

國立台灣師範大學資訊工程研究所  
碩士論文

指導教授：葉耀明 博士

利用 MapReduce 軟體架構於 Hadoop 叢集進行

地貌型直接逕流模組演算之研究

Research on The Computing of Direct Geo

Morphology Runoff on Hadoop Cluster by

Using MapReduce Framework

研究生：周建廷 撰

中華民國 一 百 年 六 月

# 摘要

利用 MapReduce 軟體架構於 Hadoop 叢集

進行地貌型直接逕流模組演算之研究

周建廷

台灣由於氣候及地形的因素，一旦下起豪大雨便常常造成河川瞬間水位暴漲，甚至釀成嚴重的災情，因此更彰顯洪水預報系統在台灣的重要性。河川流徑洪水演算是洪水預報系統最重要的一環，目的是計算流域中的各項水文相關資料以判斷流量是否超出警戒線。但河川流徑運算公式複雜，流域的相關資料量又龐大，以傳統交予大型電腦處理或者由客戶端連線至伺服器端將工作交給伺服器處理等單一主機運算的方式往往需要消耗許多時間，造成預報不夠即時。

本研究的程式開發借重於Apache軟體基金會所開發的Hadoop開放源碼平台，Hadoop提供大量資料儲存及運算的分散式運算環境，以及提供程式開發者一種專為大量資料處理所設計的軟體架構—MapReduce，以分散式運算提供整合的運算資源加速處理龐大的資料量以減少運算時間。本研究使用MapReduce架構撰寫河川流徑演算程式，將其置於Hadoop叢集上運作，透過5種情境的量測得到最佳河川流徑演算速率可提升至6倍左右，達到提高洪水預報系統的效能、讓預報更即時的目的。

關鍵字：Hadoop、MapReduce、分散式運算、大量資料處理

# ABSTRACT

Research on The Computing of Direct Geo Morphology Runoff on Hadoop Cluster

by Using MapReduce Framework

By

Chien-Ting Chou

Because of the weather and landform in Taiwan, a heavy rain often cause sudden rising of the runoff of some basins, even lead to serious disaster. That makes flood information system are highly relied in Taiwan especially in typhoon season.

Computing the runoff of a basin is the most important module of flood information system for checking whether the runoff exceeds warning level or not. However this module is complicated and data-intensive, it becomes the bottleneck when the real-time information are needed while a typhoon is attacking the basins.

The development of applications in this thesis is on "Apache Hadoop"—an open-source software that builds a distributed storage and computing environment, which allows for the distributed processing of large data sets across clusters of computers using a programming model—"MapReduce". We have developed the runoff computing module of a basin by using MapReduce framework on a Hadoop cluster. In our research, to speed up the runoff computing will increase the efficiency of the flood information system. Running our programs in an 18 nodes Hadoop cluster, we have derived the conclusion that it can speed up the execution of runoff computing by 6 times.

KeyWords : Hadoop, MapReduce, Distributed Computing, Processing for Large Data

## 誌 謝

兩年來的耕耘總算有了回報，以成就本論文。首要感謝葉耀明教授對我研究上的指導與建議，讓我更具備獨立思考的能力且促使我多方涉獵許多領域的知識。經由參加與校外合作的計畫，我接觸到許多非資訊領域的師長及業界專才，讓我更加強溝通及問題解決的能力，培養做研究的堅實基礎。

感謝楊明正教授與陳永昇教授，百忙之中能抽空前來擔任口試委員，並給予許多寶貴意見以及指正論文的缺失，惠我良多，自當謹記於心。以及感謝國立台灣大學水工試驗所李天浩教授與鄭安孺博士，對於本論文水文相關專業知識給予許多指導，方能更順利地撰寫本論文。

感謝實驗室同學冠廷、謹至、汝怡、名鈺、培然、國隆、尚儒、威傑，一同討論、修課、作業，讓我這段期間的回憶充滿歡樂與趣味，而非做研究時的苦澀；也感謝實驗室金麟、政龍等博士班學長以及學長姐嘉陳、欣培、靖山、正成、智尹、玟瑄、孟軍的帶領；還有感謝學弟妹冠緯、惠迪、永倫、芝華、瀧濱、明憲、慶瑞、孝倫於計畫上的協助。很高興自己身為 XML 實驗室的一員，與大家共度這回味無窮的兩個年頭。

還要感謝台灣國家高速網路與計算中心所架設之「Hadoop 雲端運算實驗叢集」，提供了本論文開發與運作的環境；以及該平台的負責人王耀聰副研究員，給予了本研究諸多幫助，感激不盡。

最後要感謝我的家人，在我求學期間一路相陪與支持。時至今日，總算能順利完成碩士的學業，向以上的各位獻上誠摯的感謝。

# 目錄

第一章 緒論.....	1
1.1 研究背景與動機.....	1
1.2 研究目的與意義.....	2
1.3 論文架構.....	3
第二章 文獻探討.....	4
2.1 雲端運算.....	4
2.1.1 雲端運算的定義.....	4
2.1.2 以自由軟體技術達成各種雲端服務.....	8
2.1.3 Hadoop 簡介.....	11
2.2 Hadoop 運作架構.....	13
2.2.1 與儲存相關的背景常駐程序.....	13
2.2.2 輔助背景常駐程序.....	15
2.2.3 與運算相關的背景常駐程序.....	16
2.2.4 常見的 Hadoop 叢集建構方式.....	17
2.3 MapReduce 軟體架構.....	19
2.4 淡水河流域集水區簡介.....	21
第三章 以 MapReduce 架構撰寫直接逕流模組.....	24
3.1 系統架構.....	24
3.2 直接逕流模組運算資料型態與運算流程.....	27
3.2.1 原始的及修改後的運算資料格式.....	27
3.2.2 運算流程.....	32
3.3 直接逕流模組核心演算法概要.....	36
第四章 系統實作及效能量測.....	39
4.1 程式開發及測試環境.....	39
4.2 地貌型直接逕流模組程式效能量測及比較.....	41
4.2.1 量測關鍵要素.....	42

4.2.2	單機運算測試.....	43
4.2.3	Hadoop 叢集中各種量測情境探討.....	45
第五章	結論與未來發展.....	57
5.1	結論.....	57
5.2	未來發展.....	57
附錄	.....	59
參考文獻	.....	61

## 圖目錄

圖 1.1	雲端運算概觀.....	1
圖 2.1	美國國家標準局（NIST）對於雲端運算所定義之模型.....	4
圖 2.2	雲端運算的佈署模式.....	6
圖 2.3	雲端運算的參考架構.....	7
圖 2.4	虛擬化技術.....	8
圖 2.5	虛擬化技術所對應之自由軟體.....	9
圖 2.6	虛擬化技術於雲端運算參考架構中扮演的角色.....	9
圖 2.7	Hadoop 等軟體技術於雲端運算參考架構中扮演的角色.....	10
圖 2.8	Google 公司三大關鍵技術對應的自由軟體.....	11
圖 2.9	NameNode 的網頁狀態介面（顯示 HDFS 的狀態資訊）.....	14
圖 2.10	JobTracker 的網頁狀態介面（顯示 MapReduce 的狀態資訊）.....	16
圖 2.11	較小型 Hadoop 叢集建構方式.....	18
圖 2.12	較大型 Hadoop 叢集建構方式.....	18
圖 2.13	MapReduce 架構模型.....	20
圖 2.14	Hadoop 叢集系統進行 MapReduce 工作概念圖.....	21
圖 2.15	淡水河流域內部集水區示意圖.....	22
圖 2.16	單位像元面積及流徑.....	23
圖 2.17	集水區流徑網路立體圖.....	23
圖 2.18	集水區流徑網路等高線圖.....	23
圖 3.1	地貌型逕流模式組織圖.....	24
圖 3.2	地貌型直接逕流模組示意圖.....	26
圖 3.3	套用 MapReduce 的地貌型直接逕流模組架構.....	27
圖 3.4	excess1.dat 格式.....	28
圖 3.5	huc_para.dat 格式.....	28
圖 3.6	ele 參數檔格式.....	29
圖 3.7	waterlevel 變數檔格式.....	30

圖 3.8	修改格式後之 excess1.dat 及 excess1.metadata .....	31
圖 3.9	修改格式後之 huc_para.dat 及 huc_para.metadata.....	31
圖 3.10	修改格式後之 ele 參數檔及 ele 詮釋資料檔 .....	32
圖 3.11	修改格式後之 waterlevel 變數檔及 waterlevel 詮釋資料檔 .....	32
圖 3.12	TextInputFormat 分割格式範例.....	34
圖 3.13	單一 Map 工作運算流程範例圖 .....	35
圖 3.14	流徑兩端節點斷面流量示意圖.....	36
圖 3.15	地貌型直接逕流模組核心演算法.....	37
圖 4.1	國網中心 Hadoop 實驗平台的 JobTracker 網頁狀態介面 .....	40
圖 4.2	國網中心 Hadoop 實驗平台的 NameNode 網頁狀態介面.....	40
圖 4.3	國網中心 Hadoop 實驗平台的 Ganglia 監控介面 .....	41
圖 4.4	Hadoop 實驗叢集的 TaskTracker 之子程序最大記憶體使用量設定圖 ....	43
圖 4.5	單機計算測試情境一的結果.....	44
圖 4.6	單機計算測試情境二的結果.....	45
圖 4.7	Hadoop 實驗叢集測試情境一的結果 .....	46
圖 4.8	Hadoop 實驗叢集測試情境一的 MapReduce 工作監控畫面.....	47
圖 4.9	Hadoop 實驗叢集測試情境二的結果 .....	48
圖 4.10	Hadoop 實驗叢集測試情境二的 MapReduce 工作監控畫面.....	49
圖 4.11	Hadoop 實驗叢集測試情境三的結果 .....	50
圖 4.12	Hadoop 實驗叢集測試情境三的 MapReduce 工作監控畫面.....	51
圖 4.13	Hadoop 實驗叢集測試情境四的結果 .....	52
圖 4.14	Hadoop 實驗叢集測試情境四的 MapReduce 工作監控畫面.....	53
圖 4.15	Hadoop 實驗叢集測試情境五的結果 .....	54
圖 4.16	Hadoop 實驗叢集測試情境五的 MapReduce 工作監控畫面.....	55
附錄圖 1	Muskingum.log 的格式 .....	60
附錄圖 2	huc_discharge.dat 的格式.....	60



## 表目錄

表 4.1	國網中心 Hadoop 實驗平台建置環境.....	39
表 4.2	單機計算的各種測試情境效能統整.....	56
表 4.3	Hadoop 實驗叢集計算的各種測試情境效能統整.....	56



# 第一章 緒論

## 1.1 研究背景與動機

現今網際網路發展與普及的速度越來越快，Web 2.0 概念提出後，網路使用者的上網行為有重大改變。例如從原本單純的網頁瀏覽，進化至能將個人的創作或任何資訊分享至網路上的社群。為了順應此趨勢，網路服務提供者明白傳統的大型電腦（Mainframe）以至主從式架構（Client/Server）等概念皆已不符合科技潮流，勢必需要利用網路使電腦能彼此合作無間以達到更穩定的運算及可靠的服務。因此當 Google 公司提出了「雲端運算（Cloud Computing）」這個嶄新概念並在台灣啟動「雲端運算」學術計畫後，眾多網路公司察覺到此概念將會是繼主從式架構後又一重大巨變，紛紛搭順風車投入了雲端運算服務的開發。

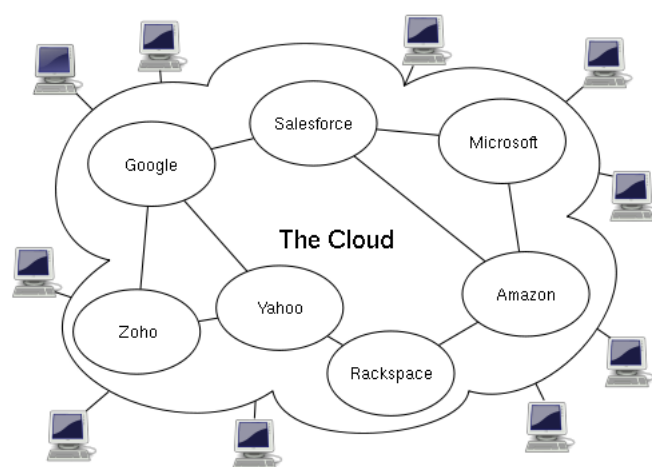


圖 1.1 雲端運算概觀

（來源：<http://zh.wikipedia.org/wiki/雲端運算>）

另一方面，台灣位於亞熱帶季風氣候區，山地多，降雨急促且雨量驚人，尤其夏秋兩季盛行颱風，夾帶的豐沛雨量常常造成河川瞬間水位暴漲，形成洪水或土石流，威脅附近居民生命財產安全及造成許多河川流域附近生態景點的破壞。

民國九十八年莫拉克颱風引起的八八水災，造成南台灣受災慘重，為台灣五十年來最大水患，使得洪水資訊預報系統的重要性更加不容忽視。而進行洪水預報最重要的一環，就是河川流徑洪水演算。要預測是否有洪水就必須先知道入流量、出流量等各種河川流域資訊，以判斷是否超出警戒線。但河川流徑運算公式複雜，流域裡各項相關資料量又龐大，以傳統單一電腦主機計算往往需要消耗許多時間，造成預報不夠即時的狀況。預報不即時，就不能防範災害於未然。

## 1.2 研究目的與意義

傳統運算模式不外乎將繁複的計算工作交予大型電腦(Mainframe)處理或者由客戶端(Client)連線至伺服器端(Server)，將工作交給伺服器端處理。但雲端運算概念的核心是利用網路使電腦彼此合作以進行工作，猶如多人通力合作勝過一人獨自完成，因此本研究之目的即借助雲端運算的概念改善河川逕流演算系統的效能。

本研究的程式開發借重於 Apache 軟體基金會 (Apache Software Foundation) 一個開放源碼雲端平台的計畫—Hadoop，Hadoop 是以 Java 開發而成、可以提供大量資料儲存並運算的分散式運算環境 (Distributed Computing Environment)。另外 Hadoop 提供程式開發者一種軟體架構 (Software Framework)—MapReduce，以分散式運算提供整合的運算資源加速處理龐大的資料量以減少運算時間。因此本研究著眼於使用 MapReduce 架構改寫逕流演算程式，利用其優點將會改善逕流演算速率，達到改善洪水預報系統的效能、讓預報更即時的目標。

## 1.3 論文架構

本論文共分為五個章節，各章節內容分別敘述如下：

### 第一章 緒論

介紹研究背景與動機、研究目的以及論文架構。

### 第二章 文獻探討

簡介近年當紅的雲端運算概念，以及一些具代表性的可建構雲端服務之自由軟體。本章重點在於針對 Hadoop 雲端平台做詳細介紹，包含 Hadoop 的運作架構以及 MapReduce 軟體架構。另外本章也對於淡水河流域之集水區做一點簡短說明。

### 第三章 以 MapReduce 架構撰寫直接逕流模組

包含系統架構、套用 MapReduce 架構之直接逕流模組運算資料型態與運算流程，以及直接逕流模組核心概要演算法。

### 第四章 系統實作及效能量測

包含程式的開發及測試環境、套用 MapReduce 架構之直接逕流模組於 Hadoop 叢集中的運算效能量測及其與單機運算效能的比較。

### 第五章 結論與未來發展

總結本研究的初步成果，並提出建議以供未來研究使用。

## 第二章 文獻探討

### 2.1 雲端運算

#### 2.1.1 雲端運算的定義

所謂「雲端」即是泛指「網路」，名稱來自工程師繪製示意圖時，常以一朵雲來代表「網路」。因此「雲端運算」等同「網路運算」。舉凡運用網路溝通多台電腦的運算工作，或是透過網路連線取得由遠端主機群提供的服務等，都可以算是一種「雲端運算」。

Visual Model Of NIST Working Definition Of Cloud Computing



圖 2.1 美國國家標準局（NIST）對於雲端運算所定義之模型

雲端運算繼承了平行運算的叢集技術、融合了分散運算的容錯特性，發展出「資料中心即電腦（Data Center as a Computer）」的新思維。根據美國國家標準局（National Institute of Standards and Technology）對於雲端運算所下的定義，所

謂「雲端運算」包含五大基礎特徵（Essential Characteristic）及四個佈署模型

（Deployment Models）與三種服務模式（Service Model）。五項基礎特徵分別為：

- On-demand self-service：不需假手他人，隨需自助服務。
- Broad network access：隨時隨地可用任何網路裝置（如手機、筆記型電腦、PDA 等）存取
- Resource pooling：多人共享資源池，如共用頻寬、儲存空間等。
- Rapid elasticity：能快速重新佈署的靈活度。
- Measured service：可被監控與量測的服務。

四個佈署模型分別為：

- 私有雲（Private Cloud）：意指由企業自行建置雲端運算平台，其建置成本較為昂貴，但因企業擁有完全的控管權限，因此能夠提供較高的安全性與隱私性防護。適用於大型企業。
- 公用雲（Public Cloud）：並不是所有企業都能負擔建置私有雲的高昂成本，對這些企業而言公用雲便是他們雲端運算的替代解決方案。就如同企業可以利用 ISP（Internet Service Provider）業者提供的服務來達成連結網際網路的需求，而非自行開發解決方案一般。企業也可以利用雲端運算服務提供者（Cloud Provider，如 Google、Amazon 等）建置的公用雲，來滿足雲端運算的需求。公用雲的好處在於企業可以依據自身的運算需求，向雲端運算服務提供者租用適量的雲端服務，並可以彈性調整租用的量。然而，由於整個資訊運用（包含可能的機敏資料）都放置在公用雲上，因此安全性與隱私性的威脅相對地大。

- 社群雲（Community Cloud）：某些團體組織相互友善，因此可合作建置共有的社群雲，以分享各組織所允許開放出來的功能及分攤雲端的維護成本。
- 混合雲（Hybrid Cloud）：以上三種模式混合了兩種以上即可稱之為混合雲。常見的狀況為私有雲混合公用雲，由於必須考慮到計算量可能超乎想像，因此私有雲若不敷使用時可動態根據計算需求調用公用雲的資源。

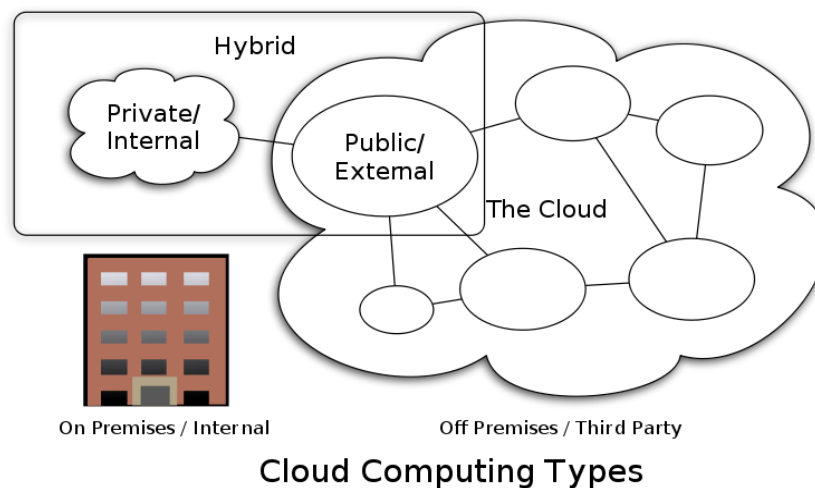


圖 2.2 雲端運算的佈署模式

（來源：[http://en.wikipedia.org/wiki/Cloud\\_computing](http://en.wikipedia.org/wiki/Cloud_computing)）

而三種服務模式則分別為：

- 基礎設施即服務（Infrastructure as a Service, IaaS）：也就是提供運算、儲存及網路等基礎設備之服務，以提供內外部使用者存取之用。為了讓資源有效管理與應用，IaaS 多半藉助虛擬化技術（Virtualization）來完成伺服器整合之基本作業。目前市面上的 IaaS 服務，以提供虛擬化機器服務的 Amazon EC2（Amazon Elastic Compute Cloud）最為大家熟知。



- 平台即服務（Platform as a Service, PaaS）：便是提供運算平台給系統管理員及開發人員，並提供整合的 API 讓開發人員構建、測試及佈署定製之應用程式更加簡便，但平台系統管理的成本較昂貴。著名的如 Microsoft Azure、Google App Engine 以及 Amazon S3（Amazon Simple Storage Service）皆屬於此類服務。而開放源碼的 Hadoop 平台的功能基本上也歸於此類。
- 軟體即服務（Software as a Service, SaaS）：用戶向服務提供商租用基於 Web 的雲端應用程式，來管理企業經營活動。存取該服務的使用者無需下載或安裝任何程式，只需直接透過瀏覽器存取雲端應用程式所提供的功能與服務，且不用對軟體進行除錯、更新等維護作業，服務提供商會全權管理和維護軟體，對於使用管理負擔及成本的降低有不小的助益。例如 Google Map、Google Docs 等 Google Apps，以及 Facebook、Twitter 等社群網站服務。

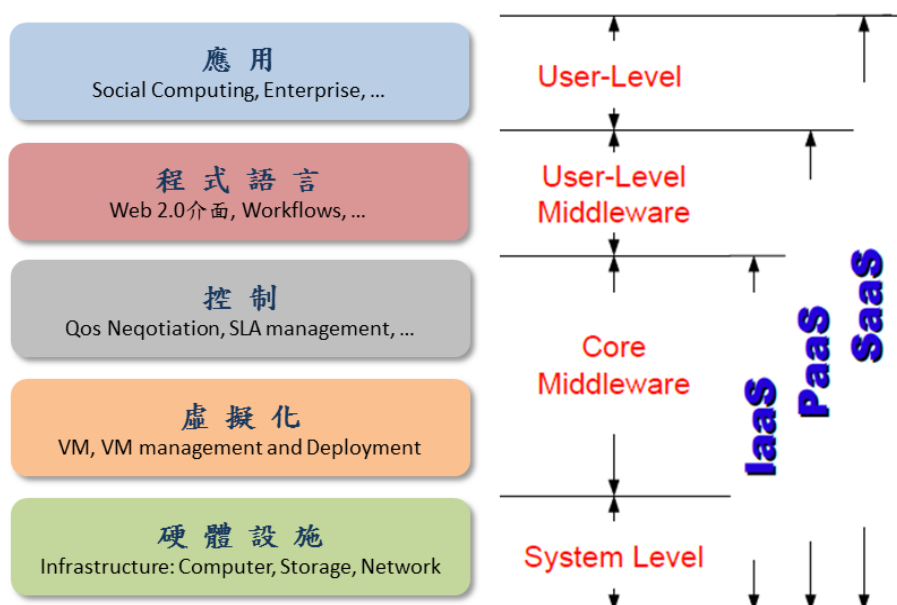


圖 2.3 雲端運算的參考架構

### 2.1.2 以自由軟體技術達成各種雲端服務

雲端運算技術的發展日益迅速，若使用者無法長期負擔向雲端服務提供者租用雲端服務的費用，亦不想錯失雲端運算帶來的效益，目前有不少開放源碼的自由軟體（Open Source）可作為替代解決方案。這些自由軟體依不同之功能可分別達成基礎設施即服務、軟體即服務、平台即服務三種服務模式。這些功能大略分類如下：

#### 1. 以虛擬化（Virtualization）技術之自由軟體達成基礎設施即服務：

虛擬化一詞最早出現於 1960 年代 IBM M44/44X 主機中，當時 IBM 公司為了讓使用者能充分地利用昂貴的大型主機資源，於是發明虛擬化技術，允許用戶在一台主機上運行多個作業系統，並將 CPU 等系統資源有效分配給這些獨立的作業系統。虛擬化技術在管理層面上提供很高的彈性與便利性，也可以有效降低主機空間、冷氣、電力的龐大需求，相當符合目前綠色節能的政策方針。

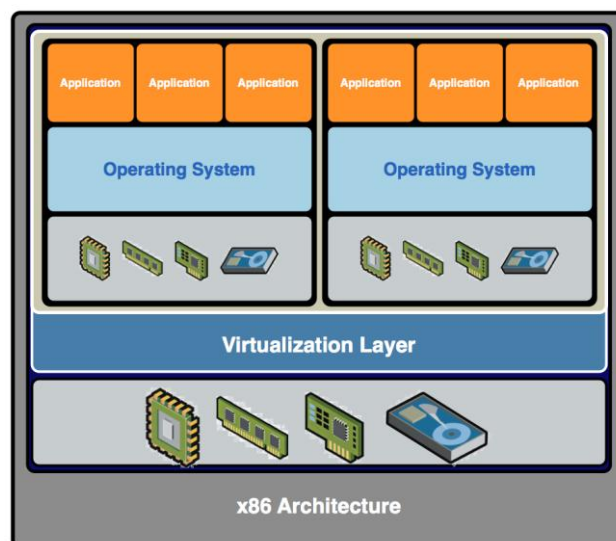


圖 2.4 虛擬化技術

（來源：<http://www.supportsd.com/services/virtualization/>）

依美國國家標準局（NIST）所定義，虛擬化技術主要包含作業系統虛擬化（OS-level Virtualization）、網路虛擬化（Network Virtualization）、儲存虛擬化（Storage Virtualization）、虛擬機器管理平台（VM Management Tool）以及狀態監控與認證收費機制（Monitoring / AAA）。圖 2.5 列出目前這些虛擬化技術所對應的自由軟體，圖 2.6 則表示虛擬化技術於雲端運算參考架構中所扮演的角色。

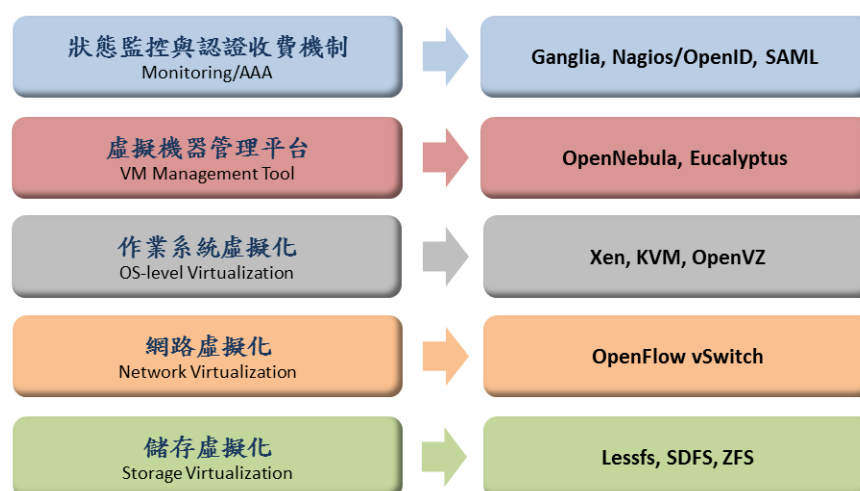


圖 2.5 虛擬化技術所對應之自由軟體

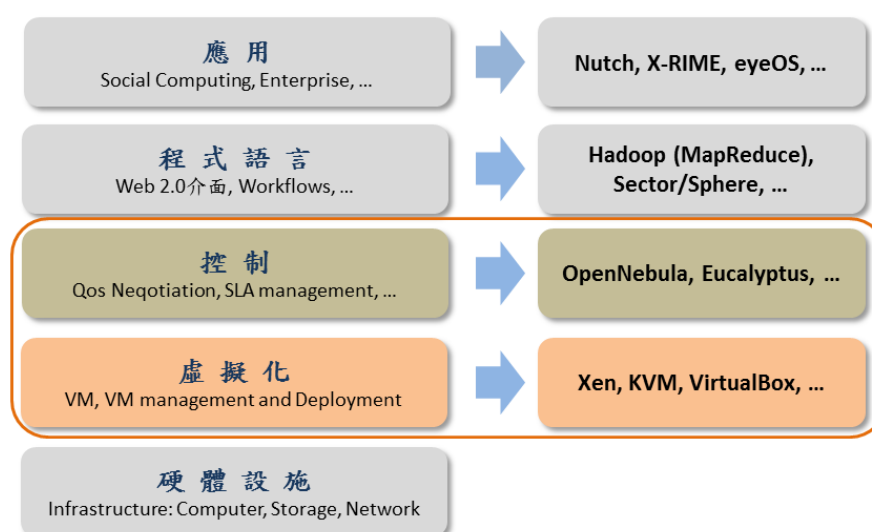


圖 2.6 虛擬化技術於雲端運算參考架構中扮演的角色

2. 以自由軟體打造平台即服務模式：

目前這方面的自由軟體以 Apache Top Level 開發專案的「Hadoop」及美國資料探勘中心所研發的「Sector/Sphere」最為著名，而功能皆為模擬出對應 Google 公司的關鍵技術－Google File System、MapReduce 及 BigTable。目的是提供使用者一個簡易撰寫並執行運算大量資料應用程式的軟體平台。

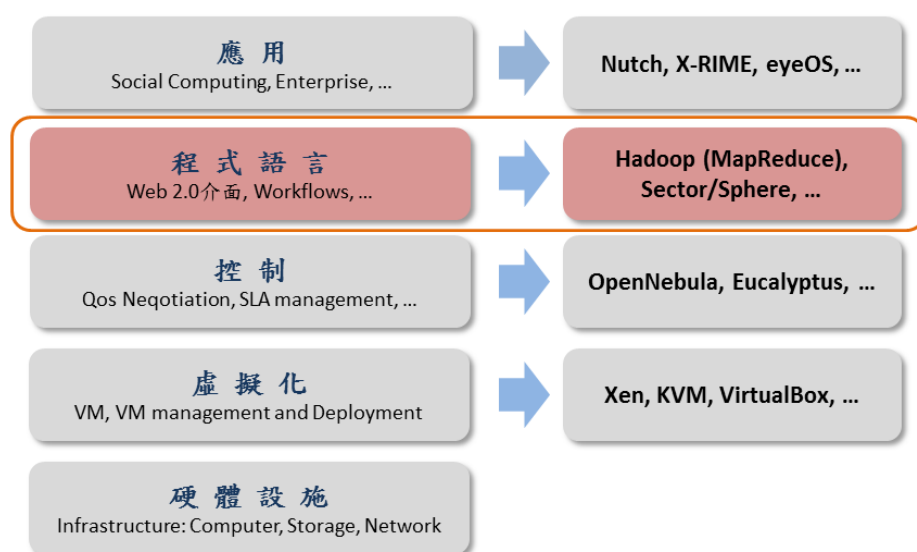


圖 2.7 Hadoop 等軟體技術於雲端運算參考架構中扮演的角色

3. 以自由軟體打造軟體即服務模式：

目前應用方面的自由軟體著名的有以 Hadoop 為基礎的相關搜尋引擎開發計畫「Nutch」及台灣國家高速網路中心所研發以 Nutch 專案為核心的叢集式搜尋引擎「Crawlzilla」。另外尚有以 Web 為基礎的虛擬桌面作業系統「eyeOS」，以及同樣以 Hadoop 為基礎，用來分析大型社交網路行為的工具「X-RIME」等等。

### 2.1.3 Hadoop 簡介

Hadoop 是 Apache 軟體基金會（Apache Software Foundation）的一個頂級開發專案（Top-level Project），以 Java 開發而成，專為開發處理大量資料之分散式應用程式的人員所設計的開放源碼雲端平台。Hadoop 包括許多子計劃，其中 Hadoop Distributed File System（HDFS）、MapReduce 與 HBase 分別對應了 Google 公司三項關鍵技術－Google File System、MapReduce 及 BigTable。MapReduce 提供分散式運算環境、HDFS 提供高容錯率的分散式儲存空間、HBase 是一個類似 BigTable 的分散式資料庫。

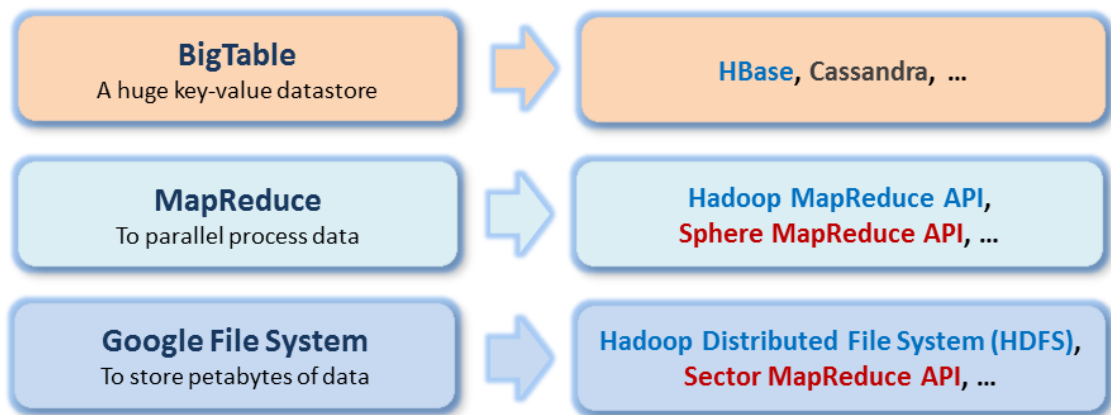


圖 2.8 Google 公司三大關鍵技術對應的自由軟體

相較於其他分散式運算平台，Hadoop 具有以下主要優點：

1. 易取得（Accessible）：運作 Hadoop 不需要昂貴或高級配備的主機，只要一台或多台一般個人電腦主機做為節點（Node）便可建構出 Hadoop 環境。

2. 可靠 (Robust) : 由於 Hadoop 能夠在一般個人電腦上運作，因此容錯能力很高。只要有某一節點發生錯誤，系統能即時自動取得備份資料以及重新佈署運算資源。
3. 巨量 (Scalable) : 擁有儲存與處理大量資料的能力，甚至可高達 Petabyte 級別的資料量。
4. 簡易 (Simple) : 提供友善的應用程式介面 (Application Programming Interface)，讓開發者能夠更快速地開發出有效率的分散式處理程式。

## 2.2 Hadoop 運作架構

所謂「Hadoop 的運作」包含了兩項功能：資料儲存（Storage）及資料運算（Computation）。資料運算即為利用前述的 MapReduce 來達成，而儲存方面則是以安裝 Hadoop 的一台或多台主機所構成之 HDFS（Hadoop Distributed File System）來做為資料儲存與管理的環境。HDFS 不但具容錯能力與效率，且增加越多節點儲存容量越大。

而這兩項功能的運作必需依靠數個扮演不同特殊角色的背景常駐程序（Daemon）來達成，且這些背景常駐程序有些又彼此有主從關係（Master/Slave），依功能區分如下：

- 與儲存相關的背景常駐程序：NameNode（Master），DataNode（Slave）
- 與運算相關的背景常駐程序：JobTracker（Master），TaskTracker（Slave）
- 輔助背景常駐程序：Secondary NameNode

以下將介紹這些背景常駐程序的功能及其扮演的角色。

### 2.2.1 與儲存相關的背景常駐程序

#### 1. NameNode：

Hadoop 的架設機制規定由多台主機構成的分散式叢集只允許一台主機運作 NameNode 背景常駐程序。NameNode 功能是負責指揮身為 slave 的 DataNode 去執行 I/O 工作，例如運算進行時對檔案的讀寫。除此之外，NameNode 還要負責追蹤資料檔案的每個儲存單位區塊（block，HDFS 預設之單位區塊為 64MB）各自儲存在哪個主機節點，以及 HDFS 的整

體情況等。這些追蹤記錄會形成詮釋資料(Metadata)儲存於 NameNode。

由於 NameNode 背景常駐程序的功能是記憶體需求密集及 I/O 需求密集的 (Memory and I/O instensive)，因此運作 NameNode 的主機節點通常不用來儲存運算用資料及進行 MapReduce 運算以避免降低效能，這也意味著運作 NameNode 背景常駐程序的主機節點不適合同時運作身為 slave 的 DataNode 或 TaskTracker 的背景常駐程序。

## NameNode 'hadoop1:9000'

Started: Fri Apr 29 23:01:22 CST 2011  
Version: 0.20.2, r911707  
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo  
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)  
[Namenode Logs](#)

---

### Cluster Summary

10 files and directories, 1 blocks = 11 total. Heap Size is 7.31 MB / 888.94 MB (0%)

Configured Capacity : 447.17 GB  
DFS Used : 1.01 MB  
Non DFS Used : 29.47 GB  
DFS Remaining : 417.7 GB  
DFS Used% : 0 %  
DFS Remaining% : 93.41 %  
[Live Nodes](#) : 2  
[Dead Nodes](#) : 0

---

### NameNode Storage:

Storage Directory	Type	State
/var/hadoop/hadoop-hadooper/dfs/name	IMAGE_AND_EDITS	Active

---

[Hadoop](#), 2011.

圖 2.9 NameNode 的網頁狀態介面 (顯示 HDFS 的狀態資訊)



## 2. DataNode：

Hadoop 由多台主機構成的分散式叢集中，除了運作 NameNode 及 JobTracker 兩個背景常駐程序的主機節點（即 Master nodes）不適合外，其他節點皆同時扮演 DataNode 與 TaskTracker 兩個角色。由於運算資料會以單位區塊的形式儲存在各 DataNode 中，因此 MapReduce 工作執行需要讀取某個運算資料檔案進行運算時，NameNode 會告知該資料檔案的所有單位區塊各自分配至哪一個 DataNode 儲存以便存取。若 DataNode 自己目前儲存的某些單位區塊有了更動，DataNode 也會將這些資訊即時回報給 NameNode。另外，DataNode 也會備份自己所儲存的所有單位區塊，將這些備份副本經由 NameNode 的指示儲存至其他的 DataNode。如此一來，假設某個 DataNode 由於某些原因無法運作時，仍舊可以保證資料在運算當中能被正確的存取。

### 2.2.2 輔助背景常駐程序

#### 1. Secondary NameNode：

為輔助 NameNode 的背景常駐程序，同樣的在 Hadoop 叢集中只能有一台主機運作 Secondary NameNode 背景常駐程序。與 NameNode 不同的是 Secondary NameNode 並不會去接收及紀錄 DataNode 內檔案單位區塊的情況，而只是單純每隔一段時間從 NameNode 中讀取 HDFS 的相關詮釋資料。一旦 NameNode 發生了不可預期的問題，管理者可以重新設定暫時以 Secondary NameNode 取代 NameNode，使系統免於停擺。

### 2.2.3 與運算相關的背景常駐程序

#### 1. JobTracker :

Hadoop 叢集同樣只允許一台主機運作 JobTracker 背景常駐程序。一旦要在叢集系統執行 MapReduce 運算工作時，JobTracker 會替我們制定一個執行計畫，此計畫包括找出運算所需的相關資料檔案、把運算工作（Map Task/Reduce Task）分派給各個 TaskTracker 以及監看所有運算的執行以確認是否有工作發生錯誤。若有某些 TaskTracker 執行工作發生了錯誤，JobTracker 會要求其他 TaskTracker 重新執行這些工作，直至到達系統設定的重試次數上限為止。

#### hadoop1 Hadoop Map/Reduce Administration

[Quick Links](#)

State: RUNNING  
Started: Fri Apr 29 23:01:31 CST 2011  
Version: 0.20.2, r911707  
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo  
Identifier: 201104292301

##### Cluster Summary (Heap Size is 36 MB/888.94 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	1	<a href="#">2</a>	4	4	4.00	<a href="#">0</a>

##### Scheduling Information

Queue Name	Scheduling Information
<a href="#">default</a>	N/A

Filter (Jobid, Priority, User, Name)   
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

##### Running Jobs

[none](#)

##### Completed Jobs

[none](#)

##### Failed Jobs

[none](#)

##### Local Logs

[Log](#) directory, [Job Tracker History](#)

[Hadoop](#), 2011.

圖 2.10 JobTracker 的網頁狀態介面（顯示 MapReduce 的狀態資訊）

## 2. TaskTracker :

與 DataNode 角色相仿，Hadoop 叢集中除了運作 NameNode 及 JobTracker 兩個背景常駐程序的節點不適合外，其他節點皆同時運作 DataNode 及 TaskTracker 背景常駐程序。TaskTracker 的角色就是各自把 JobTracker 分派下來的 MapReduce 運算工作執行完成並回覆運算結果給 JobTracker。TaskTracker 會固定每隔一段時間以心搏（heartbeat）的方式告知 JobTracker 自己尚在正常地運作，若 JobTracker 有長時間沒有接收到來自某 TaskTracker 的心搏，便會判定該 TaskTracker 已無法運作而將運算工作交付其他 TaskTracker。

### 2.2.4 常見的 Hadoop 叢集建構方式

一般在較少的主機節點所構成的較小型 Hadoop 叢集中，NameNode 及 JobTracker 可以在同一個節點上運作，而 Secondary NameNode 可以移至某個執行 DataNode 與 TaskTracker 的節點來運作（如圖 2.11）。而在較大的 Hadoop 叢集中，可以分配三個節點各自單獨運作 NameNode、JobTracker 以及 Secondary NameNode（如圖 2.12）。

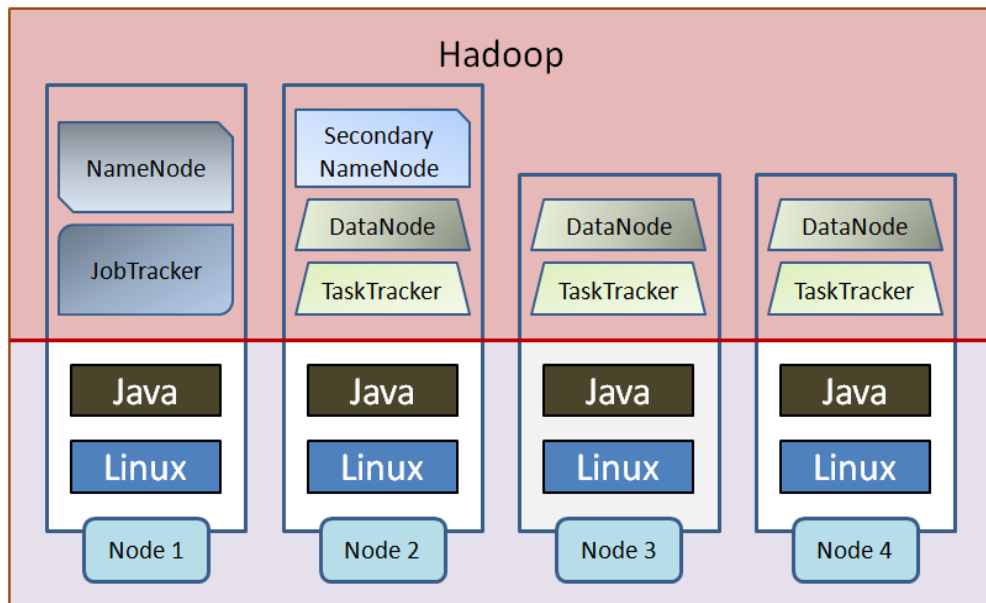


圖 2.11 較小型 Hadoop 叢集建構方式

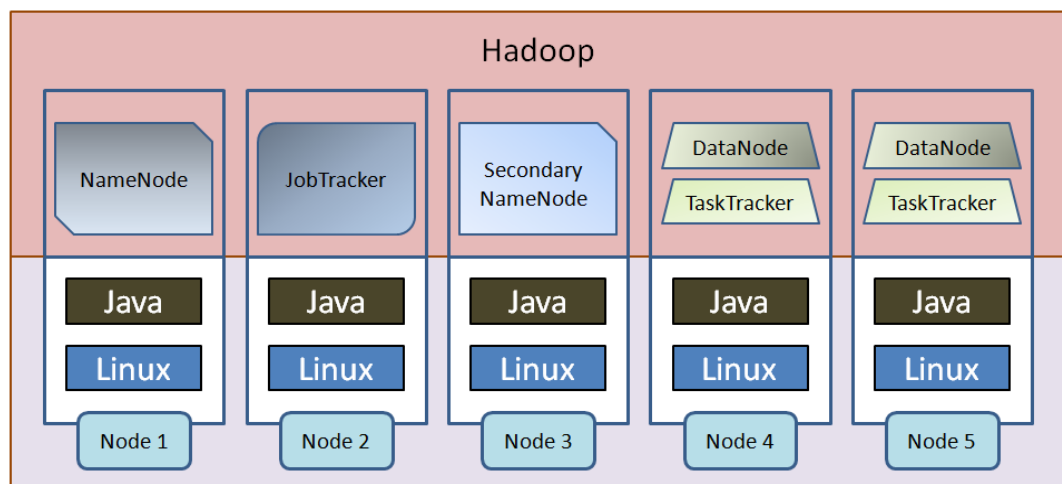


圖 2.12 較大型 Hadoop 叢集建構方式

## 2.3 MapReduce 軟體架構

「MapReduce」是合併用語，實際上包含了兩個概念—「Map」及「Reduce」。

簡單來說，若要開始進行運算時，Hadoop 叢集系統會啟動分割機制將輸入的運算資料分割成一個個切割塊，這些切割塊稱之為 Input splits。此時叢集系統中每個可以運作的 TaskTracker 背景常駐程序會各自產生子程序（Child Process），一個子程序進行一項 Map 運算工作、一項 Map 運算工作處理一個 Input split。由於 Map 工作是由各 TaskTracker 獨立處理的，因此意味著各 Map 工作的運算是平行化的。

而一個 Input split 在進入 Map 階段被處理時，會依照開發者於程式碼中指定之分割格式（分割格式是以 Java 類別型態呈現，MapReduce 架構預先已提供了數種格式，開發者也可依自身需求撰寫自訂的分割格式類別）再將該 Input split 細分成一個個 Record，而後各 Record 都會再解析（parse）成一對 Key/Value 的特殊形式來表示。接下來一項 Map 工作裡的每對 Key/Value 皆會呼叫一次 Map 工作的運算處理函數（map method），一對 Key/Value 在經過函數的處理後可能會輸出一對或數對 Key/Value，端看開發者是如何撰寫此運算函數，至此 Map 工作已告一段落。

在進入 Reduce 階段之前，經過 Map 階段所得出的所有 Key/Value 配對會依照 Key 之值來搜尋排序，若找到 Key 值相同但 Value 值互不相同的配對，便將這些配對歸類成一群。經過此一整理，使得 Reduce 階段的處理負擔大大減少，這個階段稱之為「Shuffle and Sort」。

開發者能夠在程式碼中指定 Reduce 階段所需進行的 Reduce 工作數量，之後會依照開發者指定的 Reduce 工作數來分配相同數量的 TaskTracker 進行 Reduce 工作，並將先前 Shuffle and Sort 階段所分出來的所有群組也依照 Reduce 工作數量分成同等份分配給各 TaskTracker 進行 Reduce 工作。Reduce 運算工作的進行與 Map 工作同理，同樣一項 Reduce 工作由一個 TaskTracker 程序產生一個子程序來進行處理，因此各個 Reduce 工作的運算也是平行化的。Reduce 工作結束便會輸出運算結果檔，運算結果檔數量也視 Reduce 工作數而定。

MapReduce 軟體架構對於開發者最大的好處就是完全不必考慮運算資料單位區塊在 HDFS 上是如何分佈的，開發者只需要專注在應用程式的開發即可。

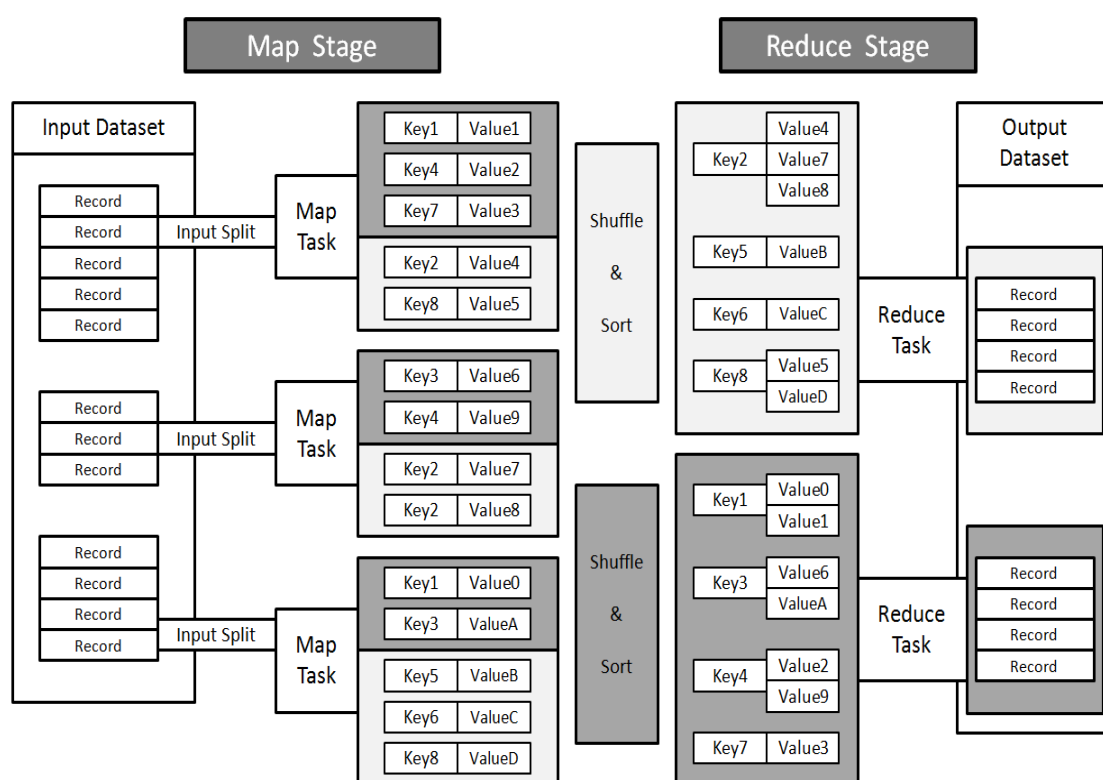


圖 2.13 MapReduce 架構模型

(本圖修改自參考文獻[5]之內容)

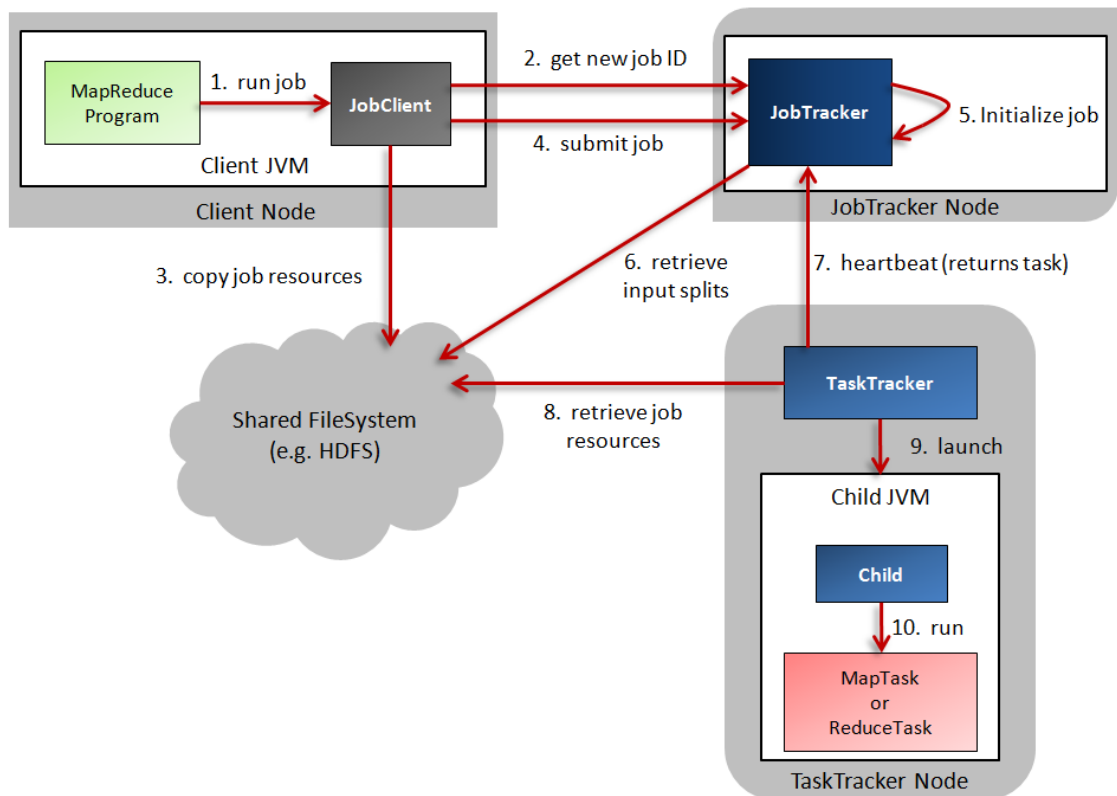


圖 2.14 Hadoop 叢集系統進行 MapReduce 工作概念圖

(本圖修改自參考文獻[4]之內容)

## 2.4 淡水河流域集水區簡介

本研究的所有水文運算資料來源為台大水工試驗所提供的實測數據，時間範圍介於西元 2009 年 8 月 3 日 13 時 5 分至西元 2009 年 8 月 10 日 2 時 0 分所量測的淡水河流域相關資料。而河川逕流演算法及其概念亦來自於該所開發的「地貌型逕流模組 (Geo Morphology Runoff)」程式。於該程式所闡述的概念中，是以「集水區 (Subbasin)」為逕流演算的處理單元，圖 2.15 為淡水河流域及內部依地貌型逕流模組程式概念劃分的集水區示意圖。

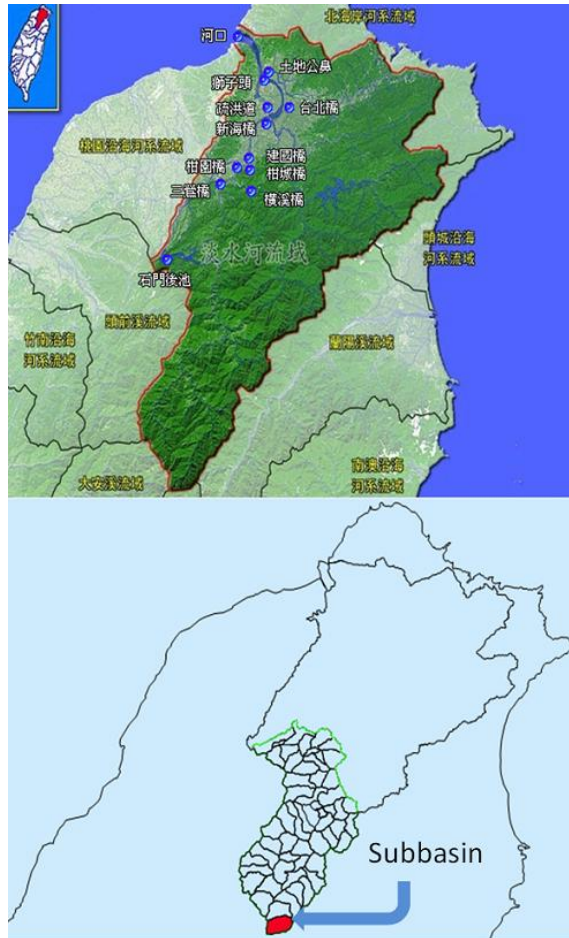


圖 2.15 淡水河流域內部集水區示意圖

另外依照演算法的概念，可將集水區視為由許多 40 公尺x40 公尺的「單位像元面積（Unit Element）」所組成的逕流網絡。如圖 2.16，此單位像元面積可以當成逕流網絡中的一個節點（Node）看待，而箭頭部份則表示流徑（Reach）中水的流向。而圖 2.17 表示一個集水區內部流徑網路的立體圖，圖 2.18 則為流徑網路的等高線圖，由此三個圖形可判斷出集水區內部的流徑網路為有向非循環之圖形（Directed Acyclic Graph）。



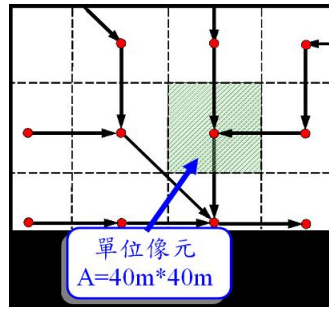


圖 2.16 單位像元面積及流徑

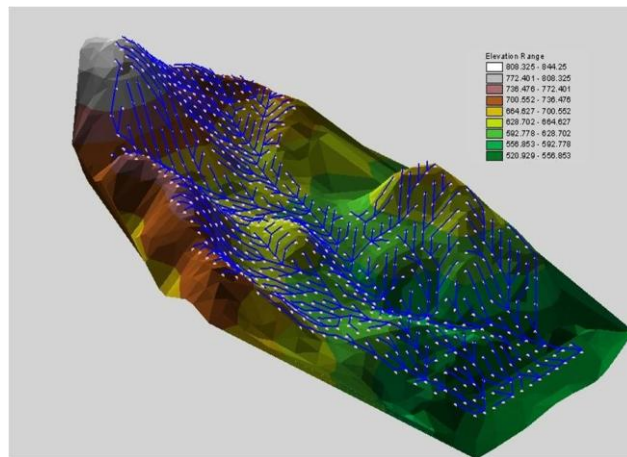


圖 2.17 集水區流徑網路立體圖

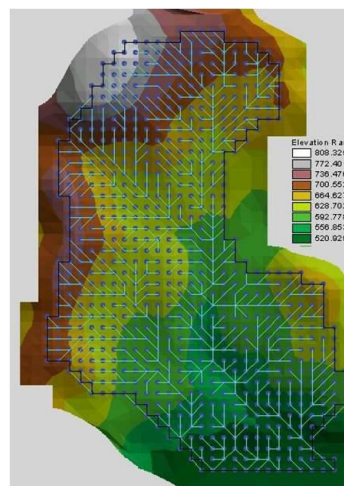


圖 2.18 集水區流徑網路等高線圖

## 第三章 以 MapReduce 架構撰寫直接逕流模組

### 3.1 系統架構

本研究的河川逕流演算法概念來自台大水工試驗所開發的洪水資訊系統—

「地貌型逕流模式」程式模組（Geo Morphology Runoff，即 GRUNOFF.exe），圖

3.1 所示為地貌型逕流模組組織圖。

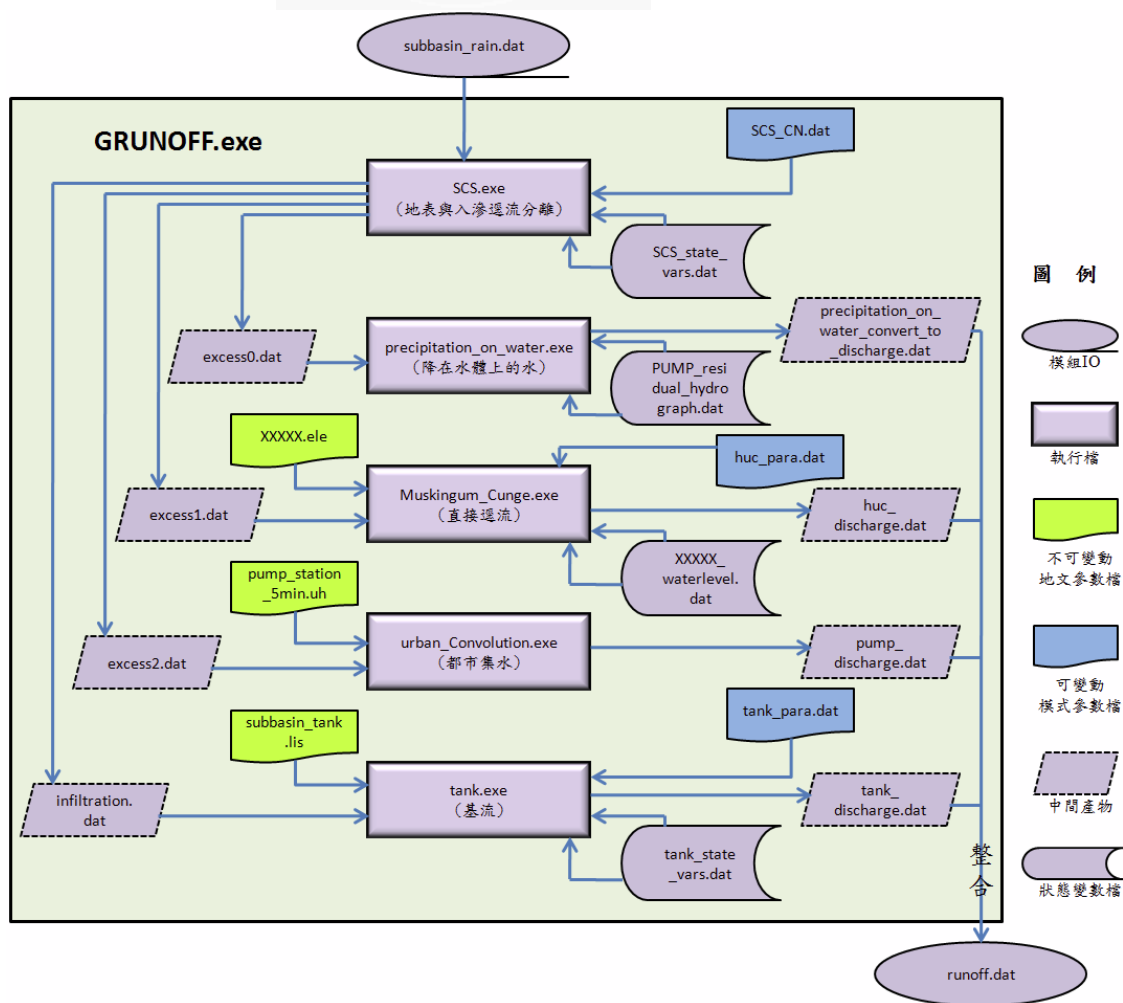


圖 3.1 地貌型逕流模式組織圖

集水區是以降雨為輸入條件，轉換為逕流歷線，此為地貌型逕流模組的運算核心。由圖 3.1 可看出地貌型逕流模組共有五項運算模組，其中三項即為模擬降雨轉換為逕流歷線的三個基本數學模式：

- ◆ 入滲損失模式（即 SCS.exe）：利用 SCS 曲線值法將降雨歷線區分為「有效降雨歷線」和「入滲損失歷線」的模式。
- ◆ 地貌型直接逕流模式（即 Muskingum\_Cunge.exe）：利用馬斯金庚—康吉法（Muskingum-Cunge Method）將集水區有效降雨歷線轉換為集水區出口直接逕流歷線的模式。先前所述將集水區簡化成由許多「單位面積像元」和許多流徑所組成的網絡即為地貌型直接逕流模式的概念。該部份為地貌型逕流模組中計算最繁雜、所耗時間最多之處，也就是本研究所著眼使用 MapReduce 軟體架構撰寫的部份。
- ◆ 基流模式（即 tank.exe）：將入滲損失量轉換為集水區基流歷線(含中間流)的模式。

本研究的目的是使用 MapReduce 軟體架構撰寫地貌型直接逕流模組（Direct Geo Morphology Runoff），地貌型直接逕流模組包含下列幾項運算步驟：

1. 決定各流徑坡向和坡度（因山區集水區有效降雨造成的地表水流方向和速度分別與坡向和坡度有關）。
2. 將落在面積像元上之有效降雨轉換為節點集中流（線性水庫法）。
3. 以「馬斯金庚—康吉法（Muskingum-Cunge Method）」由上游往下游進行演算各流徑的流量歷線。
4. 最後在各集水區出口得到代表該集水區的直接逕流歷線(Discharge Time Series)。以及獲得各集水區總體的降雨量(Rainfall)、入流量(Inflow)、

出流量（Outflow）、儲水量（Storage）以及水平衡（Water Balance）等數據結果。

下圖 3.2 紅框部份為原始地貌型直接逕流模組執行檔及相關運算的輸出入參數檔，圖 3.3 則為本研究所設計之套用 MapReduce 的新直接逕流模組架構。在此新架構中，MapReduce 的直接逕流模組運算相關的參數檔拆成前置詮釋資料檔（.metadata）及簡化了的參數檔。以下章節將詳述這些相關參數檔及套用 MapReduce 的新地貌型直接逕流模組架構。

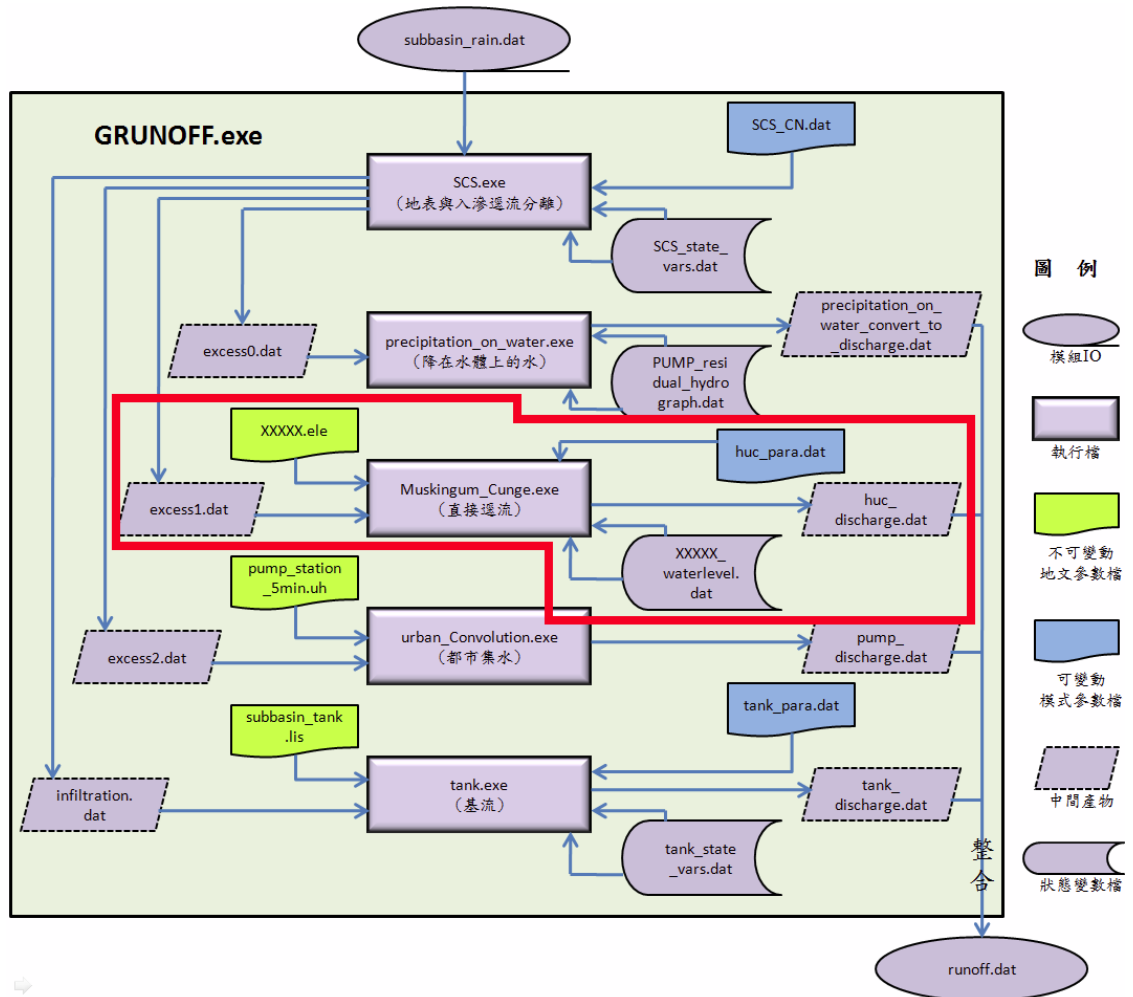


圖 3.2 地貌型直接逕流模組示意圖

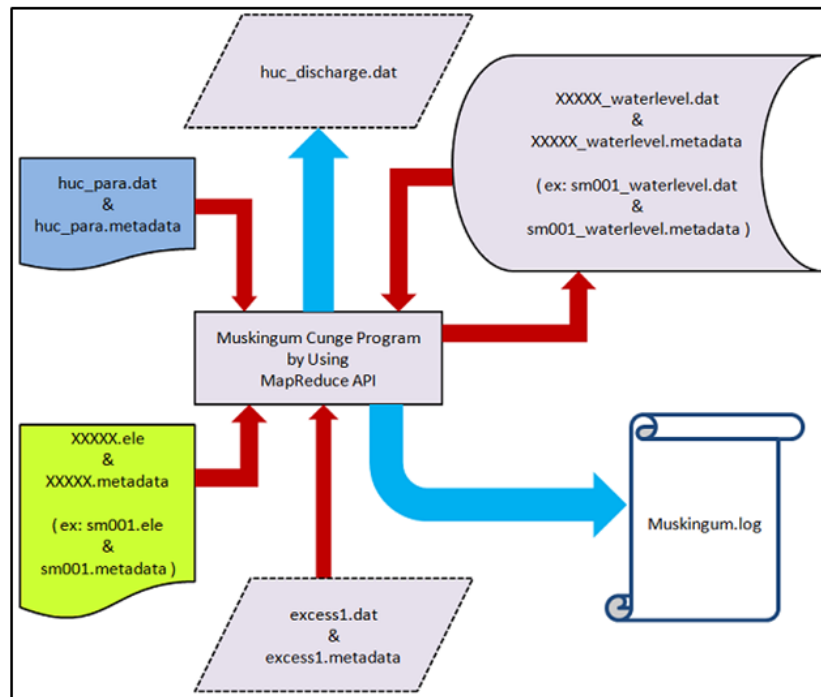


圖 3.3 套用 MapReduce 的地貌型直接逕流模組架構

## 3.2 直接逕流模組運算資料型態與運算流程

### 3.2.1 原始的及修改後的運算資料格式

由圖 3.2 紅框部份的原始直接逕流模組架構可看出進行運算的相關參數檔，列舉及說明如下：

- ✧ excess1.dat：由入滲損失模組（SCS 曲線值法）運算所得之中間產物，裡面記錄著各集水區代號及其超滲降雨歷線（Excess Rainfall Time Series），另外還有集水區數量、資料紀錄時間範圍等前置資訊。

excess1.dat ×				
首筆資料時間=	200908031305↓			
末筆資料時間=	200908100200↓			
資料時間間距(min)=	5↓			
資料筆數=	1884↓			
集水區數=	155↓			
↓				
集水區編號	超滲降雨時間序列(mm/hr)↓			
sm001	0.000000	0.000000	0.000000	0.000000
sm002	0.000000	0.000000	0.000000	0.000000
sm003	0.000000	0.000000	0.000000	0.000000
sm004	0.000000	0.000000	0.000000	0.000000
sm005	0.000000	0.000000	0.000000	0.000000
sm006	0.000000	0.000000	0.000000	0.000000
sm008A	0.000000	0.000000	0.000000	0.000000
sm009	0.000000	0.000000	0.000000	0.000000
sm010	0.000000	0.000000	0.000000	0.000000
sm011	0.000000	0.000000	0.000000	0.000000
	.	.	.	.
	.	.	.	.
	.	.	.	.

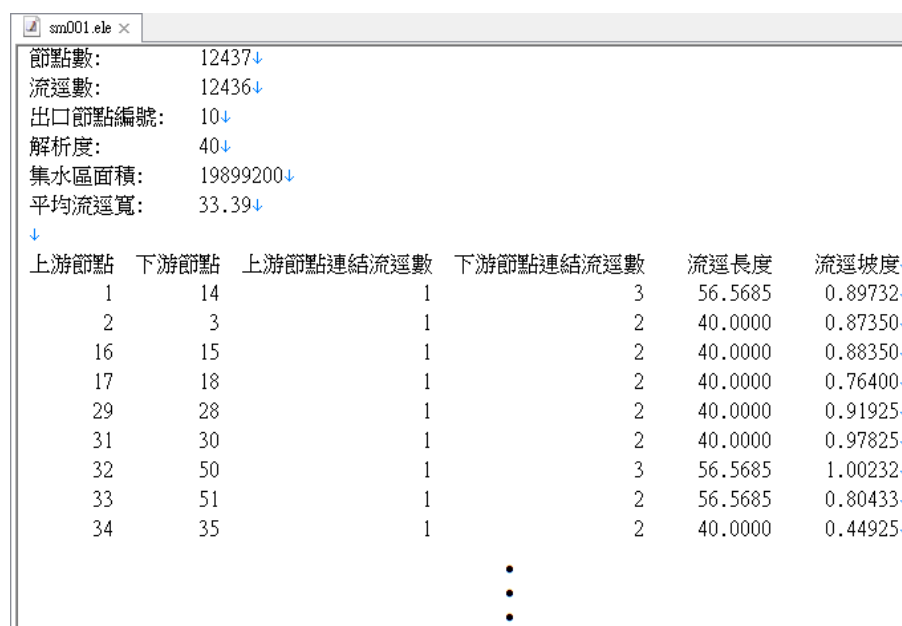
圖 3.4 excess1.dat 格式

- ☆ huc\_para.dat：為一可變動的程式參數檔，裡面記錄各集水區之曼寧值（Manning Constant）與退水常數（Recession Constant）等水文資訊，前置資訊為集水區個數。

huc_para.dat ×		
集水區數:	155↓	
↓		
集水區編號	曼寧n值	退水常數(min)↓
sm001	0.04	40↓
sm002	0.04	40↓
sm003	0.04	40↓
sm004	0.04	40↓
sm005	0.04	40↓
sm006	0.04	40↓
sm008A	0.04	40↓
sm009	0.04	40↓
sm010	0.04	40↓
sm011	0.04	40↓
	.	.
	.	.
	.	.

圖 3.5 huc\_para.dat 格式

- ✧ ele 參數檔：每個集水區皆有個別的 ele 參數檔，為不可變動之地文參數檔。記錄著集水區內每個流徑的上下游節點編號、流徑長度（Reach Slope）及坡度（Reach Length）等資訊，前置資訊記錄該集水區所有的節點數、流徑數及作為集水區出口的節點編號（Outlet）等等。



上游節點	下游節點	上游節點連結流徑數	下游節點連結流徑數	流徑長度	流徑坡度
1	14	1	3	56.5685	0.89732
2	3	1	2	40.0000	0.87350
16	15	1	2	40.0000	0.88350
17	18	1	2	40.0000	0.76400
29	28	1	2	40.0000	0.91925
31	30	1	2	40.0000	0.97825
32	50	1	3	56.5685	1.00232
33	51	1	2	56.5685	0.80433
34	35	1	2	40.0000	0.44925

圖 3.6 ele 參數檔格式

- ✧ waterlevel 變數檔：每個集水區皆有個別的 waterlevel 狀態變數檔。記錄著該集水區中被訂定之流徑編號及流徑上下游的節點水深（Nodal Water Depth），前置資訊則記錄該集水區的初始降雨強度（Initial Rainfall Intensity）、曼寧值與退水常數等。與上述其他變數檔不同的是，waterlevel 變數檔裡的數據每經過一次直接逕流模式計算，便會把計算產生的新數據取代舊數據，下回再運算時便以新的 waterlevel 變數檔計算。

sm001_waterlevel.dat ×	
紀錄時間:	yyyyMMddHHmm↓
曼寧n:	0.04↓
退水常數k(min):	40.00↓
初始降雨強度 $I_0(\text{mm/hr}) \Rightarrow r(t) = I_0 * \exp(-t/k)$ :	0.200000↓
流逕數:	12436↓
每根流逕上的節點數:	2↓
↓	
流逕編號	節點水深(m)...↓
1	0.011976 0.011976 ↓
2	0.012036 0.012036 ↓
3	0.012011 0.012011 ↓
4	0.012342 0.012342 ↓
5	0.011922 0.011922 ↓
6	0.011783 0.011783 ↓
7	0.011730 0.011730 ↓
8	0.012224 0.012224 ↓
	•
	•
	•

圖 3.7 waterlevel 變數檔格式

由上述的各類參數檔可以看出，由於每一類參數檔皆有前置詮釋資料及參數資料本身，因此參數檔內的內容格式並非整齊劃一的二維表，這樣的參數檔格式不易套用 MapReduce 架構來運算，其原因將留待之後的章節討論。所以本研究把各類原始參數檔內的前置詮釋資料與參數資料分離成兩個獨立檔案，分別以 .metadata 與 .dat 為其附檔名。參數資料檔內各欄位以 TAB 字元相隔，即為整齊劃一的二維表；前置詮釋資料檔則將原來所有前置資料的各個數據置於第一列，這些數據同樣以 TAB 字元相隔，如此前置資料各數據亦為僅一列之二維表，第二列之後則注釋這些數據分別代表之意義及此前置資料檔所對應之參數資料檔的各欄位所代表之意義。如此便易於套用 MapReduce 架構，各類參數檔的修改後之格式可見圖 3.8 至圖 3.11。



excess1.metadata ×				
200908031305	200908100200	5	1884	155↓
The parameters above respectively represent :↓				
Start Time , End Time , Duration(min) , Data Number , Subbasin Number↓				
The fields of excess1.dat respectively represent :↓				
Subbasin Name , Excess Rainfall Time Series(mm/hr)←				
excess1.dat ×				
sm001	0.000000	0.000000	0.000000	0.000000
sm002	0.000000	0.000000	0.000000	0.000000
sm003	0.000000	0.000000	0.000000	0.000000
sm004	0.000000	0.000000	0.000000	0.000000
sm005	0.000000	0.000000	0.000000	0.000000
sm006	0.000000	0.000000	0.000000	0.000000
sm008A	0.000000	0.000000	0.000000	0.000000
sm009	0.000000	0.000000	0.000000	0.000000
sm010	0.000000	0.000000	0.000000	0.000000
sm011	0.000000	0.000000	0.000000	0.000000

圖 3.8 修改格式後之 excess1.dat 及 excess1.metadata

huc\_para.metadata ×

155↓

The parameters above respectively represent :↓

Subbasin Number↓

The fields of huc\_para.dat respectively represent :↓

Subbasin Name , Manning Coefficient , Recession Constant k(min)

huc\_para.dat ×

sm001 0.04 40↓

sm002 0.04 40↓

sm003 0.04 40↓

sm004 0.04 40↓

sm005 0.04 40↓

sm006 0.04 40↓

sm008A 0.04 40↓

sm009 0.04 40↓

sm010 0.04 40↓

sm011 0.04 40↓

•

•

•

圖 3.9 修改格式後之 huc\_para.dat 及 huc\_para.metadata

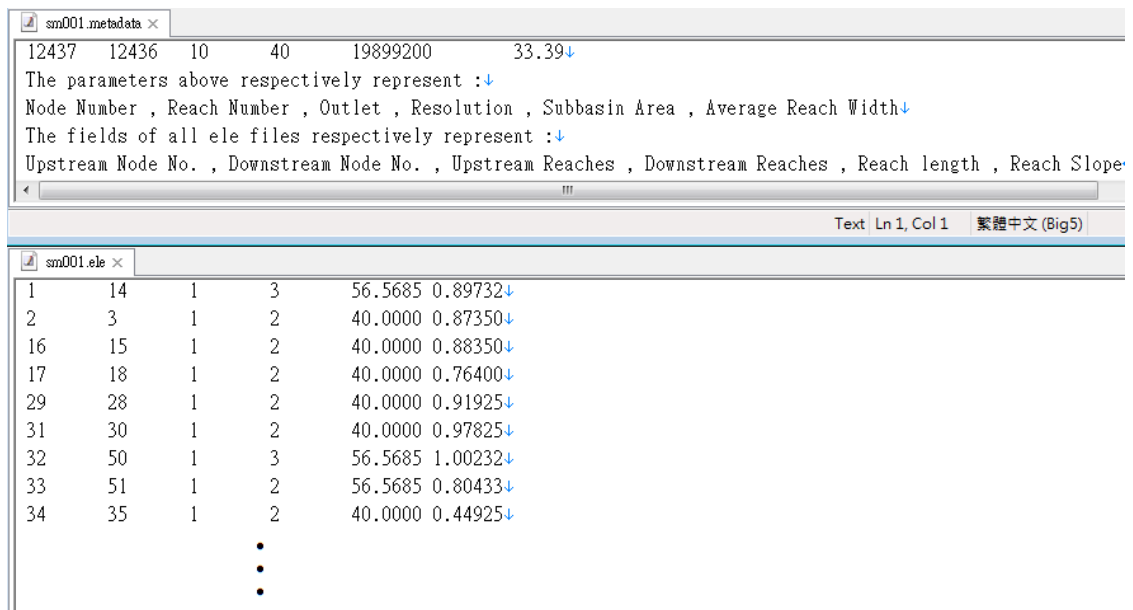


圖 3.10 修改格式後之 ele 參數檔及 ele 詮釋資料檔

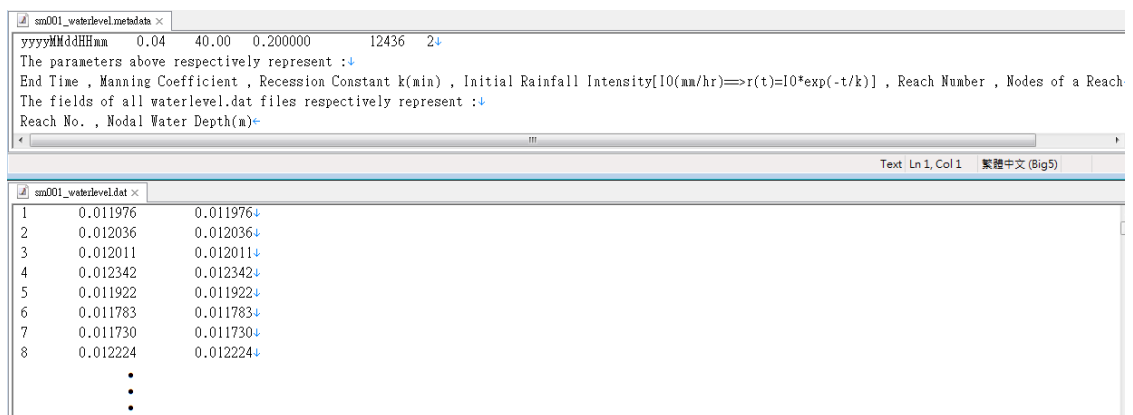


圖 3.11 修改格式後之 waterlevel 變數檔及 waterlevel 詮釋資料檔

### 3.2.2 運算流程

在 2.3 節介紹 MapReduce 軟體架構時曾經提過 Hadoop 叢集系統會啟動分割機制把起始運算輸入資料分割為 Input splits，本研究所使用之分割機制的原理為：若起始運算輸入資料有多個檔案，且這些檔案容量大於 1bytes 卻未超過 HDFS 的儲存單位區塊（block）大小（HDFS 預設之單位區塊為 64MB）時，則每一個

檔皆可視為一個 Input split。由於本研究將直接逕流演算法撰寫於 Map 階段的運算處理函數（map method）當中，因此恰可活用此原理，手動將起始運算輸入資料檔分成多份。藉由這項技巧以增加 Input split 數目，等同於增加平行化 Map 工作數量，以增進運算效能。

接著需要從上一節所介紹之四類參數檔選擇起始的運算輸入資料檔，由於起始運算輸入資料檔的記錄資訊需要包含流域中的每一個集水區，以便之後做為每一個集水區的索引文件以便分別讀取並處理各集水區個別的參數或變數檔，因此前置資料檔、ele 參數檔及 waterlevel 變數檔理所當然不在考慮之中。本研究最後選擇記錄超滲降雨歷線的 excess1.dat 做為起始的運算輸入資料檔。

在進入 Map 運算階段時，會依照開發者於應用程式中指定之分割格式再將 Input split 細分為一個個 Record，之後每個 Record 都會再解析（parse）成一對 Key/Value 的形式來表示以進行 Map 運算處理。本研究選擇 MapReduce 架構預先提供的分割格式之一—TextInputFormat，TextInputFormat 將「Input split 中的每一列字串」皆視為一個 Record，也就是說，假設一個 Input split 中有三列文字，則此 Input split 可分成三個 Record。之後 TextInputFormat 分割格式將 Record 解析成 Key 的形式為「檔案開頭至這一系列文字字串開頭的 Byte offset」，將 Record 解析成 Value 的形式為「這一系列文字字串的內容」（範例如圖 3.12）。

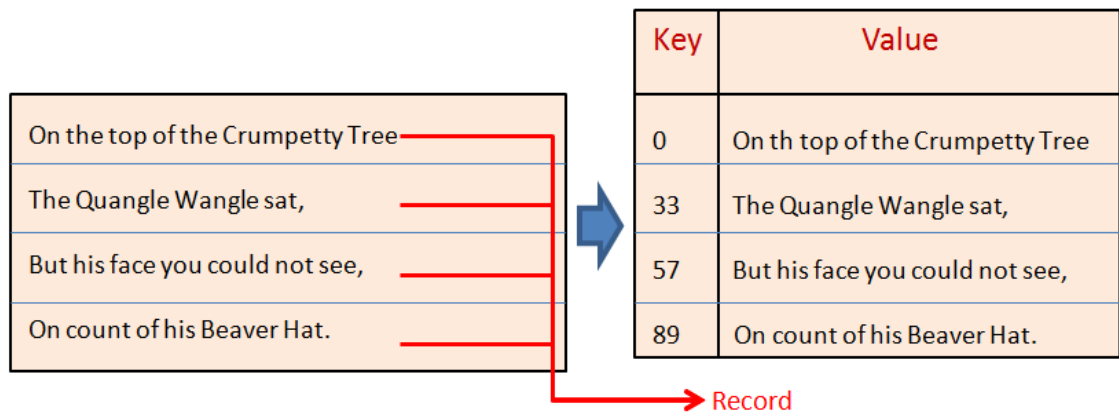


圖 3.12 TextInputFormat 分割格式範例

綜和以上所敘述的概念，我們將 excess1.dat 手動分割為相當數量的小檔（內容皆為若干集水區名稱及其超滲超滲降雨歷線）以增加 Input split 數目，之後透過 TextInputFormat 分割格式得到每個 Value 的內容為各集水區名稱及其超滲降雨歷線。也正因如此，造成 excess1.dat 不能包含前置資料，因為這將使得某些 Value 值出現有別於「集水區名稱及其超滲超滲降雨歷線」的內容，反而使運算更複雜。接著於 Map 階段透過集水區名稱索引並讀取位於 huc\_para.dat 中屬於該集水區的曼寧值、退水常數以及該集水區的 ele 參數檔與 waterlevel 變數檔，讀取到各項計算所需的數據後進行運算並獲得結果。最後在 Reduce 階段將所有集水區的五項數據結果(降雨量、入流量、出流量、儲水量以及水平衡)統整彙集成 Muskingum.log 以及每個集水區的直接逕流歷線數據彙集成 huc\_discharge.dat。圖 3.13 所示即為一項 Map 工作的運算流程範例。

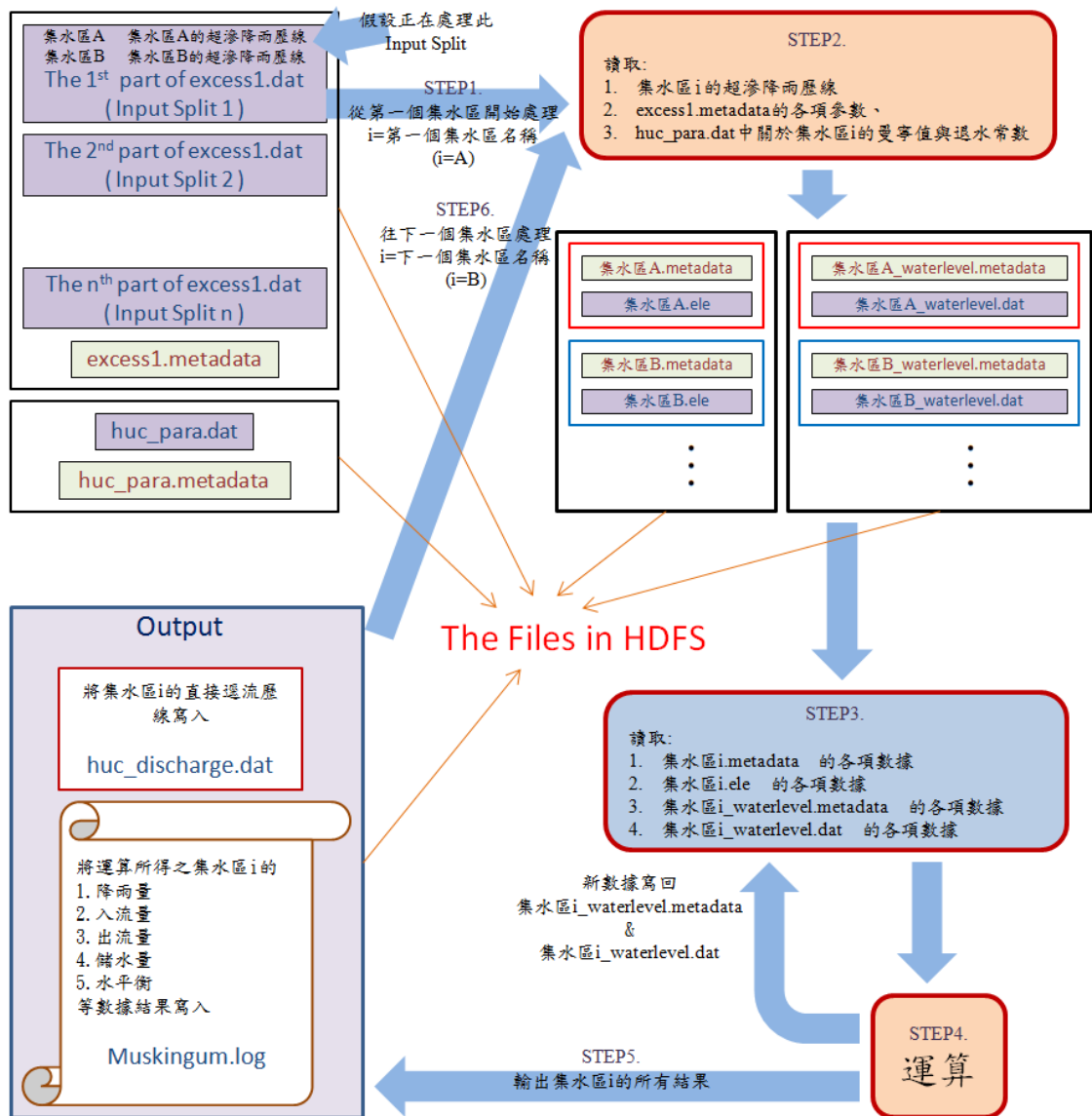


圖 3.13 單一 Map 工作運算流程範例圖

### 3.3 直接逕流模組核心演算法概要

之前系統架構一節提及地貌型直接逕流模組的運算步驟，其中各流徑坡向和坡度的決定是由水文的實測數值高程模型（DEM）資料決定。在將集水區簡化成由許多單位像元面積和許多流徑（小定型渠道）所組成的網絡之後，接下來便是直接逕流模組的演算核心—採用線性水庫方程式模擬「單位像元面積有效降雨」匯流進入「單位像元面積」逕流流徑入口（即節點集中流）的降雨流量反應歷線，以及利用馬斯金庚—康吉法由上游往下游的各流徑進行演算，將各個單位像元面積降雨轉換為集水區出口的直接逕流歷線。

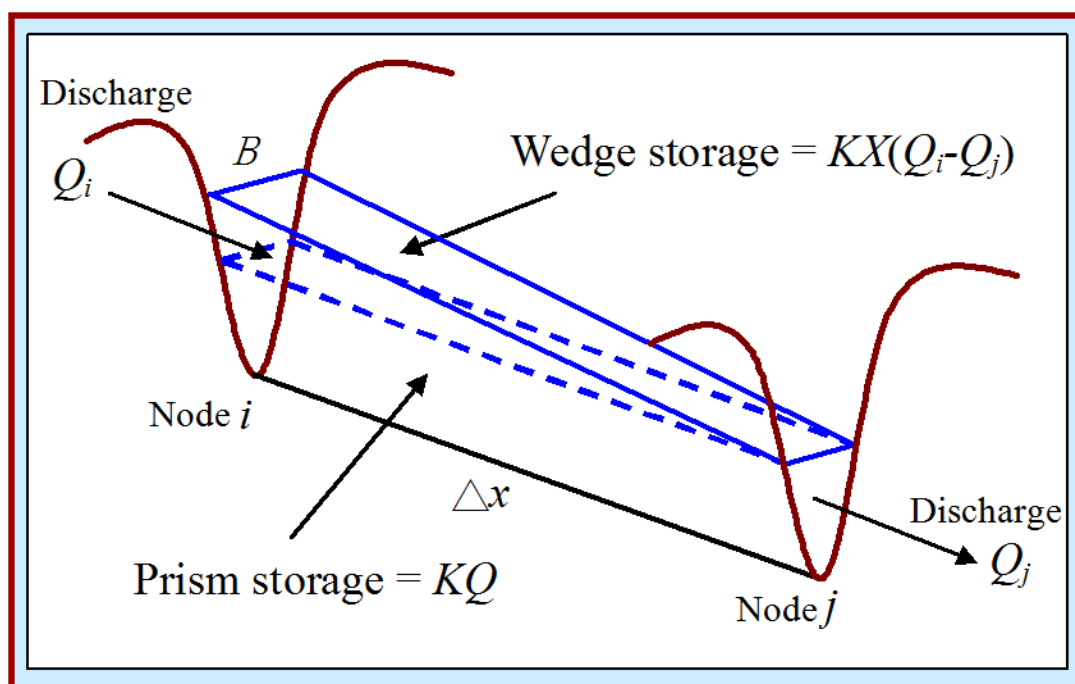


圖 3.14 流徑兩端節點斷面流量示意圖

本論文為簡化水文方面的描述，略去了許多水文專有名詞、觀念及繁雜數學公式，本研究統整了上述直接逕流模組的核心演算步驟，彙整出如圖 3.15 的核心概要演算法：

```

/* Note :
  0 ≤ X (Weight factor) < 0.5 , we set X = 0.2
  Δt : Rainfall duration
  n : Manning coefficient
  nlink : The number of reaches in the subbasin
  nt : The number of discharge time series
*/

procedure Muskingum-Cunge
for each subbasin
begin
  for p := 1 to nlink
  /* nlink also represents the number of nodes in this subbasin except the outlet */
  begin
    i := The upstream node of a reach which order is p;
    j := The downstream node relative to node i of this reach;
    yi := The nodal water depth of node i;
    yj := The nodal water depth of node j;
    Δx := The reach length from node i to j;
    S0 := The reach length from node i to j;

    Qit :=  $\frac{S_0^{\frac{1}{3}}}{2n} y_i^{\frac{8}{3}}$ ; /* The discharge of node i at time t */
    Qjt :=  $\frac{S_0^{\frac{1}{3}}}{2n} y_j^{\frac{8}{3}}$ ; /* The discharge of node j at time t */
    The 1st element of the inflow time series of node j += Qjt;

    for q := 2 to nt
    begin
      Qjt+1 := The qth element of the inflow time series of node i;

      if Qjt = 0 and Qit+1 = 0 then Qjt+1 = 0;
      else
        Qa :=  $\frac{1}{2}(Q_j^t + Q_i^{t+1})$ ; /* Average discharge */
        ck :=  $\frac{2S_0^{\frac{1}{3}}}{3n} (\frac{2nQ_a}{S_0^{\frac{1}{3}}})^{\frac{1}{4}}$ ; /* Celerity */
        K :=  $\frac{\Delta x}{c_k}$ ; /* Storage constant */

        while make sure 0.4K ≤ Δt ≤ 1.6K to ensure C1 ≥ 0 and C3 ≥ 0
        begin
          Qa :=  $\frac{1}{2}(Q_j^t + Q_i^{t+1})$ ;
          ck :=  $\frac{2S_0^{\frac{1}{3}}}{3n} (\frac{2nQ_a}{S_0^{\frac{1}{3}}})^{\frac{1}{4}}$ ;
          K :=  $\frac{\Delta x}{c_k}$ ;
          C1 :=  $\frac{\Delta t - 2KX}{2K(1-X) + \Delta t}$ ;
          C2 :=  $\frac{\Delta t + 2KX}{2K(1-X) + \Delta t}$ ;
          C3 :=  $\frac{2K(1-X) - \Delta t}{2K(1-X) + \Delta t}$ ;
          Qjt+1 := C1Qit+1 + C2Qit + C3Qjt;
          Qit := Qit+1;
          Qjt := Qjt+1;
        end
        The qth element of the inflow time series of node j += Qjt+1;
      end
    end
  end
end
end

```

圖 3.15 地貌型直接逕流模組核心演算法

我們可以看出，此核心演算法主要是描述各個集水區內部的運算內容，而套用 MapReduce 架構的目的是為了不讓集水區與集水區之間的運算完全以循序漸進的方式運作。透過切割成 Input splits，分別交由數個 Map 工作分別平行化處理各自所負責的若干集水區，將使運算效能得到顯著改善。

另外，集水區內部的運算牽涉到各節點（即單位像元面積）及流徑，每個集水區內部節點與流徑動輒上萬甚至數萬個，各節點於程式中皆以 Java 的物件表示（Object），並且每個節點本身也需要以陣列紀錄龐大的降雨流量歷線，因此可以想像集水區內部的計算需要消耗大量記憶體。這一點是 MapReduce 架構無法解決的，即使交由各 TaskTracker 產生的子程序（Child Process）進行 Map 工作（見圖 2.14）平行處理若干集水區，做為 TaskTracker 節點的主機本身還是需要有足夠記憶體來處理被分配到的若干集水區。關於本研究的記憶體使用情形及相關設定將於下一章討論。



## 第四章 系統實作及效能量測

### 4.1 程式開發及測試環境

Hadoop 及其子專案皆建構於 Java 之上，因此應用程式要充分利用 Hadoop 的功能還是必需以 Java 開發。另外目前雖有方法將 Hadoop 裝設於 Windows XP，但僅止於裝設在單一主機做為新手測試或教學用途，若要架設穩定的叢集以提供運算服務還是必需將 Hadoop 裝設於 Linux 等 Unix-Like 作業系統之上。因此為求開發方便，本研究所有程式開發皆於 Linux 終端機指令介面下進行。

本研究鑒於硬體設備以及 TaskTracker 需要有一定數量等原因，選擇台灣國家高速網路與計算中心（National Center for High-Performance Computing, NCHC）所提供的「Hadoop 雲端運算實驗叢集」做為本研究開發及測試的環境。該實驗叢集共由 18 個主機節點所組成，這 18 台主機其中只有一節點同時運作 JobTracker 及 NameNode 背景常駐程序，其他 17 個節點皆為 TaskTracker/DataNode（參考圖 4.1 與圖 4.2），每個 TaskTracker/DataNode 節點皆有一四核心 CPU 及 8GB 之記憶體。另外該實驗叢集尚安裝一套開放源碼軟體 Ganglia，可監控叢集目前整體狀態（如 CPU、記憶體、硬碟等使用情形）以網頁圖形介面方式讓管理者方便監控。

軟體名稱	版本	備註
作業系統	Debian Linux 6.0.1	
Java	1.6.0 Update 24	
Hadoop	0.20.2	
其他軟體	Ganglia 3.1.7	開放源碼的分散式叢集監控軟體

表 4.1 國網中心 Hadoop 實驗平台建置環境

## hadoop Hadoop Map/Reduce Administration

State: RUNNING

Started: Sat May 07 22:55:03 CST 2011

Version: 0.20.1+169.113, r6c765a47a9291470d3d8814c98155115d109d715

Compiled: Sun Sep 12 05:39:27 UTC 2010 by root

Identifier: 201105072255

### Cluster Summary (Heap Size is 543.56 MB/1.78 GB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	40	2611	17	68	68	8.00	0

### Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

### Running Jobs

Jobid	Priority	User	Name	Map %	Map	Ma
-------	----------	------	------	-------	-----	----

圖 4.1 國網中心 Hadoop 實驗平台的 JobTracker 網頁狀態介面

Started:	Sat May 07 22:55:03 CST 2011
Version:	0.20.1+169.113, r6c765a47a9291470d3d8814c98155115d109d715
Compiled:	Sun Sep 12 05:39:27 UTC 2010 by root
Upgrades:	There are no upgrades in progress.

[Browse the filesystem](#)

[Namenode Logs](#)

### Cluster Summary

2523854 files and directories, 3065149 blocks = 5589003 total. Heap Size is 1.68 GB / 1.78 GB (94%)

Configured Capacity	:	22.51 TB
DFS Used	:	19.94 TB
Non DFS Used	:	1.17 TB
DFS Remaining	:	1.39 TB
DFS Used%	:	88.6 %
DFS Remaining%	:	6.19 %
Live Nodes	:	17
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

### NameNode Storage:

Storage Directory	Type	State
/var/lib/hadoop/cache/hadoop/dfs/name	IMAGE_AND_EDITS	Active

圖 4.2 國網中心 Hadoop 實驗平台的 NameNode 網頁狀態介面

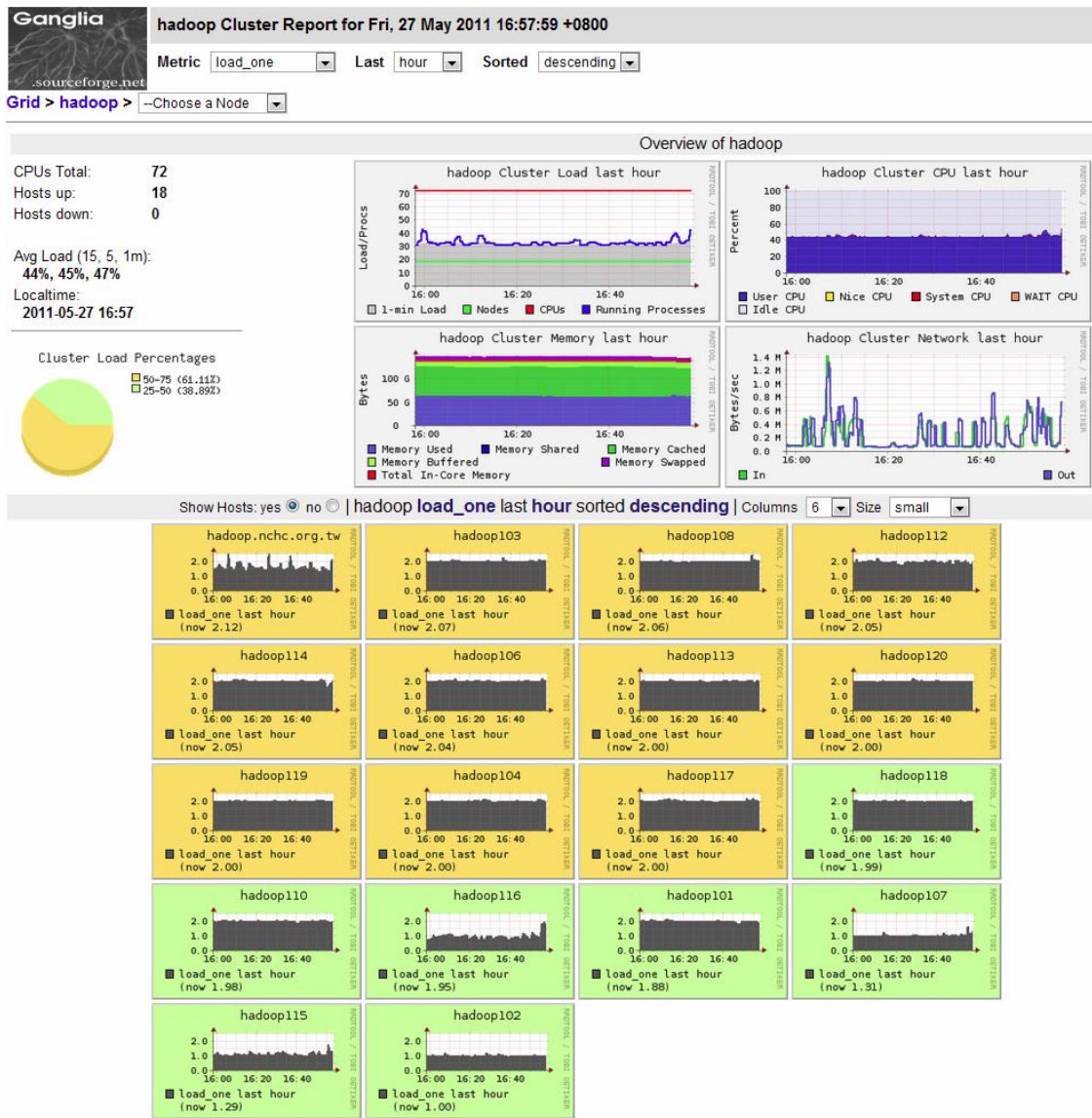


圖 4.3 國網中心 Hadoop 實驗平台的 Ganglia 監控介面

## 4.2 地貌型直接逕流模組程式效能量測及比較

本研究主旨為證實以 MapReduce 架構撰寫地貌型直接逕流模組，藉由 Hadoop 叢集平台的運算可使效能有所提升，因此勢必需要另外測量原始地貌型直接逕流模組的運算效能以做為對照組相互比較。台大水工試驗研究所開發的地貌型逕流模式所有模組（包含直接逕流模組）皆以 C 語言所撰寫，為了不使程式語言之間

的根本差異影響量測數據的可靠性，本研究將原始的地貌型直接逕流模組重新以 Java 語言改寫並在不屬於 Hadoop 叢集的另一台獨立的 Linux 主機上進行運算，使得量測數據更加地可靠。

#### 4.2.1 量測關鍵要素

前一章直接逕流核心演算法一節曾提過集水區內部的運算牽涉到內部上萬甚至數萬個節點物件，每個節點本身也需要有相當大的降雨流量歷線紀錄陣列，因此每個 TaskTracker 的任一子程序皆需配給足夠的記憶體進行一項 Map 工作，來處理一個 Input split 裡所記錄的若干集水區。Hadoop 雲端運算實驗平台的 17 個 TaskTracker/DataNode 節點皆擁有 8GB 記憶體，因此為使叢集系統穩定，國家高速網路中心原先設定叢集中每個 TaskTracker 皆最多只能產生 4 個子程序進行 Map 或 Reduce 工作，每個子程序最多能使用 2000MB 記憶體（如圖 4.4）。

由 excess1.metadata 以及 huc\_para.metadata 所記錄的資料，可以得知時間範圍介於西元 2009 年 8 月 3 日 13 時 5 分至西元 2009 年 8 月 10 日 2 時 0 分所需計算之淡水河流域集水區個數有 155 個（見附錄 1）。經由一般單機運算測試，發現 155 個集水區中有 9 個集水區內部節點超過 16000 個，計算的記憶體需求量到達 2500MB。經與 Hadoop 雲端運算實驗平台負責人溝通，平台負責人允諾給予一星期時間將每個子程序最多能使用之記憶體量調至 2500MB 以助於本研究進行。之後將針對各種測試情境及測試結果做討論，會以完整 155 個集水區、子程序最大記憶體使用量 2500MB；以及刪減過的 146 個集水區（刪減超過 16000 個內部節點的 9 個集水區，見附錄 1）、子程序最大記憶體使用量 2000MB，分屬不同測試情境進行討論。

本研究另一量測要素，便是前一章所提到將起始運算輸入資料檔 excess1.dat 手動分成數份增加 Input split 數，等同於增加平行 Map 工作數以增進運算效能。本研究為了不造成 Hadoop 實驗叢集系統太大負擔，讓同一個 TaskTracker 處理太多不同的 Map 工作，決定將 excess1.dat 分成接近 TaskTracker 數量的 16 份做為上限。另外再於刪減過的 146 個集水區（刪減超過 16000 個內部節點的 9 個集水區）、子程序最大記憶體使用量 2000MB 的測試情境中，將 excess1.dat 另外分成 8 份、4 份及 2 份亦分屬不同情境討論，詳細量測數據及與單機測試的比較結果於下面兩節敘述。

```
<property>
  <name>mapred.tasktracker.map.tasks.maximum</name>
  <value>4</value>
  <description>2010-04-21: Jazz - Increase number of mapper and reducer to fit 4-CPU's.</description>
</property>
<property>
  <name>mapred.tasktracker.reduce.tasks.maximum</name>
  <value>4</value>
  <description>2010-04-21: Jazz - Increase number of mapper and reducer to fit 4-CPU's.</description>
</property>
<property>
  <name>mapred.child.java.opts</name>
  <value>-Xmx2000m</value>
  <description>
    2011-04-28: Jazz - decrease memory limit of task to 2000MB for stability.
    2011-04-21: Jazz - Increase memory limit of task to 2500MB.
    2011-03-08: Jazz - Increase memory limit of task to 2000MB.
    2010-09-12: Jazz - Increase memory limit of task to 1000MB.</description>
</property>
```

圖 4.4 Hadoop 實驗叢集的 TaskTracker 之子程序最大記憶體使用量設定圖

## 4.2.2 單機運算測試

在探討 Hadoop 叢集中各種量測情境之前，先對於本研究於單機對各集水區進行循序運算所得到的運算效能數據作整理，以檢視不套用 MapReduce 架構的原始地貌型直接逕流模組所消耗的時間。

- ◆ 情境一：以完整 155 個集水區，並分配予最大記憶體使用量 2500MB，對各集水區進行循序運算計算。結果如圖 4.5 所示，運算效能以毫秒 (millisecond)

為單位，總共需花費 72310861 毫秒，大約 20 小時 5 分又 11 秒左右，相當地耗時。

```
hadoop@hadoop3:~$ java -Xmx2500m Muskingum_Cunge
sm001 Rainfall = 4398440.20 cubic meters
sm001 Inflow = 4371504.90 cubic meters
sm001 Storage = 26189.10 cubic meters
sm001 Outflow = 4346239.92 cubic meters
sm001 Water Balance = 1.0
sm002 Rainfall = 4002879.70 cubic meters
sm002 Inflow = 3982099.24 cubic meters
sm002 Storage = 41364.55 cubic meters
sm002 Outflow = 3941993.44 cubic meters
sm002 Water Balance = 1.0
sm003 Rainfall = 6101241.18 cubic meters
sm003 Inflow = 6080834.44 cubic meters
sm003 Storage = 27962.18 cubic meters
sm003 Outflow = 6054844.17 cubic meters
sm003 Water Balance = 1.0
.
.
.
.
sd040d Rainfall = 745649.29 cubic meters
sd040d Inflow = 744604.35 cubic meters
sd040d Storage = 5146.95 cubic meters
sd040d Outflow = 740132.86 cubic meters
sd040d Water Balance = 1.0
sd041 Rainfall = 1436027.22 cubic meters
sd041 Inflow = 1434137.65 cubic meters
sd041 Storage = 17506.53 cubic meters
sd041 Outflow = 1419486.48 cubic meters
sd041 Water Balance = 1.0
sd042 Rainfall = 2197041.63 cubic meters
sd042 Inflow = 2194411.49 cubic meters
sd042 Storage = 18699.32 cubic meters
sd042 Outflow = 2181495.26 cubic meters
sd042 Water Balance = 1.0
sd043 Rainfall = 2119105.71 cubic meters
sd043 Inflow = 2115888.91 cubic meters
sd043 Storage = 30146.90 cubic meters
sd043 Outflow = 2091686.93 cubic meters
sd043 Water Balance = 1.0
The Computer spends 72310861 milliseconds excuting this program.
```

圖 4.5 單機計算測試情境一的結果

- ◆ 情境二：以刪減過的 146 個集水區，並分配予最大記憶體使用量 2000MB，對於各集水區進行循序運算計算。結果如圖 4.6 所示，總共需花費 62245352 毫秒，大約 17 小時 17 分又 25 秒左右，同樣也消耗了不少時間。



```

hadoop@hadoop3:~$ java -Xmx2000m Muskingum_Gunge
sm001 Rainfall = 4398440.20 cubic meters
sm001 Inflow = 4371504.90 cubic meters
sm001 Storage = 26189.10 cubic meters
sm001 Outflow = 4346239.92 cubic meters
sm001 Water Balance = 1.0
sm002 Rainfall = 4002879.70 cubic meters
sm002 Inflow = 3982099.24 cubic meters
sm002 Storage = 41364.55 cubic meters
sm002 Outflow = 3941993.44 cubic meters
sm002 Water Balance = 1.0
sm003 Rainfall = 6101241.18 cubic meters
sm003 Inflow = 6080834.44 cubic meters
sm003 Storage = 27962.18 cubic meters
sm003 Outflow = 6054844.17 cubic meters
sm003 Water Balance = 1.0
.
.
.
.
.
sd040d Rainfall = 745649.29 cubic meters
sd040d Inflow = 744604.35 cubic meters
sd040d Storage = 5146.95 cubic meters
sd040d Outflow = 740132.86 cubic meters
sd040d Water Balance = 1.0
sd041 Rainfall = 1436027.22 cubic meters
sd041 Inflow = 1434137.65 cubic meters
sd041 Storage = 17506.53 cubic meters
sd041 Outflow = 1419486.48 cubic meters
sd041 Water Balance = 1.0
sd042 Rainfall = 2197041.63 cubic meters
sd042 Inflow = 2194411.49 cubic meters
sd042 Storage = 18699.32 cubic meters
sd042 Outflow = 2181495.26 cubic meters
sd042 Water Balance = 1.0
sd043 Rainfall = 2119105.71 cubic meters
sd043 Inflow = 2115888.91 cubic meters
sd043 Storage = 30146.90 cubic meters
sd043 Outflow = 2091686.93 cubic meters
sd043 Water Balance = 1.0
The Computer spends 62245352 milliseconds excuting this program.

```

圖 4.6 單機計算測試情境二的結果

### 4.2.3 Hadoop 叢集中各種量測情境探討

這裡開始探討在 Hadoop 實驗叢集中各種量測情境，測試以 MapReduce 架構改寫的地貌型直接逕流模組效能。本小節所有量測數據的展示除了終端機指令執行畫面外，尚擷取出 JobTracker 網頁監控介面中關於此 MapReduce 運算工作的監控畫面，兩相對照使得說明能更加完整。

◆ 情境一：以完整 155 個集水區、子程序最大記憶體使用量 2500MB 且

excess1.dat 分為 16 個小檔於 Hadoop 實驗叢集進行 MapReduce 運算。量測結

果如圖 4.7 指令執行畫面所示，總共需花費 13926486 毫秒，較前個小節單機

運算情境一同樣記憶體使用量及集水區數的運算速率約快了 5 倍。

```
h824@hadoop:~$ hadoop jar MRMC.jar MapReduce_Muskingum_Cunge input2 output
11/04/28 16:58:32 INFO mapred.FileInputFormat: Total input paths to process : 16
11/04/28 16:58:33 INFO mapred.JobClient: Running job: job_201104210939_0598
11/04/28 16:58:34 INFO mapred.JobClient: map 0% reduce 0%
11/04/28 17:01:58 INFO mapred.JobClient: map 1% reduce 0%
11/04/28 17:02:27 INFO mapred.JobClient: map 2% reduce 0%
11/04/28 17:03:42 INFO mapred.JobClient: map 3% reduce 0%
11/04/28 17:04:56 INFO mapred.JobClient: map 4% reduce 0%
11/04/28 17:05:24 INFO mapred.JobClient: map 5% reduce 0%
11/04/28 17:05:33 INFO mapred.JobClient: map 6% reduce 0%
11/04/28 17:06:51 INFO mapred.JobClient: map 8% reduce 0%
11/04/28 17:07:53 INFO mapred.JobClient: map 9% reduce 0%
11/04/28 17:07:54 INFO mapred.JobClient: map 10% reduce 0%
11/04/28 17:08:12 INFO mapred.JobClient: map 11% reduce 0%
11/04/28 17:08:42 INFO mapred.JobClient: map 12% reduce 0%
.
.
.
.
11/04/28 20:45:28 INFO mapred.JobClient: map 99% reduce 31%
11/04/28 20:50:22 INFO mapred.JobClient: map 100% reduce 31%
11/04/28 20:50:36 INFO mapred.JobClient: map 100% reduce 100%
11/04/28 20:50:38 INFO mapred.JobClient: Job complete: job_201104210939_0598
11/04/28 20:50:38 INFO mapred.JobClient: Counters: 19
11/04/28 20:50:38 INFO mapred.JobClient: Job Counters
11/04/28 20:50:38 INFO mapred.JobClient:   Launched reduce tasks=1
11/04/28 20:50:38 INFO mapred.JobClient:   Rack-local map tasks=18
11/04/28 20:50:38 INFO mapred.JobClient:   Launched map tasks=29
11/04/28 20:50:38 INFO mapred.JobClient:   Data-local map tasks=11
11/04/28 20:50:38 INFO mapred.JobClient: FileSystemCounters
11/04/28 20:50:38 INFO mapred.JobClient:   FILE_BYTES_READ=1823997
11/04/28 20:50:38 INFO mapred.JobClient:   HDFS_BYTES_READ=86572464
11/04/28 20:50:38 INFO mapred.JobClient:   FILE_BYTES_WRITTEN=3648596
11/04/28 20:50:38 INFO mapred.JobClient:   HDFS_BYTES_WRITTEN=36867744
11/04/28 20:50:38 INFO mapred.JobClient: Map-Reduce Framework
11/04/28 20:50:38 INFO mapred.JobClient:   Reduce input groups=155
11/04/28 20:50:38 INFO mapred.JobClient:   Combine output records=0
11/04/28 20:50:38 INFO mapred.JobClient:   Map input records=155
11/04/28 20:50:38 INFO mapred.JobClient:   Reduce shuffle bytes=1703282
11/04/28 20:50:38 INFO mapred.JobClient:   Reduce output records=155
11/04/28 20:50:38 INFO mapred.JobClient:   Spilled Records=310
11/04/28 20:50:38 INFO mapred.JobClient:   Map output bytes=1823371
11/04/28 20:50:38 INFO mapred.JobClient:   Map input bytes=2636091
11/04/28 20:50:38 INFO mapred.JobClient:   Combine input records=0
11/04/28 20:50:38 INFO mapred.JobClient:   Map output records=155
11/04/28 20:50:38 INFO mapred.JobClient:   Reduce input records=155
The Hadoop Cluster spends 13926486 milliseconds executing this program.
```

圖 4.7 Hadoop 實驗叢集測試情境一的結果



情境一的效能測試數據亦可參考圖 4.8 的 MapReduce 運算工作網頁監控介面，網頁監控畫面所顯示的運算消耗時間大約 3 小時 52 分又 5 秒左右。

同時我們亦可從網頁監控畫面看出確有 16 項 Map 工作平行執行，以及其他  
的補充數據資料。

## Hadoop job\_201104210939\_0598 on **hadoop**

User: h824

Job Name: MapReduce\_Muskingum\_Cunge

Job File: [https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job\\_201104210939\\_0598/job.xml](https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job_201104210939_0598/job.xml)

Job Setup: Successful

Status: Succeeded

Started at: Thu Apr 28 16:58:32 CST 2011

Finished at: Thu Apr 28 20:50:38 CST 2011

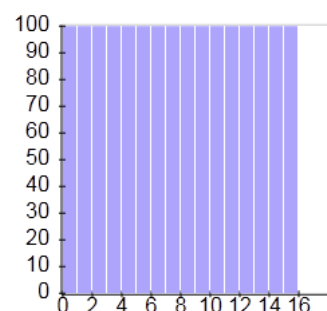
Finished in: **3hrs, 52mins, 5sec**

Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	<b>16</b>	0	0	16	0	6 / 7
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	18
	Launched map tasks	0	0	29
	Data-local map tasks	0	0	11
FileSystemCounters	FILE_BYTES_READ	0	1,823,997	1,823,997
	HDFS_BYTES_READ	86,572,464	0	86,572,464
	FILE_BYTES_WRITTEN	1,824,599	1,823,997	3,648,596
	HDFS_BYTES_WRITTEN	35,043,572	1,824,172	36,867,744
Map-Reduce Framework	Reduce input groups	0	155	155
	Combine output records	0	0	0
	Map input records	155	0	155
	Reduce shuffle bytes	0	1,703,282	1,703,282
	Reduce output records	0	155	155
	Spilled Records	155	155	310
	Map output bytes	1,823,371	0	1,823,371
	Map input bytes	2,636,091	0	2,636,091
	Map output records	155	0	155
	Combine input records	0	0	0
	Reduce input records	0	155	155

Map Completion Graph - close



Reduce Completion Graph - close

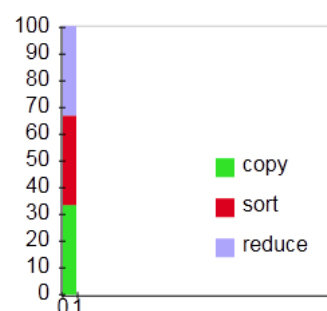


圖 4.8 Hadoop 實驗叢集測試情境一的 MapReduce 工作監控畫面

◆ 情境二：以刪減過的 146 個集水區、子程序最大記憶體使用量 2000MB 且

excess1.dat 分為 16 個小檔於 Hadoop 實驗叢集進行 MapReduce 運算。量測結

果如圖 4.9 指令執行畫面所示，總共需花費 10259880 毫秒，較前小節單機運

算情境二同樣記憶體使用量及集水區數的運算速率甚至快了約 6 倍。

```
h824@hadoop:~$ hadoop jar MRMC.jar MapReduce_Muskingum_Cunge input_incon_16parts output_incon_16parts
11/05/20 21:09:34 INFO mapred.FileInputFormat: Total input paths to process : 16
11/05/20 21:09:34 INFO mapred.JobClient: Running job: job_201105072255_1922
11/05/20 21:09:35 INFO mapred.JobClient: map 0% reduce 0%
11/05/20 21:11:31 INFO mapred.JobClient: map 1% reduce 0%
11/05/20 21:13:15 INFO mapred.JobClient: map 2% reduce 0%
11/05/20 21:13:57 INFO mapred.JobClient: map 3% reduce 0%
11/05/20 21:15:21 INFO mapred.JobClient: map 5% reduce 0%
11/05/20 21:15:49 INFO mapred.JobClient: map 6% reduce 0%
11/05/20 21:16:24 INFO mapred.JobClient: map 7% reduce 0%
11/05/20 21:16:49 INFO mapred.JobClient: map 8% reduce 0%
11/05/20 21:17:27 INFO mapred.JobClient: map 9% reduce 0%
11/05/20 21:18:40 INFO mapred.JobClient: map 10% reduce 0%
11/05/20 21:19:20 INFO mapred.JobClient: map 11% reduce 0%
11/05/20 21:19:49 INFO mapred.JobClient: map 12% reduce 0%
.
.
.
.
11/05/20 23:07:27 INFO mapred.JobClient: map 97% reduce 29%
11/05/20 23:07:41 INFO mapred.JobClient: map 97% reduce 31%
11/05/20 23:23:01 INFO mapred.JobClient: map 98% reduce 31%
11/05/20 23:37:43 INFO mapred.JobClient: map 99% reduce 31%
11/05/21 00:00:17 INFO mapred.JobClient: map 100% reduce 31%
11/05/21 00:00:25 INFO mapred.JobClient: map 100% reduce 94%
11/05/21 00:00:31 INFO mapred.JobClient: map 100% reduce 100%
11/05/21 00:00:33 INFO mapred.JobClient: Job complete: job_201105072255_1922
11/05/21 00:00:33 INFO mapred.JobClient: Counters: 19
11/05/21 00:00:33 INFO mapred.JobClient:   Job Counters
11/05/21 00:00:33 INFO mapred.JobClient:     Launched reduce tasks=1
11/05/21 00:00:33 INFO mapred.JobClient:     Rack-local map tasks=11
11/05/21 00:00:33 INFO mapred.JobClient:     Launched map tasks=22
11/05/21 00:00:33 INFO mapred.JobClient:     Data-local map tasks=11
11/05/21 00:00:33 INFO mapred.JobClient:   FileSystemCounters
11/05/21 00:00:33 INFO mapred.JobClient:     FILE_BYTES_READ=1715507
11/05/21 00:00:33 INFO mapred.JobClient:     HDFS_BYTES_READ=77156146
11/05/21 00:00:33 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=3431616
11/05/21 00:00:33 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=32908291
11/05/21 00:00:33 INFO mapred.JobClient:   Map-Reduce Framework
11/05/21 00:00:33 INFO mapred.JobClient:     Reduce input groups=146
11/05/21 00:00:33 INFO mapred.JobClient:     Combine output records=0
11/05/21 00:00:33 INFO mapred.JobClient:     Map input records=146
11/05/21 00:00:33 INFO mapred.JobClient:     Reduce shuffle bytes=1596561
11/05/21 00:00:33 INFO mapred.JobClient:     Reduce output records=146
11/05/21 00:00:33 INFO mapred.JobClient:     Spilled Records=292
11/05/21 00:00:33 INFO mapred.JobClient:     Map output bytes=1714917
11/05/21 00:00:33 INFO mapred.JobClient:     Map input bytes=2483038
11/05/21 00:00:33 INFO mapred.JobClient:     Combine input records=0
11/05/21 00:00:33 INFO mapred.JobClient:     Map output records=146
11/05/21 00:00:33 INFO mapred.JobClient:     Reduce input records=146
The Hadoop Cluster spends 10259880 milliseconds executing this program.
```

圖 4.9 Hadoop 實驗叢集測試情境二的結果

情境二的效能測試數據同樣可參考圖 4.10 的 MapReduce 工作網頁監控畫面，顯示的運算消耗時間大約 2 小時 50 分又 58 秒左右。另外從網頁監控畫面得知確有 16 項 Map 工作平行執行。

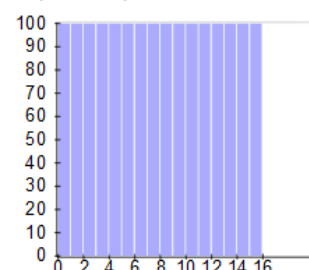
### Hadoop job\_201105072255\_1922 on [hadoop](#)

User: h824  
 Job Name: MapReduce\_Muskingum\_Cunge  
 Job File: [https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job\\_201105072255\\_1922/job.xml](https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job_201105072255_1922/job.xml)  
 Job Setup: Successful  
 Status: Succeeded  
 Started at: Fri May 20 21:09:34 CST 2011  
 Finished at: Sat May 21 00:00:33 CST 2011  
 Finished in: **2hrs, 50mins, 58sec**  
 Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	16	0	0	16	0	0 / 6
reduce	100.00%	1	0	0	1	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	11
	Launched map tasks	0	0	22
	Data-local map tasks	0	0	11
FileSystemCounters	FILE_BYTES_READ	0	1,715,507	1,715,507
	HDFS_BYTES_READ	77,156,146	0	77,156,146
	FILE_BYTES_WRITTEN	1,716,109	1,715,507	3,431,616
	HDFS_BYTES_WRITTEN	31,192,627	1,715,664	32,908,291
Map-Reduce Framework	Reduce input groups	0	146	146
	Combine output records	0	0	0
	Map input records	146	0	146
	Reduce shuffle bytes	0	1,596,561	1,596,561
	Reduce output records	0	146	146
	Spilled Records	146	146	292
	Map output bytes	1,714,917	0	1,714,917
	Map input bytes	2,483,038	0	2,483,038
	Map output records	146	0	146
	Combine input records	0	0	0
	Reduce input records	0	146	146

Map Completion Graph - [close](#)



Reduce Completion Graph - [close](#)

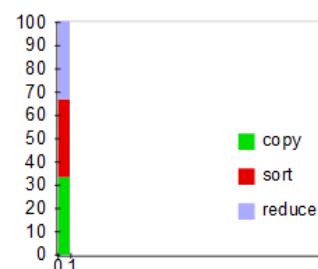


圖 4.10 Hadoop 實驗叢集測試情境二的 MapReduce 工作監控畫面

- ◆ 情境三：自此情境後皆以刪減過的 146 個集水區、子程序最大記憶體使用量 2000MB 進行測試，唯有改變 excess1.dat 的分割數量以測試若變動平行運作的 Map 工作數量是否對效能造成影響。這裡把 excess1.dat 分為 8 個小檔於 Hadoop 實驗叢集進行 MapReduce 運算，量測結果如圖 4.11 指令執行畫面所示，總共花費 13862499 毫秒，由此可知減少平行運作的 Map 工作數確實會使得運算消耗較多時間。

```
h824@hadoop:~$ hadoop jar MRMC.jar MapReduce_Muskingum_Cunge input_incom_8parts output_incom_8parts
11/05/21 01:30:02 INFO mapred.FileInputFormat: Total input paths to process : 8
11/05/21 01:30:02 INFO mapred.JobClient: Running job: job_201105072255_2154
11/05/21 01:30:03 INFO mapred.JobClient: map 0% reduce 0%
11/05/21 01:31:58 INFO mapred.JobClient: map 1% reduce 0%
11/05/21 01:37:38 INFO mapred.JobClient: map 2% reduce 0%
11/05/21 01:39:56 INFO mapred.JobClient: map 3% reduce 0%
11/05/21 01:40:21 INFO mapred.JobClient: map 4% reduce 0%
11/05/21 01:42:20 INFO mapred.JobClient: map 5% reduce 0%
11/05/21 01:43:41 INFO mapred.JobClient: map 6% reduce 0%
11/05/21 01:45:59 INFO mapred.JobClient: map 7% reduce 0%
11/05/21 01:48:42 INFO mapred.JobClient: map 9% reduce 0%
11/05/21 01:53:31 INFO mapred.JobClient: map 10% reduce 0%
11/05/21 01:54:44 INFO mapred.JobClient: map 11% reduce 0%
11/05/21 01:56:41 INFO mapred.JobClient: map 12% reduce 0%
.
.
.
.
11/05/21 04:59:09 INFO mapred.JobClient: map 98% reduce 25%
11/05/21 05:00:47 INFO mapred.JobClient: map 99% reduce 25%
11/05/21 05:00:57 INFO mapred.JobClient: map 99% reduce 29%
11/05/21 05:20:44 INFO mapred.JobClient: map 100% reduce 29%
11/05/21 05:21:02 INFO mapred.JobClient: map 100% reduce 100%
11/05/21 05:21:04 INFO mapred.JobClient: Job complete: job_201105072255_2154
11/05/21 05:21:04 INFO mapred.JobClient: Counters: 19
11/05/21 05:21:04 INFO mapred.JobClient: Job Counters
11/05/21 05:21:04 INFO mapred.JobClient:   Launched reduce tasks=1
11/05/21 05:21:04 INFO mapred.JobClient:   Rack-local map tasks=7
11/05/21 05:21:04 INFO mapred.JobClient:   Launched map tasks=9
11/05/21 05:21:04 INFO mapred.JobClient:   Data-local map tasks=2
11/05/21 05:21:04 INFO mapred.JobClient: FileSystemCounters
11/05/21 05:21:04 INFO mapred.JobClient:   FILE_BYTES_READ=1715507
11/05/21 05:21:04 INFO mapred.JobClient:   HDFS_BYTES_READ=77156154
11/05/21 05:21:04 INFO mapred.JobClient:   FILE_BYTES_WRITTEN=3431312
11/05/21 05:21:04 INFO mapred.JobClient:   HDFS_BYTES_WRITTEN=32908291
11/05/21 05:21:04 INFO mapred.JobClient: Map-Reduce Framework
11/05/21 05:21:04 INFO mapred.JobClient:   Reduce input groups=146
11/05/21 05:21:04 INFO mapred.JobClient:   Combine output records=0
11/05/21 05:21:04 INFO mapred.JobClient:   Map input records=146
11/05/21 05:21:04 INFO mapred.JobClient:   Reduce shuffle bytes=1501146
11/05/21 05:21:04 INFO mapred.JobClient:   Reduce output records=146
11/05/21 05:21:04 INFO mapred.JobClient:   Spilled Records=292
11/05/21 05:21:04 INFO mapred.JobClient:   Map output bytes=1714917
11/05/21 05:21:04 INFO mapred.JobClient:   Map input bytes=2483046
11/05/21 05:21:04 INFO mapred.JobClient:   Combine input records=0
11/05/21 05:21:04 INFO mapred.JobClient:   Map output records=146
11/05/21 05:21:04 INFO mapred.JobClient:   Reduce input records=146
The Hadoop Cluster spends 13862499 milliseconds executing this program.
```

圖 4.11 Hadoop 實驗叢集測試情境三的結果

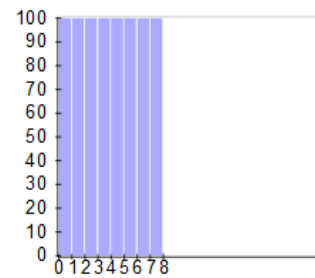
同樣地據圖 4.12 的 MapReduce 工作網頁監控圖所示，情境三的運算消耗時間大約 3 小時 51 分又 1 秒左右，另外也顯示有 8 項 Map 工作平行執行。

### Hadoop job\_201105072255\_2154 on [hadoop](#)

User: h824  
 Job Name: MapReduce\_Muskingum\_Cunge  
 Job File: [https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job\\_201105072255\\_2154/job.xml](https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job_201105072255_2154/job.xml)  
 Job Setup: [Successful](#)  
 Status: Succeeded  
 Started at: Sat May 21 01:30:02 CST 2011  
 Finished at: Sat May 21 05:21:03 CST 2011  
 Finished in: **3hrs, 51mins, 1sec**  
 Job Cleanup: [Successful](#)

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	<b>8</b>	0	0	8	0	0 / 1
reduce	100.00%	1	0	0	1	0	0 / 0

Map Completion Graph - [close](#)



	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	7
	Launched map tasks	0	0	9
	Data-local map tasks	0	0	2
FileSystemCounters	FILE_BYTES_READ	0	1,715,507	1,715,507
	HDFS_BYTES_READ	77,156,154	0	77,156,154
	FILE_BYTES_WRITTEN	1,715,805	1,715,507	3,431,312
	HDFS_BYTES_WRITTEN	31,192,627	1,715,664	32,908,291
Map-Reduce Framework	Reduce input groups	0	146	146
	Combine output records	0	0	0
	Map input records	146	0	146
	Reduce shuffle bytes	0	1,501,146	1,501,146
	Reduce output records	0	146	146
	Spilled Records	146	146	292
	Map output bytes	1,714,917	0	1,714,917
	Map input bytes	2,483,046	0	2,483,046
	Map output records	146	0	146
	Combine input records	0	0	0
	Reduce input records	0	146	146

Reduce Completion Graph - [close](#)

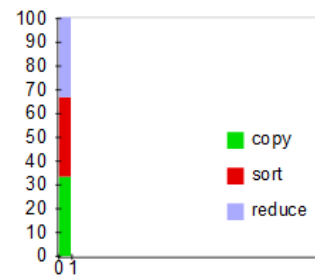


圖 4.12 Hadoop 實驗叢集測試情境三的 MapReduce 工作監控畫面





據圖 4.14 的 MapReduce 工作網頁監控畫面，顯示情境四的運算消耗時間大約 7 小時 12 分又 45 秒左右，另外也顯示有 4 項 Map 工作平行執行。

#### Hadoop job\_201105072255\_2477 on [hadoop](#)

User: h824

Job Name: MapReduce\_Muskingum\_Cunge

Job File: [hdfs://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job\\_201105072255\\_2477/job.xml](hdfs://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job_201105072255_2477/job.xml)

Job Setup: Successful

Status: Succeeded

Started at: Mon May 23 02:09:47 CST 2011

Finished at: Mon May 23 09:22:32 CST 2011

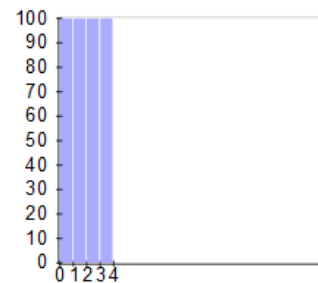
Finished in: **7hrs, 12mins, 45sec**

Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
<a href="#">map</a>	<a href="#">100.00%</a>	<b>4</b>	0	0	<a href="#">4</a>	0	0 / 0
<a href="#">reduce</a>	<a href="#">100.00%</a>	1	0	0	<a href="#">1</a>	0	0 / 0

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	4
	Launched map tasks	0	0	4
FileSystemCounters	FILE_BYTES_READ	0	1,715,507	1,715,507
	HDFS_BYTES_READ	77,156,158	0	77,156,158
	FILE_BYTES_WRITTEN	1,715,653	1,715,507	3,431,160
	HDFS_BYTES_WRITTEN	31,192,627	1,715,664	32,908,291
Map-Reduce Framework	Reduce input groups	0	146	146
	Combine output records	0	0	0
	Map input records	146	0	146
	Reduce shuffle bytes	0	1,270,520	1,270,520
	Reduce output records	0	146	146
	Spilled Records	146	146	292
	Map output bytes	1,714,917	0	1,714,917
	Map input bytes	2,483,050	0	2,483,050
	Map output records	146	0	146
	Combine input records	0	0	0
	Reduce input records	0	146	146

Map Completion Graph - [close](#)



Reduce Completion Graph - [close](#)

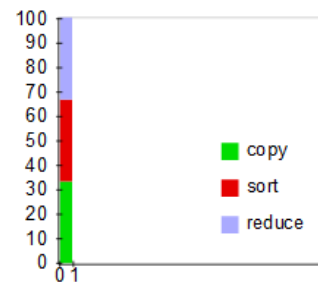


圖 4.14 Hadoop 實驗叢集測試情境四的 MapReduce 工作監控畫面

◆ 情境五：以刪減過的 146 個集水區、子程序最大記憶體使用量 2000MB 而

excess1.dat 則分為 2 個小檔於 Hadoop 實驗叢集進行 MapReduce 運算。量測

結果如圖 4.15 指令執行畫面所示，總共需花費 46797884 毫秒，明顯地運算

效能隨著 Input splits 的減少而越來越差。

```
h824@hadoop:~$ hadoop jar MRMC.jar MapReduce_Muskingum_Cunge input_incom_2parts output_incom_2parts
11/05/23 17:31:28 INFO mapred.FileInputFormat: Total input paths to process : 2
11/05/23 17:31:28 INFO mapred.JobClient: Running job: job_201105072255_2595
11/05/23 17:31:29 INFO mapred.JobClient: map 0% reduce 0%
11/05/23 17:43:42 INFO mapred.JobClient: map 1% reduce 0%
11/05/23 17:46:24 INFO mapred.JobClient: map 2% reduce 0%
11/05/23 17:54:00 INFO mapred.JobClient: map 3% reduce 0%
11/05/23 17:56:33 INFO mapred.JobClient: map 4% reduce 0%
11/05/23 18:10:19 INFO mapred.JobClient: map 5% reduce 0%
11/05/23 18:17:22 INFO mapred.JobClient: map 6% reduce 0%
11/05/23 18:24:56 INFO mapred.JobClient: map 7% reduce 0%
11/05/23 18:32:14 INFO mapred.JobClient: map 8% reduce 0%
11/05/23 18:38:11 INFO mapred.JobClient: map 9% reduce 0%
11/05/23 18:38:32 INFO mapred.JobClient: map 10% reduce 0%
11/05/23 18:48:18 INFO mapred.JobClient: map 11% reduce 0%

      .
      .
      .
      .
      .

11/05/24 06:21:30 INFO mapred.JobClient: map 98% reduce 16%
11/05/24 06:29:27 INFO mapred.JobClient: map 99% reduce 16%
11/05/24 06:31:06 INFO mapred.JobClient: map 100% reduce 16%
11/05/24 06:31:17 INFO mapred.JobClient: map 100% reduce 94%
11/05/24 06:31:23 INFO mapred.JobClient: map 100% reduce 100%
11/05/24 06:31:25 INFO mapred.JobClient: Job complete: job_201105072255_2595
11/05/24 06:31:25 INFO mapred.JobClient: Counters: 18
11/05/24 06:31:25 INFO mapred.JobClient:   Job Counters
11/05/24 06:31:25 INFO mapred.JobClient:     Launched reduce tasks=1
11/05/24 06:31:25 INFO mapred.JobClient:     Rack-local map tasks=3
11/05/24 06:31:25 INFO mapred.JobClient:     Launched map tasks=3
11/05/24 06:31:25 INFO mapred.JobClient:   FileSystemCounters
11/05/24 06:31:25 INFO mapred.JobClient:     FILE_BYTES_READ=1715507
11/05/24 06:31:25 INFO mapred.JobClient:     HDFS_BYTES_READ=77156160
11/05/24 06:31:25 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=3431084
11/05/24 06:31:25 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=32908291
11/05/24 06:31:25 INFO mapred.JobClient:   Map-Reduce Framework
11/05/24 06:31:25 INFO mapred.JobClient:     Reduce input groups=146
11/05/24 06:31:25 INFO mapred.JobClient:     Combine output records=0
11/05/24 06:31:25 INFO mapred.JobClient:     Map input records=146
11/05/24 06:31:25 INFO mapred.JobClient:     Reduce shuffle bytes=815955
11/05/24 06:31:25 INFO mapred.JobClient:     Reduce output records=146
11/05/24 06:31:25 INFO mapred.JobClient:     Spilled Records=292
11/05/24 06:31:25 INFO mapred.JobClient:     Map output bytes=1714917
11/05/24 06:31:25 INFO mapred.JobClient:     Map input bytes=2483052
11/05/24 06:31:25 INFO mapred.JobClient:     Combine input records=0
11/05/24 06:31:25 INFO mapred.JobClient:     Map output records=146
11/05/24 06:31:25 INFO mapred.JobClient:     Reduce input records=146
The Hadoop Cluster spends 46797884 milliseconds executing this program.
```

圖 4.15 Hadoop 實驗叢集測試情境五的結果



據圖 4.16 的 MapReduce 工作網頁監控畫面，顯示情境五的運算消耗時間大約 12 小時 59 分又 56 秒左右，另外也顯示有 2 項 Map 工作平行執行。

### Hadoop job\_201105072255\_2595 on [hadoop](#)

User: h824

Job Name: MapReduce\_Muskingum\_Cunge

Job File: [https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job\\_201105072255\\_2595/job.xml](https://hadoop.nchc.org.tw/var/lib/hadoop/cache/hadoop/mapred/system/job_201105072255_2595/job.xml)

Job Setup: Successful

Status: Succeeded

Started at: Mon May 23 17:31:28 CST 2011

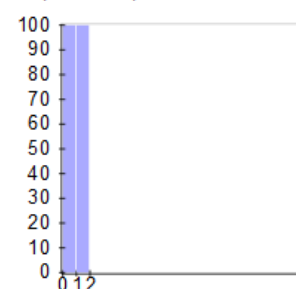
Finished at: Tue May 24 06:31:25 CST 2011

Finished in: 12hrs, 59mins, 56sec

Job Cleanup: Successful

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 1
reduce	100.00%	1	0	0	1	0	0 / 0

Map Completion Graph - [close](#)



	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	3
	Launched map tasks	0	0	3
FileSystemCounters	FILE_BYTES_READ	0	1,715,507	1,715,507
	HDFS_BYTES_READ	77,156,160	0	77,156,160
	FILE_BYTES_WRITTEN	1,715,577	1,715,507	3,431,084
	HDFS_BYTES_WRITTEN	31,192,627	1,715,664	32,908,291
Map-Reduce Framework	Reduce input groups	0	146	146
	Combine output records	0	0	0
	Map input records	146	0	146
	Reduce shuffle bytes	0	815,955	815,955
	Reduce output records	0	146	146
	Spilled Records	146	146	292
	Map output bytes	1,714,917	0	1,714,917
	Map input bytes	2,483,052	0	2,483,052
	Map output records	146	0	146
	Combine input records	0	0	0
	Reduce input records	0	146	146

Reduce Completion Graph - [close](#)

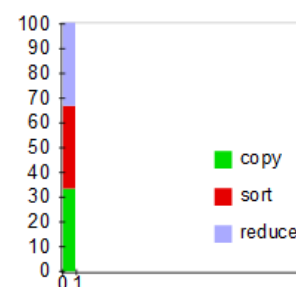


圖 4.16 Hadoop 實驗叢集測試情境五的 MapReduce 工作監控畫面

將單機運算及於 Hadoop 實驗叢集中的各種量測情境彙整成表 4.2 及表 4.3 以利進行比較，可以發現利用 MapReduce 架構及留意一些關鍵要素，在 Hadoop 實驗叢集中運算地貌型直接逕流模組的確能使得效能有相當程度的提升。但前提仍然需要視自己所設計之應用程式的功能，來考量某些硬體配備是否需要有一定的水準，如本研究首要考量的便是記憶體的需求。

另外，前一章提到最後運算的結果會將所有集水區的五項數據結果（降雨量、入流量、出流量、儲水量以及水平衡）統整彙集成 Muskingum.log，另外所有集水區的直接逕流歷線數據亦彙集成 huc\_discharge.dat，此二結果檔的格式呈現可參考附錄 2。

進行運算的集水區數 & 運算最大記憶體使用量	指令執行運算所顯示之消耗時間
155個 & 2500MB	72310861毫秒（約20小時5分又11秒）
146個 & 2000MB	62245352毫秒（約17小時17分又25秒）

表 4.2 單機計算的各種測試情境效能統整

進行運算的集水區數 & 子程序最大記憶體使用量	excess1.dat分割數量	指令執行運算 所顯示之消耗時間	MapReduce工作監控介面 所顯示之消耗時間
155個 & 2500MB	16	13926486毫秒	3小時52分又5秒
146個 & 2000MB	16	10259880毫秒	2小時50分又58秒
	8	13862499毫秒	3小時51分又1秒
	4	25966382毫秒	7小時12分又45秒
	2	46797884毫秒	12小時59分又56秒

表 4.3 Hadoop 實驗叢集計算的各種測試情境效能統整

## 第五章 結論與未來發展

### 5.1 結論

本研究以 MapReduce 架構撰寫地貌型直接逕流模組，將其置於 Hadoop 叢集系統中執行河川流徑洪水演算。盡可能地善用其軟體架構與硬體利用上的優點，並透過許多測試情境與傳統單一主機運算的效能進行比較，最後證實的確明顯減少了河川流徑洪水演算所消耗的時間。

但國家高速網路計算中心的「Hadoop 雲端運算實驗平台」有其硬體資源限制（如記憶體大小、TaskTracker 節點數量等），且該平台為開放使用者申請，有多人同時競爭硬體資源的情況，因此本研究所提供的 Hadoop 測試情境最佳效能數據尚不足以代表實際最佳之效能。若預算允許建構一具有相當規模的 Hadoop 私有雲，做為洪水資訊系統的基礎，必能使洪水資訊系統發揮最大的功用。

### 5.2 未來發展

未來「雲端」將會持續往兩個極端發展，一邊是「雲」（泛指後端服務端，如資料中心）的產業供應鏈，雲的發展是地理集中而標準化的；另一邊則是「端」（泛指前端使用者端，如行動裝置）的產業供應鏈，端的發展是位置變異而多元化的。本研究所使用的 Hadoop 開源平台及其各項子計畫目前焦點大都著重在後端「雲」的應用，不過 Hadoop 開源平台目前尚在不斷地研發創新，未來與行動手持裝置「端」的結合是必然的趨勢。藉由行動手持裝置透過連線自 Hadoop 資

料中心取得關於河川流域的即時流量資訊，將可更有效地降低洪水災害所影響的程度。

另外，雲端服務的「資料中心即電腦 (Data Center as a Computer)」的思維，使得各種資料的放置從個人電腦硬碟移至資料中心，因此雲端的隱私與安全性仍然受到不少質疑。Hadoop 雖以 Secure Shell 安全協定進行叢集中節點間的連線做為基本安全防護措施，但安全性議題多如牛毛，未來研發更新版本安全性絕對是首要的考量。

## 附錄

1. 時間範圍介於西元 2009 年 8 月 3 日 13 時 5 分至西元 2009 年 8 月 10 日 2 時 0 分所需計算之淡水河流域的 155 個集水區名稱分別如下：sm001、sm002、sm003、sm004、sm005、sm006、sm008A、sm009、sm010、sm011、sm012、sm013、sm014、sm015、sm016、sm017、sm018、sm019、sm020、sm021、sm022、sm023、sm024、sm025、sm027A、sm028、sm029、sm030、sm031、sm034A、sm035、sm036、sm037、sm038、sm039、sm040、sm042A、sm043、sm044、sm046A、sm047、sm049A、sm050、sm051、sm052、sm053、sm054、sm055、sm056、ft001、ft002、ft004A、ft005、ft006、ft007、ft008、ft009、ft010、ft013A、ft014、ft015、ft016、ft017、ft018、ft020A、ft021、ft022、kl001、kl002、kl003、kl004、kl005、kl006、kl007、kl008、kl009、kl010、kl011、kl012u、kl012d、kl013、kl014、kl015、kl016、kl017、kl019A、kl020、kl021、kl022、dh001、dh002、dh003、dh005A、dh006u、dh006d、dh007、dh008、dh009、dh010、dh011、dh012、dh013、dh014、dh015u、dh015d、dh016、dh017、dh018、dh019、dh020、dh021、dh022、dh023、dh024、dh025、sd002A、sd003、sd004、sd005、sd006、sd007、sd008、sd009、sd011A、sd012、sd013、sd014、sd015、sd016、sd017、sd018、sd020A、sd021、sd022、sd023、sd024、sd025、sd026、sd027、sd028、sd029、sd030、sd032A、sd033u、sd033d、sd034、sd035、sd037A、sd038、sd039、sd040u、sd040d、sd041、sd042、sd043。  
其中超過 16000 個內部節點的 9 個集水區分別為：sm008A、sm027A、sm034A、ft004A、dh005A、sd002A、sd011A、sd020A、sd037A。

2. Muskingum.log 的格式呈現如附錄圖 1。欄位從左至右分別為集水區名稱、降雨量（單位：立方公尺）、入流量（單位：立方公尺）、出流量（單位：立方公尺）、儲水量（單位：立方公尺）以及水平衡。

Muskingum.log ×					
dh001	4444201.39	4430913.81	4401030.55	38911.50	1.00
dh002	3696132.37	3683636.41	3645846.69	44334.81	1.00
dh003	2700903.92	2691229.26	2669996.96	26228.82	1.00
dh005A	5435100.02	5418658.93	5349578.57	78837.50	1.00
dh006d	89468.71	89220.16	89059.70	316.81	1.00
dh006u	3799362.05	3785420.01	3719531.29	74340.52	1.00
dh007	2811728.91	2802400.42	2772113.24	36990.14	1.00
dh008	1284552.11	1281961.99	1271119.08	13881.32	1.00
dh009	5235265.06	5232885.72	5233317.73	2226.25	1.00
dh010	4965289.23	4961730.16	4960650.22	3778.69	1.00
dh011	4421244.72	4415940.69	4412531.57	7175.12	1.00
dh012	4658584.22	4649017.79	4638363.56	15407.44	1.00

附錄圖 1 Muskingum.log 的格式

- huc\_discharge.dat 的格式呈現如附錄圖 2。第一欄位為集水區名稱，第二欄位之後皆表示各集水區的直接逕流歷線（Discharge Time Series）數據。

huc_discharge.dat ×						
dh001	0.931	0.931	0.931	0.931	0.930	0.929
dh002	0.804	0.804	0.804	0.803	0.801	0.800
dh003	0.598	0.598	0.598	0.597	0.596	0.595
dh005A	1.651	1.651	1.651	1.650	1.648	1.647
dh006d	0.029	0.029	0.029	0.029	0.029	0.029
dh006u	1.098	1.098	1.097	1.097	1.095	1.094
dh007	1.153	1.153	1.153	1.152	1.151	1.149
dh008	0.541	0.541	0.541	0.540	0.539	0.538
dh009	1.107	1.107	1.106	1.104	1.101	1.095
dh010	0.922	0.923	0.922	0.921	0.919	0.917
dh011	0.947	0.948	0.947	0.947	0.945	0.944
dh012	1.103	1.103	1.103	1.102	1.100	1.097

附錄圖 2 huc\_discharge.dat 的格式

## 參考文獻

- [1] Apach Hadoop. <http://hadoop.apache.org/>
- [2] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. Communications of The ACM ( Jan. 2008 ) .
- [3] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D. Stott Parker. “Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters”. ACM SIGMOD ( Jun. 2007 ) .
- [4] Tom White. “Hadoop: The Definitive Guide, 2nd edition”. O'Reilly Media, Inc ( Oct. 2010 ) .
- [5] Jason Venner. “Pro Hadoop”. Apress, Inc ( Jun. 2009 ) .
- [6] Chuck Lam. “ Hadoop in Action ”. Manning Publications Co ( Dec. 2010 ) .
- [7] Grant Mackey, Saba Sehrish, John Bent, Julio Lopez, Salman Habib, and Jun Wang. “Introducing Map-Reduce to High End Computing”. IEEE Petascale Data Storage Workshop ( Nov. 2008 ) .
- [8] Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox. “MapReduce for Data Intensive Scientific Analyses”. IEEE eScience ( Dec. 2008 ) .
- [9] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, and Russell Sears. “MapReduce Online”. EECS Department of UC Berkeley. Technical Report No. UCB/EECS-2009-136 ( Oct. 2009 ) .
- [10] Feng Wang, Jie Qiu, Jie Yang, Bo Dong, Xinhui Li, and Ying Li. “Hadoop High availability through Metadata Replication”. ACM CloudDB ( Nov. 2009 ) .

- [11] Wei Jiang, Vignesh T. Ravi, and Gagan Agrawal. “Comparing Map-Reduce and FREERIDE for Data-Intensive Applications”. IEEE Cluster Computing and Workshops ( 2009 ) .
- [12] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. “The Google File System”. ACM Symposium on Operating Systems Principles ( Oct. 2003 ) .
- [13] Hadoop Taiwan User Group. <http://www.hadoop.tw/>
- [14] NIST Cloud Computing Program. <http://www.nist.gov/itl/cloud/>