運用 PySpark 與分散式運算解決預存程序單機大量運算之延遲問題

Using PySpark and Distributed Computing to Solve the Lagging Problem of Stored Procedure with Standalone and Massive Computing

李泰穎(Tai Ying Li) [1], 陳彥宏(Yen Hung Chen)[2]
Department of Computer Science,
University of Taipei, Taiwan
m10416014@go.utaipei.edu.tw[1], yhchen@utaipei.edu.tw[2]

摘要

由於關聯式資料庫管理系統(RDBMS)與結構化查詢語言(SQL，Structured Query Language)所組成的預存程序(Stored Procedure)，在 ACID 與腳本化(Script)的基礎特性上，限制了預存程序對資料的處理，僅能在關聯式資料庫環境中以單機模式運行，即使呼叫或使用其他連結資料庫伺服器(Linked DBs/Servers)上的運算資源，原預存程序仍必須依序等待前一筆交易完成後，才能進行下一筆交易。然而，隨著資料在質量、數量、型態及需求上愈來愈廣泛，尤其當預存程序在處理大量資料如報表計算等必須消耗大量 CPU 與記憶體的工作時，所產生的延遲問題，甚至已經影響到資料庫伺服器內其他資料庫所屬應用程式或系統的正常運作，即使進行軟硬體升級、新增連結資料庫伺服器、運用索引(Index)、分割資料表(Partition)、壓縮資料表(Compression)或設定離峰時間運行等諸多資料庫效能調校動作，在資料量長期累積下，對效能提昇的幫助也相當有限。有鑑於大數據(Big Data)與資料科學的蓬勃發展，本研究運用 Spark 開源叢集運算框架中的 PySpark 與分散式運算(Distributed Computing)，提出具體有效的可行方案，解決預存程序於單機大量運算時所遭遇到的延遲問題。

1 緒論

Spark 是為了快速運行與泛用化目的而生的開源叢集運算框架[1]，目前版本主要包含了 Spark 核心與四個主要套件。其中，Spark SQL 資料處理套件搭配 Spark 核心功能如工作排程、記憶體管理、容錯及檔案系統互動等元件，正為本研究在整合關聯式資料庫與大數據架構各自優勢(如 SQL 與分散式運算)的關鍵技術上，提供了最佳的解決管道。

關聯式資料庫之所以成為目前資料庫市場的主流，很大部分的原因取決於 SQL 語法對結構化資料處理的支援程度，加以預存程序與工作排程(Job)等指令集式運行腳本的輔助，尚能應付一般性的使用需求。然而，隨著資料在質量、數量、型態及需求上愈來愈廣泛，加以關聯式資料庫於實際運作層面上仍然存在著建置維運成本過高、軟硬體升級不易，以及在處理消耗大量 CPU 與記憶體工作時所產生的延遲等問題，甚至已影響到伺服器內其他資料庫所屬應用程式或系統的正常運作，即使進行軟硬體升級、新增連結資料庫伺服器、運用索引、分割資料表、壓縮資料表或設定離峰時間運行等動作，在資料量長期累積下，對效能提昇的幫助也相當有限。因此，關聯式資料庫的效能調校往往成為資料庫管理人員的例行性工作。

預存程序與大數據架構於資料處理時的主要運行時間均包含了 CPU 運算、記憶體作業、磁碟 I/O 及網路傳輸等。基於無法在完全相同的標準下比較軟硬體等級與建立純粹的網路實驗環境等因素，本研究不討論磁碟 I/O(含預存程序第一次讀取資料表及於大數據架構中第一次讀取與最後寫入檔案之時間)及網路傳輸等因素對實驗結果的影響，而著重於將預存程序的 CPU 運算與記憶體作業轉移至大數據架構下進行分散式運算的可行性。歸納本研究之具體目標有:

1. 運用 PySpark 實作預存程序。
2. 以分散式運算取代單機運算，為預存程序於處理消耗大量CPU與記憶體工作時所產生的延遲問題，提供有效解決方案。

本研究共分五章，章節規劃如下:第一章為緒論，闡述研究背景、研究動機與研究目的；第二章為文獻探討，詳列本研究所涉及到的相關技術、知識領域與研究文獻；第三章為研究方法，規劃研究流程、設計研究架構、選擇大數據架構中適合實作預存程序之資料處理語言、說明建置大數據架構與關聯式資料庫實驗環境之步驟，並定義研究資料；第四章為實驗與討論，根據所定義之研究資料於關聯式資料庫中開發並實測預存程序、以大數據資料處理語言實作預存程序並實測，最後對實驗結果進行比較、分析與討論；第五章為本研究進行總結，說明研究發現並建議未來可研究之方向。

2 文獻探討

2.1 RDBMS、SQL & Stored Procedure

關聯式資料庫管理系統是目前資料庫管理系統的市場主流，1970 年，E. Codd 提出關聯式資料模型(Relational Data Model)[2]，藉由集合與代數等數學概念和方法來處理資料，是關聯式資料庫管理系統的理論基礎，且為保證交易的正確可靠，必須符合 ACID 四個特性。

SQL(Structured Query Language)，結構化查詢語言是一種通用於關聯式資料庫的標準語言，

1987 年，美國國家標準協會 ANSI 開始對 SQL 進行規範，並成為國際標準。但目前市場上使用率較高的關聯式資料庫系統中，其實作過程多少都對 SQL 規範進行改寫與擴充。因此，運作於不同關聯式資料庫系統的 SQL 語法，語法外觀上或許可以直觀理解，實際操作上卻無法一體適用[3]。在經過 ANSI SQL92[4]的不斷進化，SQL 儼然已成為高階的非過程化編程語言，配合關聯式資料庫的層級(Tier)概念，使用者直接在高層資料結構上作業，不需要指定資料的存放方法，也不需要了解資料處理的具體方式，即可靈活的操作、運用及管理資料。

Stored Procedure，預存程序是一種在關聯式資料庫環境中以 SQL 語法撰寫的指令或語法集的執行腳本(Script)，經編譯或修改後，即暫存於快取記憶體中，以供內外部重複使用的一種關聯式資料庫伺服器快取物件，也可以將它視為資料庫中的一種函式或子程式。預存程序可以設定接受參數與回傳值，方便資料庫管理人員或應用程式開發人員開發與應用，且具有減少網路流量、提高安全性、維護簡易、提昇效能等優勢。

2.2 Hadoop 生態系統

Hadoop 屬於 Apache 軟體基金會(Apache Software Foundation)的開放原始碼計畫(Open Source Project)之一，它是由 Dong Cutting 在參考了 Google File System[5]以及 MapReduce[6]兩篇論文後，改進了他原本使用 Lucene(高效能文件索引引擎)函式庫所開發的 Nutch 架構，最後成功運行在數十個節點的叢集之上。

Hadoop 除了分散式儲存系統與分散式運算特色外，在整個 Hadoop 生態系統中，至少包含了 HBase、Hive、Pig、Storm、Sqoop、Flume、Spark、Ambari、Flink、Zookeeper 等可依使用者需求而逕自導入的專案群，有些專案可獨立運行(如 Pig、Spark 等)，或與其他專案互相搭配(如 Ambari、Zookeeper 等)，或在 Hadoop 生態系統中共同運作。

Hadoop 具有高可靠性、高擴展性與高經濟性的特色，基於開放原始碼的精神，Hadoop 的主從架構目前均處於版本不斷進化的過程當中。過去，Hadoop 1.0[7]是由分散式儲存的 HDFS 和分散式運算的 MapReduce 所組成。其中，HDFS 可分為 NameNode 與 DataNode，而 MapReduce 則由 JobTracker 與 TaskTracker 共同運作。目前，Hadoop 2.0[8]在 HDFS 方面，針對 Hadoop 1.0 中單一 NameNode 制約了 HDFS 的擴展性問題，以 HDFS Federation 讓多個 NameNode 分管不同的目錄，進而實現存取隔離和橫向擴展。而在 MapReduce 方面，使用資源管理框架 YARN，將 JobTracker 中的資源管理和作業控制分離，分別在 ResourceManager 和 ApplicationMaster 中實現，ResourceManager 負責所有應用程序的資源分配、ApplicationMaster 則負責管理單一應用程序[9]。未來，Hadoop 3.0 有幾項新功能正在發展當中，包括使用 Erasure Coding 大量減少 HDFS 的儲存空間與 Hadoop Cluster 企業建置的硬體成本、強化 NameNode 高可用性(High Availability)、使用 YARN SharedCache 減少 Job Submission 的啟動時間和網路流量，以及 YARN Federation 支援超過一萬個以上的運算節點等[10]功能。

2.3 Spark 開源叢集運算框架

Apache Spark 是為快速運行與泛用化目的而生的開源叢集運算框架，最初是由加州大學柏克萊分校 AMPLab 所開發[11]，至 2014 年 2 月，Spark 已成為 Apache 的頂級專案(Top-Level Project)項目之一。由於 Hadoop 在分析資料時需要將過程中所產生的資料儲存在硬碟當中，因此會產生讀寫延遲的問題，而 Spark 則使用了記憶體內存的運算技術(In-Memory Computing)，預估能比 Hadoop 整體運算速度快上 10 至 100 倍。Spark 的技術特徵包括:

- 能運行在許多資源管理系統之上。
- 採用有向無環圖 DAG(Directed Acyclic Graph)的運行模式，可將資料前處理、合併、運算並整合於相同任務中。
- 使用獨特的 RDD(Resilient Distributed Dataset)資料操作方式封裝物件，開發者僅需選擇資料操作方法與核心運行演算法，RDD 便能自動處理工作背後包含分散運算與容錯等問題。
- 支援 In Memory 技術。
- 與 Hadoop HDFS 相容。
- 具備流程優化能力，能減少不必要的重複動作。
- 支援多種開發語言與直譯式語言操作，包含 Java、Scala、Python 以及 SparkR 等。
- 適合迭代運算，尤其是需求最佳解的 Machine Learning 演算法。
- Spark 可以截取儲存在 HDFS 中的任何文件，但 Hadoop 對 Spark 來說不是必須的，Spark 可以獨立運行，也可以運行在許多資源管理系統之上，包括 Hadoop YARN 與 Apache Mesos。

2.4 大數據架構上的資料處理語言

茲就目前大數據架構上重要的資料處理語言，分別探討如下:

- Pig:主要特色有對 Multi-Query 檢索循環次數的減少、支持多元資料類型，以及常見的資料操作如排序、過濾、求和、分組(Group by)或關聯(Join)等，這些優勢都讓 Pig 於資料處理上得到了廣泛的應用[12,13,14]。
- Hive:基於 HDFS 上的資料庫工具，可以將結構化、半結構化及非結構化的資料透過本身的機制映射為一個資料表，並提供類似 SQL 的查詢功能，又稱為 HiveQL 或 HQL。Hive 的優勢在於學習成本低，熟悉關聯式資料庫 SQL 語法的使用者可以直接透過類似的

SQL 語法實作，也可以用來進行大數據的萃取、轉置與載入(ETL)工作。

- SparkR:Spark 在 1.4 版本之後，開始增加了對 R 的支持，使得熟悉 R 的使用者可以結合 R 的優勢在 Spark 的分散式運算平台上工作，在 Spark SQL 套件中稱之為 SparkR。但目前 SparkR 仍是 R 在 Spark 上的輕量級版本，從 2015 年 6 月 Spark 1.4 提供 66 個 API 支援開始、Spark 1.5 提供了 197 個 API 支援，到了 Spark 1.6 版本，已經提供了超過 225 個 API 支援。
- Python:是一種物件導向、直譯式的程式語言[15]，在結合了處理複雜資料的採礦能力後，迅速成為了資料分析語言的主流，Python 語法相較於其他語言更加簡單也更加直觀，背後強大的社群也呈現不可思議地快速成長。Python 結合 Spark 核心功能如工作排程、記憶體管理、容錯及檔案系統互動等元件，更讓使用者可以輕易的搭配 PySpark 函式庫來進行資料運算處理與資料圖表呈現。
- Java:擁有跨平台、物件導向、泛型程式設計的特性，廣泛運用於企業級 Web 和行動應用程式開發中。Java 不同於一般的編譯語言或直譯語言，它實作了一次編寫、到處運行的跨平台特性。Java 在大數據資料處理上雖然沒有 SparkR 與 Python 一樣好的視覺化功能，也不是統計建模所推薦的最佳工具，但凡是基於 Hadoop 的程式開發，最後都必須實作於 Java 之上。
- Scala:為 Spark 的原生語言，在 Spark 上的支援與效能相對於其他語言更具優勢。Scala 以獨立編譯、動態載入的編譯模型兼容於 Java，所以可以直接呼叫 Java 的類別庫，根據原生於 Java 的特性，當然也能在 Hadoop 生態系統上通行無礙 [16]。

3 研究方法

3.1 研究流程、研究工具與系統架構

研究流程:本研究之主題範圍清晰、變數關係明確，研究方法即採行實驗研究法。自變項包括程式編碼與運行環境；依變項是程式運行結果與運行時間；控制變項有磁碟 I/O、網路傳輸、環境設備及軟硬體等級等。規劃研究流程如圖 3-1。

圖 3-1、研究流程圖

研究工具:基於軟硬體的一致性、擴充性與調整便利性，雲端建置方案最符合本研究實驗之需求。然而，雲端建置方案的選擇性相當多，包括 Amazon 的 EMR、IBM 的 SmartCloud、Microsoft 的 Azure 以及 Google 的 Google Cloud Platform 等，因應 Microsoft 為關聯式資料庫 SQL Server 的開發廠商，研究工具採用 Microsoft Azure，一併處理大數據叢集環境與關聯式資料庫的建置問題。

研究架構:大數據架構主要包含 Hadoop 與 Spark 二種模式，由於 Hadoop MapReduce 上所有的工作都要經由 Map、Shuffle 與 Reduce 等三個階段，且每次運算都會在磁碟上讀取或寫入資料，加以整個運算架構不排除網路傳輸等，都將導致延遲。因此，本研究的實驗環境在大數據架構上選擇 Spark，繪製系統架構圖，如圖 3-2。

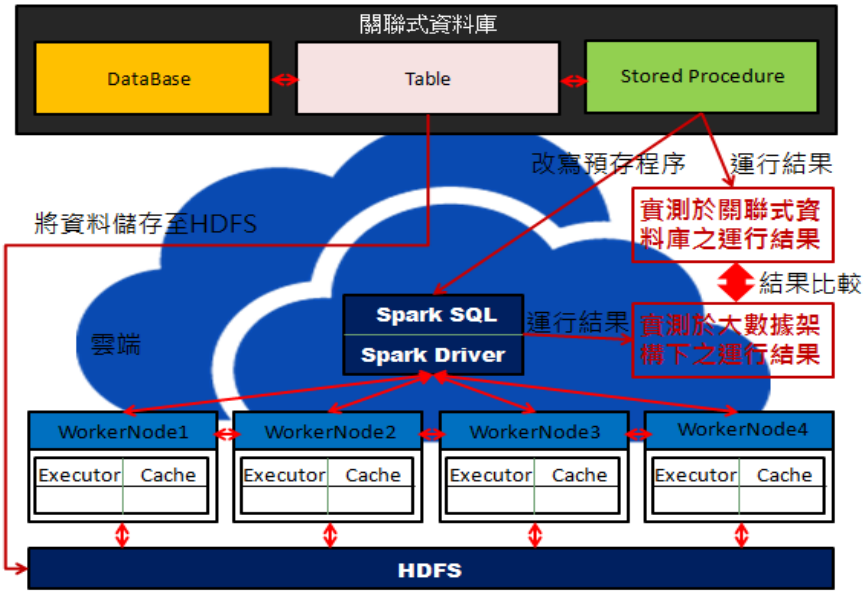

圖 3-2、系統架構圖

3.2 大數據架構之資料處理語言分析

本研究欲運用大數據架構上之資料處理語言實作關聯式資料庫之預存程序，以分散式運算的優勢來解決預存程序單機大量運算之延遲問題。歸納其首要工作，即在選擇合適之大數據資料處理語言。本章節彙整目前 Hadoop 與 Spark 架構上重要之資料處理語言，依本研究實驗需求之角度分析 Pig、Hive、Python、Java、Scala 與 SparkR 目前版本於大數據架構中之現況，根據社群力量、支援 SQL 能力、資料處理能力、統計分析能力、支援圖表能力、程式碼簡潔度、連結關聯式資料庫能力以及網站互動式介面等因素，是本研究採用 Python 與 PySpark 函式庫，作為本研究實作預存程序之大數據資料處理語言的主要原因。

3.3 大數據叢集環境的設定與建置

Microsoft Azure HDInsight 上的 Spark 叢集採用 Spark on Hadoop 模式，也就是基於 YARN 資源管理系統的叢集環境，並提供可快速擴展的技術堆疊 [17] 與 99.9% 的 SLA(Service Level Agreement)支援。雲端大數據叢集環境的設定與建置步驟如下:

Step1:進入 Microsoft Azure，選擇 HDInsight 之 Hadoop 叢集。

Step2:選擇 Spark 2.0 版本。

Step3:設定帳號密碼與 SSH，以利後續研究使用 PuTTY 指令模式登入，SSH 的公鑰與私鑰可經由下載運行 PuTTY gen 程式產生。

Step4:設定雲端資源位置，叢集位置選擇距離較近的東亞。

Step5:設定工作節點數量與硬體規格，本研究共建立四個 8 核心(規格為 D12v2)背景工作節點與二個 4 核心(規格為 D4v2)前端節點 [18]。

Step6:設定完成後，系統隨即開始部署。

3.4 關聯式資料庫環境的設定與建置

根據 Microsoft Azure 於官網上之說明，Microsoft Azure 的雲端關聯式資料庫系統提供可預測的效能、無停機時間的延展性、商務持續性和資料保護等功能 [19]，並可支援現有的 Microsoft SQL Server 工具、程式庫與 API。雲端關聯式資料庫環境的設定與建置步驟如下:

Step1:選擇 Microsoft Azure 的資料庫並建立 SQL 資料庫。

Step2:設定資料庫名稱 utaipei 與帳號密碼。

Step3:選擇資料庫硬體規格(本研究設定為 S2 標準級別/50 DTUs)與定序。DTU 是 Microsoft Azure SQL DataBase 資源的度量單位，是 CPU、記憶體、資料 I/O 和交易紀錄檔 I/O 的混合量值[20]。

Step4:設定 Microsoft SQL Server 防火牆與用戶端連線 IP。

Step5:使用 SQL Server Management Studio 連線。

3.5 定義研究資料

本研究定義之研究資料來源下載自行政院環境保護署環境資源開放資料庫網站[21]，在資料下載區選擇大氣選項中三十天觀測資料內所有資料，下載的時間範圍從 2013 年 10 月份到 2016 年 12 月份，解壓縮後共有 39 個 CSV 檔案，將下載解壓縮後的資料匯入雲端關聯式資料庫系統主機 utaipei、建立 weather 資料庫，並將資料儲存至 dbo.Weather_Taiwan 中，產生[StationName]、[MonitorDate]、[Pressure]、[Temperature]、[Humidity]、[WindDirection]、[WindSpeed]、[Precipitation]、[SunshineHour]共 9 個欄位，總計 742,589 筆資料。

所匯入之資料於欄部分位並無固定，遺漏值主要集中在[WindSpeed]、[Precipitation]與[SunshineHour]等3個欄位。此外，資料型態中的[StationName]、[WindDirection]為string、[MonitorDate]為datetime，其餘欄位則為float。

4 實驗與討論

4.1 開發並實測預存程序

4.1.1 預存程序內容說明

拆解資料庫欄位中的[MonitorDate]，新增為[YearNum]、[MonthNum]及[DayNum]三個時間欄位以利時間區別；個別計算[Pressure]、[Temperature]、[Humidity]、[WindSpeed]、[Precipitation]及[SunshineHour]的年平均值、月平均值與日平均值，分別使用#temp_Weather_Taiwan、#temp_AY_Weather(年平均值)、#temp_AM_Weather(月平均值)、#temp_AD_Weather(日平均值)等四個暫存表儲存，再由四個暫存表交叉計算(Table Join)產生十三個新增欄位:

1. [Pressure_MRate]-以大氣壓力除以月平均大氣壓力，所產生的月平均大氣壓力比率。
2. [Pressure_YRate]-以大氣壓力除以年平均大氣壓力，所產生的年平均大氣壓力比率。
3. [Temperature_DRate]-以溫度除以日平均溫度，所產生的日平均溫度比率。
4. [Temperature_MRate]-以溫度除以月平均溫度，所產生的月平均溫度比率。
5. [Temperature_YRate]-以溫度除以年平均溫度，所產生的年平均溫度比率。
6. [Humidity_DRate]-以溼度除以日平均溼度，所產生的日平均溼度比率。
7. [Humidity_MRate]-以溼度除以月平均溼度，所產生的月平均溼度比率。
8. [Humidity_YRate]-以溼度除以月平均溼度，所產生的年平均溼度比率。
9. [WindSpeed_MRate]-以風速除以月平均風速，所產生的月平均風速比率。
10. [WindSpeed_YRate]-以風速/年平均風速，所產生的年平均風速比率。
11. [Precipitation_YRate]-以降水量除以年平均降水量，所產生的年平均降水量比率。
12. [SunshineHour_YRate]-以日照時數除以年平均日照時數，所產生的年平均日照時數比率。
13. [Pressure_DRate]-以大氣壓力除以日平均大氣壓力，所產生的日平均大氣壓力比率。

4.1.2 預存程序開發步驟

Step1:統計資料筆數。
Step2:拆解[MonitorDate]欄位並做遺漏值修補，組成#temp_Weather_Taiwan 暫存表。
Step3:建立各項數據的年平均值欄位，組成#temp_AY_Weather 暫存表。
Step4:建立各項數據的月平均值欄位，組成#temp_AM_Weather 暫存表。
Step5:建立各項數據的日平均值欄位，組成#temp_AD_Weather 暫存表。
Step6:運用 Table Join 交叉計算方式產生暫存表 temp_Weather_Taiwan_Report，並以 Select 語法顯示所有資料。

4.1.3 實測預存程序

在 Microsoft Azure 雲端所建立的關聯式資料庫 SQL Server 於沒有其他內存或應用程式的干擾下，預存程序 Weather_Taiwan_Report 在 S2 標準級別(50 DTUs)下的運行結果，如圖 4-1。

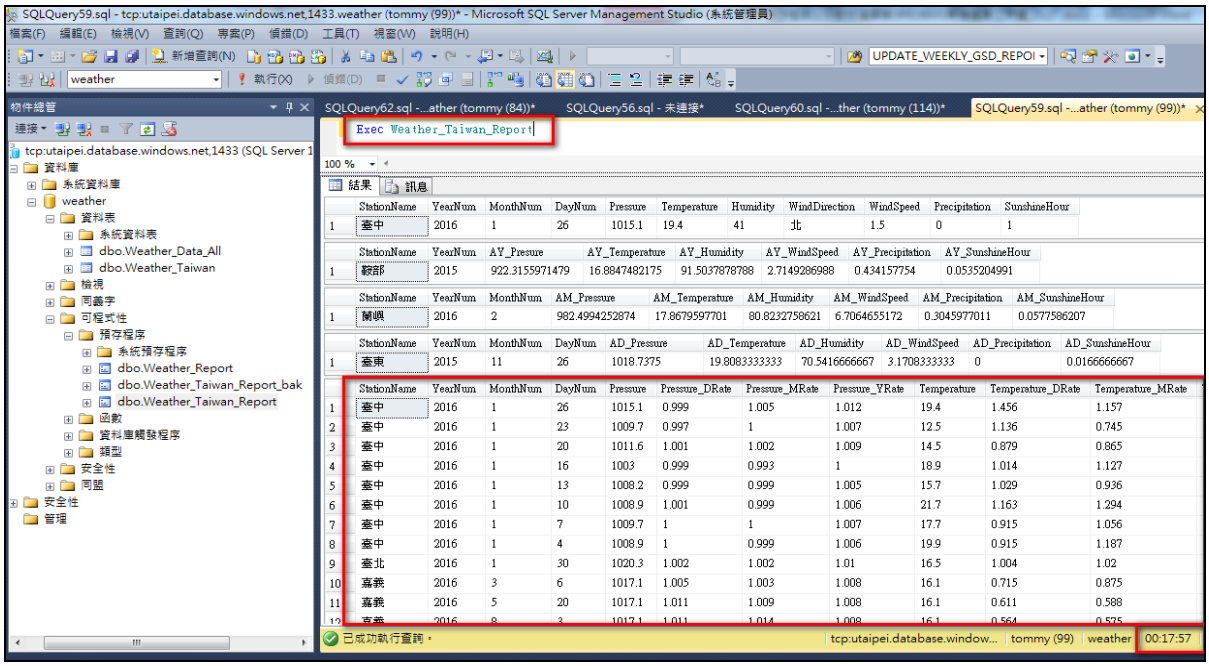

圖 4-1、預存程序運行結果

4.2 運用 PySpark 函式庫實作預存程序

4.2.1 從雲端資料庫匯製.csv 檔並儲存至 HDFS 之步驟

Step1:基於本研究不討論磁碟 I/O(含預存程序第一次讀取資料表及於大數據架構中第一次讀取與最後寫入檔案之時間)及網路傳輸等因素對實驗結果的影響，因此，本研究使用檔案傳遞的方式，從雲端關聯式資料庫 weather 中將資料匯成 Weather_Taiwan.csv 檔，再使用 WinSCPPortable 將檔案上傳至雲端 Spark on Hadoop 叢集架構中。

Step2:使用 PuTTY SSH 連接 Spark on Hadoop 叢集架構，並以 ls 指令查看 Weather_Taiwan.csv 是否已傳遞成功。

Step3: 以 hdfs dfs –put Weather_Taiwan.csv /HdiSamples/Weather_Taiwan.csv 指令將檔案置入 HDFS，再以 hdfs dfs –ls /HdiSamples 指令確認檔案已存在。

4.2.2 於 Jupyter 上運用 PySpark 實作預存程序之步驟

Step1:於 Jupyter 上新增 PySpark3(使用 Python3.x 版本)專案。

Step2:撰寫使用者定義函數以修補遺漏值。

Step3:讀取 HDFS 中的 Weather_Taiwan.csv 檔案，統計並確認筆數(742,589)，再呼叫使用者定義函數做遺漏值修補。「Wasbs:///」為 Spark on Hadoop 中 HDFS 的預設路徑。

Step4:顯示拆解[MonitorDate]欄位與遺漏值修補後的暫存表。此步驟與預存程序中顯示暫存表#temp_Weather_Taiwan 步驟相同，並需比對兩者之計算結果是否一致。

Step5:顯示各項數據的年平均值暫存表。此步驟與預存程序中顯示暫存表#temp_AY_Weather 步驟相同，並需比對兩者之計算結果是否一致。

Step6:顯示各項數據的月平均值暫存表。此步驟與預存程序中顯示暫存表#temp_AM_Weather 步驟相同，並需比對兩者之計算結果是否一致。

Step7:顯示各項數據的日平均值暫存表。此步驟與預存程序中顯示暫存表#temp_AD_Weather 步驟相同，並需比對兩者之計算結果是否一致。

Step8:以 PySpark 註冊完各項數據的年、月、日平均值暫存表後，即可開始運行最後產生報表的主要程式碼區段。

Step9:將程式碼存檔。

4.2.3 實測以分散式運算方式運行之結果

本研究於 Microsoft Azure HDInsight 上共建立四個 8 核心(規格為 D12v2)背景工作節點與二個 4 核心(規格為 D4v2)前端節點之 Spark on Hadoop 叢集架構。實測以四個 executors 採分散式運算方式運行，並於 Jupyter 上 import datetime 函式庫以記錄運行時間，如圖 4-2。

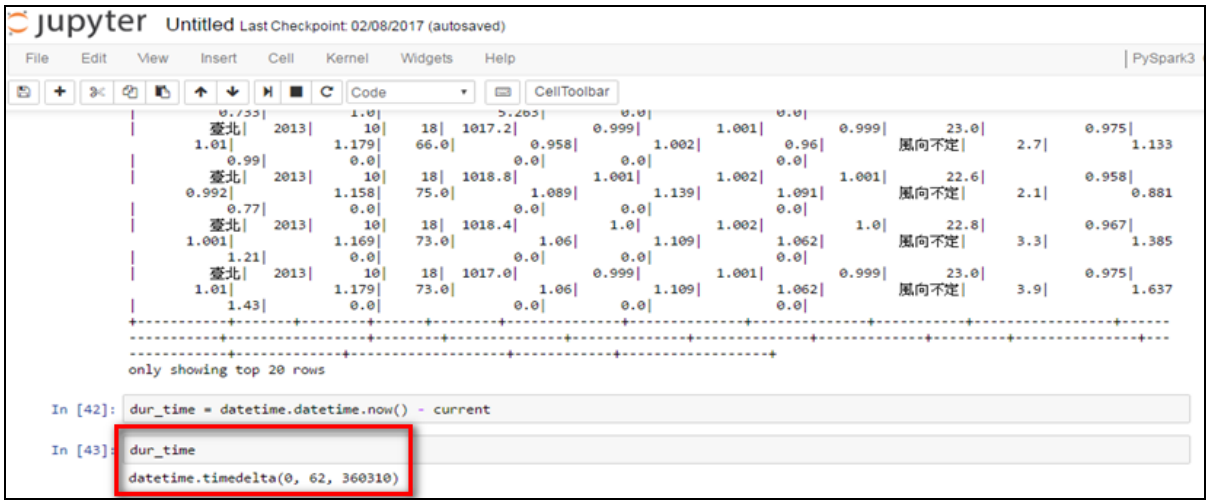

圖 4-2、以四個 executors 採分散式運算方式運行

4.3 實驗結果分析與討論

4.3.1 比對預存程序與經 PySpark 實作後之運算結果

經比對預存程序執行後所產生的結果報表，與經 PySpark 實作後於 Spark on Hadoop 叢集架構上運行 Weather_Report.ipynb 所產生儲存於 HDFS 中的結果檔案，在 742,589 筆運算數量與數據上完全相同，如圖 4-3，確認預存程序以 PySpark 實作並移轉至 Spark on Hadoop 叢集架構上運行之可行性。

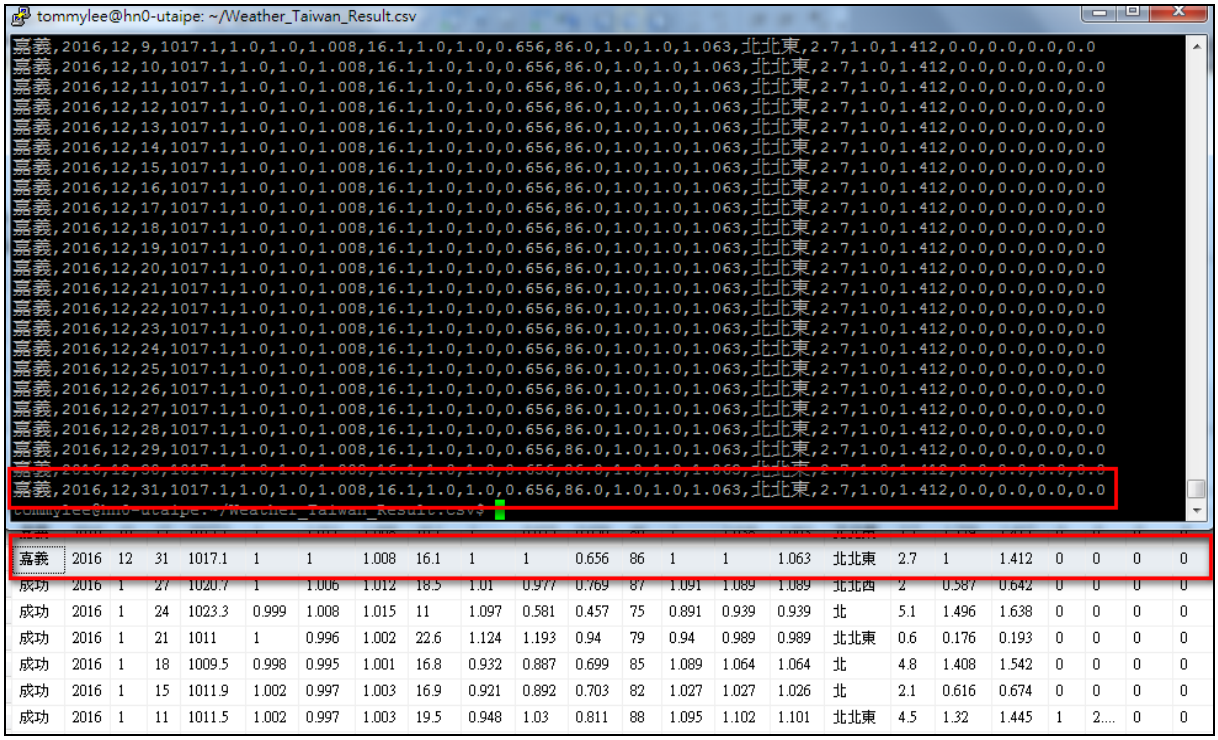

圖 4-3、比對實作後之運算結果相同

4.3.2　數據分析

- 預存程序於本實驗之雲端關聯式資料庫運行十次之紀錄，如表 4-1。

表 4-1、預存程序運行十次之紀錄表

預存程序於本實驗之雲端關聯式資料庫運行實測數據										
實測序號	1	2	3	4	5	6	7	8	9	10
實測時間	17 分 57 秒	10 分 21 秒	10 分 52 秒	14 分 03 秒	9 分 58 秒	11 分 13 秒	13 分 38 秒	13 分 06 秒	11 分 10 秒	12 分 12 秒
總秒數	1077	621	652	843	598	673	818	786	670	732

- 以 PySpark 實作預存程序於本實驗之雲端 Spark on Hadoop 叢集架構運行十次之紀錄，如表 4-2。

表 4-2、以 PySpark 實作預存程序運行十次之紀錄表

運用 PySpark 實作預存程序於本實驗之雲端 Spark on Hadoop 叢集架構實測數據										
實測序號	1	2	3	4	5	6	7	8	9	10
實測時間	1 分 02 秒	58 秒	1 分 11 秒	1 分 03 秒	56 秒	1 分 03 秒	1 分 02 秒	1 分 09 秒	1 分 04 秒	1 分 05 秒
總秒數	62	58	71	63	56	63	62	69	64	65

- 以 SPSS 計算兩者之平均數、標準差與變異數等，如表 4-3、表 4-4

表 4-3、SPSS 觀察值處理摘要

	觀察值					
	包括		排除		總和	
	個數	百分比	個數	百分比	個數	百分比
預存程序	10	100.0%	0	0.0%	10	100.0%
運用 PySpark 實作	10	100.0%	0	0.0%	10	100.0%

表 4-4、SPSS 報表

	預存程序	運用 PySpark 實作
平均數	747.00	63.30
個數	10	10
標準差	142.755	4.473
平均數的標準誤	45.143	1.415
最小值	598	56
最大值	1077	71
總和	7470	633

4.3.3 結果討論

1. 從各自實測十次的平均數來看，在 Microsoft Azure HDInsight 上之 Spark on Hadoop 叢集架構(四個 8 核心規格為 D12v2 的背景工作節點與二個 4 核心規格為 D4v2 的前端節點)以四個 executors 採分散式運算方式運行，比預存程序在 Microsoft Azure SQL Server S2 標準級別(50 DTUs)下的運行結果，於十次實測的平均速度上，約莫快上 11.8(747/63.3)倍。若有更大的叢集環境，可藉由調整 executor 與 cache 來尋求最佳化效能。
2. 從各自實測十次的標準差來看，在 Microsoft Azure HDInsight 上之 Spark on Hadoop 叢集架構(四個 8 核心規格為 D12v2 的背景工作節點與二個 4 核心規格為 D4v2 的前端節點)以四個 executors 採分散式運算方式運行，較預存程序在 Microsoft Azure SQL Server S2 標準級別(50 DTUs)下的運行結果，就與平均值的分散程度上，前者顯得較為集中。
3. 從各自實測十次的最大最小值來看，在 Microsoft Azure HDInsight 上之 Spark on Hadoop 叢集架構(四個 8 核心規格為 D12v2 的背景工作節點與二個 4 核心規格為 D4v2 的前端節點)以四個 executors 採分散式運算方式運行的最慢速度，比預存程序在 Microsoft Azure SQL Server S2 標準級別(50 DTUs)下運行的最快速度，約莫快上 8.4(598/71)倍。
4. 就資料處理語言而言，PySpark 是目前支援 SQL 語法最廣泛的函式庫之一。雖然本研究於實驗過程中，曾遭遇 PySpark 尚未完全支援 SQL 語法 SubQuery 功能的問題，導致必須改以 Table Join 的方式計算，但相信 PySpark 正處於發展的活躍時期，PySpark 所支援的 SQL 語法，將會更具全面性。
5. 就開發工具而言，Python 具有語法簡潔、物件導向與直譯式互動編程等特色，與腳本化(Script)的 SQL 互動與執行模式相仿，並提供參數輸入。此外，JDBC 更解決了關聯式資料庫與大數據架構的連結問題，不必如同本實驗的檔案傳遞方式或以 Sqoop 等其他方式連結關聯式資料庫。
6. 就整體環境而言，在關聯式資料庫環境中，預存程序常內嵌於例行工作(Job)中運行，Hadoop 與 Spark 均架設於 Linux 環境之上，同樣可運用 Linux 的例行性工作排程 Crontab 執行.py 檔來實現。

5 結論與建議

有鑑於大數據(Big Data)與資料科學的蓬勃發展，本研究運用 PySpark 實作預存程序、以分散式運算取代單機運算，為預存程序於處理消耗大量 CPU 與記憶體工作時所產生的延遲問題，提供有效解決方案，並為關聯式資料庫程式開發與管理人員、大數據系統架構師、資料工程師、資料科學家以及相關資料科學研究人員提供可行模式與結果參考，針對本研究可供未來研究建議有:

1. 索引(Index)是關聯式資料庫用來加快資料查詢效能的重要功能與方法，在大數據架構之 NoSQL 資料庫尚未發展索引功能的情況下，僅能使用記憶體內存(In Memory)技術來輔助提昇效能。若能同時運用索引與記憶體內存技術的優勢，將是未來資料庫系統轉型的關鍵技術。
2. 有向無環圖 DAG(Directed Acyclic Graph)與彈性分散資料集 RDD(Resilient Distributed Dataset)是 Spark SQL 於資料處理上重要的二個特色，其與關聯式資料庫 ACID 特性間的運作關聯，值得後續的追蹤與研究。

Acknowledgements

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 105-2221-E-845-002。Corresponding author: Yen Hung Chen。

Reference

[1] Holden Karau. Andy Konwinski. Patrick Wendell. Matei Zaharia 著. 許致軒譯. 碁峰資訊. "SPARK 學習手冊". 2016.
[2] https://zh.wikipedia.org/wiki/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93
[3] https://zh.wikipedia.org/zh-tw/SQL
[4] JIM MELTON. Sybase. Inc.. Sandy. Utah.

Sql language summary. 1996.
[5] Google. "OSDI 2006:Bigtable: A Distributed Storage System for Structured Data. 2006.
[6] Google. "OSDI 2004:MapReduce：Simplifed Data Processing on Large Cluster. 2004.
[7] http://hadoop.apache.org/docs/r0.23.11/
[8] http://hadoop.apache.org/docs/r2.7.3/
[9] https://read01.com/R5nJDO.html
[10] http://hadoop.apache.org/docs/r3.0.0-alpha1/
[11] SPARK 亞太研究院 王家林. 上奇資訊股份有限公司. "大數據分析處理 SPARK 技術應用與性能質化". 2016.
[12] Pig Latin Reference Manual 1. http://pig.apache.org/docs/r0.8.1/piglatin_ref1.html
[13] Pig Latin Reference Manual 2. http://pig.apache.org/docs/r0.8.1/piglatin_ref2.html#Flatten+Operator
[14] Z. Zhang, L.Cherkasova, A.Verma, B.T. Loo, Optimizing Completion Time and Resource Provisioning of Pig Programs. In 12th ACM *International Symposium on Cluster Cloud and Grid Computing*, *2012*.
[15] https://zh.wikipedia.org/wiki/Python
[16] https://zh.wikipedia.org/wiki/Scala
[17] https://azure.microsoft.com/zh-tw/services/hdinsight/
[18] https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-sizes
[19] https://docs.microsoft.com/zh-tw/azure/sql-DataBase/sql-DataBase-technical-overview
[20] https://docs.microsoft.com/zh-tw/azure/sql-database/sql-database-what-is-a-dtu
[21] http://erdb.epa.gov.tw/FileDownload/FileDownload.aspx