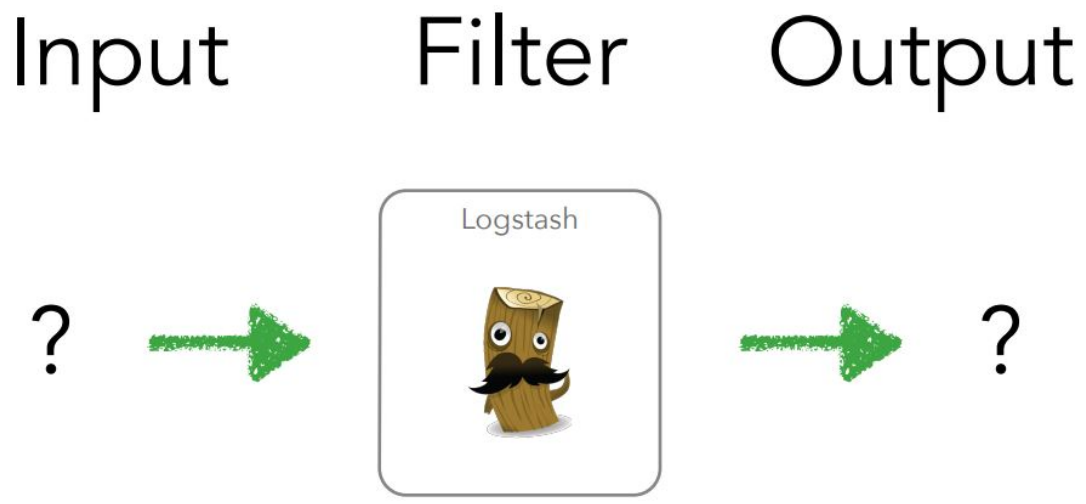


FIGURE 2.3: Logstash 輸入輸出示意圖

Log Analysis 日誌分析是理解日誌和提取有用信息的過程。其中一種開源日誌框架是由 Apache Software Foundation 在 Java 中開發的 Log4j [22]。它已經被用於記錄諸如時間戳，日誌級別，錯誤消息以及 IP 地址，用戶名稱和所請求的 URL 之類的資訊。一旦這些日誌被捕獲，然後進行日誌分析，就可以進一步對網站用戶上網行為進行分析與調查。

Log Collector 負責收集日誌，然後解析日誌，然後記錄 Windows 和 Linux 機器的事件。記錄、登錄、取消和安全相關事件的記錄。在安全日誌中，有助於為操作系統活動安全配置。從多個節點收集日誌後，對日誌進行格式正規化並發佈日誌，並且日誌收集器必須能夠處理大量，高速率和各種日誌 [24]。

Kibana 是一個開源的數據視覺化界面，用於即時彙總和繪製資料數據，起初 Kibana 有另一個名字叫 Elasticsearch Dashboard，隨著功能演進逐漸成為一個基於 Web 化圖形管理及資料呈現介面，可以用於搜索、分析及視覺化呈現存

放於 Elasticsearch Index(索引) 中的數據。利用 Elasticsearch REST API 來檢索數據，使用者可以依據自己的需求創造個人化的視覺儀表板，通過提供不同的視覺化效果，如長條圖，原餅圖，線條點和地圖如 Figure 2.4，幫助使用者方便理解大量的數據集。它還提供了可以自行定義儀表板中的呈現不同可視化 [23]。

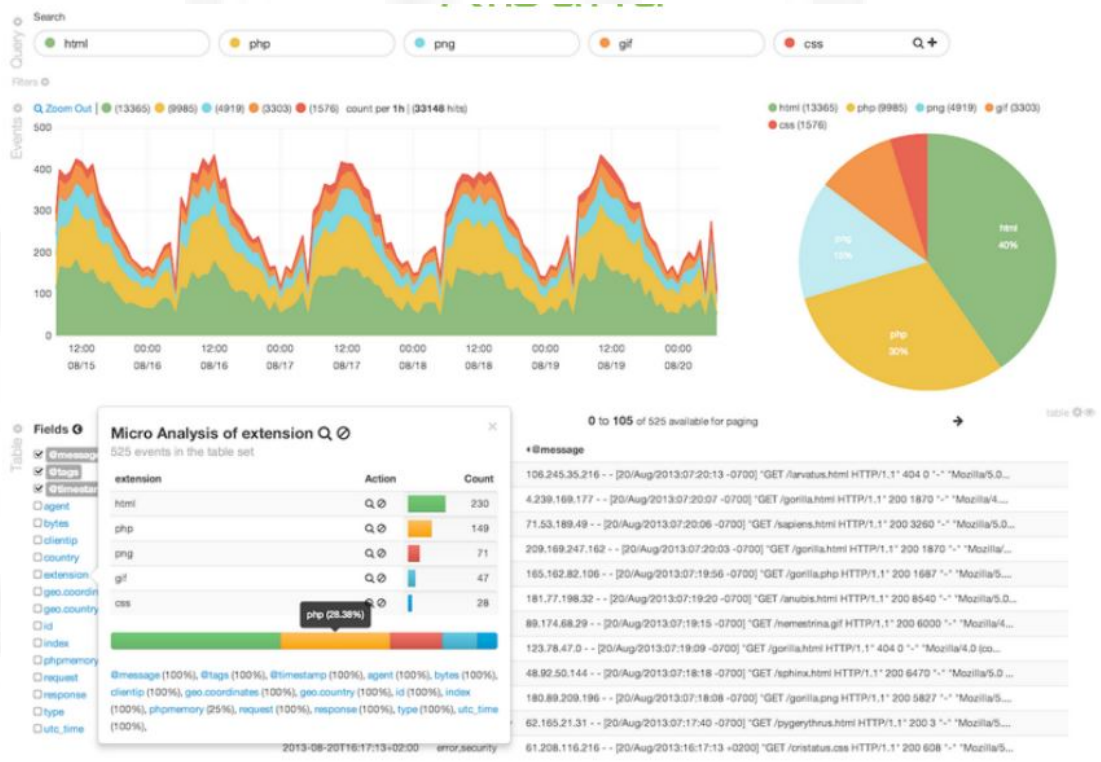


FIGURE 2.4: Kibana Dashboard 示意圖

2.1.4 交通資料蒐集支援系統

TDCS 是交通資料蒐集支援系統 (Traffic Data Collection System)，其資料庫資料來源為「高快速公路整體路網交通管理系統」與「高速公路電子收費系統」，彙整國道即時及歷史交通資訊，資訊內容是免費對外開放使用的，可由政府資料開放平台取得。國道高速公路局每年也利用這些資料，辦理資料應用創意競賽，每年都有不同的主題，主要透過創意比賽的手段，結合各界的共同參與，激盪出高速公路 ETC 資料在交通管理之創意發想與應用技術。 TDCS 產生之檔案清單及說明如下: [33]

- M03A: 各類車種通行量統計，依車輛種類通過單一偵測站統計之交通量，依每 5 分鐘為時階統計產出報表，示意圖如 Figure 2.5。

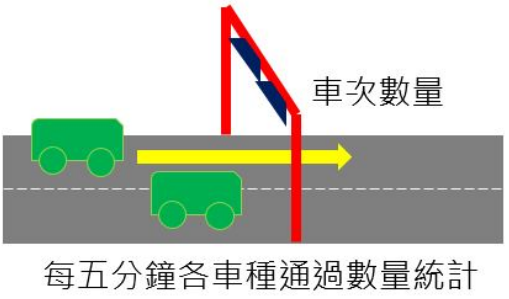


FIGURE 2.5: M03A 各類車種通行量統計示意圖

- M04A: 站間各車種平均旅行時間，依報表時階設定依各車種統計上下游偵測站之平均旅行時間，依每 5 分鐘為時階統計產出報表，示意圖如 Figure 2.6。

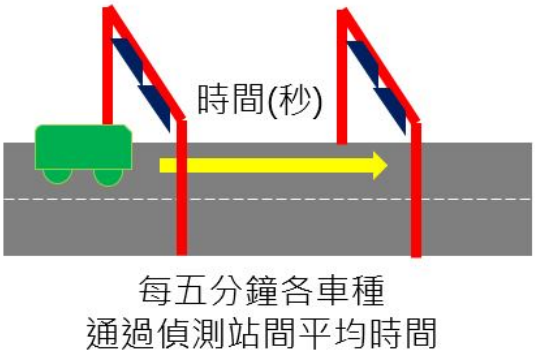


FIGURE 2.6: M04A 站間各車種平均旅行時間示意圖

- M05A: 站間各車種平均行駛車速，依報表時階設定依各車種統計上下游偵測站之平均旅行時間，依每 5 分鐘為時階統計產出報表，詳細內容及示意圖於第三章說明。

- M06A: 各旅次路徑原始資料，各旅次之完整行駛路徑資料，旅次係指一車輛自進入交流道到離開交流道稱為 1 個旅次。依每一小時為時階統計產出報表，系統於隔日 6 時以後開始計算前一日的所有旅次，並一次產生該日所有檔案，詳細內容及示意圖於第三章說明。

- M07A: 各類車種旅次平均長度，依報表時階設定統計各車種於各偵測站出發之平均旅行長度，計算依某偵測站為起點之旅次，示意圖如 Figure 2.7，依每一小時為時階統計產出報表，每天產製一份報表。

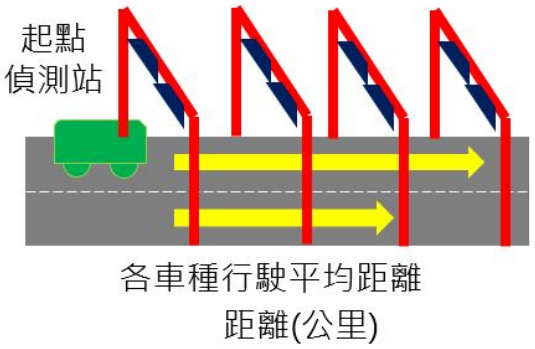


FIGURE 2.7: M07A 各類車種旅次平均長度示意圖

- M08A: 各類車種旅次數量，依報表時階設定統計各旅次起迄對之統計交通量，依每 5 分鐘為時階統計產出報表，每天產製一份報表，示意圖如 Figure 2.8。

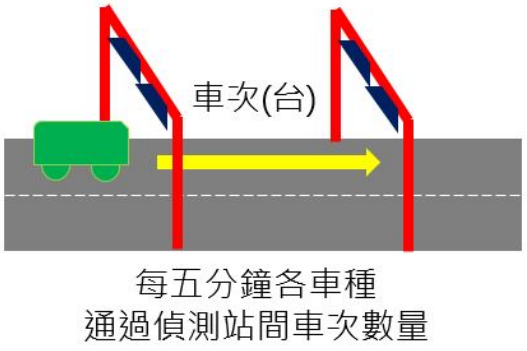


FIGURE 2.8: M08A 各類車種旅次數量示意圖

2.1.5 MariaDB

MariaDB 是從 MySQL 獨立出來的另一套衍生資料庫，而且都來自相同的創始人 Michael Widenius 和其他創始人，他們在 1994 年開始開發 MySQL，2008 年將 MySQL 賣給了昇陽電腦，傳為開源軟體商業化的經典案例。2 年後，MySQL 更推出大受歡迎的 5.5 版，但甲骨文卻收購了昇陽電腦。MySQL 二度

易主，MySQL 社群擔心甲骨文箝制而紛紛出走，Michael Widenius 因而推出了與 MySQL 相容的 MariaDB。MariaDB 的目的是完全相容 MySQL，包括 API 和命令列，使之能輕鬆成為 MySQL 的代替品。[25]

2.2 相關研究

隨著網路普及率的上升，網站瀏覽 log 交易量巨大，其 log 包含了巨大商業價值的隱藏信息，利用 ELK stack 開源軟體平台，可以有效地通過 log 來識別網站用戶流量，對這些 log 的分析不僅有助於公司的決策，也有助於改善他們的產品和服務。然而，大多數小規模公司都無力負擔昂貴的日誌分析管理系統，ELK stack 提供了一個開源的日誌分析管理系統平台 [26]。

除了使用 Elasticsearch、Logstash、Kibana 這三個開源軟體的搭配外，也可以 Apache Flume 來代理收集日誌事件，存放於 Hadoop ecosystem 中的 Hbase，然後 ElasticSearch 根據搜索條件獲取 Hbase 數據並呈現於 Kibana 視覺儀表板 [27]。

ELK stack 也應用於系統監視，透過收集系統中的資源信息，例如：CPU 使用率、存儲器使用量、磁盤 I/O 使用等)，以便用於故障識別。從科學的分佈式系統收集信息，透過 SVM 分類演算法來做為故障檢測方法，準確率高達 90% [28]。

網絡服務已經在社會，商業，政府和學術界等各個領域成為不可或缺的一部分。雲端技術的出現，使得放在雲端的應用程序也可通過 Web 界面進行訪問和控制。因此，網絡安全是非常重要且具有挑戰性的。例如常見的安全問題就是 SQL-injection。現有用於檢測這些攻擊的大多數解決方案，大多都使用日誌分析，利用模式匹配方法來檢測這些攻擊類型，而這個日誌分析系統就是由 ELK Stack 組合而成 [19]。

義大利國家核物理研究所 (INFN) 杜林分部計算中心的私有雲，為各種不同的科學計算應用提供 IaaS 服務。基礎服務架構是由 OpenNebula 所構成的 [29]，

除了追蹤資源使用情況之外，也需要為使用者動態分配資源，並對資源使用情況進行詳細的監控和計算。所以管理者建立了一個監控系統來檢查各節點活動，無論是 IaaS 還是託管虛擬機上運行的應用程序，這一個監控系統使用了 Elasticsearch，Logstash 和 Kibana 三個開源軟體。將不同的系統運作資訊送到不同的 MySQL 數據庫，並通過 Logstash 插件將資料送到 Elasticsearch，最後設置了一些預先定義查詢條件在 Kibana 儀表板上呈現，以便在每種情況下監控相關資訊。[30]。

Twitter 作為世界上用戶數最多的社交媒體之一，Twitter 提供了一個 API，使我們能夠即時觀看和使用 Twitter 數據。Elasticsearch 是一個有能力分析大數據的工具。有兩種方法可以將 Twitter 數據輸入到 Elasticsearch。第一個是通過 Twitter River，第二個是通過 Logstash 這個元件。輸入數據的準確性和效率以及數據的存儲方式對於支持大數據系統是非常重要的。在這篇文章中，介紹了從 Twitter API 中輸入 Twitter 數據的 Twitter River 和 Logstash 性能評估。這項研究監測兩個 HPC 服務器上的 Elasticsearch 叢集，它們同時從 Twitter API 抓取數據。比較參數是 CPU 進程，RAM 使用率，空間使用率，Twitter 輸入數據和輸入字段數量。這項研究的結果顯示，Twitter River 平均每天的 CPU 進程為 33.96%，Logstash 為 34.95%。Twitter River 平均每天的內存使用率為 32.7%，而 Logstash 則為 39.9%。此外，Twitter River 平均每天的空間使用量為 431 MB，Logstash 為 544 MB。對於 Twitter 輸入數據，Twitter River 在一周內輸入比 Logstash 多 191 條推文。結果表明，Logstash 在很多數值上比 Twitter 原生的 API 程式 Twitter River 不理想。[31]

隨著物聯網普及率的提高，大量的日誌文件持續在成長，其中包含具有巨大商業價值的隱藏信息。為了挖掘這些隱藏的價值，日誌管理系統有助於做出業務決策。現在有許多的日誌管理系統的解決方案，但是他們有些不能橫向擴展或成本高昂。ELK 生態系統，即 Elasticsearch，Logstash 和 Kibana 聚集在一起，有效地分析日誌文件，並提供交互式且易於理解的見解。建立在 ELK Stack 上的日誌管理系統需要分析大型日誌數據集，同時通過交互式界面使整個計算過程易於監控。從開源社區開始，ELK Stack 有許多有用的功能用於日誌分析。Elasticsearch 被用作索引，儲存和檢索引擎。當 Kibana 使用儀表板執行

數據可視化時，Logstash 充當日誌輸入切片器和切片機和輸出寫入器。通過實施 ELK 生態系統，我們有效地使用日誌對網站用戶流量進行地理標識 [32]。

Chapter 3

系統設計與實作

3.1 系統架構

3.1.1 MariaDB 與 Elasticsearch 查詢測試架構

MariaDB 資料庫是 MySQL 資料庫的一個分支，屬於關聯式資料的一種。Elasticsearch 資料庫則屬於分散式資料庫的一種，而這兩者就結構上來看其實有些類似，其對應的關係如 Table 3.1 。為了解單機運作的 Elasticsearch 資料庫與M ariaDB 資料庫在執行查詢的回應時間差異，所以設計了這個簡單的測試實驗，測試架構如 Figure 3.1，主要使用兩台虛擬機分別匯入相同筆數及內容的 M06A 資料，查詢資料表內含有特定偵測站代碼的筆數，並記錄其執行時間。

Table 3.1: MariaDB 與 Elasticsearch

	MarriaDB	Elasticsearch
資料庫	Databases	Indices
資料表	Tables	Types
列	Rows	Documents
欄位	Columns	Fields

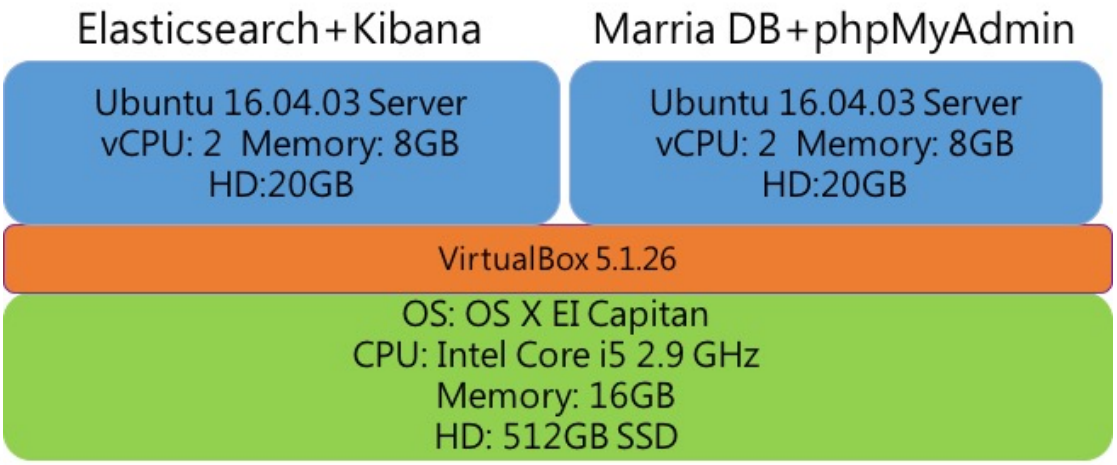


FIGURE 3.1: 資料庫查詢測試架構

3.1.2 分析系統架構

在文獻的探討內容中，可以發現 ELK Stack 多數應用於 log、硬體或系統的問題分析，然而卻很少應用於開放性資料分析或一般性資料的分析。所以此論文以 ELK Stack 為底層基礎架構，來分析即呈現國道電子收費系統資料，系統整合架構如 Figure 3.2。

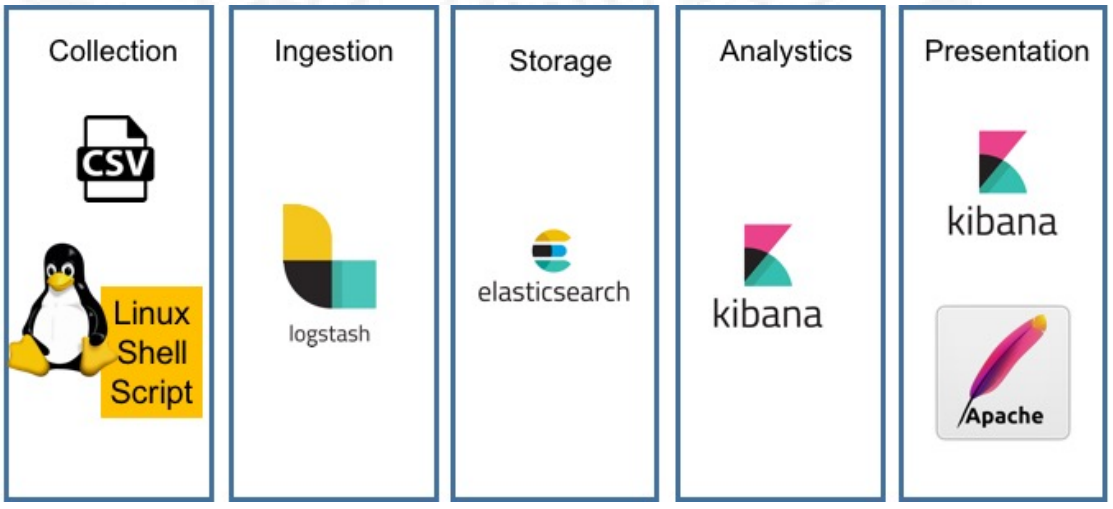


FIGURE 3.2: 資料視覺化系統架構圖

本架構作業流程圖如 Figure 3.3，主要利用建置在虛擬環境上虛擬機，透過排程執行 Linux shell script 定期定時取得政府開放資料平台資料，抓取頻

率分為兩種：M05A 按發佈時間每五分鐘抓取一次、M06A 則為早上 10:00 抓取，在抓取的資料的過程中同時進行資料的清洗及過濾。經過清洗後的資料由 Logstash 讀取，並依照 M05A 或 M06A 不同類別進行資料的過濾、分類、類型轉換及欄位處理，處理過後的資料依照類別輸出儲存於 Elasticsearch 的資料庫中，最後使用 Kibana 設定關鍵字及過濾條件，將分析過濾後的資料即時呈現於 Kibana Dashboard，最後利用 apache http server 呈現整合畫面。

主要以台中市內國道一號及國道三號交流道分析為主，透過這些分析來了解台中市的車都是由哪來的，又到哪去。也透過每五分鐘站間各車種平均行駛車速資料，來了解車速與車流量的對比關係，藉由 ELK Stack 這個開源解決方案，來實作與以往不同應用。

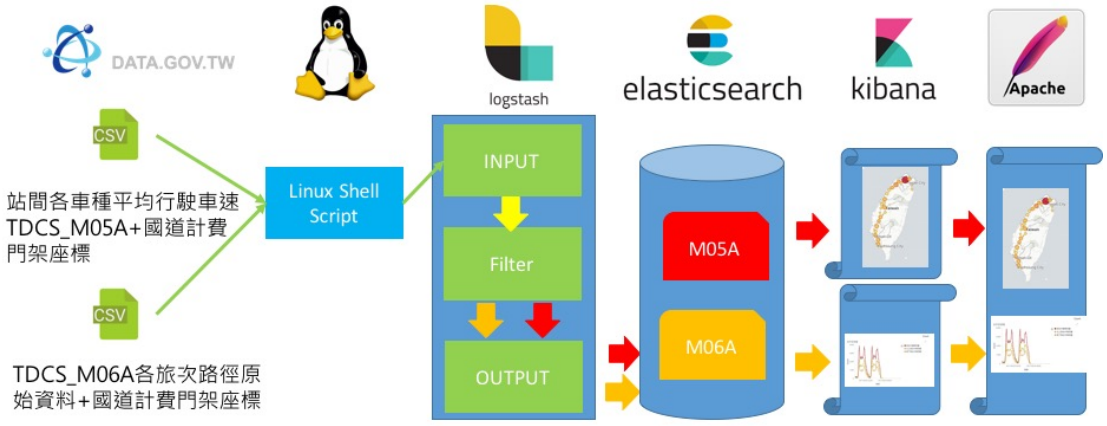


FIGURE 3.3: 資料視覺化系統流程示意圖

3.2 實驗方法

3.2.1 資料庫查詢測試

此實驗使用單機運作的 Elasticsearch 資料庫與 MariaDB 資料庫進行查詢執行時間比較，使用兩台虛擬機分別匯入相同筆數及內容，資料庫主要分為 Elasticsearch、MariaDB 沒有建立 index 與 MariaDB 有建立 index 三個類型，利用 kibana 與 phpMyAdmin 提供的查詢平台進行測試，語法為搜尋特定偵測站編號資料進行查詢，同一資料庫類型每次查詢的偵測站編號皆不相同，每

種資料量筆數皆執行十次查詢，曲執行時間的平均值為結果，資料量分別為 10 萬筆、50 萬筆、100 萬筆、500 萬筆及 1000 萬筆，語法及查詢結果示意如 Figure3.4與3.5。

FIGURE 3.4: MariaDB 查詢語法及結果

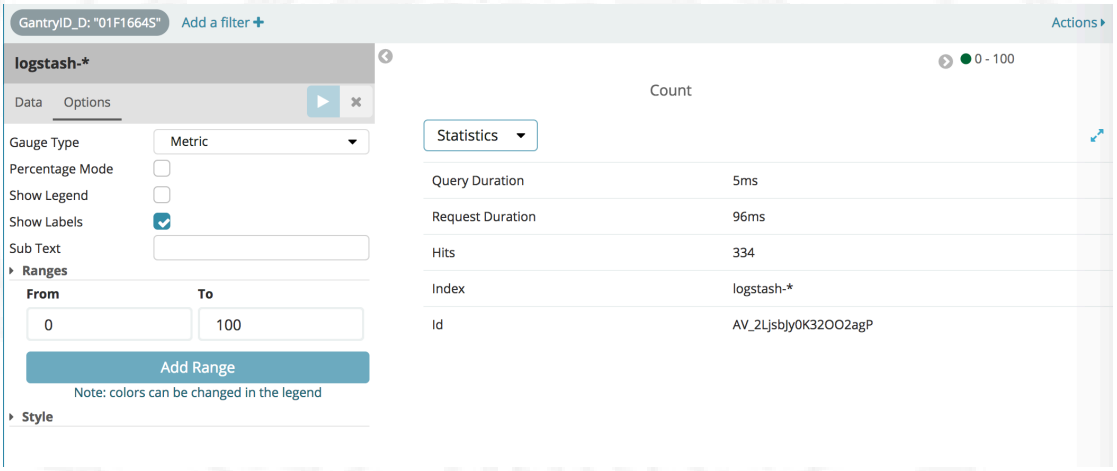


FIGURE 3.5: Elasticsearch 查詢結果

3.2.2 TDCS 資料取得及處理

此分析系統架構有別於以往 ELK Stack 單純應用於 syslog 分析，syslog 在網際網路工程任務組 (IETF) 的 RFC 5424 中定義，使標準化的系統得以生成、過濾、記錄和分析記錄檔訊息，也由於標準化藉此可減少開發人員程式設計及解讀上的困難。但此系統架構主要同時處理兩個不同格式的資料來源，並依資料來源不同進行不同的資料過濾並存放在不同類型的資料庫中，此部分有別於現有 syslog 分析。資料主要來源為政府開放資料平台內的 TDCS 交通資料庫，使用資料列表如下：

國道計費門架座標及里程牌價表 104.09.04 版

站間各車種平均行駛車速（M05A）

各旅次路徑原始資料 (M06A)

由於 TDCS 交通資料庫內的資料總共分成兩種類型檔案，一種是一個月內的資料，這種類型的資料主要是以逗點符號分隔的 csv 檔，其命名方式為 TDCS 與資料類型組合並加上時間為檔案名稱如 Table 3.2，另一種則為時間距今超過一個月的資料，這種類型則是先將原始資料的 csv 檔案及目錄階層架構資料打包成 tar 檔，再壓縮成 gzip。針對這兩種類型的檔案分別撰寫 Linux Shell Script 搭配排成定期抓取資料、處理指定日期的資料或處理已打包的壓縮檔，其應用示意圖如 Figure 3.6。

TABLE 3.2: TDCS 檔案名稱清單

項目	檔案名稱	說明
M03A	TDCS_M03A_YYYYMMDD_hhmmss.csv	各類車輛通行輛統計
M04A	TDCS_M04A_YYYYMMDD_hhmmss.csv	站間各車種平均旅行時間
M05A	TDCS_M05A_YYYYMMDD_hhmmss.csv	站間各車種平均行駛車速
M06A	TDCS_M06A_YYYYMMDD_hhmmss.csv	各旅次路徑原始資料 (日報)
M07A	TDCS_M07A_YYYYMMDD_hhmmss.csv	各車種旅次平均長度 (日報)
M08A	TDCS_M08A_YYYYMMDD_hhmmss.csv	各車種全日旅次交通量 (日報)

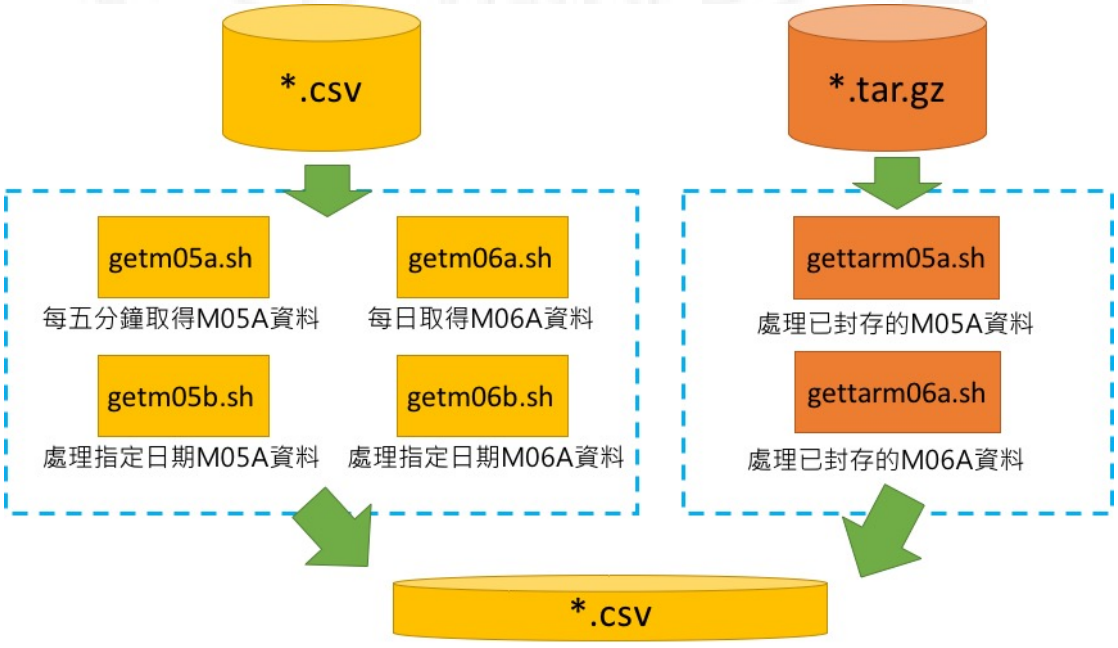


FIGURE 3.6: Linux shell Script 資料處理示意圖

國道計費門架座標及里程牌價表資料，主要記錄方向、編號、收費區設定里程、小車牌價、起迄交流道、緯度 (北緯)、經度 (東經)，經過清洗只選取偵測站編號、起迄交流道，並將緯度和經度合併為一個經緯度欄位資料，其示意如 Figure 3.7，透過這些資料就可以於視覺化介面上呈現地理位置，並將偵測站編號替換為較容易判讀的中文交流道資訊。

方向	編號	收費區設定里程	小車牌價	起訖交流道	起訖交流道	緯度代號	緯度（北緯）	經度代號	經度（東經）
N	01F-000.5N	1.1	1.3	基隆(長庚醫院)	基隆端(基隆港)	N	25.11831	E	121.7316
N	01F-001.7N	1.2	1.4	八堵	基隆(長庚醫院)	N	25.10957	E	121.7259
N	01F-002.9N	2.8	3.3	大華系統(連接台62)	八堵	N	25.10311	E	121.7174

⇩ 資料欄位篩選過濾及整理後

編號	起訖交流道	起訖交流道	經緯度
01F0005N	基隆(長庚醫院)	基隆端(基隆港)	25.11831, 121.7316
01F0017N	八堵	基隆(長庚醫院)	25.10957, 121.7259
01F0029N	大華系統(連接台62)	八堵	25.10311, 121.7174

FIGURE 3.7: 國道計費門架座標資料清洗示意圖

國道電子收費系統車種係依照高公局之定義，(1) 小型車包括：小客車、小貨車、小客貨兩用車、代用小客車、小型特種車及小型車附掛輕型拖車；(2) 大型車：包含大客車、大貨車、大客貨兩用車、代用大客車及大型特種車。(3) 聯結車：包含曳引車、兼供曳引大貨車、全聯結車及半聯結車。收費方式也依這三個類型來進行不同級距的收費，但依國道高速公路計程電子收費 TDCS 使用手冊 [33]，識別車種主要分為 5 種，分別為小客車、小貨車、大客車、大貨車及聯結車，其對應代號如 Figure3.8，這五種車輛主要應用於 TDCS 內各項資料的統計及識別。

TABLE 3.3: TDCS 各車種名稱列表

項目	車種代號	說明
1	31	小客車
2	32	小貨車
3	41	大客車
4	42	大貨車
5	5	聯結車

M05A 資料係指每 5 分鐘產出的一個報表，報表內容為五分鐘內各車種統計上下游偵測站之間的交通資訊，資訊包含主要報表產製時間 (TimeInterval)、上游偵測站編號 (GantryFrom)、下游偵測站編號 (GantryTo)、車種 (VehicleType)、平均車速 (SpaceMeanSpeed)、交通量，其單一車輛計算車速公式為相鄰上下游偵測站之旅行時間/門架距離 (單位:KM/Hour)，平均車速為每 5 分鐘之各車輛車速加總/交通量，其示意如 Figure3.8。將 M05A 資料進行整理，使用國道計費門架座標資料對應偵測站編號，在原始資料加入各偵測站編號之中文描述資訊及經緯度，其示意如 Figure3.9，並可利用經緯度在地圖上定位各偵測站間車速資訊。

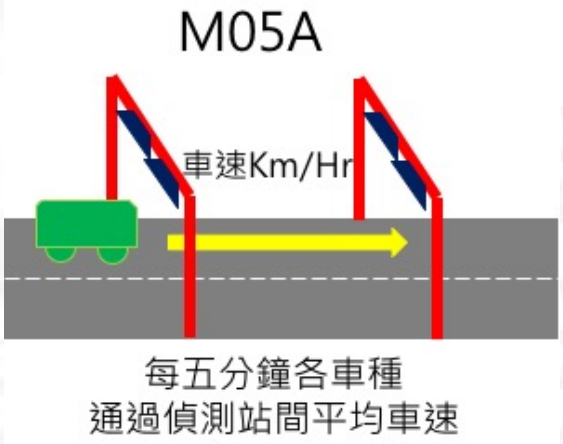


FIGURE 3.8: M05A 站間各車種平均行駛車速

報表產生時間	上游偵測站編號	下游偵測站編號	車種	平均車速	交通量
2017/8/29 15:45	01F0017N	01F0005N	31	92	47
2017/8/29 15:45	01F0017N	01F0005N	32	90	13
2017/8/29 15:45	01F0017N	01F0005N	41	90	3

⇩ 資料欄位篩選過濾及整理後

報表產生時間	上游偵測站編號	偵測站座標位置	偵測站區段名稱	下游偵測站編號	車種	平均車速	交通量
2017/8/29 15:45	01F0017N	25.10956667, 121.7259056	八堵-基隆(長庚醫院)	01F0005N	31	92	47
2017/8/29 15:45	01F0017N	25.10956667, 121.7259056	八堵-基隆(長庚醫院)	01F0005N	32	90	13
2017/8/29 15:45	01F0017N	25.10956667, 121.7259056	八堵-基隆(長庚醫院)	01F0005N	41	90	3

FIGURE 3.9: M05A 資料欄位處理示意圖

M06A 每天早上 6:30 產製前一天資料，每一小時為一個報表，此資料記錄各旅次之完整旅次係指一車輛自進入交流道到離開交流道稱為 1 個旅次，行駛路徑資料包含車種 (VehicleType)、車輛通過本旅次第 1 個偵測站時間 (DetectionTime_O)、車輛通過本旅次第 1 個偵測站編號 (GantryID_O)、車輛通過本旅次最後 1 個偵測站時間 (DetectionTime_D)、車輛通過本旅次最後 1 個偵測站編號 (GantryID_D)、本旅次之行駛距離 (TripLength)、旅次結標記 (TripEnd)、本旅次經過的各個偵測站之通過時間及偵測站編號 (TripInformation)，其示意如 Figure 3.10。將 M06A 欄位進行整理，使用國道計費門架座標資料對映偵測站編號，在原始資料加入各偵測站編號之中文描述資訊及經緯度，其示意如 Figure 3.11。

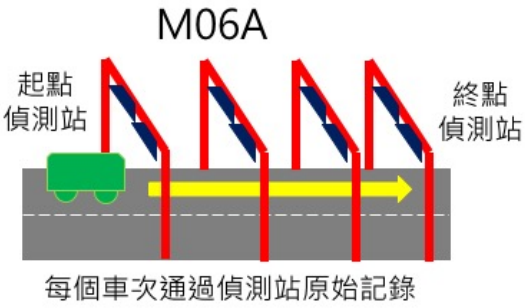


Figure 3.10: M06A 各旅次路徑原始資料

車種	第一個偵測站經過時間	第一個偵測站編號	最後一個偵測站經過時間	最後一個偵測站編號	行駛距離	是否正常結束	旅程紀錄
31	2017/8/28 22:49	03F0158S	2017/8/28 22:50	05F0000S	5.2	Y	2017-08-28 22:49:15+03F0158S;2017-08-28 22:50:02+05F0000S
41	2017/8/28 22:47	01F0005S	2017/8/28 22:47	01F0005S	1.1	Y	2017-08-28 22:47:30+01F0005S
32	2017/8/28 22:46	03F0337S	2017/8/28 22:46	03F0337S	4.8	Y	2017-08-28 22:46:29+03F0337S

資料欄位篩選過濾及整理後

車種	車種中文	第一個偵測站經過時間	第一個偵測站編號	第一個偵測站座標位置	第一個偵測站交流道名稱	最後一個偵測站經過時間	最後一個偵測站編號	最後一個偵測站座標位置	最後一個偵測站交流道名稱	行駛距離
31	小客車	2017/8/28 22:49	03F0158S	25.036533, 121.621386	南港(連接環東大道)	2017/8/28 22:50	05F0000S	25.03508889, 121.6229306	石碇	5.2
41	大客車	2017/8/28 22:47	01F0005S	25.11878611, 121.7317639	基隆端(基隆港)	2017/8/28 22:47	01F0005S	25.11878611, 121.7317639	基隆(長庚醫院)	1.1
32	小貨車	2017/8/28 22:46	03F0337S	24.978881 ,121.502381	安坑(中央、安康路)	2017/8/28 22:46	03F0337S	24.978881 ,121.502381	中和(連接台64)	4.8

Figure 3.11: M06A 資料欄位處理示意圖

此實驗主要使用政府資料開放平台的交通資料庫 M05A 與 M06A 兩種資料，其資料格式有 csv 與 tar.gz 兩種，而依照取得資料的方式可以分為系統排程取得、取得指定日期即取得已封存的檔案共三種方式，所以如 Figure 3.6所示，共撰寫了六個 Linux Shell Script，其用途主要是從政府資料開放平台的交通資料庫取得資料，然後保留需要的欄位，並加入較容易識別的中文描述與座標等資訊，最後再將處理過後的資料移動到指定目錄。

首先是 getm05a.sh，主要設定於排程中每五分鐘取得 M05A 資料並進行處理，首先利用 date 指令取得現在的時間，配合檔案命名的規則分別計算出符合格式的年月日時分，最後再與固定的網路目錄路徑結合在一起，因為交通部資料庫 M05A 報表產製時間雖然為五分鐘為一次，但偶爾會發生時間差的問題，所以先利用 curl 指令確認檔案是否存在，若不存在則等待一分鐘後再測試直到檔案存在，檔案取得後利用 awk 保留需要的欄位，並將他寫入暫存檔中，最後利用國道計費門架座標資料加上 sed 指令將檔案內的座標名稱加入中文名稱及座標等數值，整個 Shell Script 流程如 Figure 3.12。

getm05b.sh 主要是下載指定日期的 M05A 檔案，而這些日期的資料通常是一整日的資料而且是尚未封存的，所以需要一一到最底層的路徑取得檔案，所以一開始依據執行的指令取得日期參數，再利用 for 迴圈產生 01 至 23 的小時參數，其中還包含了一個 for 迴圈，這個迴圈主要是產生每 5 分鐘的間隔從 00,05,10,15 一直到 55，利用這兩個迴圈產生的參數設定為檔案目錄路徑及檔案名稱，依序取得指定日期內的所有檔案，每下載一個檔案就利用 sed 指令將檔案內的座標名稱加入中文名稱及座標等數值，並存放到一個暫存檔，當指定日期整日的檔案都處理完成後，再將這個檔案移動到 logstash 讀取的目錄，整個 Shell Script 流程如 Figure 3.13。

gettarm05a.sh 則是針對距離現在已超過一個月的封存資料進行處理，一開始在執行時也是需要給予指定日期，程式在取得指定日期後，再利用這個日期產生下載的目錄位置，檔案下載完成都會是一個副檔名為 tar.gz 的檔案，所以需要解壓縮，利在在解壓縮的過程當中，把解壓縮後的檔案名稱及路徑讀取到一個暫存列表，後續我們依這個列表一一讀取檔案並利用 sed 指令將檔案內的

座標名稱加入中文名稱及座標等數值，並存放到一個暫存檔，當列表內的檔案都處理完成後，再將這個檔案移動到 logstash 讀取的目錄，並刪除產生的所有暫存檔及壓縮檔，整個 Shell Script 流程如 Figure 3.16。

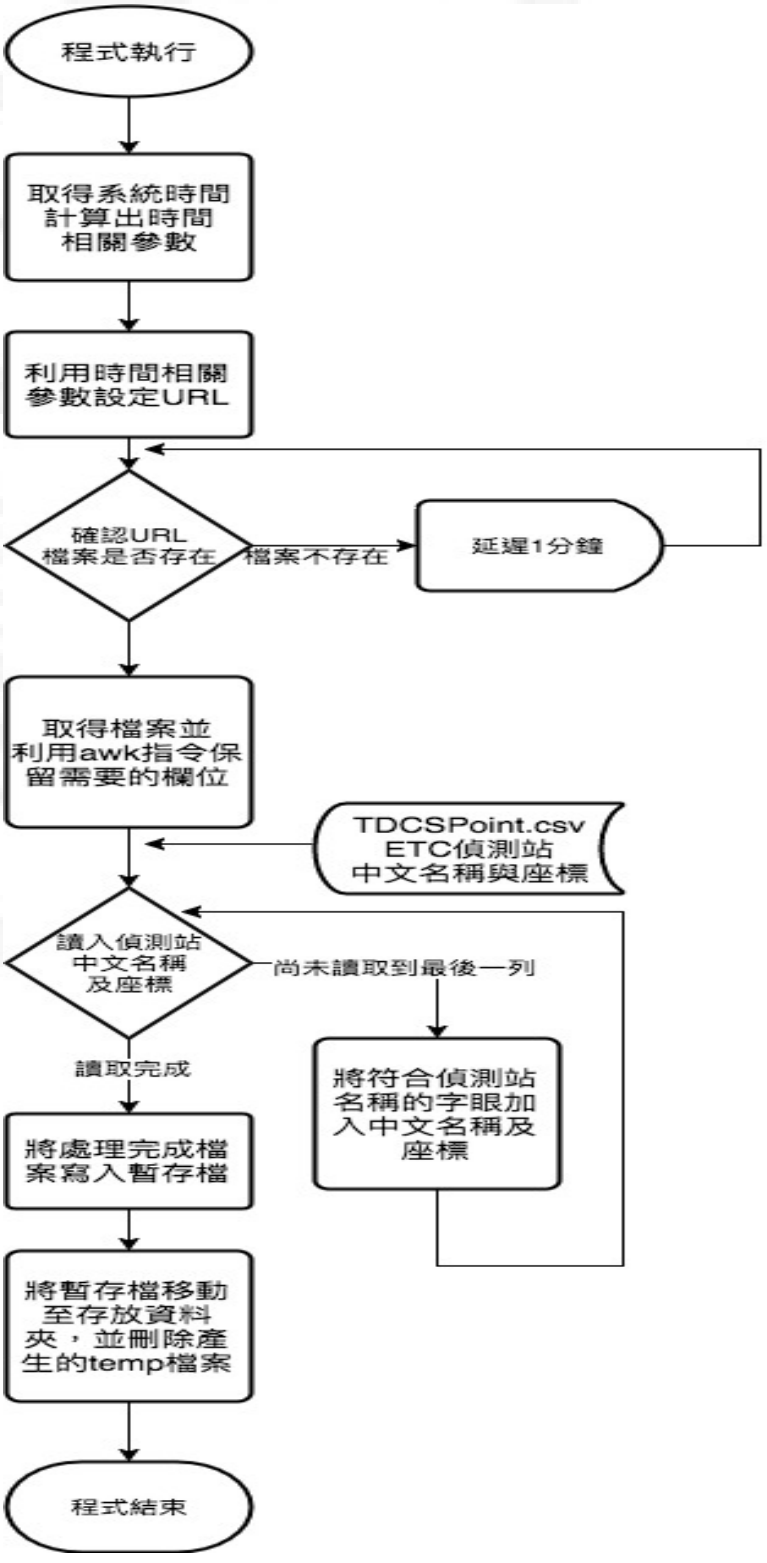


FIGURE 3.12: 每五分鐘取得 M05A 資料 getm05a.sh 流程示意圖

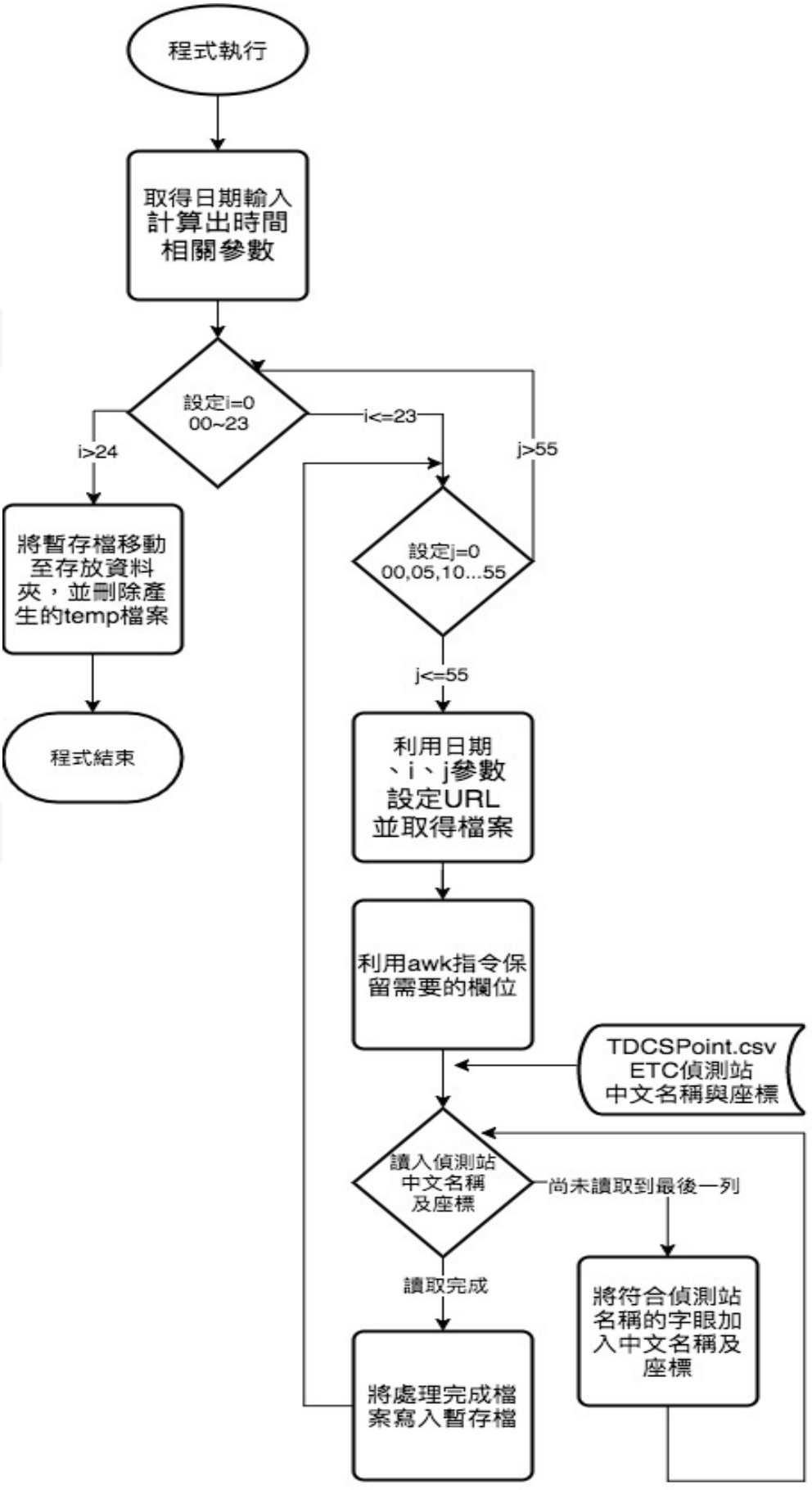


FIGURE 3.13: 處理指定日期 M05A 資料 getm05b.sh 流程示意圖

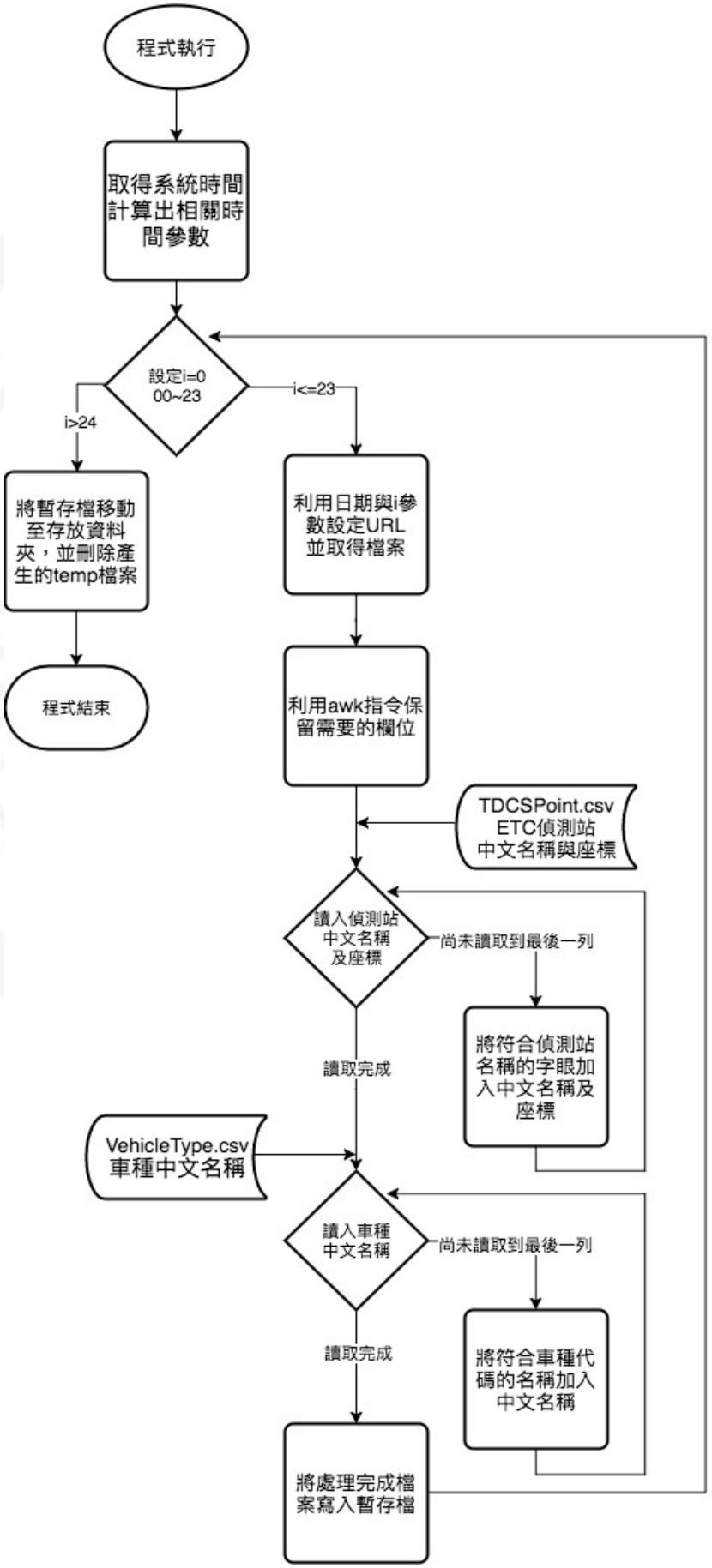


FIGURE 3.14: 每日取得 M06A 資料 getm06a.sh 流程示意圖

getm06a.sh，主要設定於排程中每天早上 10:00 取得 M06A 資料並進行處理，雖然交通部資料庫產製的時間為上午 6:30，但經過資料處理到上傳完成也是需要一段時間。程式首先利用 date 指令取得現在的時間，再利用 for 迴圈產生 00 至 23 的小時參數，和 getm05a.sh 一樣配合檔案命名的規則分別計算出符合格式的年月日時分，最後再與固定的網路目錄路徑結合在一起，因為 M06A 是一個小時一個檔案，所以也利用剛才 for 迴圈一一下載檔案，每一次下載檔案後就立即進行處理，將檔案後利用 awk 保留需要的欄位，並將他寫入暫存檔中，利用國道計費門架座標資料加上 sed 指令將檔案內的座標名稱加入中文名稱及座標等數值，由於國道計費門架座標資料中並沒有車種的對應資訊，所以在一次利用車種對應資訊加上 sed 指令將檔案內的車種代表加上中文描述等資訊，處理完成的資料都寫入到另一個暫存檔，當 24 小時內的檔案都處理完成後，再將這個檔案移動到 logstash 讀取的目錄，並刪除產生的所有暫存檔及壓縮檔，整個 Shell Script 流程如 Figure 3.14。

getm06b.sh 主要是用於下載及處理指定日期的 M06A 檔案，程式流程與 getm06a.sh 是一致的，主要差別只在於日期參數是由系統自動產生還是執行指令時所指定的，整個 Shell Script 流程如 Figure 3.15。

gettarm06a.sh 則是針對距離現在已超過一個月的 M06A 封存資料進行處理，一開始在執行時也是需要給予指定日期，程式在取得指定日期後，再利用這個日期產生下載的目錄位置，檔案下載完成都會是一個副檔名為 tar.gz 的檔案，所以需要解壓縮，利在在解壓縮的過程當中，把解壓縮後的檔案名稱及路徑讀取到一個暫存列表，後續我們依這個列表一一讀取檔案，將檔案後利用 awk 保留需要的欄位，並將他寫入暫存檔中，利用國道計費門架座標資料加上 sed 指令將檔案內的座標名稱加入中文名稱及座標等數值，最後利用車種對應資訊加上 sed 指令將檔案內的車種代表加上中文描述等資訊，處理完成的資料都寫入到另一個暫存檔，當列表內的檔案都處理完成後，再將這個檔案移動到 logstash 讀取的目錄，並刪除產生的所有暫存檔及壓縮檔，整個 Shell Script 流程如 Figure 3.17。

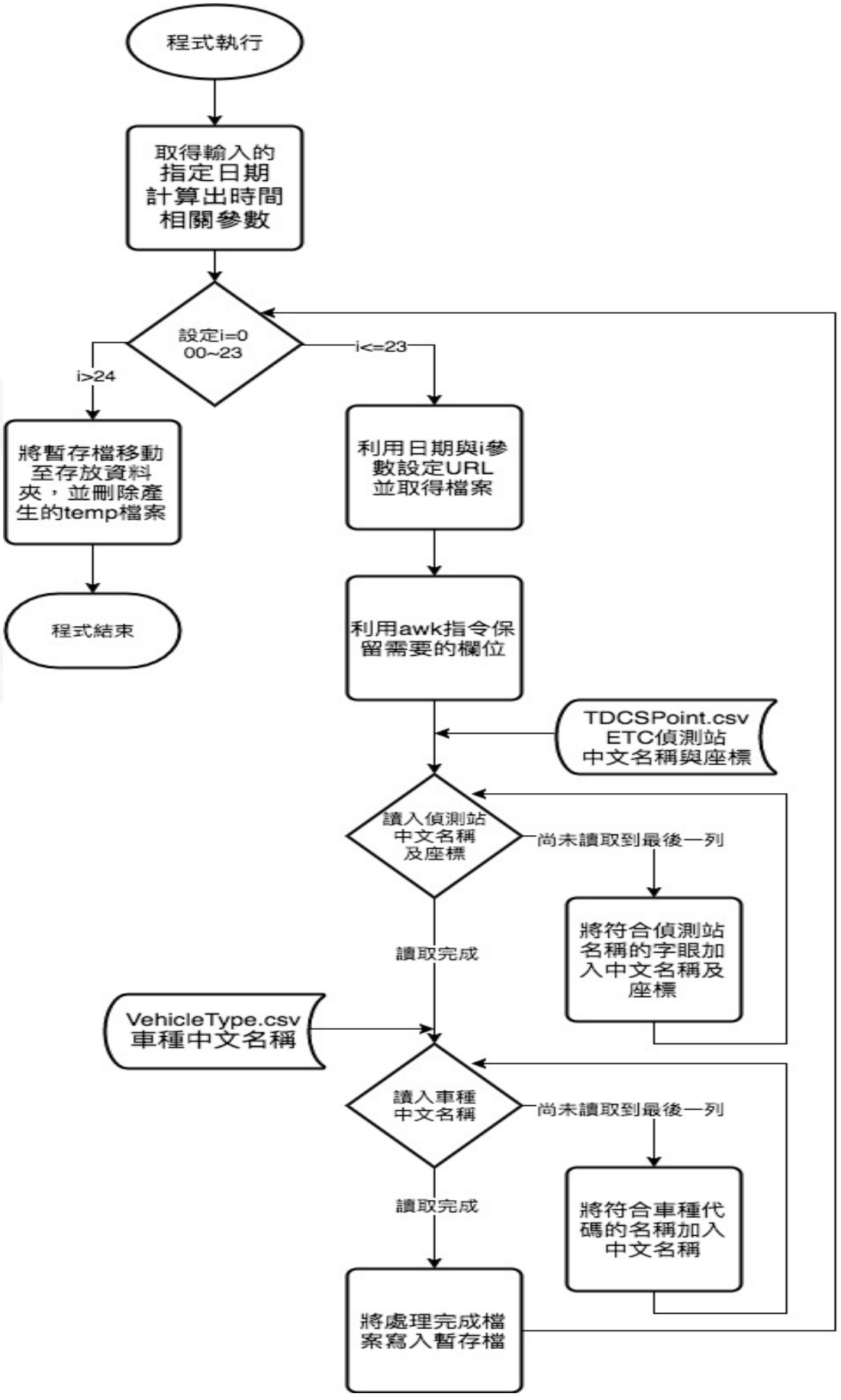


FIGURE 3.15: 處理指定日期 M06A 資料 getm06b.sh 流程示意圖

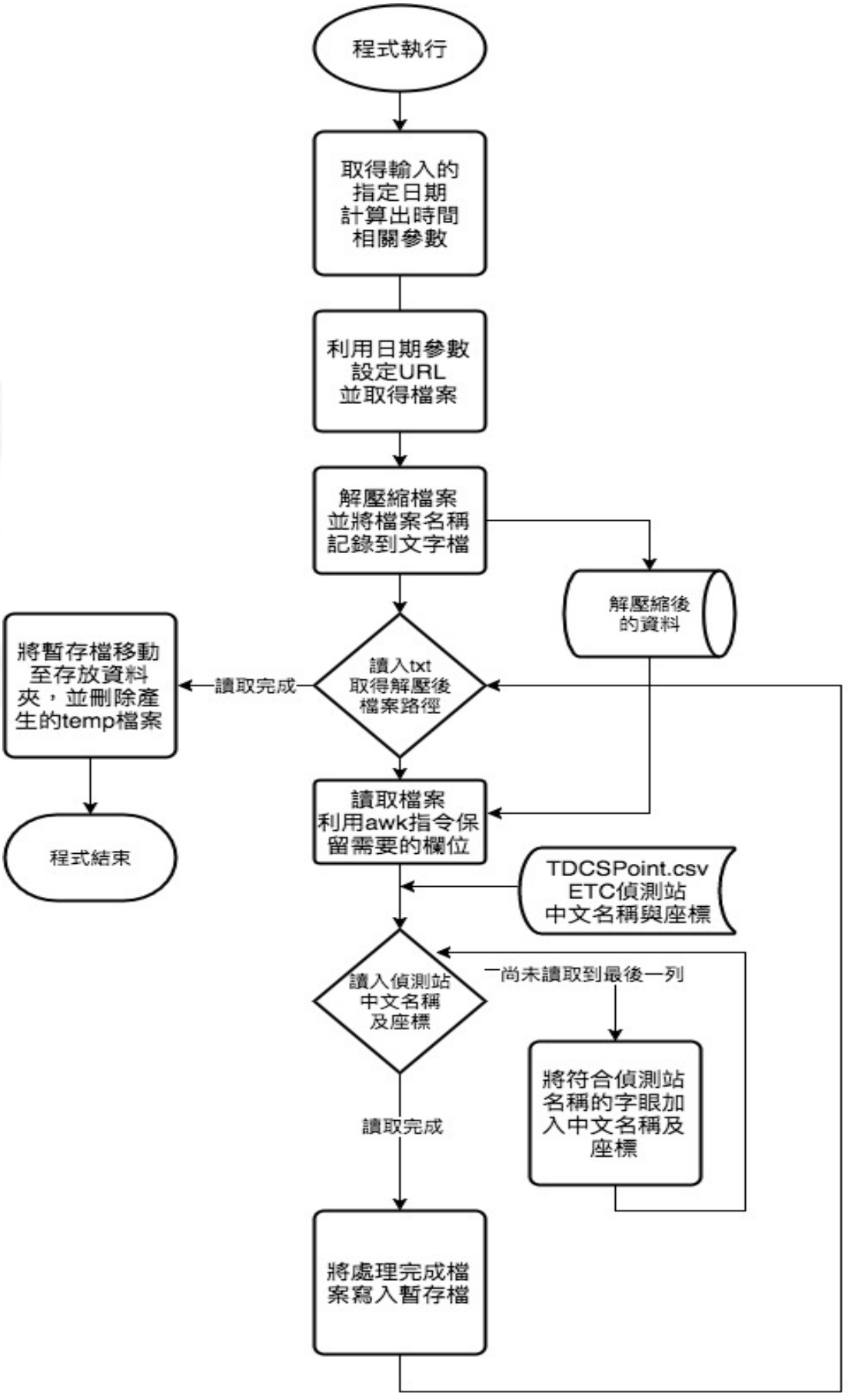


FIGURE 3.16: 處理已封存的 M05A 資料 gettarm05a.sh 流程示意圖

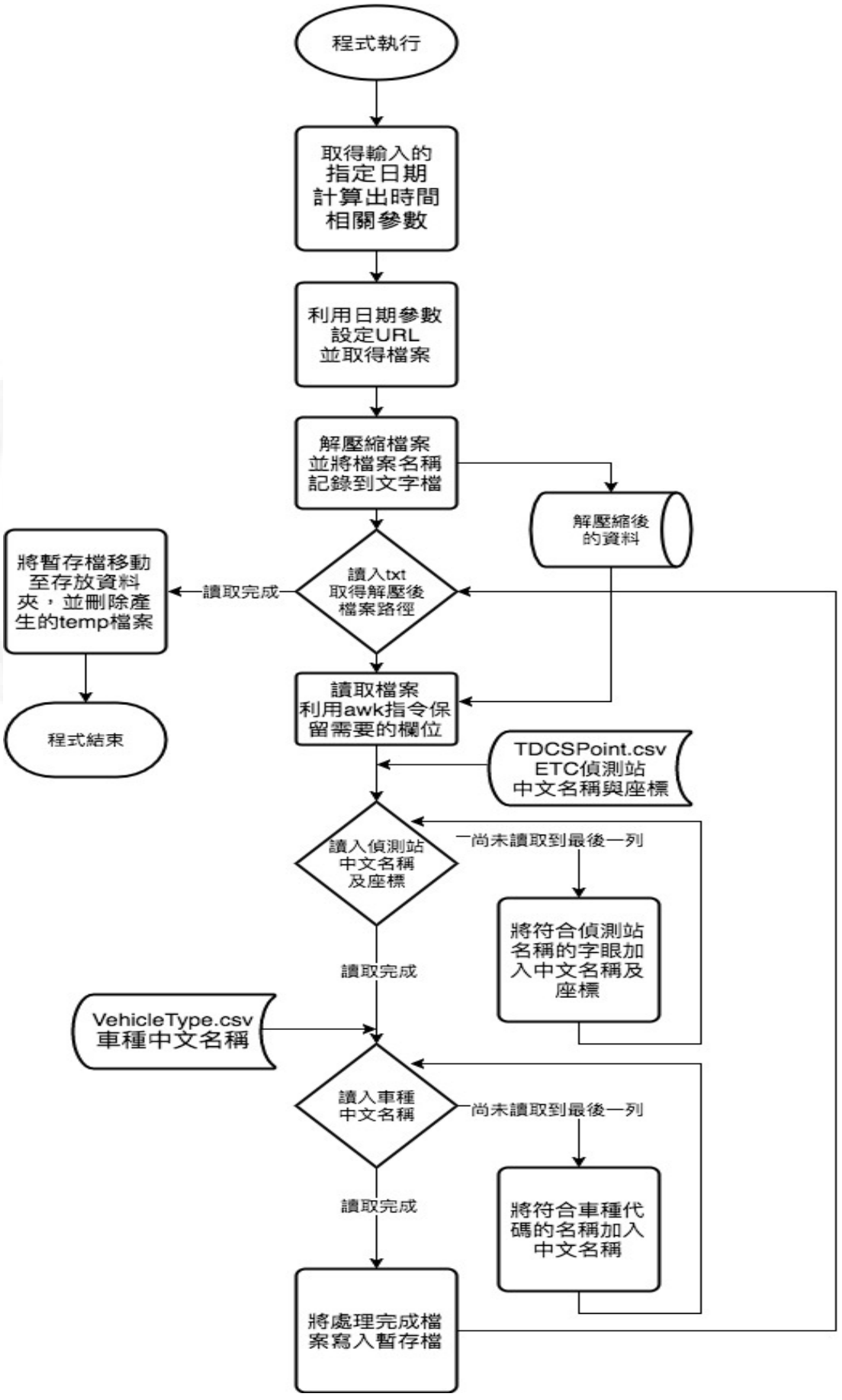


FIGURE 3.17: 處理已封存的 M06A 資料 gettarm06a.sh 流程示意圖

3.2.3 ELK 資料處理流程

經過清洗及處理整理後的資料，由於資料主要分為兩大類為每五分鐘更新一次的 M05A 及每天 6:30 批次產生的 M06A 兩種，呈現方式也有所不同，M05A 主要以地圖方式呈現各區段車速及分析，M06A 主要應用於各項分析，所以在資料的處理最後，都會將處理完成檔案移動到指定位置，Logstash 透過讀入資料的位置來加增資料類型為 M05A 或 M06A 的標籤，所 input 階段藉由讀取位置不同加上類別標籤，接下來 Filter 階段依類別標籤執行不同的欄位對應設定及欄位屬性宣告，透過欄位指定方式，將預設 @timestamp 欄位資料更新為偵測站觸發時間，@timestamp 欄位為 Kibana 時間軸主要依據，若不進行資料替換會造成時間不一致的問題，另外根據讀入的欄位將車流量、車速及車行距離等資料轉換為數值，因為預設 logstash 讀入的資料都會將它視為字串，為了能讓分析順利，所以需要進行這些設定，最後 Output 階段依類別標籤將資料傳輸到 Elasticsearch 對應的資料表位置，而 lindex 則是由資料類型與時間當為名稱，方便日後的管理，其示意如 Figure 3.18。

Elasticsearch 我們採用單機運作的架構來實驗，並依 logstash 所要求的類別標籤加上日期標籤為名稱，每天將輸入的資料分別存放對應資料表。架構最後為 Kibana 主要用來呈現視覺化圖表。每五分鐘偵測站間車速熱感圖，利用篩選類別標籤為 M05A，建立 Coordinate Maps，以偵測站座標位置標示地圖位置，以各車種平均車速數據顯示，此圖表可以快速了解全台車速狀態，當滑鼠移到偵測站位置顯示對應車速資訊，除此之外利用長期收集到 M05A 資料來統計分析車流量與車速的對比關係。而 M06A 資料主要以台中交流道做為主要分析點，利用篩選類別標籤為 M06A，建立線性圖表、圓餅圖等來分析交流道車流量及車輛來源及目的地，並依時間區段顯示對應資料，再透過得到的結果在進行車流的行為分析，另外比較平常日與假日每小時車流的差異，以及了解國道多人上下的交流道位置。

由於在實驗階段所以並未加入安全性的機制，但仍利用 apache http server 將 kibana 產生的圖表利用 ifeame 語法坎入到網頁中，這樣可以更快提供相關

性的畫面給使用者，並模擬對網頁讀取的控管，減少不必要的人員操作 Kibana console 介面。

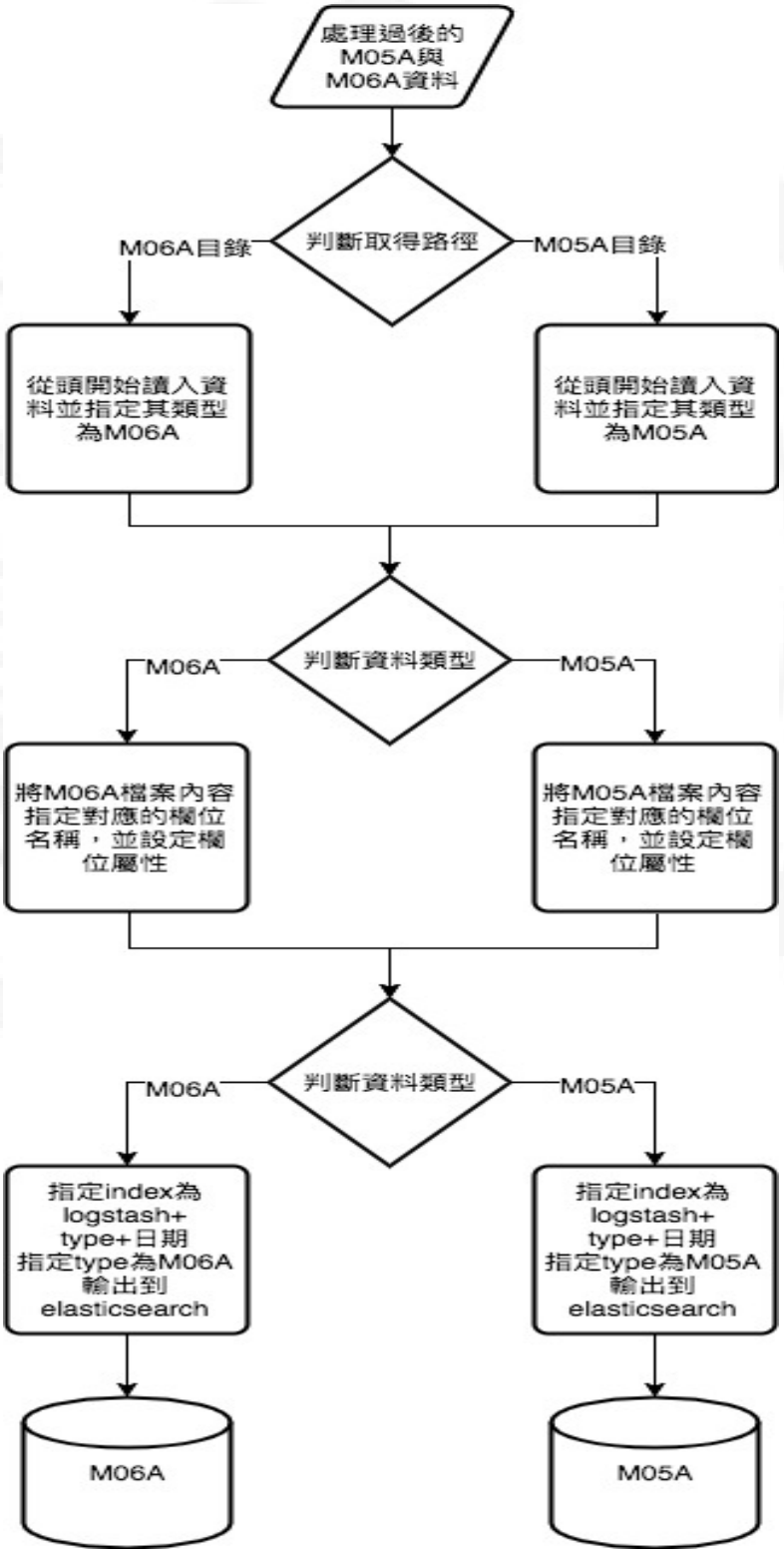


FIGURE 3.18: logstash configuration 資料處理流程示意圖

Chapter 4

實驗環境與結果

4.1 實驗環境

第一個實驗環境主要是以筆記型電腦搭配虛擬化軟體，以虛擬機建置實驗環境，進行資料庫查詢測試，其軟硬體系統及套件版本如 Table 4.1 至4.3。第二個實驗為分析系統整合架構建置，由於需處理大量的資料寫入及運算，故使用桌上型電腦來進行，其軟硬體系統及套件版本如 Table 4.4。

Table 4.1: 資料庫查詢測試實體機規格表

項目	規格
處理器	Intel Core i5 2.9 GHz
記憶體	16GB
硬碟	SSD 512GB
作業系統	OS X EI Capitan
虛擬化軟體	VirtualBox 5.1.26

Table 4.2: Elasticsearch 虛擬機規格表

項目	規格
虛擬機核心數	2 vCPU
虛擬機記憶體	8GB
虛擬機作業系統	Ubuntu server 16.04.03
Elasticsearch	5.5.1
Logstash	5.5.1
Kibana	5.5.1
Java Version	OpenJDK 1.8.0.131

Table 4.3: MariaDB 虛擬機規格表

項目	規格
虛擬機核心數	2 vCPU
虛擬機記憶體	8GB
虛擬機作業系統	Ubuntu server 16.04.03
MariaDB	10.0.31
Apache	2.4.18
PHP	7.0.22
phpMyAdmin	4.5.4

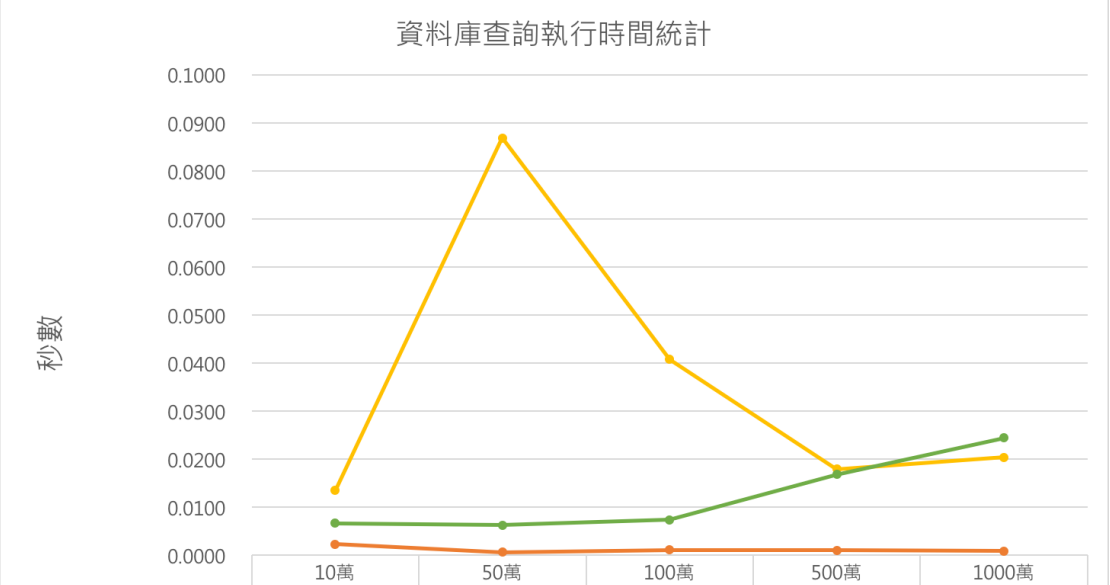
Table 4.4: 分析系統整合架構實驗環境規格表

項目	規格
處理器	Intel Core i7-7700 3.6 GHz
記憶體	16GB
硬碟	SATA 1TB
虛擬化軟體	vSphere 6.5
虛擬機核心數	8 vCPU
虛擬機記憶體	12GB
虛擬機作業系統	Ubuntu server 16.04.03
Elasticsearch	6.1.1
Logstash	6.1.1
Kibana	6.1.1
Java Version	OpenJDK 1.8.0.151

4.2 實驗結果

4.2.1 資料庫查詢執行時間結果

實驗結果主要以 Excel 試算表統計結果如 Figure 4.1，從統計結果可以發現 MariaDB 有沒有建立 index 查詢執行時間差異很大。另外從實驗中發現，MariaDB 在沒有建立 index 情形下，若查詢的數值剛好位於排序後的資料表最下方，其執行時間就會拉長，例如資料量為 50 萬筆時，有兩筆查詢的數值都位於資料表最後端，故執行時間相對較長。另外可以發現 Elasticsearch 資料庫查詢時間優於沒有建立 index 的 MariaDB，但隨著資料量的成長，查詢時間也會線性成長。



	10萬	50萬	100萬	500萬	1000萬
MariaDB index	0.0023	0.0006	0.0011	0.0010	0.0009
MariaDB non-index	0.0135	0.0868	0.0407	0.0179	0.0203
Elasticsearch	0.0066	0.0062	0.0073	0.0168	0.0244

FIGURE 4.1: 資料庫查詢執行時間統計圖

4.2.2 ELK Stack 圖形化控制台

在 ELK Stack 三個套件安裝完成後，可以透過瀏覽器開啟 Kibanan 所提供的圖形化控制頁面，開啟網址進入後首先會看到全文檢索結果，點選 Discover 會依照上方時間區段列出所有符合條件的結果，左上方顯示符合的條件筆數，左方可以指定要顯示的欄位名稱，下面則顯示符合條件的時間及資料內容，透過此介面可以執行資料搜尋如 Figure 4.2。

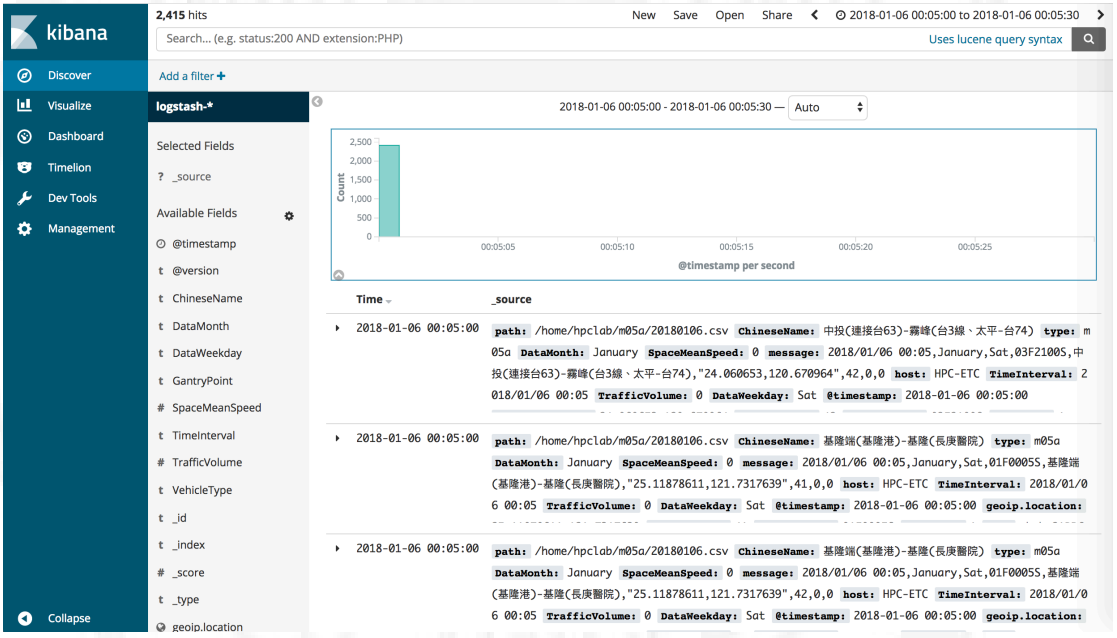


FIGURE 4.2: 全文檢索結果畫面

Kibana 右上可以選擇要搜尋的時間區段，這一個區段主要是以 elasticsearch 中的@ timestamp 為依據，而這個欄位預設是紀錄 elasticsearch 寫入資料庫的時間，而在我們的分析系統中，特別將此欄位改成偵測站觸發的時間。Kibana 預設的時間工具列可以利用以現在時間為主往前計算，或著選擇特定的時間點如 Figure 4.3與 Figure 4.4。

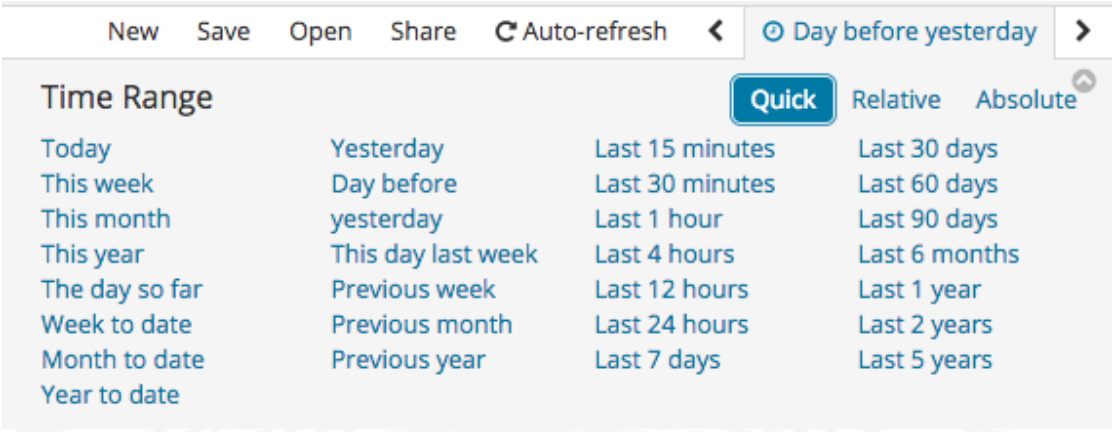


FIGURE 4.3: 快選時間工具列

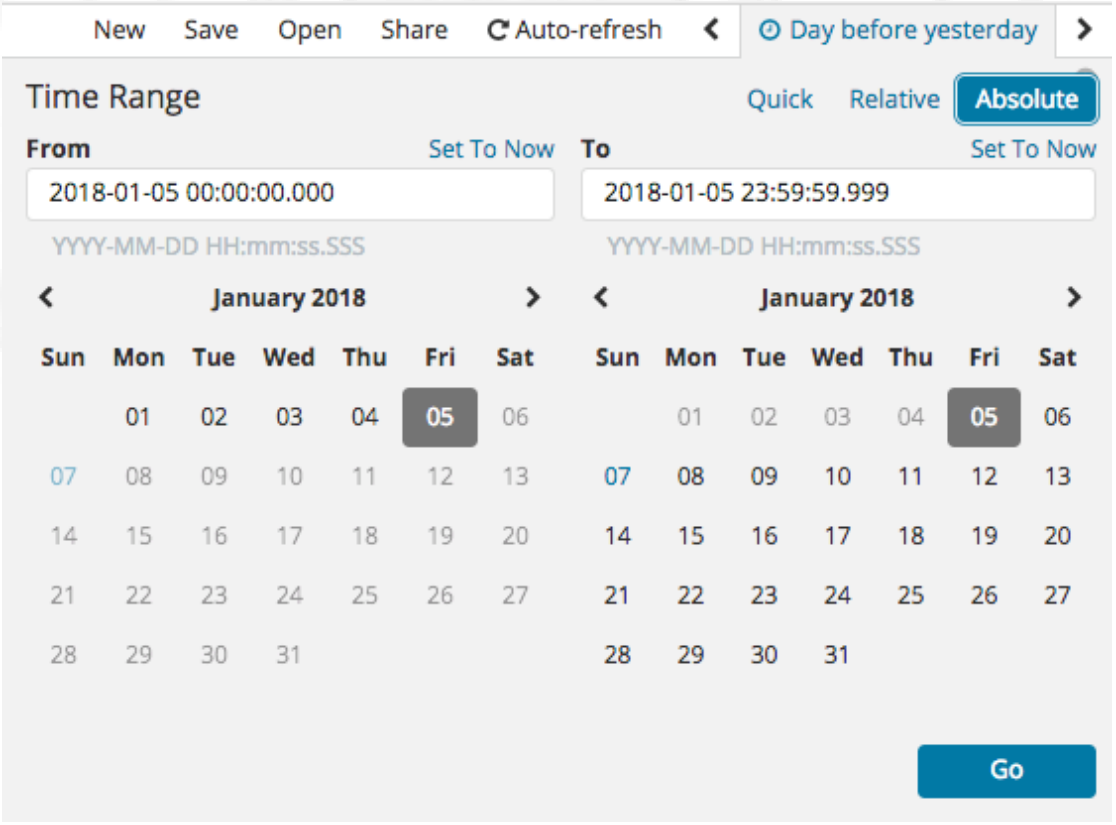


FIGURE 4.4: 指定時間工具列

Visualize 主要可以利用範本來建立視覺化圖表如 Figure 4.5，這些建立的圖表可以置放於 Dashboard 上呈現。預設 Visualize 包含了折線圖、直條圖、圓餅圖、資料表、數值百分比及地圖等，透過欄位的過濾、篩選及設定，就可以產生對應的圖表。

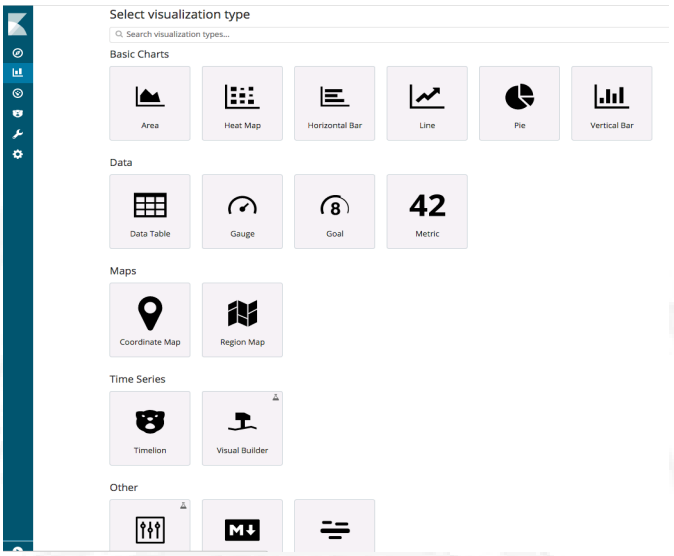


FIGURE 4.5: Visualize 畫面

Dashboard 主要將已建立的 Visualize 圖表集中呈現在同一個頁面，可以依需求呈現不同的圖表集合如 Figure 4.6。

FIGURE 4.6: Dashboard 畫面

4.2.3 M05A 每五分鐘 ETC 偵測站間車速

實驗結果主要以 Kibana 視覺化圖表呈現，可利用內建 Dashboard 將同一類型圖表聚集在同一個頁面觀看。第一個圖表顯示每五分鐘 ETC 偵測站間車速熱感圖，這個圖表利用偵測站間平均車速資料呈現，透過指標移動到地圖對應位置可以顯示平均車速資訊如 Figure 4.7，圖示紅色代表車速超過 110KM，橘色代表車速在 100KM，顏色越淺表示車速越慢，黃色代表車速約 90KM，綠色代表車速約 80KM，顏色越淡車速越慢。也可以指定特定的時間來顯示歷史資料如 Figure 4.8。使用地圖方式呈現車速，並以數值來表示，其資訊較容易判讀，與高公局 1968 APP 以文字描述位置及使用車速級距不同，因為在國道上每個車種車速限制是不同的，透過車種的篩選更容易掌握各車種行駛速度，提供簡單明瞭的介面。

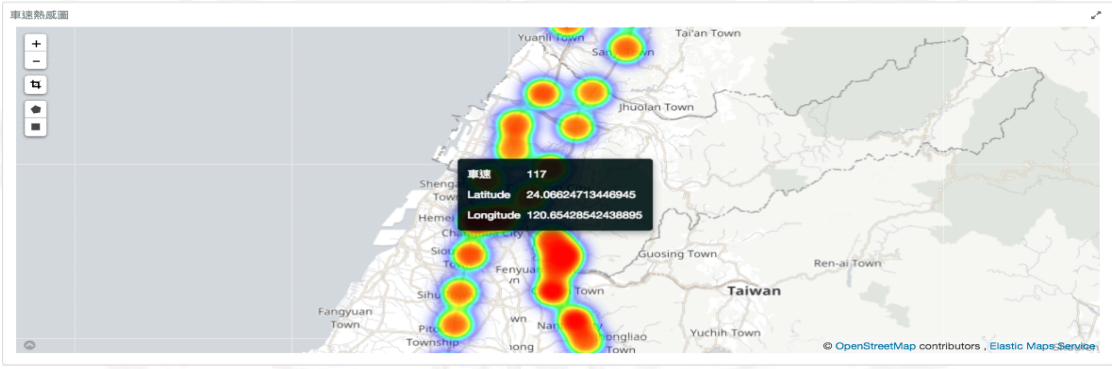


FIGURE 4.7: 偵測站間車速熱感圖

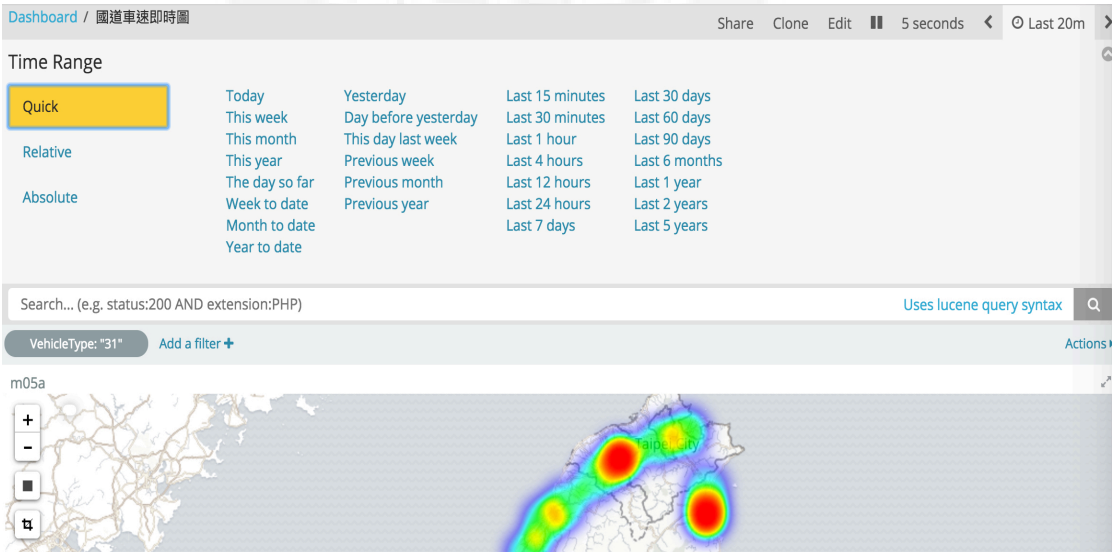


FIGURE 4.8: 車速熱感圖歷史資料

4.2.4 利用 M05A 資料呈現車速與車流量關係

利用過去一個的資料來呈現車流量與車速的對比關係，如 Figure 4.9所示。統計 106 年 12 月份 M05A 資料，X 軸為車流量間距為三十輛車次，Y 軸為車速間距為每小時 10 公里，統計數值為出現的次數，並篩選過濾條件設定為小客車，將出現次數分為四個級距，0 到 100、100 到 2000、2000 到 20000、以及 20000 以上，從圖表中可以發現當偵測站間的車流量小於 30 輛，有可能會出現 0 到 150 公里都有可能出現，而且還會有不少車輛超過國道速限，另外可以發現在偵測站間車流量低於 210 時，多數的車輛速度維持在 80 到 110 之間，另外一個就當車流量大於 150 時，車速就開始會降低，當車流量超過高達 270 時，明顯可以發現符合的筆數開始變少，這也可以代表這個車流量是偵測站間的極限值。

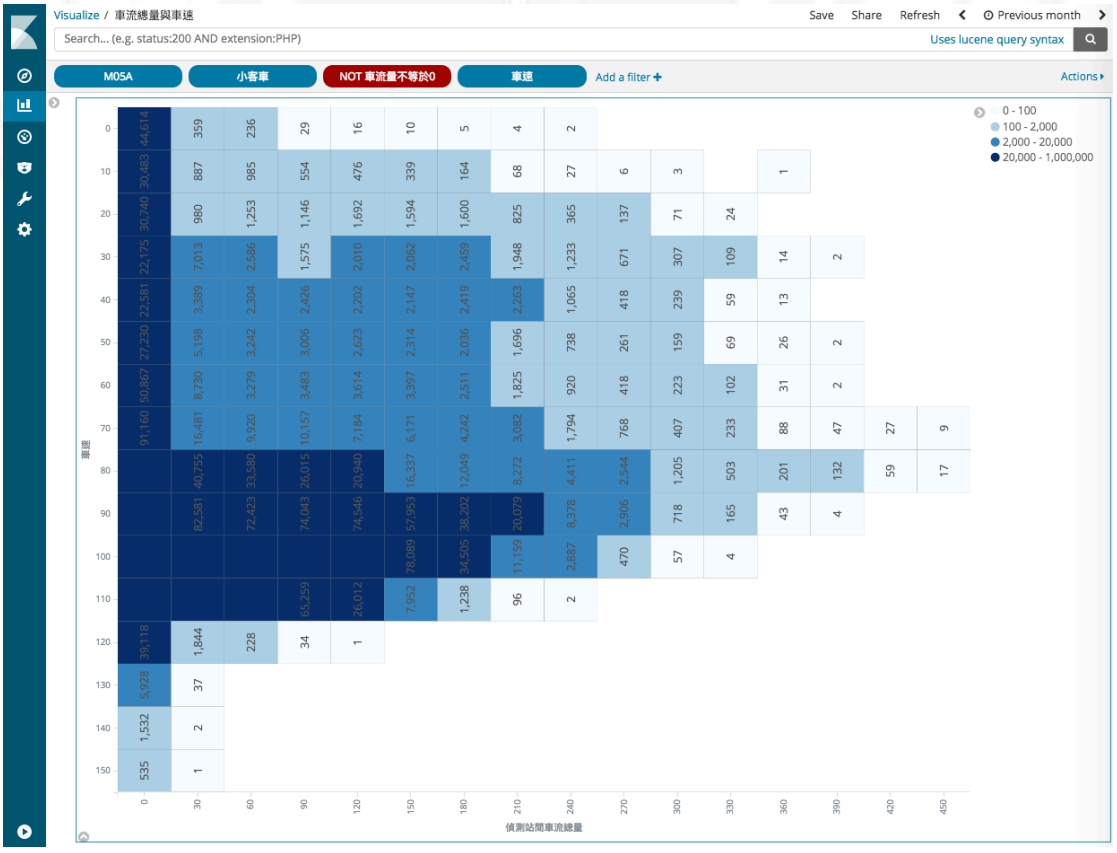


FIGURE 4.9: 車流量與車速關係圖

4.2.5 M06A 台中交流道車流分析

另外的圖表主要使用高公局所每天提供的使用 TDCS M06A 來分析資訊，此實驗以台中交流道分析為例，透過上方時間區段篩選歷史資料，分析到台中交流道的車流如 Figure 4.10，這張圖表主要是分析 106 年 12 月份到台中交流道下車的車流，X 軸是每日的時間軸，Y 軸代表下交流道的車次，圖表曲線共有星期一到星期日七天的平均值，從 12 月份來看，平常日的尖峰時段幾乎都是重疊的，早上的峰值出現在 7 點到 8 點，晚間則是出現 16 點-18 點，除了星期五下班的峰值所有不同，星期一到星期四曲線幾乎吻合，另外假日的峰值則出現在 10 點到 12 點間，而且 12 點到 23 點的平均車流量略高於平常日，由假日的峰值比平常日峰值晚，可以推斷平常日的峰值有可能是上班的車潮，

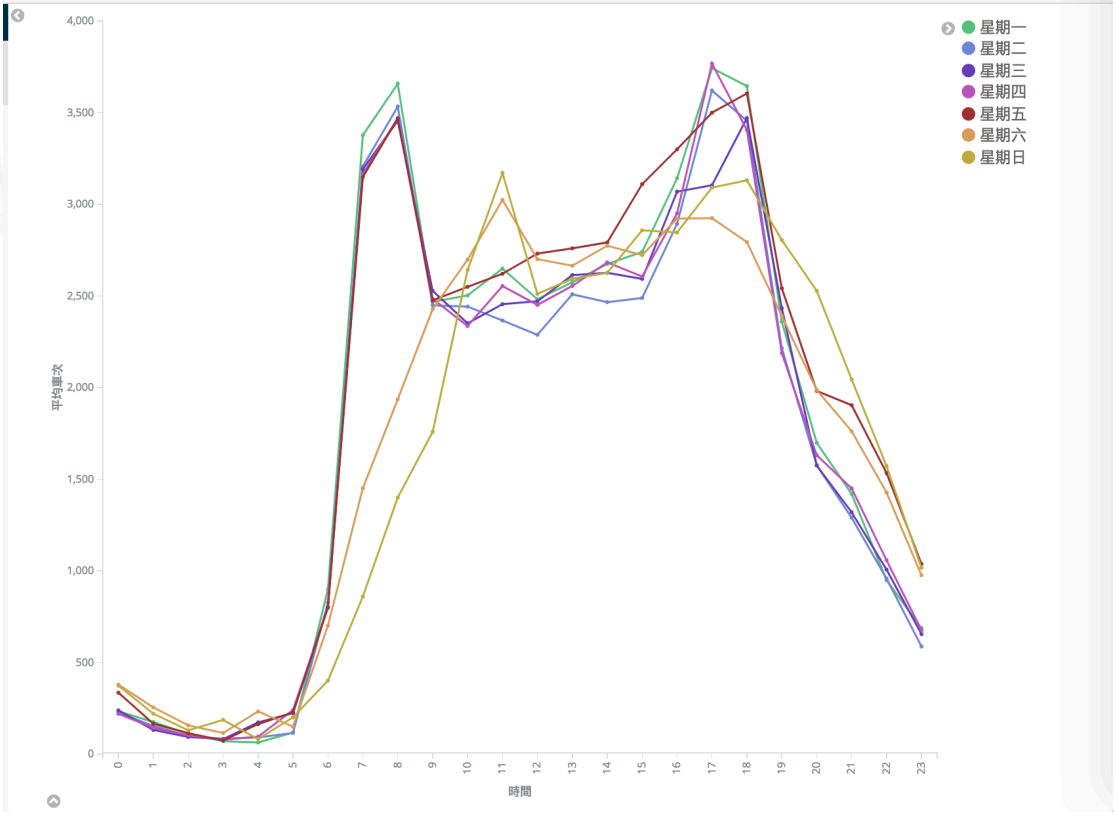


FIGURE 4.10: 到台中交流道的每日平均車流

另外透過上方時間指定平常日來進行分析，統計 106 年 12 月到台中交流道的車流如 Figure 4.11，上半部圖表主要呈現折線圖，下方圖表主要為平均車次的排序，透過上方圖表呈現很快可以知道平常上班日尖峰時段集中於 7 點-8 點

與 16 點-18 點這兩個時段，這也符合一般公司上下班時間，而比較特別的是可以發現從南下到台中交流道的車較多。

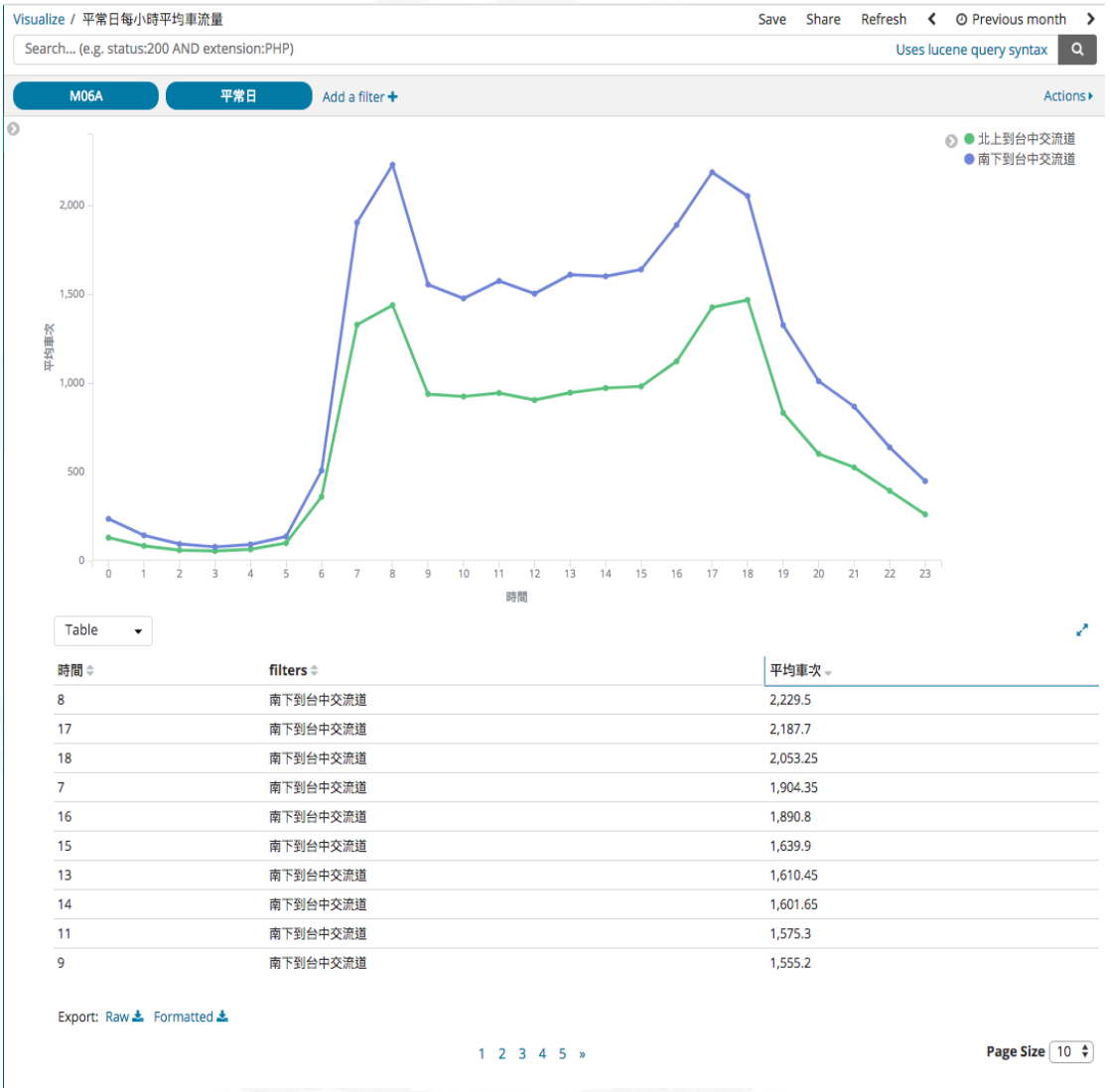


FIGURE 4.11: 平常日到台中交流道的平均車流量

另外的圖表主要使用篩選呈現交流道出入車流量及車輛起訖目的端地分析，分析從何處出發到台中交流道前十名如 Figure 4.12，可以發現從彰化到后里這個區段範圍內出發的車輛較多，更深入了解 7 點到 8 點車流量其分析結果如 Figure 4.13，可以發現上班尖峰時段，從彰化到后里開車到台中上班的車流與整日統計比例一致，由於目前高公局提供的資料已經去識別化，所以沒辦法利用車輛識別碼來確認是否為常態行為，但藉由此系統也可以達到分析與發現問題。

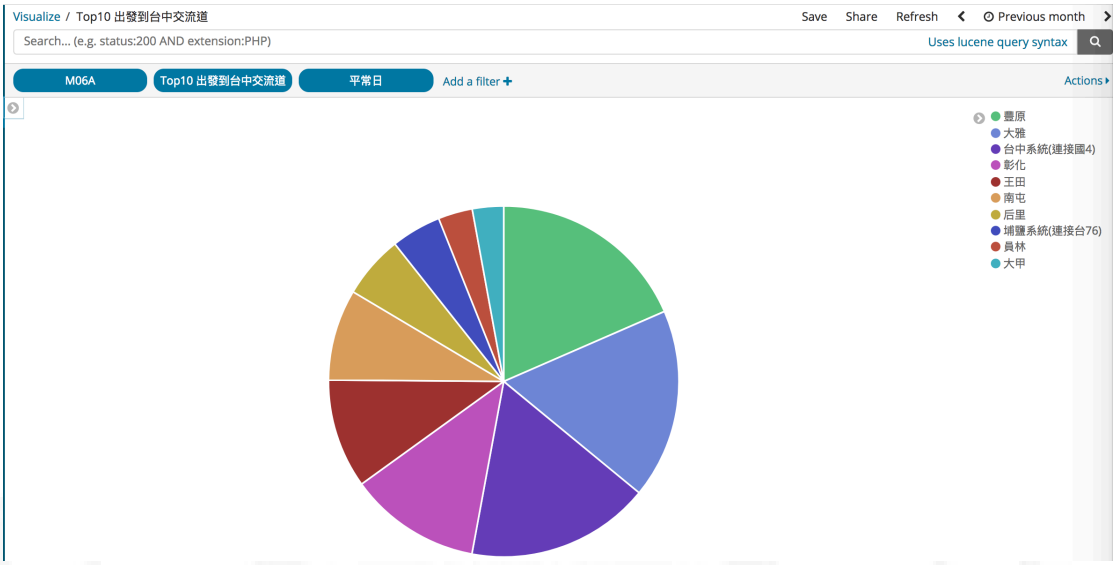


FIGURE 4.12: Top 10 出發到台中交流道

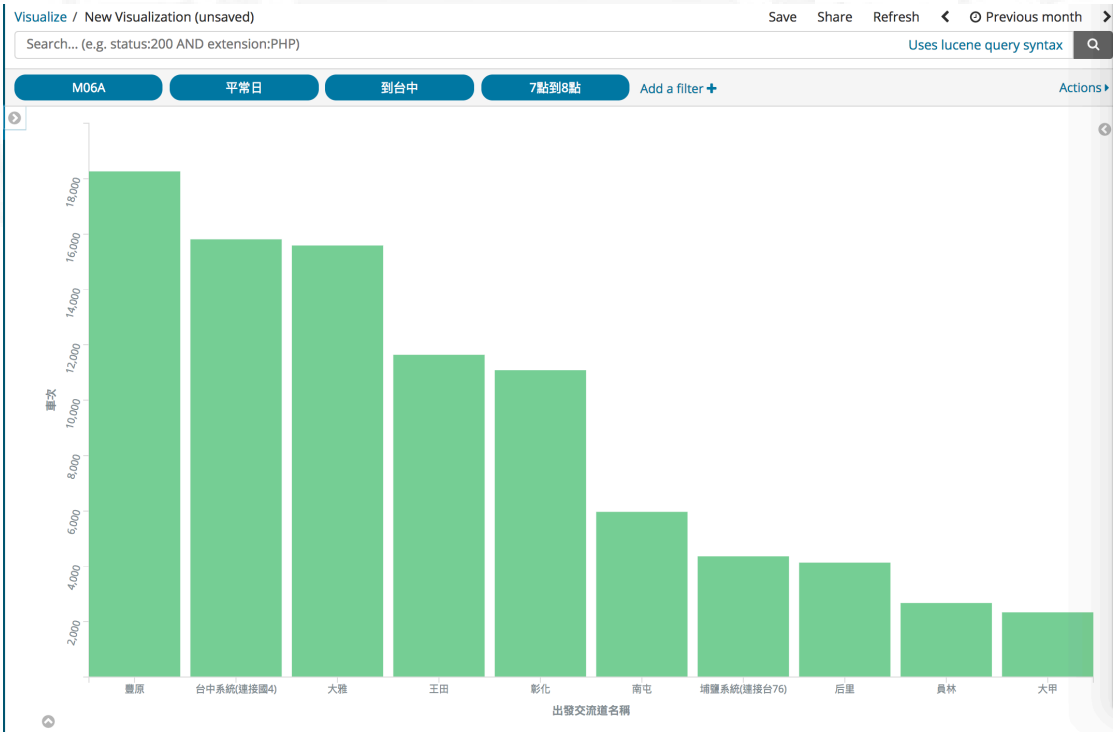


FIGURE 4.13: 上班尖峰時段車流分析

另外從台中交流道出發的車流量與 Top10 目的地分析如 Figure 4.14與 Figure 4.15，透過圖表也可得到尖峰時段集中在 7 點與 16 點-18 點這兩個區段，而 Top10 目的地與下台中交流道結果雷同，透過下班時段 16-18 點 Top10 目的

地圖表如 Figure 4.16，可以更確定平常日上下班時段的車流，是由鄰近的縣市開車到台中市區上班族產生的。

FIGURE 4.14: 從台中交流道出發的車流量

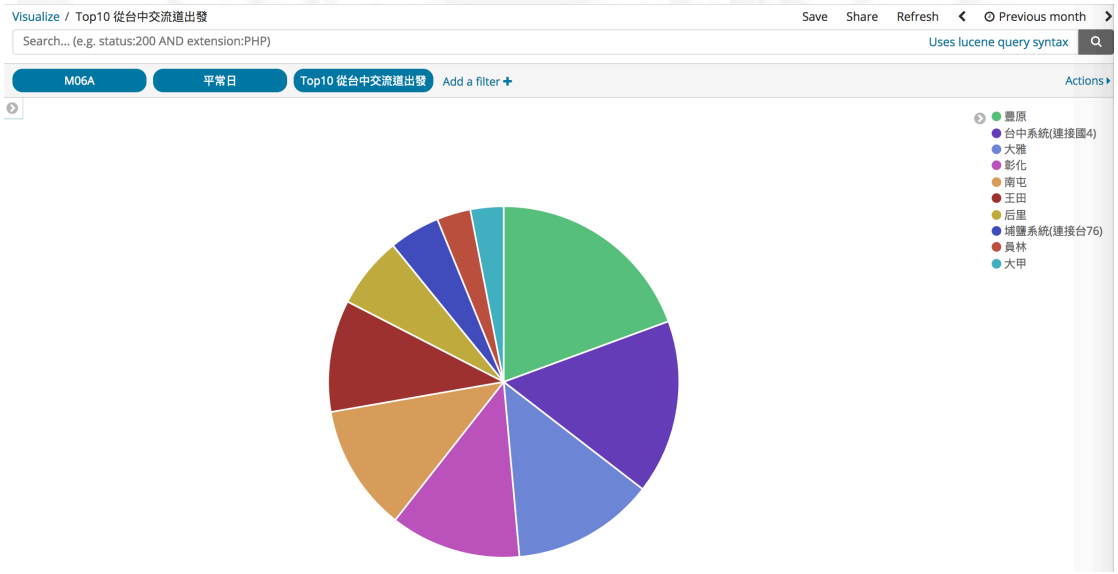


FIGURE 4.15: Top 10 從台中交流道出發

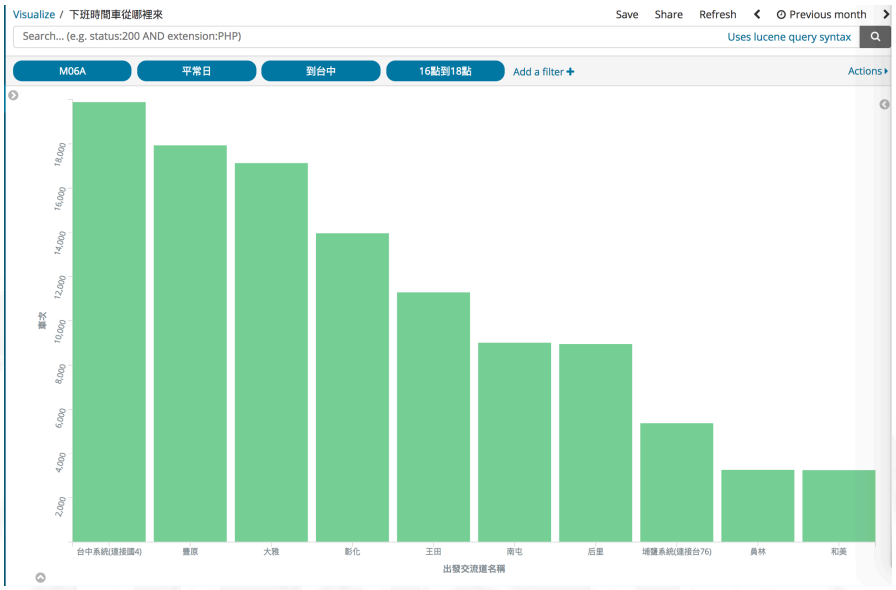


FIGURE 4.16: 下班尖峰時段車流分析

在假日由每日車流量及 Top10 來源及目的端交流道圖表如 Figure 4.17到 Figure 4.18，與平常日很明顯的不同，出發到台中的車流量，並沒有上班時段 7-8 點的尖峰車流量，取而代之的是 10-13 點與 16-17 點這兩個尖峰時段，而且從 10-17 都有維持著每小時 1800 輛以上的車流量。另外從 Figure 4.19與 Figure 4.20發現一個很特別的情形，台中到新竹科學園區兩地往返車流皆有進入 Top10。

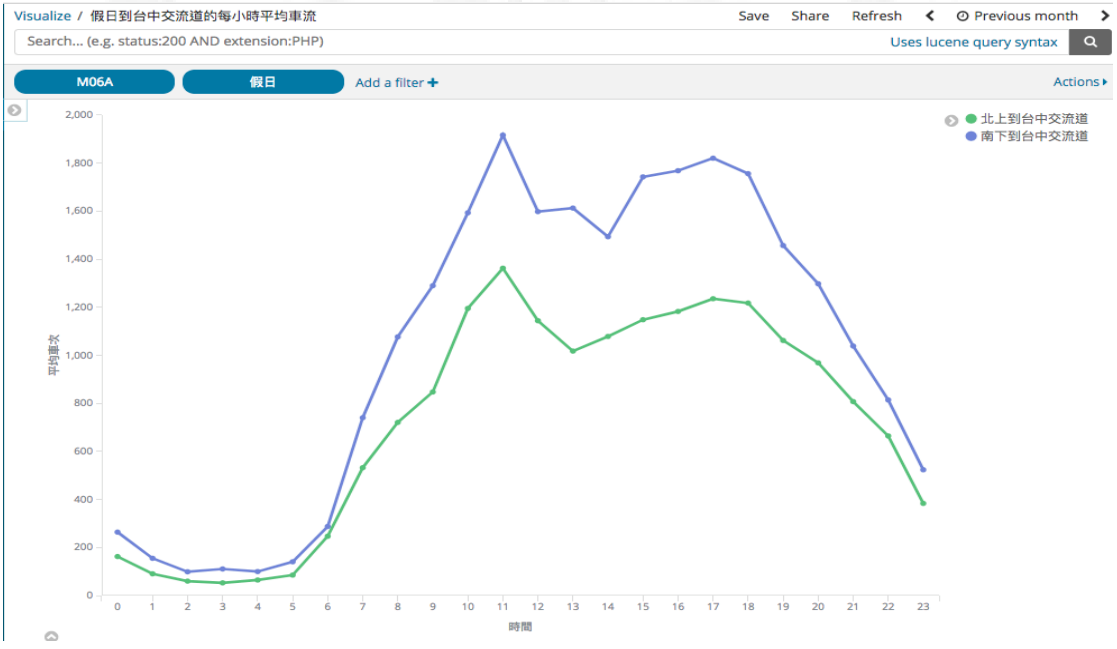


FIGURE 4.17: 假日到台中交流道的車流量

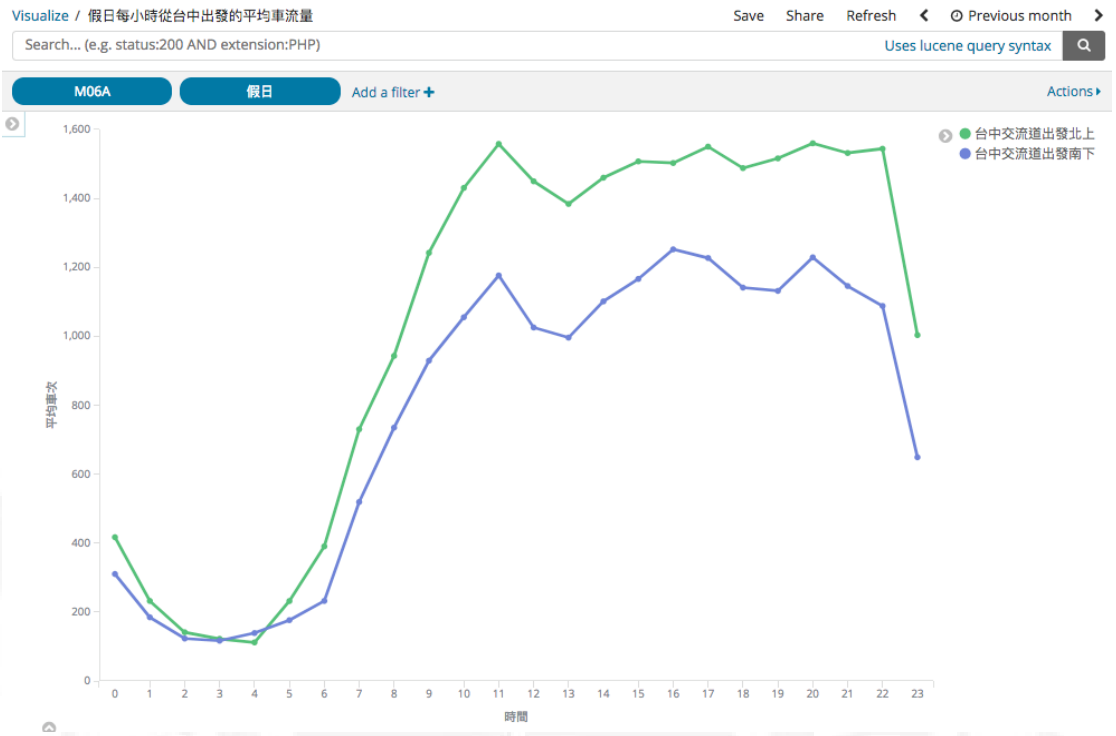


FIGURE 4.18: 假日從台中交流道出發的車流量

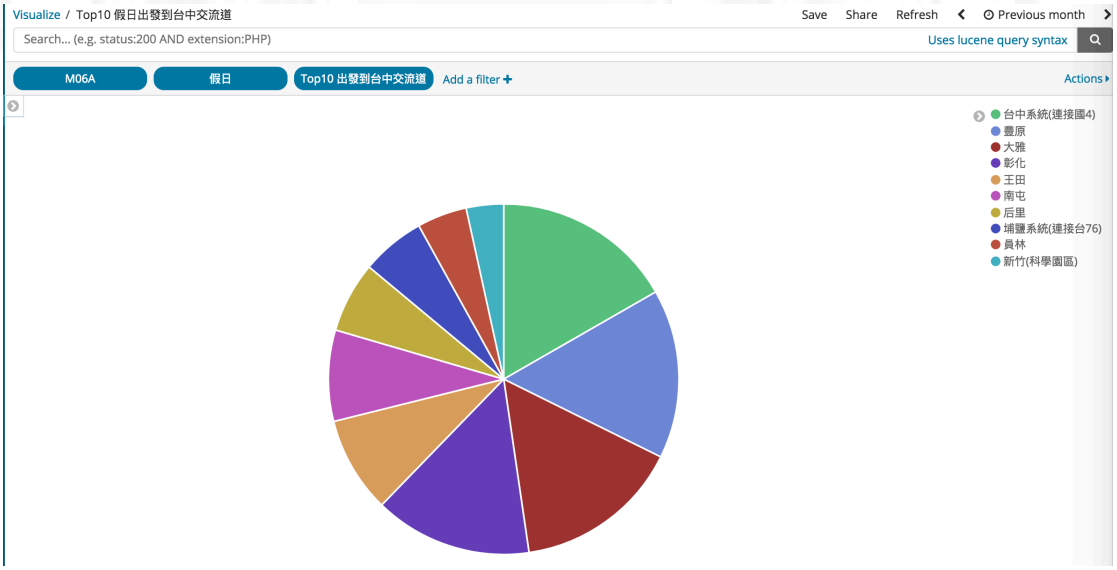


FIGURE 4.19: Top 10 假日出發到台中交流道

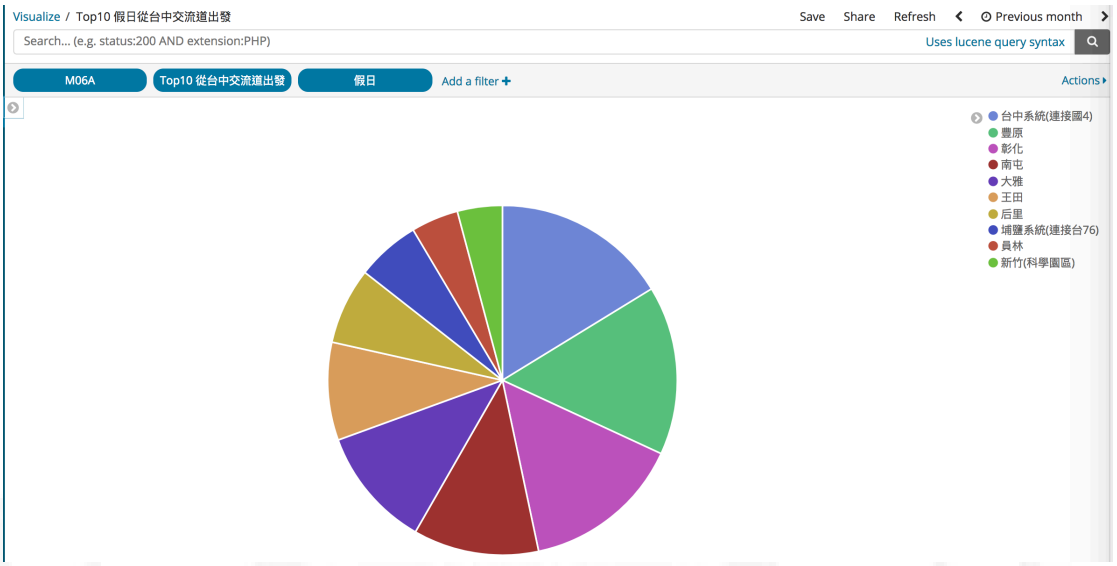


FIGURE 4.20: Top 10 假日從台中交流道出發

對於國道 20 公里免費路程是否將取消，其實國內學者都有探討，那已 106 年 12 月來看如 Figure 4.21，單趟旅程小於 20 公里的車輛就佔了 65.89%，隨然沒辦法利用識別碼來判斷是否車過優惠，但從圖表上就可以了解短程使用者佔了絕大多數。

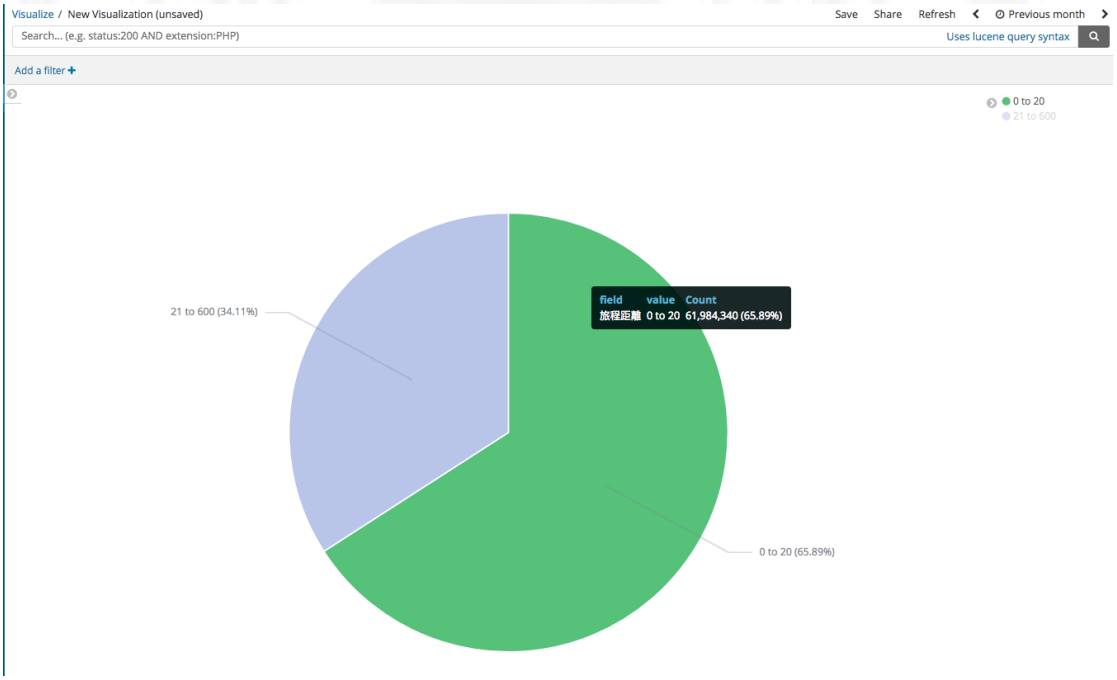


FIGURE 4.21: 旅程距離比率

4.2.6 台中交流道與新竹科學園區來往車流分析

以時間為 X 軸每小時為一個統計單位深入分析，統計各車種從台中出發到新竹科學園區如 Figure 4.22、從新竹科學園區到台中如 Figure 4.23，從小型車的車流可以發現在星期一早上及星期日 17 點過後出發到新竹科學園區如 Figure 4.24，以 106 年 12 月資料分析後排序，每小時平均車流量最高，發生在星期日有 428 輛車次，從排名中也可以發現星期日 17 點過後平均都有 300 輛車次前往新竹科學園區。而星期五 17 點過後到星期六都有車流從新竹科學園區到台中如 Figure 4.25，以 106 年 12 月資料分析後排序，每小時平均車流量最高，發生在星期六中午 11 點有 313 輛車次，而在星期五 17 點過後都有 200 輛以上車次前往台中。若分析平常日也發現每天下班時間 17-18 點都會有固定的車流從新竹科學園區返回台中，透過這些現象可以了解到有部分車輛在上下班、放假日會固定往返兩地，由於沒有車輛識別資訊可以進一步分析，若有此方面資訊可以進一步分析是否為居住於台中的居民到新竹科學園區工作產生的車流。

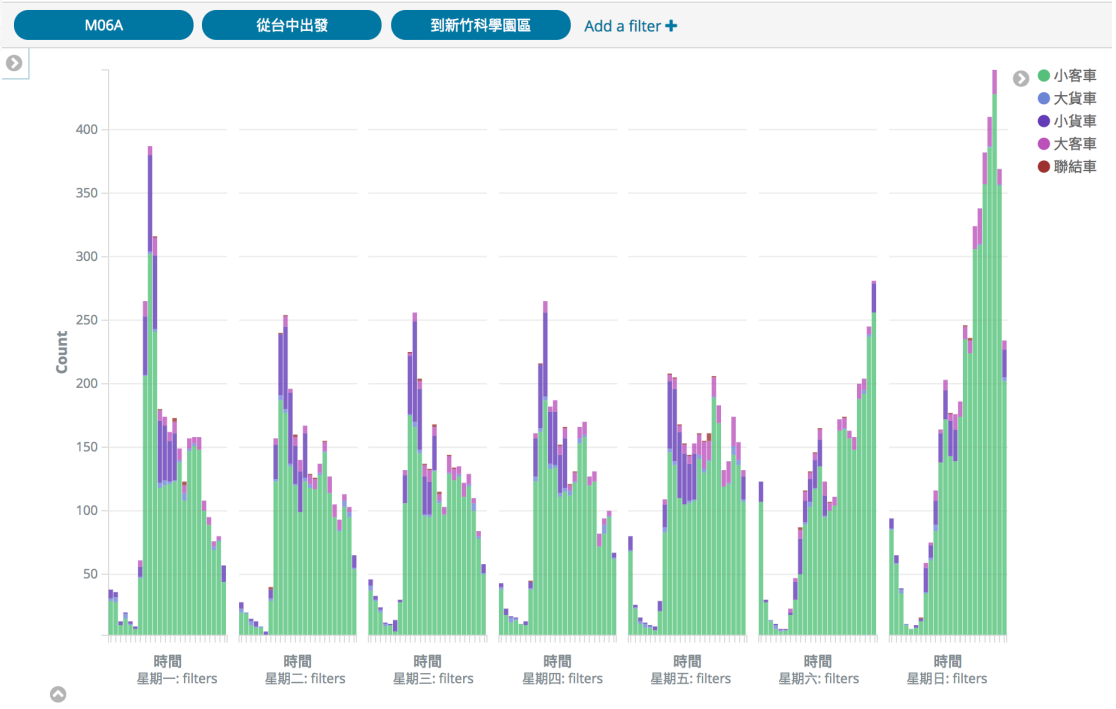


FIGURE 4.22: 從台中出發到新竹科學園區車流量

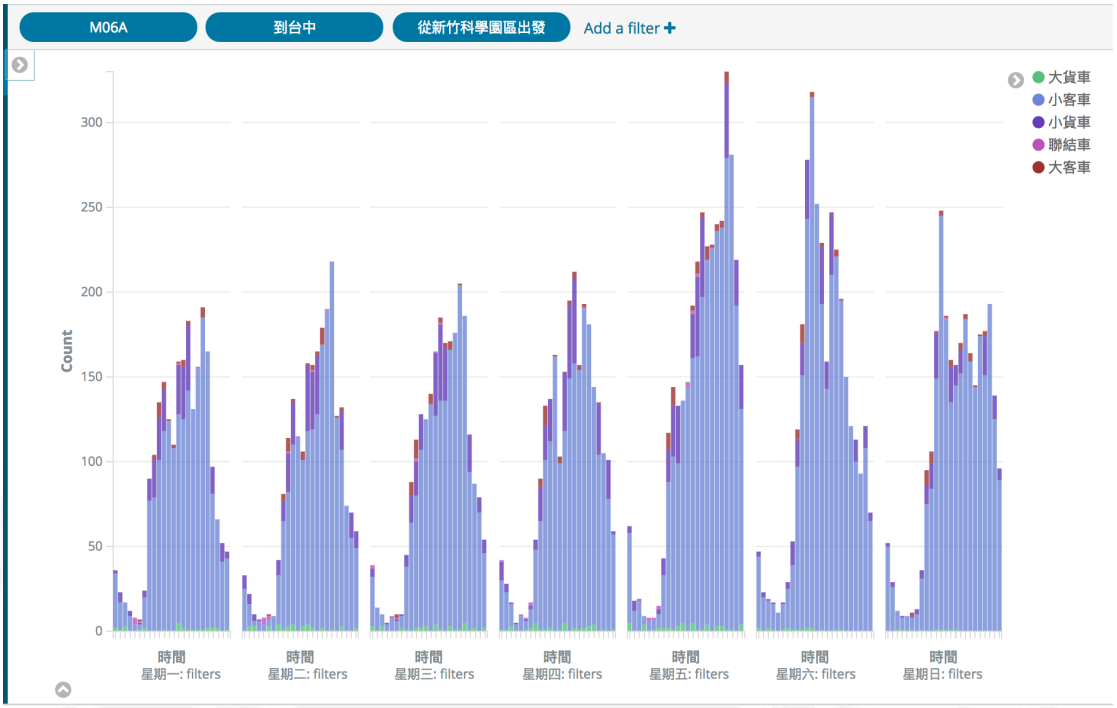


FIGURE 4.23: 從新竹科學園區到台中車流量

FIGURE 4.24: 從台中出發到新竹科學園區車流量排名

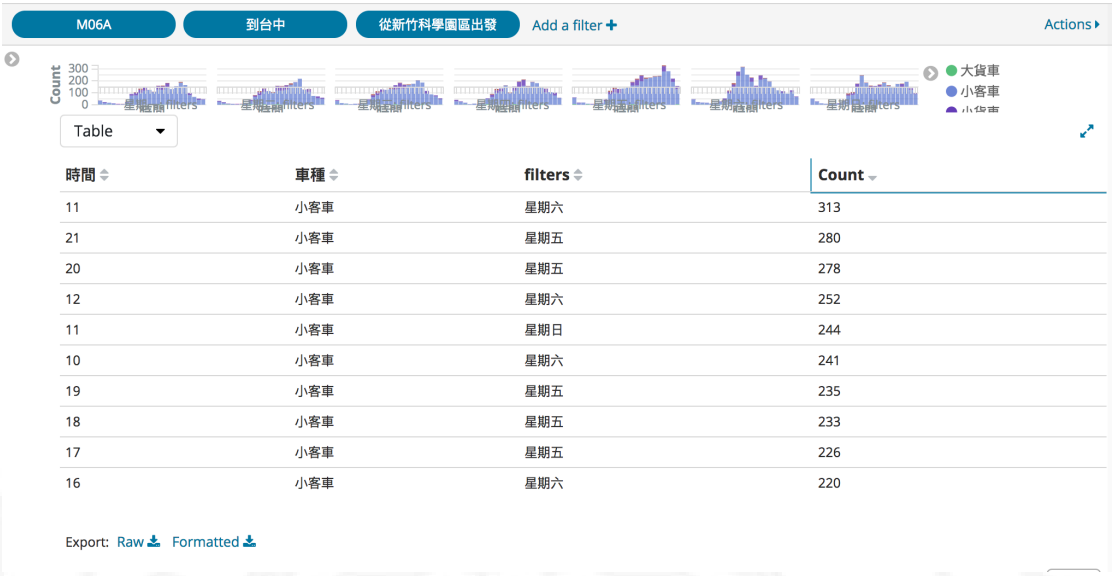


FIGURE 4.25: 從新竹科學園區到台中車流量排名

4.2.7 國道收費區段車流量最大的交流道

利用 106 年 12 月份資料分析，統計國道收費區段最多人出入的前 10 個交流道，如 Figure 4.26與 Figure 4.27。從圖表上可以最多人上下交流道的位置是銜接機場系統的出口最多人上下交流道的位置是銜接機場系統的出口，單月總車流可以接近 460 萬車次，而且主要集中在台北及高雄這兩個大城市。

FIGURE 4.26: Top10 國道最多車上的交流道

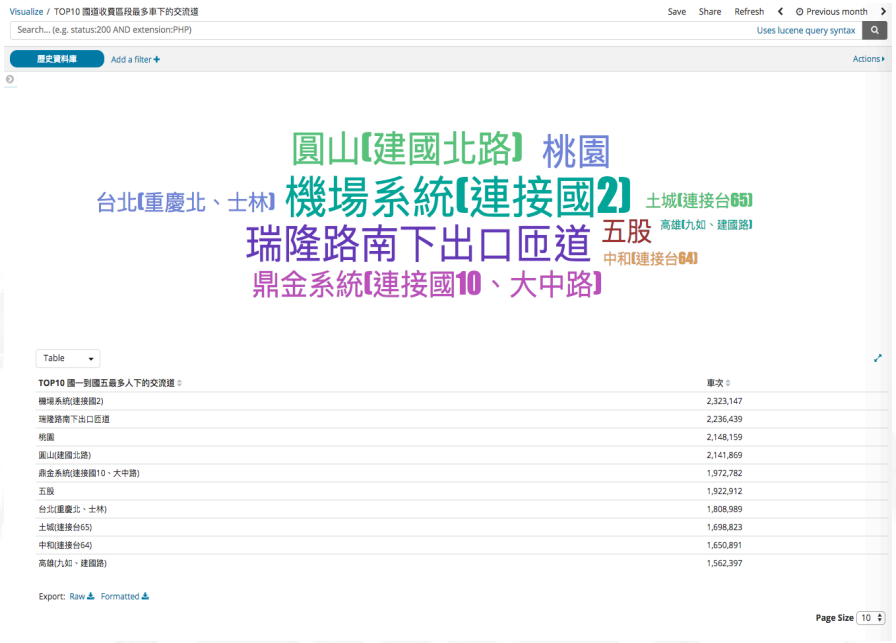


FIGURE 4.27: Top10 國道最多車下的交流道

此系統架構最重要的分析功能，使用者可以自訂想要分析的資料區間及索引資料，更重要的是搜尋的反應時間，在超過 5 億筆資料中搜尋符合條件的資料顯示圖表，查詢及回應時間加總在 1 秒以下如 Figure 4.28及 Figure 4.29，凸顯由 ELK Stack 環境建置而成的系統擁有快速的分析能力。

FIGURE 4.28: 資料總筆數

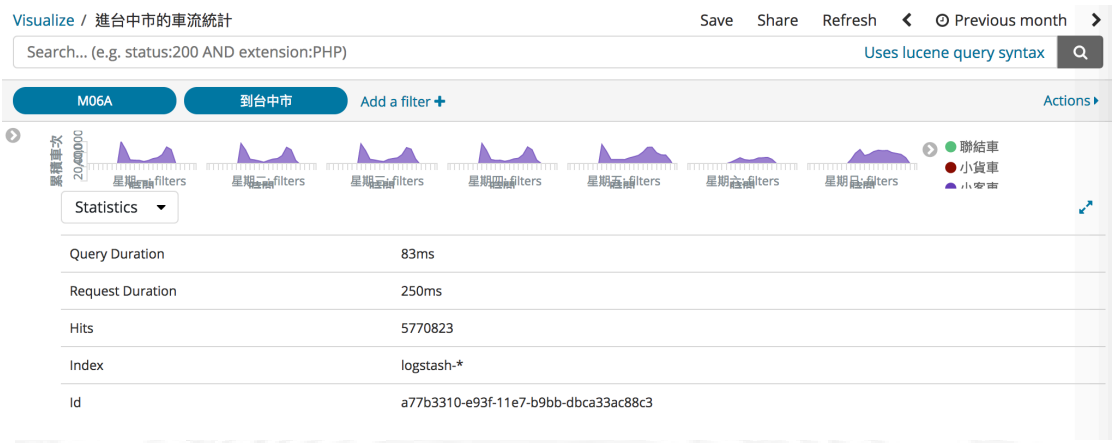


FIGURE 4.29: 查詢執行時間

最後就是利用 apache 來呈現整體的畫面，此部分主要是將 Kibana 管理介面隱藏起來避免被誤操作，而且呈現給使用者或著主管最後的整理畫面即可，其畫面可以使用 Kibana Dashboard 的圖表，利用 html iframe 語法坎入到網頁，實作後畫面如 Figure 4.30與 Figure 4.31等。

FIGURE 4.30: 利用 apache 呈現圖表

FIGURE 4.31: 子網頁-進出台中車流統計

Chapter 5

結論與未來方向

5.1 結論

資料的應用已是現在主流，透過對資料的分析和應用，可以從中挖掘出未知的趨勢及有價值的資訊。本論文主要應用 ELK Stack 建置一個分析系統，運用高公局公開於政府開放平台的開放性資料-國道高速公路電子收費交通資料蒐集支援系統，進行車流分析及即時車速顯示。本論文進行資料庫查詢測試，其結果可以發現 Elasticsearch 在單機運作時隨著資料量越大，其搜尋執行時間會相對成長，在對 MySQL 欄位建立 index 後，其搜尋執行時間比 Elasticsearch 表現更佳好。在 ELK Stack 所建置的環境資料分析呈現部分主要分為兩個部分。第一部分是以地圖方式呈現 M05A 每五分鐘各偵測站間車流速度，此部分比現有高公局 1968 APP 以文字描述位置及車速級距來顯示各區段車速，來得容易辨識，而且可以根據車種類別篩選得知不同車種的行車速度。第二部分主要是利用 M06A 各旅次原始資料來進行分析，實驗以分析台中交流道為例，可以發現清楚了解各時段車流狀況及得到上下班時段車輛都來自於附近縣市，而且這些車流都是國道 20 公里免收費的優惠範圍內，利用圖表也可以發現近 6 成 6 的車輛都是短程行駛，若國道取消 20 公里優惠里程，勢必對用路人會造成一定的衝擊。另外也發現同一個交流道進出車流量是具有差異的，由於無法識別車輛，但此部分推測有可能為車輛利用其他交流道進出所造成。透過週間車流

量發現台中到新竹科學園區有固定的車流在往返，比較遺憾的是因為去識別化後，無法識別這個行為是不是特定車輛週期性行為。此論文提出的分析系統架構有別於過往應用於系統事件日誌收集與處理，而是同時進行多種不同格式的開放性資料的收集與處理，也省去程式開發人員製作客製化報表的時間，只需要了解資料欄位就可以自行組合想要呈現的圖表。此系統具備資訊收集、處理及即時呈現視覺化圖表，提供使者分析應用，實現快速檢索、即時視覺化及歷史資料分析。

5.2 未來方向

這套系統目前僅具備資料分析功能，未來希望能結合天候、空氣品質、交通狀況等情形，利用機器學習技術達到車流預測及雍塞預警，提供更多元化的決策方向，例如尖峰時段前加強疏導，達到車流順暢。是否可利用歷史資料進行各偵測站間最大車流量與車速的對應關係的研究，分析出各區段間車流是否達到某個值後車速就會降低。結合影像車牌辨識資料與現有系統 M06A 車次原始資料結合，研究車流週期性行為。若搭配市府單位於市區重要路段架設 ETC 感測器設備，即時將路況資料傳回系統，透過圖表的呈現，就可以達到市區交通網的掌控。

參考文獻

[1] Rajkumar Buyya, Rodrigo N Calheiros, and Amir Vahid Dastjerdi. *Big Data: Principles and Paradigms*. Morgan Kaufmann, 2016.

[2] Apache. Apache hadoop,http://hadoop.apache.org, 2017.

[3] Apache. Apache spark,https://spark.apache.org, 2017.

[4] 國家發展委員會. 政府資料開放平臺,https://data.gov.tw, 2017.

[5] D. Mintz, United States. Environmental Protection Agency. Office of Air Quality Planning, and Standards. *Technical Assistance Document for the Reporting of Daily Air Quality - the Air Quality Index (AQI)*. U.S. Environmental Protection Agency, Office of Air Quality Planning and Standards, 2009.

[6] 交通部臺灣區國道高速公路局. 國道高速公路電子收費交通資料蒐集支援系統,http://tisvcloud.freeway.gov.tw/history/TDCS, 2017.

[7] 交通部臺灣區國道高速公路局. 第三屆高速公路 etc 資料在交通管理之應用創意競賽,http://www.freeway.gov.tw, 2017.

[8] Andrej Trnka. Big data analysis. *European Journal of Science and Theology*, 10(1):143–148, 2014.

[9] Gema Bello-Orgaz, Jason J Jung, and David Camacho. Social big data: Recent achievements and new challenges. *Information Fusion*, 28:45–59, 2016.

[10] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115, 2015.

[11] Daniel E O'Leary. Artificial intelligence and big data. *IEEE Intelligent Systems*, 28(2):96–99, 2013.

[12] Jules J Berman. *Introduction in Principles of big data: preparing, sharing, and analyzing complex information*. Newnes, 2013.

[13] CT Yang, SC Chou, CJ Chen, SH Cao, and Y Sun. Implementation of wi-fi log analysis system with elk stack. In *International Conference on Internet Studies (NETs 2016)*. ATISR academic, 2016.

[14] Elastic. Elastic stack,https://www.elastic.co, 2017.

[15] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. " O'Reilly Media, Inc.", 2015.

[16] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000.

[17] Abdelkader Lahmadi and Frédéric Beck. Powering monitoring analytics with elk stack. In *9th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2015)*, 2015.

[18] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W Godfrey. Mining modern repositories with elasticsearch. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 328–331. ACM, 2014.

[19] Melody Moh, Santhosh Pininti, Sindhusha Doddapaneni, and Teng-Sheng Moh. Detecting web attacks using multi-stage log analysis. In *Advanced*

Computing (IACC), 2016 IEEE 6th International Conference on, pages 733–738. IEEE, 2016.

[20] Xiwei Xu, Ingo Weber, Len Bass, Liming Zhu, Hiroshi Wada, and Fei Teng. Detecting cloud provisioning errors using an annotated process model. In *Proceedings of the 8th Workshop on Middleware for Next Generation Internet Computing*, page 5. ACM, 2013.

[21] Alexander Reelsen. Using elasticsearch, logstash and kibana to create realtime dashboards. *Dostupné z: https://secure. trifork. com/dl/goto-berlin-2014/GOTO_Night/logstash-kibana-intro. pdf*, 2014.

[22] Ceki Gülcü. *The complete log4j manual*. QOS. ch, 2003.

[23] Elastic. Kibana: Explore, visualize, discover data,",https://www.elastic.co/products/kibana, 2017.

[24] Sushma Sanjappa and Muzameel Ahmed. Analysis of logs by using logstash. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications*, pages 579–585. Springer, 2017.

[25] 楊智傑. Google 棄甲骨文 mysql，將大規模導入 mariadb,https://www.ithome.com.tw/node/82854, 2013.

[26] Tarun Prakash, Misha Kakkar, and Kritika Patel. Geo-identification of web users through logs using elk stack. In *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, pages 606–610. IEEE, 2016.

[27] Jun Bai. Feasibility analysis of big log data real time search based on hbase and elasticsearch. In *Natural Computation (ICNC), 2013 Ninth International Conference on*, pages 1166–1170. IEEE, 2013.

[28] Khanasin Yamnual, Phond Phunchongharn, and Tiranee Achalakul. Failure detection through monitoring of the scientific distributed system. In *Applied*

System Innovation (ICASI), 2017 International Conference on, pages 568–571. IEEE, 2017.

[29] S Bagnasco, D Berzano, S Lusso, M Masera, and S Vallero. Managing competing elastic grid and cloud scientific computing applications using opennebula. In *Journal of Physics: Conference Series*, volume 664, page 022004. IOP Publishing, 2015.

[30] S Bagnasco, D Berzano, A Guarise, S Lusso, M Masera, and S Vallero. Monitoring of iaas and scientific applications on the cloud using the elasticsearch ecosystem. In *Journal of Physics: Conference Series*, volume 608, page 012016. IOP Publishing, 2015.

[31] Pingkan PI Langi, Warsun Najib, Teguh Bharata Aji, et al. An evaluation of twitter river and logstash performances as elasticsearch inputs for social media analysis of twitter. In *Information & Communication Technology and Systems (ICTS), 2015 International Conference on*, pages 181–186. IEEE, 2015.

[32] Tarun Prakash, Misha Kakkar, and Kritika Patel. Geo-identification of web users through logs using elk stack. In *Cloud System and Big Data Engineering (Confluence), 2016 6th International Conference*, pages 606–610. IEEE, 2016.

[33] 交通部臺灣區國道高速公路局. 國道高速公路計程電子收費階段交通資料蒐集支援系統使用手冊 *V3.1.* 2017.

附錄 A

ELK Stack 安裝步驟

A.1 基本環境設定

步驟 1. 更新 vim 避免舊版 vim 上下左右鍵異常，為時間錯亂安裝 NTP。

```
sudo apt-get update
sudo apt-get install -y vim ntp curl ssh
```

步驟 2. 將所有主機加入本機 IP 對應表。

```
sudo vim /etc/hosts
#加入下列參數
192.168.56.103 master
```

步驟 3. 安裝 open JDK。

```
sudo apt-get install -y openjdk-8-jdk
sudo ln -s /usr/lib/jvm/java-8-openjdk-amd64 /usr/lib/jvm/jdk
```

步驟 4. 編輯環境參數。

```
vim .bashrc
#在最後面加入jdk.hadoop等相關路徑
export JAVA_HOME=/usr/lib/jvm/jdk/
#匯入環境變數
source .bashrc
```

A.2 Elasticsearch 安裝

步驟 1. 加入 Elastic 套件庫來源。

```
https://packages.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
```

步驟 2. 建立 Elastic Stack 來源表。

```
sudo apt-get install -y apt-transport-https
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | sudo tee -a /
    etc/apt/sources.list.d/elastic-5.x.list
```

步驟 3. 更新套件庫列表及安裝 Elasticsearch。

```
sudo apt-get update
sudo apt-get install -y elasticsearch
```

步驟 4. 編輯 elasticsearch.yml 讓外部可以存取。

```
sudo vi /etc/elasticsearch/elasticsearch.yml
```

步驟 5. 更新參數，將下列幾行參數 # 移除並修改如下。

```
cluster.name    : hpc
network.host    : 192.168.56.103
http.port          : 9200
```

步驟 6. 重啟服務並將 elasticsearch 設定成開機就啟動。

```
sudo systemctl restart elasticsearch
sudo systemctl enable elasticsearch
```

A.3 Logstash 安裝

步驟 1. 下載安裝 logstash。

```
sudo apt-get install -y logstash
```

步驟 2. 簡易功能測試。

```
cd /usr/share/logstash
sudo bin/logstash -e 'input {stdin{ } } output {stdout{ } }'
輸入Hello
螢幕畫面會顯示master Hello則表示基本功能正常
```

步驟 3. 建立設定檔。

```
sudo vi /etc/logstash/conf.d/first-pipeline.conf
#加入下列參數進行測試，實際加入參數請參考附錄D “Logstash configuration file”

input {
                file {
                        path => "/var/log/*.log"
                        }
                }
output {
                elasticsearch {
                        hosts => " 192.168.56.103:9200"
                        manage_template => false
                        index => "%{[@metadata][beat]}-%{+YYYY.MM.dd}"
                        document_type => "%{[@metadata][type]}"
                        }
                }
```

步驟 4. 重啟服務並將 logstash 設定成開機就啟動。

```
sudo systemctl restart logstash
sudo systemctl enable logstash
```

A.4 Kibana 安裝

步驟 1. 下載安裝 Kibana。

```
sudo apt-get install -y kibana
```

步驟 2. 編輯 kibana.yml。

```
sudo vi /etc/kibana/kibana.yml
```

步驟 3. 更新參數，將下列幾行參數#移除並修改如下。

```
server.port: 5601
server.host: "192.168.56.103"
elasticsearch_url :http://192.168.56.103:9200
```

步驟 6. 重啟服務並將 kibana 設定成開機就啟動。

```
sudo systemctl restart kibana
sudo systemctl enable kibana
```

附錄 B

Get M05A Shell Script

B.1 getm05a.sh 每五分鐘取得 M05A 資料

getm05a.sh

```
#/bin/bash
#配合TDCS檔案名稱格式設定參數
#let date format to 20170720...
Dataday=$(date -d '10 min ago' +%Y%m%d)
Datahour=$(date -d '10 min ago' +%H)
Dataminute=$(date -d '10 min ago' +%M)
Dataminute=$(expr "$Dataminute" / 5 \* 5)
Dataminute=$(printf "%02d\n" $Dataminute)
Dataweekday=$(date -d '10 min ago' +%a)
Datamonth=$(date -d '10 min ago' +%B)

#設定TDCS M05A下後檔案名稱及下載位置
Url="http://tisvcloud.freeway.gov.tw/history/TDCS/M05A/$Dataday/$Datahour/TDCS_M05A_"
    $Dataday"_"$Datahour$Dataminute"00.csv"
Filename="TDCS_M05A_"$Dataday"_"$Datahour$Dataminute"00.csv"
filename2=$Dataday".csv"
TMPF="tmp1.csv"
cd /tmp

#自動檢測下載檔案是否存在，若不存在則等待60秒
res=$(curl -s -o /dev/null -w '%{http_code}\n' $Url)
while [ $res -eq "404" ]
do
```

```
sleep 60
res=$(curl -s -o /dev/null -w '%{http_code}\n' $Url)
done

#下載TDCS M05A檔案
wget $Url

#進行檔案的過濾及清洗，去除不必要的欄位
cat $Filename | awk -F"," 'BEGIN{OFS=","}{print $1,$2,$4,$5,$6}' > $TMPF
IFS=','

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入資料
while read GantryPoint oChineseName dChineseName Gps
do
sed -i "s/$GantryPoint/$Datamonth","$Dataweekday","$GantryPoint","$oChineseName"-"
    $dChineseName","$Gps/g" $TMPF
done < /home/hpclab/TDCSPoint.csv

#將處理過後資料輸出至檔案
cat $TMPF >> /home/hpclab/m05a/$filename2

#清除暫存檔案
rm -f $Filename
rm -f *.csv*
cd ~
```

B.2 getm05b.sh 處理指定日期 M05A 資料

getm05b.sh

```
#/bin/bash

#配合TDCS檔案名稱格式設定參數
#let date format to 20170720...
#讀取日期參數
Dataday=$1
Dataweekday=$(date -d $Dataday +%a)
Datamonth=$(date -d $Dataday +%B)
TMPF="tmp1.csv"
TMPF2="$Dataday.csv"
cd /home/hpclab/tempm05a/

#設定小時參數為00~23
for ((i=0; i<=23; i++))
do
#let times format to 00 01 02 ~23
Datahour=$(printf "%02d\n" $i)

#設定分鐘參數為00、05、10....55
for ((j=0; j<=55; j=j+5))
do
Dataminute=$(printf "%02d\n" $j)

#設定TDCS M05A下後檔案名稱及下載位置
Url="http://tisvcloud.freeway.gov.tw/history/TDCS/M05A/$Dataday/$Datahour/TDCS_M05A_"
    $Dataday"_"$Datahour$Dataminute"00.csv"
Filename="TDCS_M05A_"$Dataday"_"$Datahour$Dataminute"00.csv"

#下載TDCS M05A檔案
wget $Url

#進行檔案的過濾及清洗，去除不必要的欄位
cat $Filename | awk -F"," 'BEGIN{OFS=","}{print $1,$2,$4,$5,$6}' > $TMPF
IFS=','

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入資料
while read GantryPoint oChineseName dChineseName Gps
do
sed -i "s/$GantryPoint/$Datamonth","$Dataweekday","$GantryPoint","$oChineseName"-"
    $dChineseName","$Gps/g" $TMPF
done < /home/hpclab/TDCSPoint.csv
```

```
#將處理過後資料輸出至檔案
cat $TMPF >> $TMPF2
done
done
mv  $TMPF2 /home/hpclab/m05a/

#清除暫存檔案
rm -rf *.csv
cd ~
```

B.3 gettarm05a.sh 處理已封存的指定日期資料

gettarm05a.sh

```
#/bin/bash

#配合TDCS檔案名稱格式設定參數
#讀入日期參數
#let date format to 20170720...
Dataday=$1
Dataweekday=$(date -d $Dataday +%a)
Datamonth=$(date -d $Dataday +%B)
TMPF='tmp1.csv'
TMPF2="$Dataday.csv"
cd /home/hpclab/tempm05a/
rm -f *.csv

#設定TDCS M05A下後檔案名稱及下載位置
filename="M05A_$Dataday.tar.gz"
Url="http://tisvcloud.freeway.gov.tw/history/TDCS/M05A/"$filename

#下載TDCS M05A檔案
#download opendate from ETC M05A
wget $Url

#解壓縮TDCS M05A資料，並過濾含有csv字眼資料輸出到tarlist檔案
tar -xzvf $filename | grep csv > tarlist.txt

#讀取tarlis檔案，一一將檔案進行清洗
while read mfilepwd
do
#進行檔案的過濾及清洗，去除不必要的欄位
cat $mfilepwd | awk -F"," 'BEGIN{OFS=","}{print $1,$2,$4,$5,$6}' > $TMPF
IFS=','

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入資料
while read GantryPoint oChineseName dChineseName Gps
do
sed -i "s/$GantryPoint/$Datamonth","$Dataweekday","$GantryPoint","$oChineseName"-"
    $dChineseName","$Gps/g" $TMPF
done < /home/hpclab/TDCSPoint.csv
#將處理過後資料輸出至檔案
cat $TMPF >> $TMPF2
done  < tarlist.txt
```

```
#將處理過後資料移動存放位置
mv $TMPF2 /home/hpclab/m05a/
#清除暫存檔案
rm -rf *.csv
rm -rf  M05A/
rm -rf *.gz
cd ~
```

附錄 C

Get M06A Shell Script

C.1 getm06a.sh 每日取得 M06A 資料

getm06a.sh

```
#/bin/bash

cd /home/hpclab/temp

#配合TDCS檔案名稱格式設定參數
#let date format to 20170720...
Dataday=$(date -d '1 days ago' +%Y%m%d)
Dataweekday=$(date -d $Dataday +%a)
Datamonth=$(date -d $Dataday +%B)

#清除暫存檔案
rm -rf *.csv

#設定進行清洗時暫存檔位置
#create new file for row data for all data
TMPF1='tmp1.csv'
TMPF2='tmp2.csv'
TMPF3='tmp3.csv'
TMPF4="$Dataday.csv"

#配合TDCS檔案名稱格式設定參數
for ((i=0; i<=23; i++))
do
```

```
#let times format to 00 01 02 ~23
Datatimes=$(printf "%02d\n" $i)

#將TDCS M06A下載路徑設定為一個參數
Url="http://tisvcloud.freeway.gov.tw/history/TDCS/M06A/$Dataday/$Datatimes/TDCS_M06A_"
    $Dataday"_"$Datatimes"0000.csv"

#下載TDCS M06A檔案
#download opendate from ETC M06A
wget $Url

#進行檔案的過濾及清洗，去除不必要的欄位
#merge every hour data to all day datas
#only get "VehicleType", "DetectionTime_O", "GantryID_O", "DetectionTime_D", "
    GantryID_D","TripLength"
cat "/home/hpclab/temp/TDCS_M06A_"$Dataday"_"$Datatimes"0000.csv" | awk -F, 'BEGIN{OFS
    ="," }{print "vt"$1,$2,"o"$3,$4,"d"$5,$6}' > $TMPF1
IFS=','

#進行檔案的過濾及清洗，將時間參數取出，為了後續能加入容易辨識的月份及星期名稱
cat "/home/hpclab/temp/TDCS_M06A_"$Dataday"_"$Datatimes"0000.csv" | awk -F, 'BEGIN{OFS
    ="," }{print $2}' > $TMPF2
cat "/home/hpclab/temp/TDCS_M06A_"$Dataday"_"$Datatimes"0000.csv" | awk -F, 'BEGIN{OFS
    ="," }{print $4}' >> $TMPF2
sort $TMPF2 | cut -c1-13 | uniq > $TMPF3

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入交流道資料
while read GantryPoint oChineseName dChineseName Gps
do
sed -i "s/"o"$GantryPoint/$GantryPoint","$oChineseName","$Gps/g" $TMPF1
sed -i "s/"d"$GantryPoint/$GantryPoint","$dChineseName","$Gps/g" $TMPF1
done < /home/hpclab/TDCSPoint.csv

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入車種資料
#find VehicleType add chinese name
while read Vt VehicleType VehicleTypeCN
do
sed -i "s/$Vt/$Datamonth","$Dataweekday","$VehicleType","$VehicleTypeCN/g" $TMPF1
done < /home/hpclab/VehicleType.csv

#進行檔案的過濾及清洗，將容易辨識月份及星期名稱加入資料
while read DetectionTime
do
DataHour=$(echo $DetectionTime | cut -c12-13 )
sed -i "s/$DetectionTime/$DataHour","$DetectionTime/g" $TMPF1
```

```
done < $TMPF3
cat $TMPF1 >> $TMPF4
done

#清除暫存檔案
mv $TMPF4 /home/hpclab/m06a/
rm -rf *.csv
cd ~
```

C.2 getm06b.sh 處理指定日期 M06A 資料

getm06b.sh

```
#/bin/bash

cd /home/hpclab/temp

#配合TDCS檔案名稱格式設定參數
#讀取日期參數
#let date format to 20170720...
Dataday=$1
Dataweekday=$(date -d $Dataday +%a)
Datamonth=$(date -d $Dataday +%B)

#清除暫存檔案
#remove old temp filw
rm -rf *.csv

#設定進行清洗時暫存檔位置
#create new file for row data for all data
TMPF1='tmp1.csv'
TMPF2='tmp2.csv'
TMPF3='tmp3.csv'
TMPF4="$Dataday.csv"

#配合TDCS檔案名稱格式設定參數
for ((i=0; i<=23; i++))
do
#let times format to 00 01 02 ~23
Datatimes=$(printf "%02d\n" $i)

#將TDCS M06A下載路徑設定為一個參數
Url="http://tisvcloud.freeway.gov.tw/history/TDCS/M06A/$Dataday/$Datatimes/TDCS_M06A_"
    $Dataday"_"$Datatimes"0000.csv"

#下載TDCS M06A檔案
#download opendate from ETC M06A
wget $Url

#進行檔案的過濾及清洗，去除不必要的欄位
#merge every hour data to all day datas
#only get "VehicleType", "DetectionTime_O", "GantryID_O", "DetectionTime_D", "
    GantryID_D","TripLength"
```

```
cat "/home/hpclab/temp/TDCS_M06A_"$Dataday"_"$Datatimes"0000.csv" | awk -F, 'BEGIN{OFS
    =","}{print "vt"$1,$2,"o"$3,$4,"d"$5,$6}' > $TMPF1
IFS=','

#進行檔案的過濾及清洗，將時間參數取出，為了後續能加入容易辨識的月份及星期名稱
cat "/home/hpclab/temp/TDCS_M06A_"$Dataday"_"$Datatimes"0000.csv" | awk -F, 'BEGIN{OFS
    =","}{print $2}' > $TMPF2
cat "/home/hpclab/temp/TDCS_M06A_"$Dataday"_"$Datatimes"0000.csv" | awk -F, 'BEGIN{OFS
    =","}{print $4}' >> $TMPF2
sort $TMPF2 | cut -c1-13 | uniq > $TMPF3

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入交流道資料
while read GantryPoint oChineseName dChineseName Gps
do
sed -i "s/"o"$GantryPoint/$GantryPoint","$oChineseName","$Gps/g" $TMPF1
sed -i "s/"d"$GantryPoint/$GantryPoint","$dChineseName","$Gps/g" $TMPF1
done < /home/hpclab/TDCSPoint.csv

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入車種資料
#find VehicleType add chinese name
while read Vt VehicleType VehicleTypeCN
do
sed -i "s/$Vt/$Datamonth","$Dataweekday","$VehicleType","$VehicleTypeCN/g" $TMPF1
done < /home/hpclab/VehicleType.csv

#進行檔案的過濾及清洗，將容易辨識月份及星期名稱加入資料
while read DetectionTime
do
DataHour=$(echo $DetectionTime | cut -c12-13 )
sed -i "s/$DetectionTime/$DataHour","$DetectionTime/g" $TMPF1
done < $TMPF3
cat $TMPF1 >> $TMPF4
done

#清除暫存檔案
mv $TMPF4 /home/hpclab/m06a/
rm -rf *.csv
cd ~
```

C.3 gettarm06a.sh 處理已封存的指定日期資料

gettarm06a.sh

```
#/bin/bash

cd /home/hpclab/tarm06a

#配合TDCS檔案名稱格式設定參數
#讀取日期參數
#let date format to 20170720...
Dataday=$1
Dataweekday=$(date -d $Dataday +%a)
Datamonth=$(date -d $Dataday +%B)

#清除暫存檔案
#remove old temp filw
rm -rf *.csv

#設定進行清洗時暫存檔位置
#create new file for row data for all data
TMPF1='tmp1.csv'
TMPF2='tmp2.csv'
TMPF3='tmp3.csv'
TMPF4="$Dataday.csv"
filename="M06A_$Dataday.tar.gz"

#將TDCS M06A下載路徑設定為一個參數
Url="http://tisvcloud.freeway.gov.tw/history/TDCS/M06A/"$filename

#下載TDCS M06A檔案
#download opendate from ETC M06A
wget $Url

#解壓縮TDCS M06A資料，並過濾含有csv字眼資料輸出到tarlist檔案
tar -xzvf $filename | grep csv > tarlist.txt

#讀取tarlis檔案，一一將檔案進行清洗
while read mfilepwd
do

#進行檔案的過濾及清洗，去除不必要的欄位
#merge every hour data to all day datas
#only get "VehicleType", "DetectionTime_O", "GantryID_O", "DetectionTime_D", "
    GantryID_D","TripLength"
```

```
cat $mfilepwd | awk -F, 'BEGIN{OFS=","}{print "vt"$1,$2,"o"$3,$4,"d"$5,$6}' > $TMPF1
#find GantryPoint add ChineseName,Gps
IFS=','

#進行檔案的過濾及清洗，將時間參數取出，為了後續能加入容易辨識的月份及星期名稱
cat $mfilepwd | awk -F, 'BEGIN{OFS=","}{print $2}' > $TMPF2
cat $mfilepwd | awk -F, 'BEGIN{OFS=","}{print $4}' >> $TMPF2
sort $TMPF2 | cut -c1-13 | uniq > $TMPF3

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入交流道資料
while read GantryPoint oChineseName dChineseName Gps
do
sed -i "s/"o"$GantryPoint/$GantryPoint","$oChineseName","$Gps/g" $TMPF1
sed -i "s/"d"$GantryPoint/$GantryPoint","$dChineseName","$Gps/g" $TMPF1
done < /home/hpclab/TDCSPoint.csv

#進行檔案的過濾及清洗，將容易辨識的中文名稱加入車種資料
#find VehicleType add chinese name
while read Vt VehicleType VehicleTypeCN
do
sed -i "s/$Vt/$Datamonth","$Dataweekday","$VehicleType","$VehicleTypeCN/g" $TMPF1
done < /home/hpclab/VehicleType.csv

#進行檔案的過濾及清洗，將容易辨識月份及星期名稱加入資料
while read DetectionTime
do
DataHour=$(echo $DetectionTime | cut -c12-13 )
sed -i "s/$DetectionTime/$DataHour","$DetectionTime/g" $TMPF1
done < $TMPF3
cat $TMPF1 >> $TMPF4
done < tarlist.txt

#清除暫存檔案
mv $TMPF4 /home/hpclab/m06a/
rm -rf *.csv
rm -rf var/
rm -rf *.gz
cd ~
```

附錄 D

Logstash configuration file

logstash getTDCS.conf

```
input {
        file {
                #設定讀取M05A檔案的目錄位置
                path => "/home/hpcuser/m05a/*.csv"
                start_position => beginning
                type => "m05a"
                }
        file {
                #設定讀取M06A檔案的目錄位置
                path => "/home/hpcuser/TDCS/*.csv"
                start_position => beginning
                type => "m06a"
                }
        }

filter{
        #分別依M05A或M06A類別進行資料格式轉換及欄位參數設定
        if [type] == "m05a"{
                csv {
                        columns => ["TimeInterval", "DataMonth", "DataWeekday", "
    GantryPoint", "ChineseName", "geoip.location", "VehicleType", "SpaceMeanSpeed", "
    TrafficVolume"]
                        }
                date {
                        match => [ "TimeInterval", "yyyy/MM/dd HH:mm" ]
                        timezone => "Asia/Taipei"
                        target => "@timestamp"
```

```
                        }
                mutate{
                        convert => ["SpaceMeanSpeed", "integer"]
                        convert => ["TrafficVolume", "integer"]
                        }
        }
        if [type] == "m06a"{
                csv {
                columns => ["DataMonth", "DataWeekday", "VehicleType", "VehicleCN", "
    DataHour_O", "DetectionTime_O", "GantryID_O", "GantryCN_O", "GpsLocation_O", "
    DataHour_D", "DetectionTime_D", "GantryID_D", "GantryCN_D", "GpsLocation_D", "
    TripLength"]
                        }
                date {
                        match => [ "DetectionTime_D", "yyyy-MM-dd HH:mm:ss" ]
                        timezone => "Asia/Taipei"
                        target => "@timestamp"
                        }
                mutate{
                        convert => ["TripLength", "integer"]
                        convert => ["DataHour_O", "integer"]
                        convert => ["DataHour_D", "integer"]
                        }
                }
        }

output {
        #分別依M05A或M06A類別輸出到不同的資料表
        if [type] == "m05a"{
                elasticsearch {
                        hosts => "192.168.56.103:9200"
                        index => "logstash-%{type}-%{+YYYY.MM.dd}"
                        document_type => "%{type}"
                        }
                }
        if [type] == "m06a"{
                elasticsearch {
                        hosts => "192.168.56.103:9200"
                        index => "logstash-%{type}-%{+YYYY.MM.dd}"
                        document_type => "%{type}"
                        }
                }
        }
```

附錄 E

TDCSPoint file

TDCSPoint.csv 是由國道計費門架座標及里程牌價表清洗過後的資料，主要保留偵測站編號、上下流交流道名稱及偵測站座標經緯度。

```
01F0005S,基隆端(基隆港),基隆(長庚醫院),"25.11878611,121.7317639"
01F0017S,基隆(長庚醫院),八堵,"25.10956667,121.7259056"
01F0029S,八堵,大華系統(連接台62),"25.10311111,121.7173694"
01F0061S,大華系統(連接台62),五堵,"25.08839444,121.6934833"
01F0099S,五堵,汐止、汐止系統(連接國3),"25.07626389,121.6594778"
01F0147S,汐止、汐止系統(連接國3),東湖,"25.06564444,121.6135194"
01F0155S,東湖,內湖(南京東、成功路),"25.06475278,121.6059028"
01F0182S,內湖(南京東、成功路),圓山(建國北路),"25.06898611,121.5802306"
01F0248S,圓山(建國北路),台北(重慶北、士林),"25.07763056,121.5166028"
01F0264S,台北(重慶北、士林),三重,"25.07746389,121.5013861"
01F0293S,三重,五股,"25.07479722,121.4729611"
01F0339S,五股,高公局,"25.068239,121.428764"
01F0376S,高公局,林口(文化一路),"25.058667,121.397306"
01F0413S,林口(文化一路),林口(文化北路),"25.065081,121.363444"
01F0467S,林口(文化北路),桃園,"25.047075,121.317761"
01F0509S,桃園,機場系統(連接國2),"25.027206,121.283072"
01F0532S,機場系統(連接國2),中壢服務區,"25.012944,121.266194"
01F0557S,中壢服務區,內壢,"24.99858333,121.2477222"
01F0578S,內壢,中壢,"24.984414,121.233197"
01F0633S,中壢,平鎮系統(連接台66),"24.949281,121.195594"
01F0664S,平鎮系統(連接台66),幼獅,"24.929544,121.176919"
01F0681S,幼獅,楊梅,"24.917967,121.167586"
01F0699S,楊梅,校前路,"24.901297,121.154230"
01F0750S,校前路,湖口,"24.893436,121.111731"
```

01F0880S,湖口,竹北,"24.848719,121.019031"
01F0928S,竹北,新竹(光復路),"24.808453,121.010306"
01F0950S,新竹(光復路),新竹(科學園區),"24.789622,121.005247"
01F0980S,新竹(科學園區),新竹系統(連接國3),"24.763586,120.998861"
01F1045S,新竹系統(連接國3),頭份,"24.727131,120.951897"
01F1123S,頭份,頭屋,"24.67931667,120.9042667"
01F1292S,頭屋,苗栗,"24.550442,120.838925"
01F1389S,苗栗,銅鑼,"24.485919,120.781903"
01F1465S,銅鑼,三義,"24.420514,120.775731"
01F1572S,三義,后里,"24.339147,120.72085"
01F1621S,后里,台中系統(連接國4),"24.301186,120.697325"
01F1664S,台中系統(連接國4),豐原,"24.262647,120.691956"
01F1725S,豐原,大雅,"24.212956,120.671375"
01F1774S,大雅,台中(台灣大道),"24.184592,120.635611"
01F1802S,台中(台灣大道),南屯,"24.163706,120.621933"
01F1839S,南屯,王田,"24.130767,120.617247"
01F1906S,王田,彰化系統(連接國3),"24.113572,120.568131"
01F1960S,彰化系統(連接國3),彰化,"24.085622,120.52785"
01F2011S,彰化,埔鹽系統(連接台76),"24.040639,120.522786"
01F2089S,埔鹽系統(連接台76),員林,"23.971794,120.506117"
01F2156S,員林,北斗,"23.912769,120.49765"
01F2249S,北斗,西螺,"23.830569,120.484881"
01F2322S,西螺,虎尾(斗六聯絡道),"23.766328,120.469342"
01F2394S,虎尾(斗六聯絡道),斗南(158縣道),"23.702603,120.474917"
01F2425S,斗南(158縣道),雲林系統(連接台78),"23.677375,120.462281"
01F2483S,雲林系統(連接台78),大林,"23.629158,120.440264"
01F2514S,大林,民雄,"23.601989,120.435417"
01F2603S,民雄,嘉義,"23.526739,120.405583"
01F2674S,嘉義,水上,"23.469403,120.3777"
01F2714S,水上,嘉義系統(連接台82),"23.436367,120.361561"
01F2827S,嘉義系統(連接台82),新營服務區,"23.347469,120.322192"
01F2866S,新營服務區,新營,"23.320994,120.298703"
01F2930S,新營,下營系統(連接台84),"23.272589,120.265769"
01F3019S,下營系統(連接台84),麻豆,"23.205089,120.236675"
01F3083S,麻豆,安定,"23.148131,120.231303"
01F3126S,安定,台南系統(連接國8),"23.104817,120.247681"
01F3185S,台南系統(連接國8),永康,"23.051311,120.253025"
01F3227S,永康,大灣,"23.014172,120.249997"
01F3252S,大灣,台南(仁德),"22.991814,120.248175"
01F3286S,台南(仁德),仁德系統(連接台86),"22.960903,120.249822"
01F3366S,仁德系統(連接台86),路竹,"22.892619,120.27215"
01F3398S,路竹,高科,"22.867419,120.284989"
01F3460S,高科,岡山,"22.818039,120.312731"
01F3525S,岡山,楠梓(旗楠路),"22.761672,120.330806"
01F3561S,楠梓(旗楠路),楠梓(鳳楠路),"22.728956,120.333764"

01F3590S,楠梓(鳳楠路),鼎金系統(連接國10、大中路),"22.703525,120.328219"
01F3640S,鼎金系統(連接國10、大中路),高雄(九如、建國路),"22.658892,120.332081"
01F3676S,高雄(九如、建國路),高雄(中正、三多路),"22.627819,120.336222"
01F3686S,高雄(中正、三多路),瑞隆路南下出口匝道,"22.618206,120.336853"
01F3736S,高雄端(中山四路),高雄端(新生路),"22.581933,120.32315"
01F0005N,基隆(長庚醫院),基隆端(基隆港),"25.11831111,121.7316361"
01F0017N,八堵,基隆(長庚醫院),"25.10956667,121.7259056"
01F0029N,大華系統(連接台62),八堵,"25.10311111,121.7173694"
01F0061N,五堵,大華系統(連接台62),"25.08839444,121.6934833"
01F0099N,汐止、汐止系統(連接國3),五堵,"25.07604444,121.6594222"
01F0147N,東湖,汐止、汐止系統(連接國3),"25.06564444,121.6135194"
01F0155N,內湖(南京東、成功路),東湖,"25.06475278,121.6059028"
01F0213N,圓山(建國北路),內湖(南京東、成功路),"25.073053,121.550358"
01F0233N,台北(重慶北、士林),圓山(建國北路),"25.07301944,121.5307028"
01F0256N,三重,台北(重慶北、士林),"25.07806667,121.5091056"
01F0293N,五股,三重,"25.07479722,121.4729611"
01F0339N,高公局,五股,"25.068239,121.428764"
01F0376N,林口(文化一路),高公局,"25.058667,121.397306"
01F0413N,林口(文化北路),林口(文化一路),"25.065081,121.363444"
01F0467N,桃園,林口(文化北路),"25.047075,121.317761"
01F0509N,機場系統(連接國2),桃園,"25.027206,121.283072"
01F0532N,中壢服務區,機場系統(連接國2),"25.012944,121.266194"
01F0557N,內壢,中壢服務區,"24.998583,121.247722"
01F0584N,中壢,內壢,"24.980558,121.228944"
01F0633N,平鎮系統(連接台66),中壢,"24.949281,121.195594"
01F0664N,幼獅,平鎮系統(連接台66),"24.929544,121.176919"
01F0681N,楊梅,幼獅,"24.917967,121.167586"
01F0699N,校前路,楊梅,"24.901297,121.154230"
01F0750N,湖口,校前路,"24.893436,121.111731"
01F0880N,竹北,湖口,"24.848719,121.019031"
01F0928N,新竹(光復路),竹北,"24.808453,121.010306"
01F0956N,新竹(科學園區),新竹(光復路),"24.783944,121.003725"
01F0979N,新竹系統(連接國3),新竹(科學園區),"24.764414,120.999897"
01F1045N,頭份,新竹系統(連接國3),"24.727131,120.951897"
01F1123N,頭屋,頭份,"24.67931667,120.9042667"
01F1292N,苗栗,頭屋,"24.550442,120.838925"
01F1389N,銅鑼,苗栗,"24.485919,120.781903"
01F1465N,三義,銅鑼,"24.420514,120.775731"
01F1572N,后里,三義,"24.339147,120.72085"
01F1621N,台中系統(連接國4),后里,"24.301186,120.697325"
01F1664N,豐原,台中系統(連接國4),"24.262647,120.691956"
01F1725N,大雅,豐原,"24.212956,120.671375"
01F1774N,台中(台灣大道),大雅,"24.184592,120.635611"
01F1802N,南屯,台中(台灣大道),"24.163706,120.621933"
01F1839N,王田,南屯,"24.130767,120.617247"

01F1906N,彰化系統(連接國3),王田,"24.113572,120.568131"
01F1960N,彰化,彰化系統(連接國3),"24.085622,120.52785"
01F2011N,埔鹽系統(連接台76),彰化,"24.040639,120.522786"
01F2089N,員林,埔鹽系統(連接台76),"23.971794,120.506117"
01F2156N,北斗,員林,"23.912769,120.49765"
01F2249N,西螺,北斗,"23.830569,120.484881"
01F2322N,虎尾(斗六聯絡道),西螺,"23.766328,120.469342"
01F2394N,斗南(158縣道),虎尾(斗六聯絡道),"23.702603,120.474917"
01F2425N,雲林系統(連接台78),斗南(158縣道),"23.677375,120.462281"
01F2483N,大林,雲林系統(連接台78),"23.629158,120.440264"
01F2514N,民雄,大林,"23.601989,120.435417"
01F2603N,嘉義,民雄,"23.526739,120.405583"
01F2674N,水上,嘉義,"23.469403,120.3777"
01F2714N,嘉義系統(連接台82),水上,"23.436367,120.361561"
01F2827N,新營服務區,嘉義系統(連接台82),"23.347469,120.322192"
01F2866N,新營,新營服務區,"23.320994,120.298703"
01F2930N,下營系統(連接台84),新營,"23.272589,120.265769"
01F3019N,麻豆,下營系統(連接台84),"23.205089,120.236675"
01F3083N,安定,麻豆,"23.148131,120.231303"
01F3126N,台南系統(連接國8),安定,"23.104817,120.247681"
01F3185N,永康,台南系統(連接國8),"23.051311,120.253025"
01F3227N,大灣,永康,"23.014172,120.249997"
01F3252N,台南(仁德),大灣,"22.991814,120.248175"
01F3286N,仁德系統(連接台86),台南(仁德),"22.960903,120.249822"
01F3366N,路竹,仁德系統(連接台86),"22.892619,120.27215"
01F3398N,高科,路竹,"22.867419,120.284989"
01F3460N,岡山,高科,"22.818039,120.312731"
01F3525N,楠梓(旗楠路),岡山,"22.761672,120.330806"
01F3559N,楠梓(鳳楠路),楠梓(旗楠路),"22.730903,120.334286"
01F3590N,鼎金系統(連接國10、大中路),楠梓(鳳楠路),"22.703525,120.328219"
01F3640N,高雄(九如、建國路),鼎金系統(連接國10、大中路),"22.658892,120.332081"
01F3676N,高雄(中正、三多路),高雄(九如、建國路),"22.627819,120.336222"
01F3696N,瑞隆路南下出口匝道,高雄(中正、三多路),"22.609428,120.339608"
01F3736N,高雄端(新生路),高雄端(中山四路),"22.581933,120.32315"
01H0163S,汐止、汐止系統(連接國1平面),堤頂,"25.064006,121.598733"
01H0206S,堤頂,環北,"25.07275,121.557739"
01H0305S,環北,五股(高架),"25.072747,121.450939"
01H0334S,五股(高架),高公局,"25.068728,121.433281"
01H0447S,高公局,機場系統(連接國2),"25.060125,121.331858"
01H0579S,機場系統(連接國2),中壢,"24.984094,121.232531"
01H0610S,中壢,楊梅,"24.96525,121.209867"
01H0174N,堤頂,汐止、汐止系統(連接國1平面),"25.065544,121.588044"
01H0200N,下塔悠北上出口匝道,堤頂,"25.070994,121.562519"
01H0208N,環北,下塔悠北上出口匝道,"25.072261,121.555014"
01H0271N,五股(高架),環北,"25.075256,121.494019"

01H0333N,高公局,五股(高架),"25.068147,121.435133"
01H0447N,機場系統(連接國2),高公局,"25.059242,121.332072"
01H0579N,中壢,機場系統(連接國2),"24.983667,121.232981"
01H0608N,楊梅,中壢,"24.965878,121.211614"
03F0006S,基金,瑪東系統(連接台62),"25.137106,121.708908"
03F0087S,瑪東系統(連接台62),汐止系統(連接國1),"25.089994,121.653556"
03F0116S,汐止系統(連接國1),新台五路,"25.065611,121.643467"
03F0136S,新台五路,南港(連接環東大道),"25.052742,121.631281"
03F0158S,南港(連接環東大道),南港系統(連接國五),"25.036533,121.621386"
03F0201S,南港系統(連接國五),木柵(連接國3甲),"25.007436,121.595981"
03F0217S,木柵(連接國3甲),新店(中興路),"24.993567,121.592933"
03F0301S,新店(中興路),安坑(中央、安康路),"24.96255,121.530572"
03F0337S,安坑(中央、安康路),中和(連接台64),"24.978881,121.502381"
03F0394S,中和(連接台64),土城(連接台65),"24.975247,121.454925"
03F0447S,土城(連接台65),樹林,"24.950503,121.412681"
03F0498S,樹林,三鶯,"24.943825,121.365275"
03F0525S,三鶯,鶯歌系統(連接國2),"24.936025,121.3419"
03F0559S,鶯歌系統(連接國2),大溪(連接台66),"24.930225,121.312458"
03F0648S,大溪(連接台66),龍潭,"24.878792,121.250839"
03F0698S,龍潭,關西服務區,"24.856558,121.208581"
03F0783S,關西服務區,關西,"24.797339,121.179128"
03F0846S,關西,竹林,"24.791439,121.125997"
03F0961S,竹林,寶山,"24.758442,121.030425"
03F0996S,寶山,新竹系統(連接國1),"24.755811,120.997269"
03F1022S,新竹系統(連接國1),茄苳,"24.757914,120.972267"
03F1051S,茄苳,香山,"24.753094,120.945675"
03F1128S,香山,西濱(連接台61),"24.711383,120.889639"
03F1161S,西濱(連接台61),竹南,"24.698006,120.860086"
03F1215S,竹南,大山,"24.655964,120.838242"
03F1257S,大山,後龍,"24.627161,120.814006"
03F1332S,後龍,西湖服務區,"24.579642,120.764417"
03F1395S,西湖服務區,通霄,"24.531125,120.733253"
03F1485S,通霄,苑裡,"24.467308,120.681217"
03F1633S,苑裡,大甲,"24.341119,120.651567"
03F1651S,大甲,中港系統(連接國4),"24.334719,120.635578"
03F1710S,中港系統(連接國4),清水服務區,"24.292881,120.601964"
03F1739S,清水服務區,沙鹿,"24.269464,120.590806"
03F1779S,沙鹿,龍井,"24.234647,120.593306"
03F1860S,龍井,和美,"24.177831,120.548244"
03F1944S,和美,彰化系統(連接國1),"24.119011,120.536692"
03F1991S,彰化系統(連接國1),快官(連接台74),"24.102808,120.577661"
03F2066S,快官(連接台74),烏日,"24.070333,120.639711"
03F2079S,烏日,中投(連接台63),"24.067778,120.652361"
03F2100S,中投(連接台63),霧峰(台3線、太平-台74),"24.060653,120.670964"
03F2129S,霧峰(台3線、太平-台74),霧峰系統(連接國6),"24.037392,120.675775"

03F2152S,霧峰系統(連接國6),草屯,"24.019992,120.663222"
03F2194S,草屯,中興系統(台76-八卦山隧道),"23.985844,120.652714"
03F2235S,中興系統(台76-八卦山隧道),中興,"23.955503,120.661267"
03F2261S,中興,南投,"23.9376,120.677039"
03F2306S,南投,南投服務區,"23.906781,120.702828"
03F2336S,南投服務區,名間,"23.882031,120.709186"
03F2415S,名間,竹山,"23.814836,120.700003"
03F2447S,竹山,南雲,"23.787486,120.704519"
03F2535S,南雲,斗六,"23.775581,120.6259"
03F2614S,斗六,古坑(文化路)、古坑系統(連接台78),"23.724425,120.601786"
03F2709S,古坑(文化路)、古坑系統(連接台78),古坑(朝陽路),"23.649142,120.57295"
03F2747S,古坑(朝陽路),古坑服務區,"23.618203,120.558567"
03F2777S,古坑服務區,梅山,"23.606792,120.532511"
03F2840S,梅山,竹崎(竹崎連絡道),"23.558125,120.505061"
03F2899S,竹崎(竹崎連絡道),竹崎(159縣道),"23.511683,120.486328"
03F2923S,竹崎(159縣道),中埔,"23.492078,120.494586"
03F2985S,中埔,水上系統(連接台82),"23.442486,120.4812"
03F3101S,水上系統(連接台82),白河,"23.361253,120.436356"
03F3187S,白河,東山服務區,"23.296692,120.404164"
03F3211S,東山服務區,柳營,"23.279281,120.390253"
03F3259S,柳營,烏山頭,"23.244994,120.367378"
03F3307S,烏山頭,官田系統(連接台84),"23.203275,120.355461"
03F3392S,官田系統(連接台84),善化,"23.132064,120.333711"
03F3445S,善化,新化系統(連接國8),"23.085272,120.329033"
03F3496S,新化系統(連接國8),關廟(連接台86),"23.040203,120.325172"
03F3588S,關廟(連接台86),關廟服務區,"22.969758,120.340164"
03F3670S,關廟服務區,田寮,"22.901969,120.354767"
03F3743S,田寮,燕巢系統(連接國10),"22.845122,120.385428"
03F3854S,燕巢系統(連接國10),九如,"22.769197,120.441839"
03F3916S,九如,長治,"22.749792,120.496839"
03F4018S,長治,麟洛,"22.682775,120.563086"
03F4142S,麟洛,竹田系統(連接台88),"22.579586,120.526161"
03F4168S,竹田系統(連接台88),崁頂,"22.557186,120.524558"
03F4232S,崁頂,南州,"22.501367,120.525775"
03F4263S,南州,林邊(大鵬灣端),"22.474661,120.530844"
03F0021N,瑪東系統(連接台62),基金,"25.129514,121.698631"
03F0054N,汐止系統(連接國1),瑪東系統(連接台62),"25.103747,121.681789"
03F0116N,新台五路,汐止系統(連接國1),"25.065611,121.643467"
03F0140N,南港(連接環東大道),新台五路,"25.050319,121.628156"
03F0150N,南港系統(連接國五)、南深路出口匝道,南港(連接環東大道),"25.042389,121.624789"
03F0201N,木柵(連接國3甲),南港系統(連接國五)、南深路出口匝道,"25.007375,121.596381"
03F0217N,新店(中興路),木柵(連接國3甲),"24.993744,121.593367"
03F0301N,安坑(中央、安康路),新店(中興路),"24.96255,121.530572"
03F0338N,中和(連接台64),安坑(中央、安康路),"24.978586,121.501744"
03F0394N,土城(連接台65),中和(連接台64),"24.975247,121.454925"

03F0447N,樹林,土城(連接台65),"24.950503,121.412681"
03F0498N,三鶯,樹林,"24.943825,121.365275"
03F0525N,鶯歌系統(連接國2),三鶯,"24.936025,121.3419"
03F0559N,大溪(連接台66),鶯歌系統(連接國2),"24.930225,121.312458"
03F0648N,龍潭,大溪(連接台66),"24.878792,121.250839"
03F0698N,關西服務區,龍潭,"24.856558,121.208581"
03F0783N,關西,關西服務區,"24.797339,121.179128"
03F0846N,竹林,關西,"24.791439,121.125997"
03F0961N,寶山,竹林,"24.758442,121.030425"
03F0996N,新竹系統(連接國1),寶山,"24.755811,120.997269"
03F1022N,茄苳,新竹系統(連接國1),"24.757914,120.972267"
03F1051N,香山,茄苳,"24.753094,120.945675"
03F1128N,西濱(連接台61),香山,"24.711383,120.889639"
03F1161N,竹南,西濱(連接台61),"24.698006,120.860086"
03F1215N,大山,竹南,"24.655964,120.838242"
03F1257N,後龍,大山,"24.627161,120.814006"
03F1332N,西湖服務區,後龍,"24.579642,120.764417"
03F1395N,通霄,西湖服務區,"24.531125,120.733253"
03F1485N,苑裡,通霄,"24.467308,120.681217"
03F1633N,大甲,苑裡,"24.341119,120.651567"
03F1651N,中港系統(連接國4),大甲,"24.334719,120.635578"
03F1710N,清水服務區,中港系統(連接國4),"24.292881,120.601964"
03F1739N,沙鹿,清水服務區,"24.269464,120.590806"
03F1779N,龍井,沙鹿,"24.234647,120.593306"
03F1860N,和美,龍井,"24.177831,120.548244"
03F1941N,彰化系統(連接國1),和美,"24.120947,120.534547"
03F1992N,快官(連接台74),彰化系統(連接國1),"24.102525,120.578747"
03F2066N,烏日,快官(連接台74),"24.070333,120.639711"
03F2078N,中投(連接台63),烏日,"24.067958,120.651433"
03F2100N,霧峰(台3線、太平-台74),中投(連接台63),"24.060428,120.671533"
03F2125N,霧峰系統(連接國6),霧峰(台3線、太平-台74),"24.040994,120.677853"
03F2153N,草屯,霧峰系統(連接國6),"24.01894444,120.6631611"
03F2194N,中興系統(台76-八卦山隧道),草屯,"23.985844,120.652714"
03F2231N,中興,中興系統(台76-八卦山隧道),"23.957103,120.657614"
03F2260N,南投,中興,"23.937928,120.676853"
03F2306N,南投服務區,南投,"23.906781,120.702828"
03F2336N,名間,南投服務區,"23.882031,120.709186"
03F2415N,竹山,名間,"23.814836,120.700003"
03F2447N,南雲,竹山,"23.787486,120.704519"
03F2535N,斗六,南雲,"23.775581,120.6259"
03F2614N,古坑(文化路)&古坑系統(連接台78),斗六,"23.724425,120.601786"
03F2709N,古坑(朝陽路),古坑(文化路)、古坑系統(連接台78),"23.64927778,120.5729111"
03F2747N,古坑服務區,古坑(朝陽路),"23.618203,120.558567"
03F2777N,梅山,古坑服務區,"23.606792,120.532511"
03F2840N,竹崎(竹崎連絡道),梅山,"23.558125,120.505061"

03F2899N,竹崎(159縣道),竹崎(竹崎連絡道),"23.511683,120.486328"
03F2923N,中埔,竹崎(159縣道),"23.492078,120.494586"
03F2985N,水上系統(連接台82),中埔,"23.442486,120.4812"
03F3101N,白河,水上系統(連接台82),"23.361253,120.436356"
03F3187N,東山服務區,白河,"23.296692,120.404164"
03F3211N,柳營,東山服務區,"23.279281,120.390253"
03F3259N,烏山頭,柳營,"23.244994,120.367378"
03F3307N,官田系統(連接台84),烏山頭,"23.203275,120.355461"
03F3392N,善化,官田系統(連接台84),"23.132064,120.333711"
03F3445N,新化系統(連接國8),善化,"23.085272,120.329033"
03F3496N,關廟(連接台86),新化系統(連接國8),"23.040203,120.325172"
03F3588N,關廟服務區,關廟(連接台86),"22.969758,120.340164"
03F3670N,田寮,關廟服務區,"22.901969,120.354767"
03F3743N,燕巢系統(連接國10),田寮,"22.845122,120.385428"
03F3854N,九如,燕巢系統(連接國10),"22.769197,120.441839"
03F3916N,長治,九如,"22.749792,120.496839"
03F4021N,麟洛,長治,"22.680347,120.563311"
03F4142N,竹田系統(連接台88),麟洛,"22.579586,120.526161"
03F4168N,崁頂,竹田系統(連接台88),"22.557408,120.524761"
03F4232N,南州,崁頂,"22.501367,120.525775"
03F4259N,林邊(大鵬灣端),南州,"22.477464,120.531386"
03A0015S,台北端,萬芳(動物園、信義快),"25.012144,121.562128"
03A0041S,萬芳(動物園、信義快),木柵(連接國3),"25.003864,121.586039"
03A0015N,萬芳(動物園、信義快),台北端,"25.012144,121.562128"
03A0041N,木柵(連接國3),萬芳(動物園、信義快),"25.003864,121.586039"
05F0000S,南港系統(連接國3),石碇,"25.03508889,121.6229306"
05F0055S,石碇,坪林行控專用道,"24.99646667,121.6520583"
05F0287S,坪林行控專用道,頭城,"24.84256944,121.7889667"
05F0309S,頭城,宜蘭(四城、大福),"24.82370556,121.7862139"
05FR113S,宜蘭(四城、大福),宜蘭(壯圍),"24.73394444,121.7819889"
05F0439S,宜蘭(壯圍),羅東,"24.71102778,121.7894778"
05F0494S,羅東,蘇澳,"24.66258056,121.79985"
05F0001N,石碇,南港系統(連接國3),"25.03497222,121.6248611"
05F0055N,坪林行控專用道,石碇,"24.99615556,121.6520972"
05F0287N,頭城,坪林行控專用道,"24.84263889,121.7892861"
05F0309N,宜蘭(四城、大福),頭城,"24.82370556,121.7862139"
05FR143N,宜蘭(壯圍),宜蘭(四城、大福),"24.73391667,121.78235"
05F0438N,羅東,宜蘭(壯圍),"24.71092222,121.7895972"
05F0528N,蘇澳,羅東,"24.63271667,121.8071667"

附錄 F

MariaDB 與 phpMyAdmin 安裝步驟

步驟 1. 更新 vim 避免舊版 vim 上下左右鍵異常，為時間錯亂安裝 NTP。

```
sudo apt-get update
sudo apt-get install -y vim ntp curl ssh
```

步驟 2. 將所有主機加入本機 IP 對應表。

```
sudo vim /etc/hosts
#加入下列參數
192.168.56.101 master
```

步驟 3. 更新套件庫，安裝 MariaDB、Apache、PHP 等套件。

```
sudo apt-get update
sudo apt install mariadb-server mariadb-client apache2 php7.0 libapache2-mod-php7.0
```

步驟 4.MariaDB 參數調整，主要為了使 phpMyAdmin 可以順利運作及修改資料庫 root 密碼。

```
sudo mysqladmin -u root password
#變更允許Mariadb root登入phpmyadmin
sudo mysql -u root -p mysql

#登入MariaDB調整權限設定，並新增phpmyadmin管理帳號
```

```
update user set plugin='' where user='root';
flush privileges;
create database phpmyadmin;
grant all on phpmyadmin.* to 'phpmyadmin'@'localhost' identified by '123456' with
    grant option;
exit;
```

步驟 5. 更新套件庫，安裝 phpMyAdmin 套件。

```
sudo apt update
sudo apt-get install phpmyadmin
```

步驟 6. 開啟瀏覽器輸入 http://192.168.56.101/phpmyadmin 確認運作狀況。

附錄 G

Elasticsearch 與 MariaDB 簡易查詢測試數據

Table G.1: 總資料量 10 萬筆測試結果

搜尋偵測站編號	MarriaDB index	MarriaDB non-index	Elasticsearch
01F1774S	0.003	0.0034	0.005
01F1774N	0.0028	0.0092	0.008
01F1802S	0.0025	0.0037	0.006
01F1802N	0.0004	0.0084	0.008
03F1779S	0.0033	0.0115	0.005
03F1779N	0.0037	0.0149	0.004
03F1710S	0.0003	0.0395	0.006
03F1710N	0.0024	0.0188	0.006
03F1651S	0.0025	0.0115	0.01
03F1651N	0.002	0.0136	0.008

Table G.2: 總資料量 50 萬筆測試結果

搜尋偵測站編號	MarriaDB index	MarriaDB non-index	Elasticsearch
03F2078N	0.0006	0.0315	0.005
03F2079S	0.0009	0.0161	0.007
01F2322S	0.0007	0.0055	0.005
01F2322N	0.001	0.0115	0.006
01F2514N	0.0004	0.0171	0.006
01F2514S	0.0005	0.0038	0.005
03F0006S	0.0006	0.0223	0.008
03F0021N	0.0003	0.0209	0.004
03F2899N	0.0004	0.3629	0.008
03F2899S	0.0003	0.3763	0.008

Table G.3: 總資料量 100 萬筆測試結果

搜尋偵測站編號	MarriaDB index	MarriaDB non-index	Elasticsearch
01F0950S	0.0032	0.0493	0.015
01F0956N	0.0009	0.0333	0.004
01F0509N	0.0007	0.0093	0.008
01F0509S	0.0005	0.0031	0.007
03F0337S	0.0007	0.0148	0.006
03F0338N	0.0005	0.0169	0.003
03F0846N	0.0005	0.0183	0.003
03F0846S	0.0019	0.0154	0.012
03F3187N	0.0006	0.1047	0.01
03F3187S	0.0015	0.1419	0.005

Table G.4: 總資料量 500 萬筆測試結果

搜尋偵測站編號	MarriaDB index	MarriaDB non-index	Elasticsearch
01F0017N	0.0008	0.0197	0.016
01F0017S	0.001	0.0493	0.017
01F0256N	0.0008	0.0103	0.028
01F0264S	0.0006	0.0021	0.019
01F0293N	0.0007	0.0133	0.018
01F0293S	0.0018	0.0016	0.024
03F0648N	0.0007	0.0189	0.014
03F0648S	0.0017	0.0163	0.014
03F2078N	0.0009	0.0321	0.009
03F2079S	0.0008	0.0152	0.009

Table G.5: 總資料量 1000 萬筆測試結果

搜尋偵測站編號	MarriaDB index	MarriaDB non-index	Elasticsearch
03F2260N	0.0008	0.0701	0.053
03F2261S	0.0008	0.0198	0.054
03F3854N	0.001	0.0274	0.019
03F3854S	0.0008	0.0172	0.01
01F3286N	0.0007	0.0149	0.027
01F3286S	0.0009	0.0074	0.005
01F3590S	0.0009	0.0078	0.02
01F3640N	0.0008	0.0149	0.012
01F3686S	0.0007	0.0082	0.029
01F3696N	0.0011	0.0157	0.015

附錄 H

Apache 網站 Html 語法

index.html 是為了隱藏 Kibana 管理介面，以及管理可存取權限所特別獨立出來的網頁，透過 Apache 這個服務也可以讓 Kibana 的圖表更加多元化的呈現，以下是簡單的範例檔。

```
<!DOCTYPE html>
<html>
<head>
<title>
THU HPC LAB ETC Open Data Test
</title>
<link rel="Shortcut Icon" type="image/x-icon" href="hpcicon.ico" />
</head>
<body>
<br>
<iframe src="http://etc.thu.edu.tw:5601/app/kibana#/dashboard/ff4b3400-ecaf-11e7-994a-
    d3e8612df750?embed=true&_g=(refreshInterval%3A(display%3AOff%2Cpause%3A!f%2Cvalue
    %3A0)%2Ctime%3A(from%3Anow-1y%2Cmode%3Aquick%2Cto%3Anow))" height="600" width
    ="100%"></iframe>
<br>
<a href="http://etc.thu.edu.tw/m05aheatmap.html"><img src="m05aheatmap.jpg" alt="
    m05aheatmap" height="20%" width="20%"border="0"></a>
<a href="http://etc.thu.edu.tw/taichung.html"><img src="taichung.jpg" alt="taichung"
    height="20%" width="20%"border="0"></a>
<a href="http://etc.thu.edu.tw/txgweekday.html"><img src="txgweekday.jpg" alt="
    txgweekday" height="20%" width="20%"border="0"></a>
<a href="http://etc.thu.edu.tw/txgholiday.html"><img src="txgholiday.jpg" alt="
    txgholiday" height="20%" width="20%"border="0"></a>
```

```
<br>
<a href="http://etc.thu.edu.tw/carspeed.html"><img src="carspeed.jpg" alt="carspeed"
    height="20%" width="20%"border="0"></a>
<a href="http://etc.thu.edu.tw/txgandhsc.html"><img src="txgandhsc.jpg" alt="txgandhsc
    " height="20%" width="20%"border="0"></a>
<a href="http://etc.thu.edu.tw/topupdown.html"><img src="topupdown.jpg" alt="topupdown
    " height="20%" width="20%"border="0"></a>
</body>
</html>
```

附錄 I

Kibana JSON 語法

Kibana 圖表除了可以使用網頁方式篩選產生，也可以使用 JSON 語法方式產生，下列為此論文車速熱感圖 JSON 語法。

I.1 車速熱感圖 JSON 語法


```
[
{
"_id": "33fa4c30-e94b-11e7-b9bb-dbca33ac88c3",
"_type": "dashboard",
"_source": {
"title": "車速熱感圖",
"hits": 0,
"description": "",
"panelsJSON": "[{\"panelIndex\":\"3\",\"gridData\":{\"x\":0,\"y\":2,\"w\":12,\"h
    \":5,\"i\":\"3\"},\"id\":\"08871920-e94b-11e7-b9bb-dbca33ac88c3\",\"type\":\"
    visualization\",\"version\":\"6.1.1\"},{\"panelIndex\":\"4\",\"gridData\":{\"x
    \":0,\"y\":0,\"w\":12,\"h\":2,\"i\":\"4\"},\"version\":\"6.1.1\",\"type\":\"
    visualization\",\"id\":\"53caae70-ecbf-11e7-994a-d3e8612df750\"}]",
"optionsJSON": "{\"darkTheme\":false,\"hidePanelTitles\":false,\"useMargins\":true}",
"uiStateJSON": "{\"P-3\":{\"mapCenter\":[24.03517651018698,120.57495117187501],\"
    mapZoom\":10,\"spy\":{\"mode\":{\"fill\":false,\"name\":null}}},\"P-4\":{\"vis
    \":{\"params\":{\"sort\":{\"columnIndex\":null,\"direction\":null}}}}}",
"version": 1,
"timeRestore": false,
"kibanaSavedObjectMeta": {
```

```
"searchSourceJSON": "{\"query\":{\"language\":\"lucene\",\"query\":\"\"},\"filter
    \":[{\"$state\":{\"store\":\"appState\"},\"meta\":{\"alias\":\"M05A\",\"disabled
    \":false,\"index\":\"a77b3310-e93f-11e7-b9bb-dbca33ac88c3\",\"key\":\"_type\",\"
    negate\":false,\"params\":{\"query\":\"m05a\",\"type\":\"phrase\"},\"type\":\"
    phrase\",\"value\":\"m05a\"},\"query\":{\"match\":{\"_type\":{\"query\":\"m05a
    \",\"type\":\"phrase\"}}}},{\"$state\":{\"store\":\"appState\"},\"meta\":{\"alias
    \":\"車流量不為0\",\"disabled\":false,\"index\":\"a77b3310-e93f-11e7-b9bb-
    dbca33ac88c3\",\"key\":\"TrafficVolume\",\"negate\":true,\"params\":{\"query
    \":0,\"type\":\"phrase\"},\"type\":\"phrase\",\"value\":\"0\"},\"query\":{\"match
    \":{\"TrafficVolume\":{\"query\":0,\"type\":\"phrase\"}}}},{\"$state\":{\"store
    \":\"appState\"},\"meta\":{\"alias\":\"小客車\",\"disabled\":false,\"index\":\"
    a77b3310-e93f-11e7-b9bb-dbca33ac88c3\",\"key\":\"VehicleType\",\"negate\":false,\"
    params\":{\"query\":\"31\",\"type\":\"phrase\"},\"type\":\"phrase\",\"value
    \":\"31\"},\"query\":{\"match\":{\"VehicleType\":{\"query\":\"31\",\"type\":\"
    phrase\"}}}},{\"$state\":{\"store\":\"appState\"},\"meta\":{\"alias\":\"車速大於0
    \",\"disabled\":false,\"index\":\"a77b3310-e93f-11e7-b9bb-dbca33ac88c3\",\"key
    \":\"SpaceMeanSpeed\",\"negate\":true,\"params\":{\"query\":0,\"type\":\"phrase
    \"},\"type\":\"phrase\",\"value\":\"0\"},\"query\":{\"match\":{\"SpaceMeanSpeed
    \":{\"query\":0,\"type\":\"phrase\"}}}}],\"highlightAll\":true,\"version\":true}"
}
}
}
]
```

I.2 進出台中市車流量統計 JSON 語法


```
[
{
"_id": "7e7bb300-fc62-11e7-918c-3187a4f329f2",
"_type": "dashboard",
"_source": {
"title": "進出台中市車流量統計",
"hits": 0,
"description": "",
"panelsJSON": "[{\"panelIndex\":\"1\",\"gridData\":{\"x\":0,\"y\":2,\"w\":12,\"h
    \":5,\"i\":\"1\"},\"id\":\"77ffa120-fc59-11e7-918c-3187a4f329f2\",\"type\":\"
    visualization\",\"version\":\"6.1.1\"},{\"panelIndex\":\"2\",\"gridData\":{\"x
    \":0,\"y\":7,\"w\":12,\"h\":5,\"i\":\"2\"},\"id\":\"b860da90-fc5e-11e7-918c-3187
    a4f329f2\",\"type\":\"visualization\",\"version\":\"6.1.1\"},{\"panelIndex
    \":\"3\",\"gridData\":{\"x\":0,\"y\":0,\"w\":12,\"h\":2,\"i\":\"3\"},\"id\":\"
    eff22d70-fc62-11e7-918c-3187a4f329f2\",\"type\":\"visualization\",\"version
    \":\"6.1.1\"},{\"panelIndex\":\"4\",\"gridData\":{\"x\":0,\"y\":12,\"w\":12,\"h
    \":4,\"i\":\"4\"},\"version\":\"6.1.1\",\"type\":\"visualization\",\"id\":\"65
    a7bbb0-fc64-11e7-918c-3187a4f329f2\"}]",
"optionsJSON": "{\"darkTheme\":false,\"hidePanelTitles\":false,\"useMargins\":true}",
"uiStateJSON": "{}",
"version": 1,
"timeRestore": false,
"kibanaSavedObjectMeta": {
"searchSourceJSON": "{\"query\":{\"language\":\"lucene\",\"query\":\"\"},\"filter
    \":[],\"highlightAll\":true,\"version\":true}"
}
}
}
]
```

I.3 平常日台中交流道車流狀況 JSON 語法


```
[
{
"_id": "85b6f490-041e-11e8-8a2e-dd0f8815297c",
"_type": "dashboard",
"_source": {
"title": "平常日台中交流道車流狀況",
"hits": 0,
"description": "",
"panelsJSON": "[{\"panelIndex\":\"1\",\"gridData\":{\"x\":0,\"y\":0,\"w\":6,\"h\":3,\"
    i\":\"1\"},\"version\":\"6.1.1\",\"type\":\"visualization\",\"id\":\"b2049820-f3c2
    -11e7-a6a0-9b79a4d8a16b\"},{\"panelIndex\":\"2\",\"gridData\":{\"x\":6,\"y\":0,\"w
    \":6,\"h\":3,\"i\":\"2\"},\"version\":\"6.1.1\",\"type\":\"visualization\",\"id
    \":\"890ae1d0-f3c8-11e7-9e29-1333f142bb2a\"},{\"panelIndex\":\"3\",\"gridData
    \":{\"x\":0,\"y\":3,\"w\":6,\"h\":3,\"i\":\"3\"},\"version\":\"6.1.1\",\"type\":\"
    visualization\",\"id\":\"e05aec80-f447-11e7-a08d-2173df0ff768\"},{\"gridData\":{\"
    w\":6,\"h\":3,\"x\":6,\"y\":3,\"i\":\"4\"},\"version\":\"6.1.1\",\"panelIndex
    \":\"4\",\"type\":\"visualization\",\"id\":\"3f2ec870-f449-11e7-a08d-2173df0ff768
    \"}]",
"optionsJSON": "{\"darkTheme\":false,\"useMargins\":true,\"hidePanelTitles\":false}",
"uiStateJSON": "{}",
"version": 1,
"timeRestore": false,
"kibanaSavedObjectMeta": {
"searchSourceJSON": "{\"query\":{\"query\":\"\",\"language\":\"lucene\"},\"filter
    \":[],\"highlightAll\":true,\"version\":true}"
}
}
}
]
```

I.4 假日台中交流道車流狀況 JSON 語法


```
[
{
"_id": "b1c0c250-041e-11e8-8a2e-dd0f8815297c",
"_type": "dashboard",
"_source": {
"title": "假日台中交流道車流狀況",
"hits": 0,
"description": "",
"panelsJSON": "[{\"panelIndex\":\"1\",\"gridData\":{\"x\":0,\"y\":0,\"w\":6,\"h\":3,\"
    i\":\"1\"},\"version\":\"6.1.1\",\"type\":\"visualization\",\"id\":\"6f100570-f3c4
    -11e7-a6a0-9b79a4d8a16b\"},{\"panelIndex\":\"2\",\"gridData\":{\"x\":6,\"y\":0,\"w
    \":6,\"h\":3,\"i\":\"2\"},\"version\":\"6.1.1\",\"type\":\"visualization\",\"id
    \":\"3b14e290-f3c9-11e7-9e29-1333f142bb2a\"},{\"panelIndex\":\"3\",\"gridData
    \":{\"x\":0,\"y\":3,\"w\":6,\"h\":3,\"i\":\"3\"},\"version\":\"6.1.1\",\"type\":\"
    visualization\",\"id\":\"ac2a6920-f449-11e7-a08d-2173df0ff768\"},{\"gridData\":{\"
    w\":6,\"h\":3,\"x\":6,\"y\":3,\"i\":\"4\"},\"version\":\"6.1.1\",\"panelIndex
    \":\"4\",\"type\":\"visualization\",\"id\":\"faaa7ff0-f448-11e7-a08d-2173df0ff768
    \"}]",
"optionsJSON": "{\"darkTheme\":false,\"useMargins\":true,\"hidePanelTitles\":false}",
"uiStateJSON": "{}",
"version": 1,
"timeRestore": false,
"kibanaSavedObjectMeta": {
"searchSourceJSON": "{\"query\":{\"query\":\"\",\"language\":\"lucene\"},\"filter
    \":[],\"highlightAll\":true,\"version\":true}"
}
}
}
]
```