

東海大學

資訊工程研究所

碩士論文

指導教授：楊朝棟博士

基於 Hadoop 與 HBase 平台之 LCD
Flicker 雲端儲存系統之實作

The Implementation of Cloud Storage System for LCD

Flicker based on Hadoop and HBase Platform

研究生：謝岳霖

中華民國一〇四年一月

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 謝 岳 霖 所提之論文

基於 HBase 平台之 LCD Flicker 雲端

儲存系統之實作

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

楊武

簽章

委

員

許慶賢

呂芳如

賴廷州

指 導 教 授

楊朝棟

簽章

中華民國 104 年 1 月 19 日

摘要

本論文，提出一個 LCD Flicker 雲端儲存系統，工廠可以上傳 LCD Flicker 測試結果到 HBase 資料庫，然後即時產生反饋讓工程師來控制產線變異。LCD Flicker 雲端儲存系統的目的是為了克服當前使用系統的缺點，它沒有能力來保存從生產線收集的數據。因為測量數據只儲存在測試設備中，無法回報給工程師，發生錯誤時，這個系統不可能用來分析和觀察變異的趨勢。因此，LCD Flicker 雲端儲存系統具有線上即時監控量測數據的關鍵作用。此系統為私有雲 (Private Cloud) 架構，主要使用於工廠與工廠之間的資料共享，並使用 LCD Flicker 資料來做不良率分析。本文實驗使用虛擬化平台搭配 Hadoop 與 HBase，來做 flicker 資料的傳送與訪問並分析系統效能，並藉此提升整體產品品質與效能。

關鍵字：私有雲、虛擬化、Hadoop、HBase、Flicker

Abstract

In this thesis, a LCD Flicker cloud storage system is proposed. This data storage system is based on Hadoop and Hbase architecture, which factory could upload measurement and test results of LCD flicker into HBase, then generate online and real-time feedback for engineers to control those variations. The purpose of LCD Flicker cloud storage system is to overcome the disadvantages of currently used system, which has no capability to save data collected from production lines. Because that measured data are only store locally within testing equipment and unable to report to the engineer, when errors occurred, it is impossible for system to analyze and observe the trend of variations. Therefore, LCD Flicker cloud storage system has a critical role in order to monitor measured data online and real-time. This system is build by private cloud architecture, mainly used between the factory and the factory data sharing, and use the LCD Flicker data to perform failure rate analysis. In this thesis, the experiments are conducted by using virtualization platform with Hadoop and HBase. We performe flicker data transferring and accessing, and analyzing the performance, also enhance the overall product quality and performance.

Keyword : Private Cloud 、 Virtualization 、 Hadoop 、 HBase 、 Flicker.

致謝詞

對於出社會後邁向工作第十一年的我，能夠修習碩士學位一直是我在出社會後想完成的目標之一，修畢二技學位後心中還是渴望能夠在唸個研究所，若能夠拿到碩士學位對我來說更是無比的興奮，學生時代半工半讀的完成二技學位，但沒有好好把該學習的技能學好，直到考上研究所後，才真正的努力學習並且實際應用在工作上。

在忙碌的工作環境中，要專心唸書真的辛苦的多，除了讀書之外，要工作及照顧家庭，時常是蠟燭多頭燒的狀況，現在這些事情對我來說都是個人來說都是一份責任與義務，只能努力完成。

十年的工作經驗，但是都專注在面板產業中，接觸的領域屬於 LCD 螢幕的設計與開發，寫的程式都是 C 語言與 FPGA，所以對於研究的雲端運算領域說實在話是個門外漢，學習起來也格外辛苦，雖然都跟電腦有關，但卻是不同的領域。

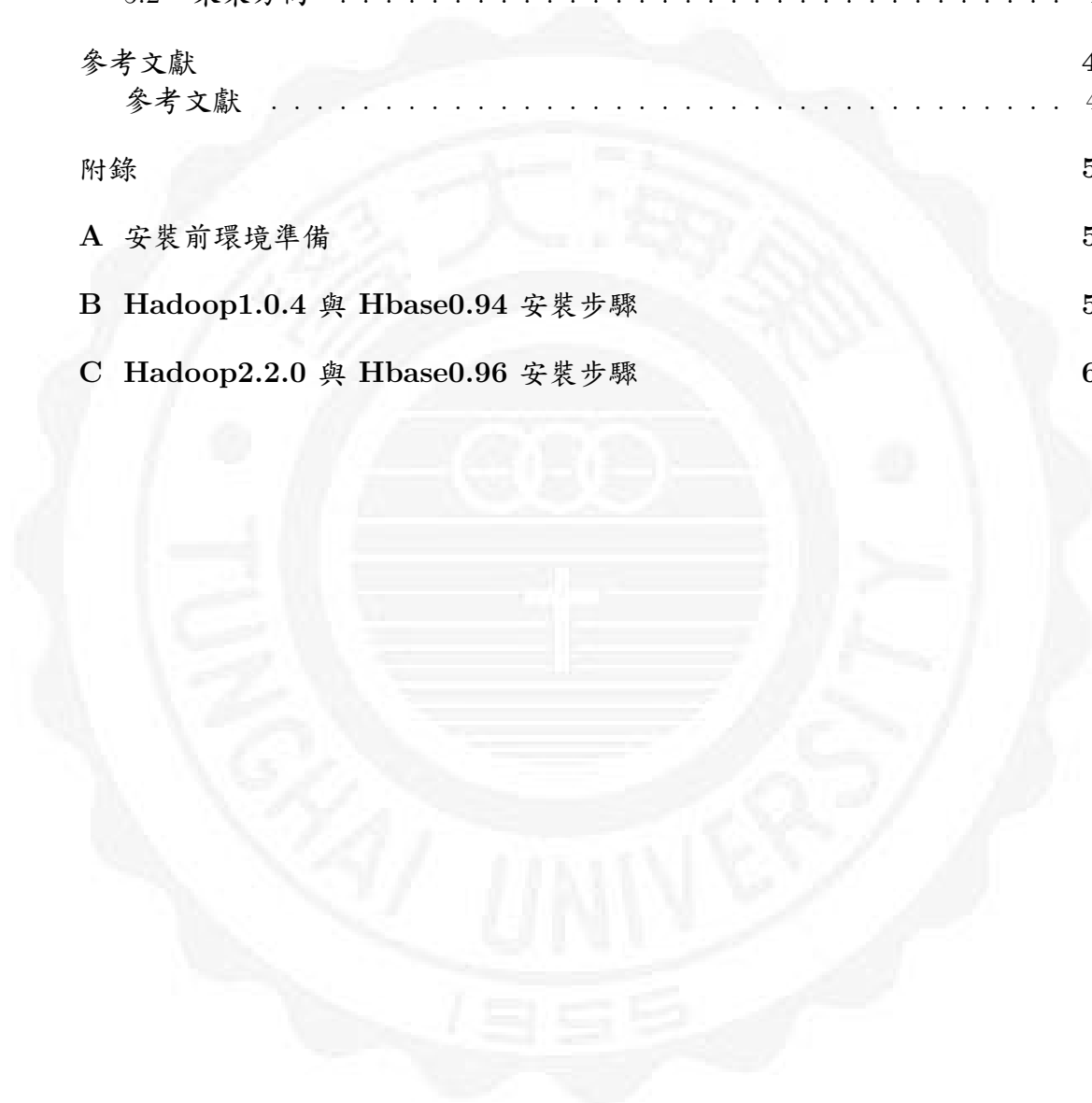
在求學期間，非常感謝我的指導老師楊朝棟教授，教授指引我研究的方向，教導我雲端運算領域的相關知識，這些知識剛好能夠讓我學以致用，將雲端知識應用在我的工作中，建立 LCD 儲存系統並可以將工作的數據有效儲存與分析，在大數據的時代讓我可以將所學運用到工作中，使得念研究所更有價值。

另外就是實驗室一起學習與成長的學長與同學們，除了在知識與技術上的相互支援外，大家也都變成了好朋友，也成為我念碩士時的另一種收穫。

Table of Contents

摘要	I
Abstract	II
致謝詞	III
Table of Contents	IV
List of Figures	VI
List of Tables	VIII
1 簡介	1
1.1 研究背景	1
1.2 研究動機	2
1.3 論文架構	3
2 研究背景	4
2.1 大數據 (Big Data)	4
2.2 虛擬化 (Virtualization)	6
2.3 Hadoop	6
2.4 Hadoop1.X 架構	8
2.5 Hadoop 2.X 架構	10
2.6 HBase	13
2.7 ZooKeeper	14
2.8 Flicker	16
3 系統設計與實作	17
3.1 系統架構	17
3.2 系統安裝	22
4 實驗環境與結果	26
4.1 實驗環境	26
4.2 實驗架構	30
4.3 實驗方法	32
4.4 實驗結果	34

5 結論與未來方向	46
5.1 結論	46
5.2 未來方向	47
參考文獻	48
參考文獻	48
附錄	52
A 安裝前環境準備	52
B Hadoop1.0.4 與 Hbase0.94 安裝步驟	54
C Hadoop2.2.0 與 Hbase0.96 安裝步驟	63



List of Figures

2.1	大數據 3V 原則圖	5
2.2	虛擬化的架構圖	6
2.3	Hadoop 組成模組	7
2.4	Hadoop1.X 與 Hadoop2.X 版本差異圖	8
2.5	Hadoop 的 HDFS 架構圖	9
2.6	Hadoop1.x MapReduce 架構圖	10
2.7	Hadoop2.x 架構圖	11
2.8	Hadoop2.X YARN 架構圖	12
2.9	Hadoop 與 HBase 層次圖	13
2.10	Zookeeper 數據結構	15
2.11	LCD Flicker 微笑曲線圖	16
3.1	系統架構圖	18
3.2	HBase 寫入工作階段圖	19
3.3	HStore 之 Compaction 和 Split 的過程圖	21
3.4	Flicker 數據寫入到 HBase 流程圖	22
3.5	Hadoop JPS 結果	23
3.6	Hadoop 與 Hbase JPS 結果	24
3.7	Hadoop 網頁	24
3.8	HBase 網頁	25
4.1	VMware Workstation 之實驗系統環境圖	27
4.2	一台 Host Server 之實驗系統環境圖	27
4.3	三台 Host Servers 之實驗系統環境圖	28
4.4	VMware Workstation 與 VMware vSphere 實驗系統架構圖	30
4.5	連線環境示意圖	31
4.6	NB 環境無 Buffer 之比較圖	34
4.7	三種實驗環境中無 Buffer 之比較圖	35
4.8	NB 實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 比較圖	36
4.9	三種實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 比較圖	37
4.10	開啟 Buffer 之 10 萬筆 Flicker Data 比較圖	38
4.11	三種實驗環境開啟 Buffer 之 10 萬筆 Flicker Data 比較圖	39
4.12	開啟 Buffer 之十五萬筆 Flicker Data 比較圖	40
4.13	三種實驗環境開啟 Buffer 之 15 萬筆 Flicker Data 比較圖	41
4.14	開啟 Buffer 之 20 萬筆 Flicker Data 直條圖	42
4.15	三種實驗環境開啟 Buffer 之 20 萬筆 Flicker Data 比較圖	43

4.16 NB 環境下開啟 Buffer 各筆數 Flicker Data 比較圖 44
4.17 三個實驗環境下開啟 Buffer 各筆數 Flicker Data 比較圖 45



List of Tables

2.1	Hadoop version support matrix	14
4.1	VMware Workstation 與 VMware vSphere 系統配置表	29
4.2	Client 端硬體規格表	29
4.3	Hadoop Name Node 虛擬機硬體規格表	29
4.4	Hadoop Data Node 虛擬機硬體規格表	29
4.5	Flicker 資料內容表	32
4.6	NB 環境無 Buffer 之效能差異表	34
4.7	三種實驗環境中無 Buffer 之效能差異表	35
4.8	NB 實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 時間表	36
4.9	三種實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 時間表	36
4.10	開啟 Buffer 之 10 萬筆 Flicker Data 時間表	37
4.11	三種實驗環境之開啟 Buffer 之 10 萬筆 Flicker Data 時間表	38
4.12	開啟 Buffer 之十五萬筆 Flicker Data 時間表	39
4.13	三種實驗環境開啟 Buffer 之 15 萬筆 Flicker Data 時間表	40
4.14	開啟 Buffer 之 20 萬筆 Flicker Data 時間表	41
4.15	三種實驗環境之開啟 Buffer 之 20 萬筆 Flicker Data 時間表	42

Chapter 1

簡介

1.1 研究背景

雲端運算 [31] 是現今作熱門的議題，不管是軟體或是硬體廠商均開始將產品逐漸移轉到雲端，發展雲端相關產品。其中屬於儲存領域的「雲端儲存」[19]，便是雲端應用中重要的一項服務。雲端儲存 (Cloud storage)，簡單來說，就是將資源或是資料放到網路上供人存取的一種模式。資料在雲端上，使用者可以在任何時間、任何地方，透過任何可上網的裝置快速容易地存取資料。但是現有的雲端儲存服務對於資料安全性是否能夠確保資料的安全無虞，在公有雲的雲端環境中資料安全性其實是有很大的風險，因為資料都是交由提供雲端服務之廠商儲存如果遇到不肖的雲端廠商或是駭客，都有可能發生資料外洩的情況。另外在私有雲的環境中除了安全性較公有雲安全外，但在設備的購買與安裝、設定與維護上，會對企業來說是一項不小的負擔，畢竟設備都需要定期維護，才能確保在存取資料中不會發生異常狀況。在現今的雲端服務爆發的時代，雲端儲存的方案處處可見，但對於中小型公司或是大企業要儲存一些特殊的資料數據的雲端儲存方案就少之又少。

為了能夠減少企業購買硬體與軟體的負擔，並且還需要將資源最大化利用，虛擬化技術 [20, 21] 就顯得非常重要，在等級較好的伺服器或是電腦上可以開啟多部虛擬機器，也就是說以一台電腦的耗電量，等同於開啟多台電腦的耗電量，

故虛擬化技術在節能省電上以及減少碳排放是效果是驚人的，另外在公司 IT 資源的使用成本也可以大大的降低。另外虛擬化技術允許多作業系統與相關應用軟體可同時運行在單一實體機器中的特性，可協助企業加速完成雲端基礎架構即服務 (Infrastructure as a Service, IaaS) 方案的部署。所謂 IaaS 資源大致是指儲存、網路與運算等三種資源而言，使用者可針對特定屬性的虛擬機器，指定搭配不同用量的資源配置，如果主要是使用儲存資料使用，則可以將電腦硬碟加大來做為雲端儲存系統使用。

在軟體方面，如果使用現有 IT 廠商或是提供雲端服務之公司所開發的套裝雲端系統，除了收費貴之外，每年還是需要支付一筆維護費用給服務提供廠商，若能夠使用 Open Source 軟體來建置雲端儲存平台，將可以有效控制與降低購買與建置費用，並可以根據企業本身之特殊儲存需求來做雲端儲存平台的規畫，來滿足企業需求。

1.2 研究動機

隨著網際網路服務快速發展與行動裝置規格的提升，可以隨時隨地的連線上網，網際網路服務與雲端服務已經成為企業界的主流，雲端運算 (Cloud Computing) 的概念也順應而生；在雲端運算快速發展下，政府機關、民間企業紛紛開始使用與建置雲端服務，例如：公有雲 (Public Cloud)，私有雲 (Private Cloud) 以及混和雲 (Hybrid Cloud)，在面板產業中會以使用私有雲 (Private Cloud) 為主。

在企業內部單位與單位之間經常有資料無法共享之問題，無法立即了解生產中的產品是否有不良率太高問題，或是資料量累積的數量不夠多，造成無法長期觀察產品數據之不良成因，也因為工廠產線無法架設大型資料系統來收集所有產線生產的 LCD Flicker 數據，故需要建置一個雲端儲存系統來解決上述問題，所以在軟體方面利用開放原始碼軟體來建置雲端儲存系統，在硬體系統建置方面，以工廠內現有的電腦為主要硬體設備，畢竟面板產業目前是非常競爭的產業，對於控制預算方面較為嚴格把關，故本實驗有二個實驗限制：

1、沿用原工廠電腦主機：主要是不增加公司購買硬體負擔。

2、使用 Open Source 軟體建構平台：使用 Open Source 即可不增加軟體購買費用。

本研究使用 Hadoop 與 HBase 來建置 LCD Flicker 雲端資料系統，並架構在虛擬機環境中，作業系統 Ubuntu 12.04.01 Desktop 為主，並在此作業系統安裝 Hadoop 再搭配 HBase 做為雲端儲存系統主要架構，此架構包含一個 Name Node 與一個 Data Node，來分析傳送與寫入效能，來提供企業應用平台的最佳方案選擇。

1.3 論文架構

本論文主要在介紹雲端儲存系統的效能評估，並運用底層虛擬化技術與上層 Hadoop 檔案系統與 HBase 資料庫之跨層次效能評估。第一章針對雲端運算及雲端儲存作介紹，並說明研究動機。第二章說明本論文的研究背景並介紹目前 Open Source 所提供之雲端運算相關技術。第三章描述系統的建置方法及實驗方法。第四章說明實際實驗的結果，並依照實驗的結果說明 Hadoop 新版與舊版兩個系統的優劣。第五章說明本論文依照實驗結果做出結論及未來的研究方向。

Chapter 2

研究背景

2.1 大數據 (Big Data)

大數據 (Big Data) [1,24] 又稱巨量資料或是海量資料，這是近年來與雲端運算一樣熱門的話題，不管是個人或是企業均能會直接使用到或是間接觸到 Big Data 的應用，尤其是雲端與大數據分析這一塊，包括氣象學、基因學、類神經網路、即時交通路況，以及生物和環境研究等等。大數據分析無時無刻在生活周遭，包含最近的伊波拉病毒疫情，電信公司提供通話數據，運用 Big Data 繪製移動地圖，追蹤疫情擴散地區。所以大量的數據資料裡，將資料作分析，就可以提出一些有跡可尋的參考指標。

Big Data 有三種特性: Volume、Velocity、Variety) [26]，如下圖所示

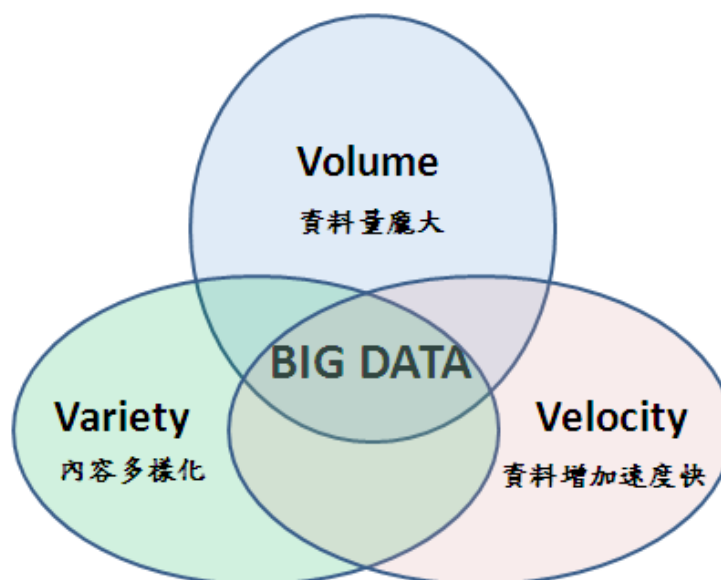


FIGURE 2.1: 大數據 3V 原則圖

- 資料量 (Volume)：單日資料量以數十、數百 TB 的速度增加，而總資料量也達到了 PB (Petabyte) 等級。
- 處理速度 (Velocity)：資料量越來越大，資料的處理速度也必須越來越快，諸如 Mobile Computing、臉書與社交網路的風行，使得資料增加的速度比傳統的企業應用程式來得快很多，一旦資料增生速度越快，資料處理、分析的速度也就得跟上，從批次、即時到串流，線上廣告與銀行授信系統均需要反應時間非常短的狀況下決定回應內容或是計算。
- 多樣性內容 (Variety)：指資料的種類多樣性，現在網際網路不是單單只看看資訊，同時我們不斷在產出資料，例如：上傳照片、上傳影片、寫網路文章等，而另一方面，資訊 IT 設備深入生活中的各個層面，各式各樣的監控器、感應器也不停地產出機器資訊，資料的型式已不像過去那麼單純，資料的樣式包括結構化、非結構化與半結構化，以及三種型式的組合。

前面所述的 3V 的綜合需求，現今資料的複雜性該如解決資料處理問題則是
大數據的一大挑戰。

2.2 虛擬化 (Virtualization)

虛擬化 [2,11] 是一個很廣義的技術概念，可以是指個別裝置、伺服器、作業系統、應用程式，或者是網路的虛擬化。以雲端運算為基礎架構並平台化廠商很多，包括：VMware、Microsoft Hyper-V、Citrix 以及 RHEV，可以依據需求去調整虛擬化的應用並調配雲端虛擬資源服務速度、彈性與靈活性。而本實驗利用虛擬化技術建置雲端儲存平台如圖 Figure 2.2所示：

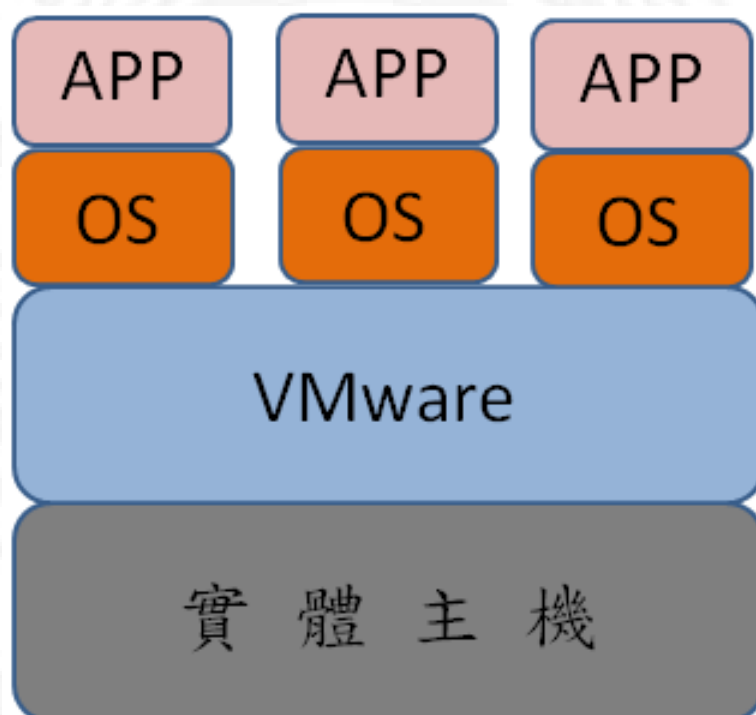


FIGURE 2.2: 虛擬化的架構圖

2.3 Hadoop

Hadoop [3,22] 是 Apache 軟體基金會底下的計畫之一，它是屬於一個開放原始碼計畫。主要是用來處理大量資料的分散式檔案系統，它是利用 JAVA 下去做開發，而主要的概念是來自於 Google 在 2006 年所發表的 Big Table 跟 Google File System 文章概念所建立而成。

Apache Hadoop 專案是提供開放源程式碼的軟體，來建立穩定可擴充的分散式運算平台，它可從單一伺服器擴展到數千台伺服器，提供本地運算和存儲服務。如圖 Figure 2.3 Hadoop 組成模組所示，Hadoop 包含 Common、HDFS、MapReduce 三種主要項目，在 Hadoop2.0 版本之後有了 YARN，讓各種處理與分析的方法或工具可以與 Map/Reduce 平等運作，都能讀取與寫入 HDFS。另外 Apache 的 Hadoop 的相關子項目包括: Core、Avro、Pig、ZooKeeper、Hive、Chukwa。

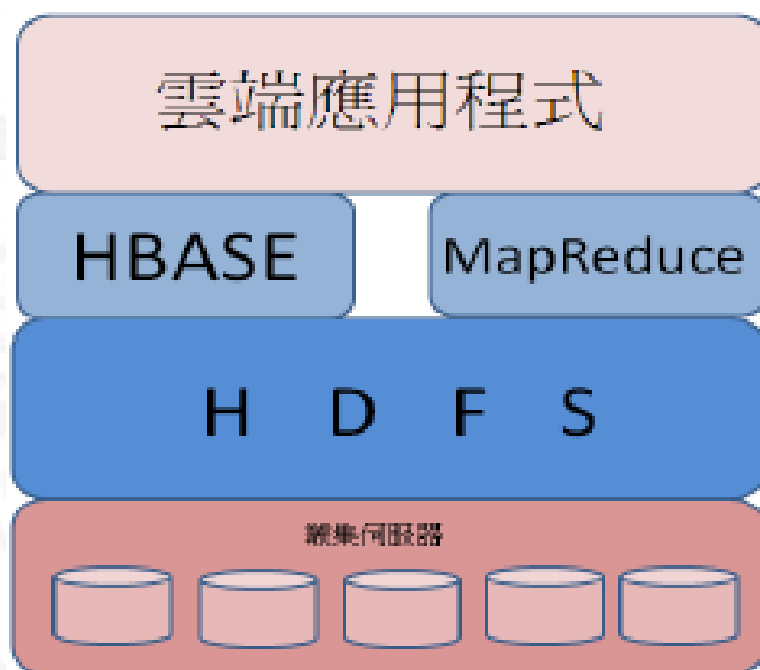


FIGURE 2.3: Hadoop 組成模組

Hadoop2.X 相較於 Hadoop1.X 來說，HDFS 的架構與 MapReduce 架構上有顯著的變化，如圖 Figure 2.4 HADOOP1.X 與 HADOOP2.X 版本差異圖 [28] 在 Hadoop2.X 中有兩個重要的改變：

- HDFS 的 NameNodes 可以以群集的方式佈署，增強了 NameNodes 的水平擴展能力和可用性。
- MapReduce 將 JobTracker 中的資源管理及任務生命週期管理，拆分成兩個獨立的模組，並更名為 YARN (Yet Another Resource Negotiator) [27]。

MrappMaster 是 MapReduce [4] 的 ApplicationMaster 實現，它使得 MapReduce 計算框架可以運行於 YARN 之上。在 YARN 中，MrappMaster 負責管理 MapReduce 作業的生命週期，包括創建 MapReduce 作業，向 ResourceManager 申請資源，與 NodeManager 通信要求其啟動 Container，監控作業的運行狀態，當任務失敗時重新啟動任務等。

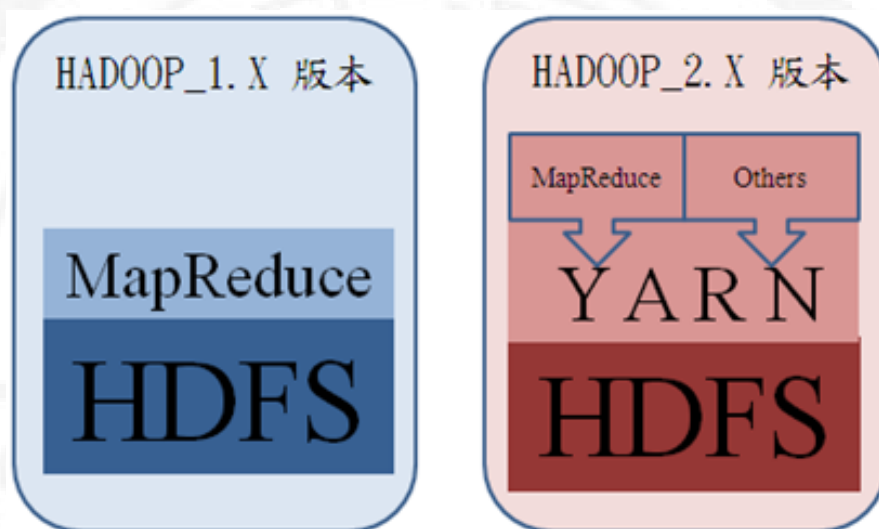


FIGURE 2.4: Hadoop1.X 與 Hadoop2.X 版本差異圖

2.4 Hadoop1.X 架構

如圖 Figure 2.5，在 Hadoop1.X [15] 的架構中主要區分為 Namenode 與 Datanodes 兩部分，通常 Hadoop1.X 只有一個 Namenode 與多個 Datanodes 的配置，NameNode 它負責管理 HDFS 和相關檔的中繼資料資訊，DataNode 負責實際的資料存儲，並將資訊定期傳輸給 NameNode 用戶端寫入資料到 HDFS 時，系統會將要儲存的檔案切割成固定大小的區塊，再將區塊資料儲存到檔案系統內。

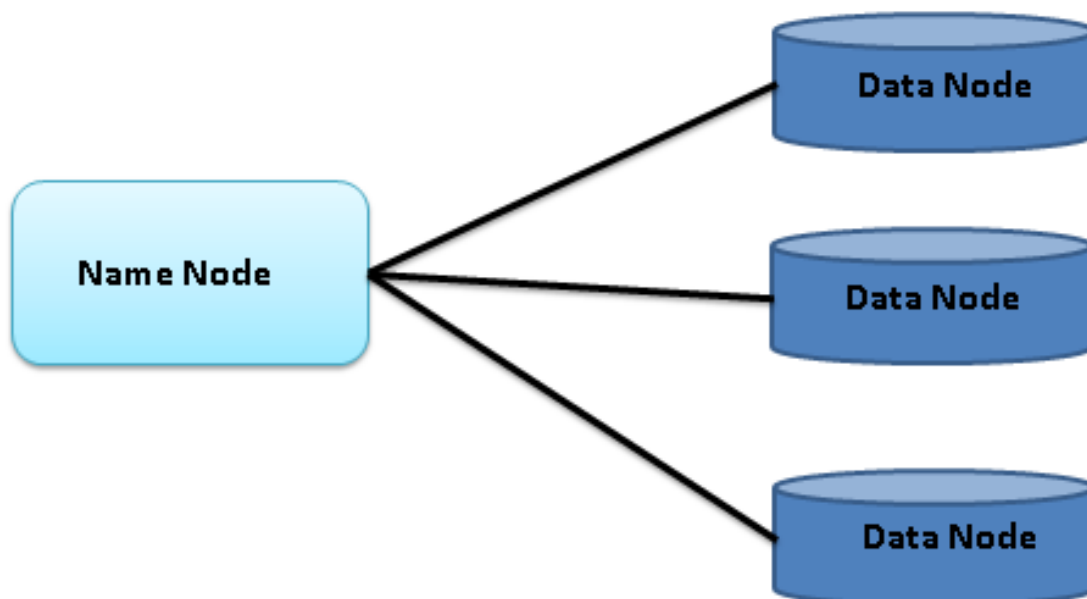


FIGURE 2.5: Hadoop 的 HDFS 架構圖

如圖 Figure 2.6，Job Tracker 是 MapReduce 架構的中心，首先 JobClient 提交了一個 Job，Job 的資訊會發送到 Job Tracker 中，Job Tracker 需要與集群中的機器定時傳遞訊息，需要管理任務應該在哪個機器上執行，並且需要管理所有 Job 失敗、重啟等操作，在 MapReduce 集群中每台機器都有 TaskTracker，主要在監視機器的資源情況與 tasks 運行狀況，而 TaskTracker 需要把這些資訊通過訊息的溝通傳送給 JobTracker，JobTracker 會搜集這些訊息，當新的 Job 進來時，分配在適合的機器上運行。

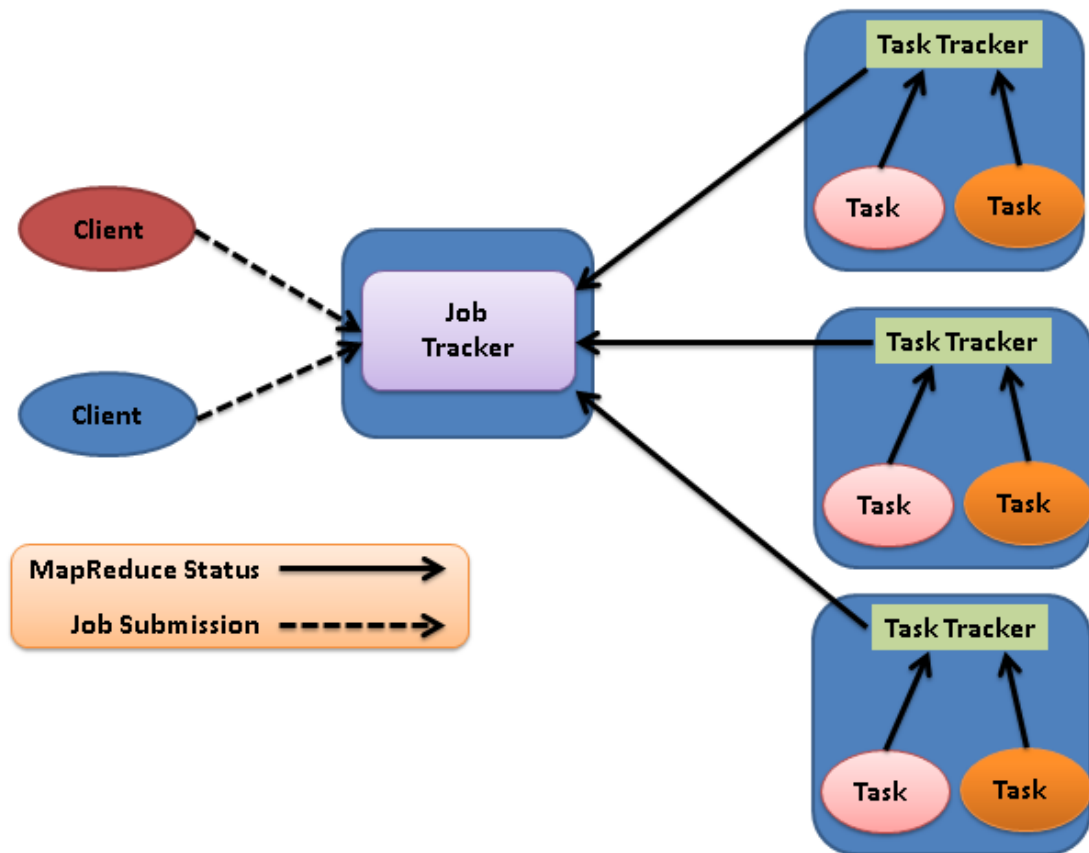


FIGURE 2.6: Hadoop1.x MapReduce 架構圖

2.5 Hadoop 2.X 架構

從分散式系統的變化趨勢和 Hadoop 架構的長遠發展來看，MapReduce 的 JobTracker/TaskTracker 機制需要大規模的調整來修復它在可擴展性，電腦記憶體消耗，可靠性等。如圖 Figure 2.7，在 Hadoop2.X 的架構中 Namenode 分為 Active 與 Standby 與 Datanodes，Hadoop2.X 為了解決 hadoop1.x 版本的一些問題如：namenode 單點故障問題與 namenode 記憶體壓力過大難以擴展問題。

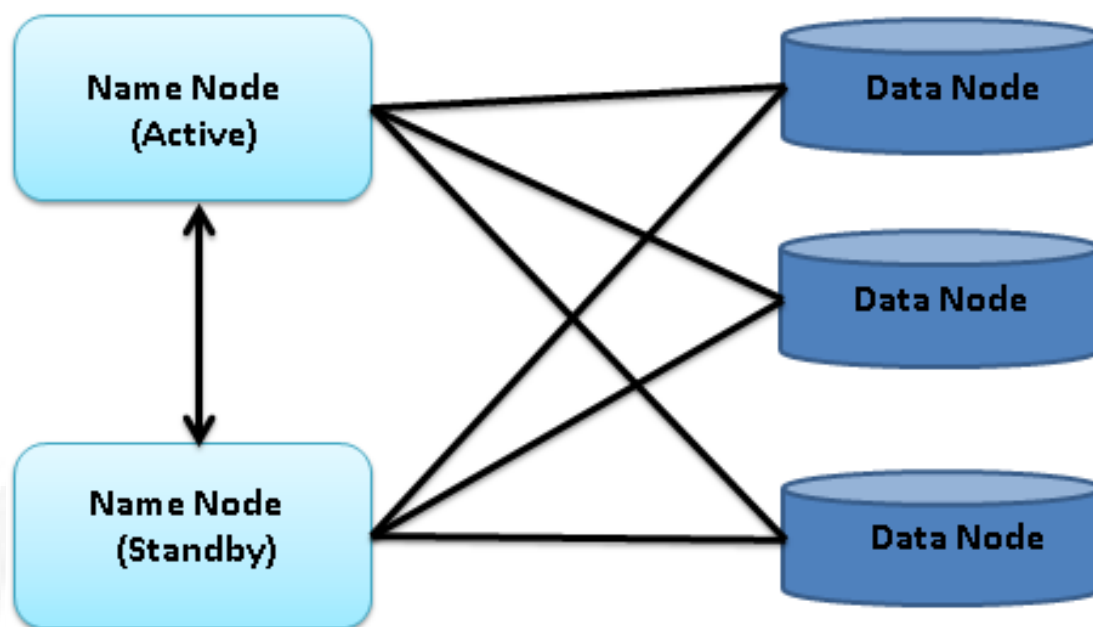


FIGURE 2.7: Hadoop2.x 架構圖

為從根本上解決舊 MapReduce 架構上的瓶頸，也必須讓 Hadoop 架構有更長遠發展，從 Hadoop2.x 版本開始，Hadoop 的 MapReduce 架構完全重新架構，將 ResourceManager 從 MapReduce 獨立出來，讓 MapReduce 單純架構在 YARN 上面的 Batch Job Computing Framwork，新的 Hadoop MapReduce 框架命名為 MapReduceV2 或者叫 YARN [12]。

如圖 Figure 2.8 ResourceManager 跟 Node Manager 主要負責協調資源的調度，ApplicationMaster 負責與 ResourceManager 協調資源並跟 Node Manager 合作執行 Container，監督 Container 和資源消耗狀況，另外也負責 Schedule 中的各個 Task。

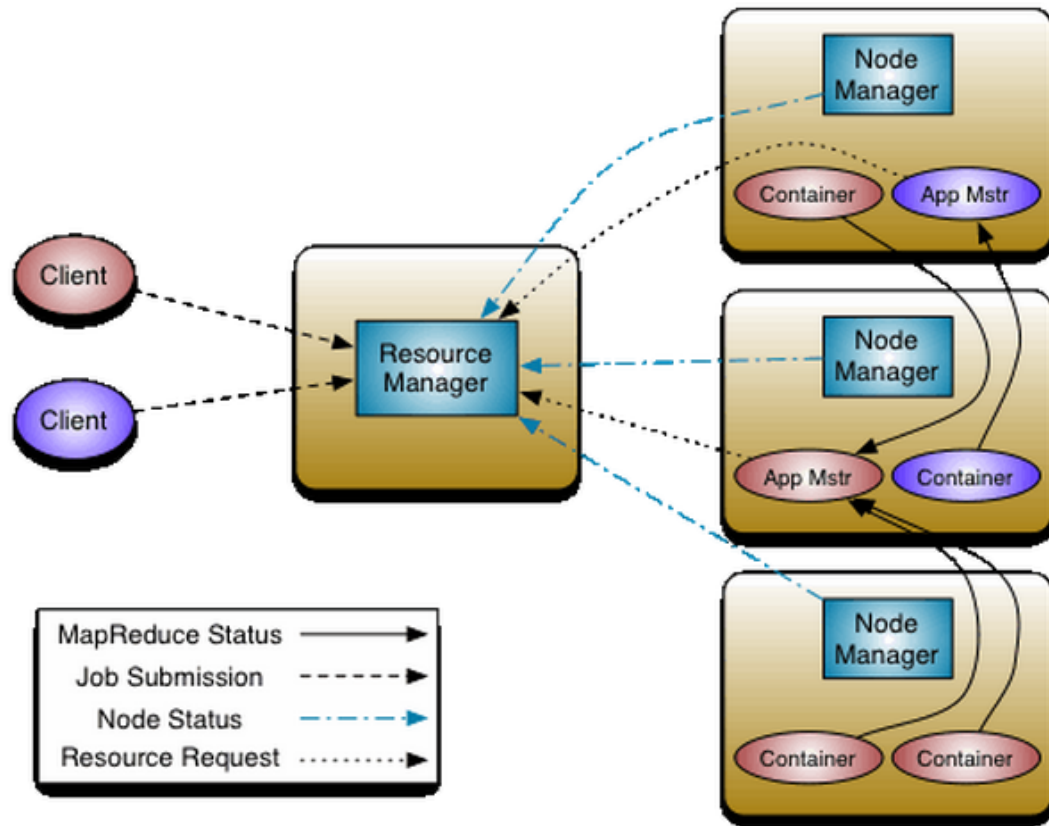


FIGURE 2.8: Hadoop2.X YARN 架構圖

YARN 架構相對於 MapReduce 架構優勢如下：

- 減小了 JobTracker (ResourceManager) 的資源消耗。
- 舊有架構中，JobTracker 有一個很大的負擔就是監控 Job 下的 tasks 的運行狀況，現在這個部分交由 ApplicationMaster 來處理。
- Container 是 YARN 為了將資源隔離而提出的一個架構，後續能支持更多的資源調度和控制，既然資源表示成記憶體量，那就沒有了之前的 map slot/reduce slot 分開造成集群資源閒置的情況發生。
- 對於資源的表示以記憶體為單位。
- Yarn 架構中，ApplicationMaster 是一個可變更的部分，使用者可以對不同的程式設計模型。

2.6 HBase

HBase [16, 17] 是一個分散式開源資料庫，基於 Hadoop 分散式文件系統，模仿並提供了基於 Google 文件系統的 Bigtable 資料庫的所有功能。Hbaes 的目標是處理非常龐大的表，可以用普通的電腦處理超過 10 億行資料，並且有數百萬列元素組成的資料表。Hbase 可以直接使用本地文件系統或者 Hadoop 作為資料存儲方式，不過為了提高資料可靠性和系統的健壯性，發揮 HBase 處理大資料量等功能，需要使用 Hadoop 作為文件系統。

HBase [23] 是 Apache Hadoop 的資料庫，能夠對大量資料提供隨機、即時的讀與寫存取，目前已經是 Apache 眾多開放原始碼項目中的一個頂級專案。如圖 Figure 2.9 HBase 是架構在 HDFS 上的分散式資料庫，與一般關聯式資料庫 (relational database) 不同。HBase 使用列 (row) 與行 (column) 為索引儲存資料值，因此查詢的時候比較像在使用 map 容器 (container)；Hbase 的另一個特點是每一筆資料都有一個時間戳記 (timestamp)，因此同一欄位可依不同時間存在多筆資料。

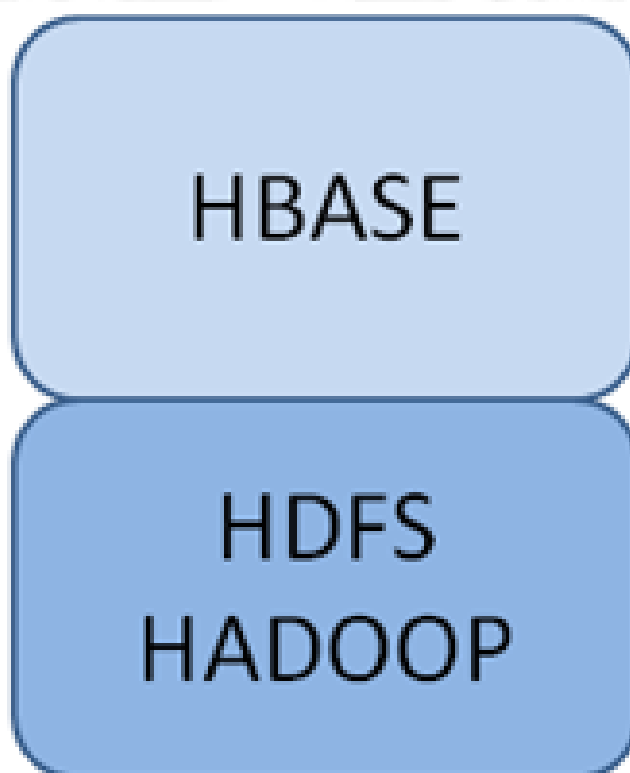


FIGURE 2.9: Hadoop 與 HBase 層次圖

HBase 是建構在 Hadoop 上來運作，也就是說必須先有 Hadoop 才能建置 HBase，另外 Hadoop 各版本對應 HBase 各版本之搭配表，如表 Table 2.1 [30] 來解釋：

- S = 支持和測試，
- X = 不支援，
- NT = 它應該運行，但未經測試不夠
- d= 為了能夠在 HBase-0.94.X 到 Hadoop-2.2.0 上運行，需要改變的 Hadoop-2.2.0 裡面的 protobuf 的版本。

TABLE 2.1: Hadoop version support matrix

版本	Hbase094.X	Hbase0.96.X
Hadoop-1.0.3+	S	S
Hadoop-1.1.X	S	S
Hadoop-2.0.X-alpha	NT	X
Hadoop-2.1.0-alpha	NT	S
Hadoop-2.2.0	NT[d]	S
Hadoop-2.3.0	NT	S
Hadoop-2.4.0	NT	S
Hadoop-2.5.0	NT	S

2.7 ZooKeeper

Zookeeper [29] 分散式服務框架是 Apache Hadoop 的一個子專案，它主要是用來解決分散式應用中經常遇到的一些資料管理問題，如：統一命名服務、狀態同步服務、集群管理、分散式應用配置項的管理等。Zookeeper 是一個開放原始碼，具有高效能和高可靠的協同工作系統，在一個分佈式的環境中，我們需要一個 Master 實例或存儲一些配置信息，確保文件寫入的一致性，在分散式的檔案系統中，假如硬體損壞時，必須要有一個機制來處理當硬體出現問題時，要如何配置回相同的資料。Hadoop 使用 Zookeeper 的事件處理確保整個集群只有一個 NameNode，存儲配置信息，而 HBase 使用 Zookeeper 的事件處理確

保整個集群只有一個 HMaster, 察覺 HRegionServer 聯機和宕機, 存儲訪問控制列表等。 Zookeeper 是一個具有層次關係的資料結構, 它非常類似於一個標準的檔案系統, 如 Figure 2.10 所示:

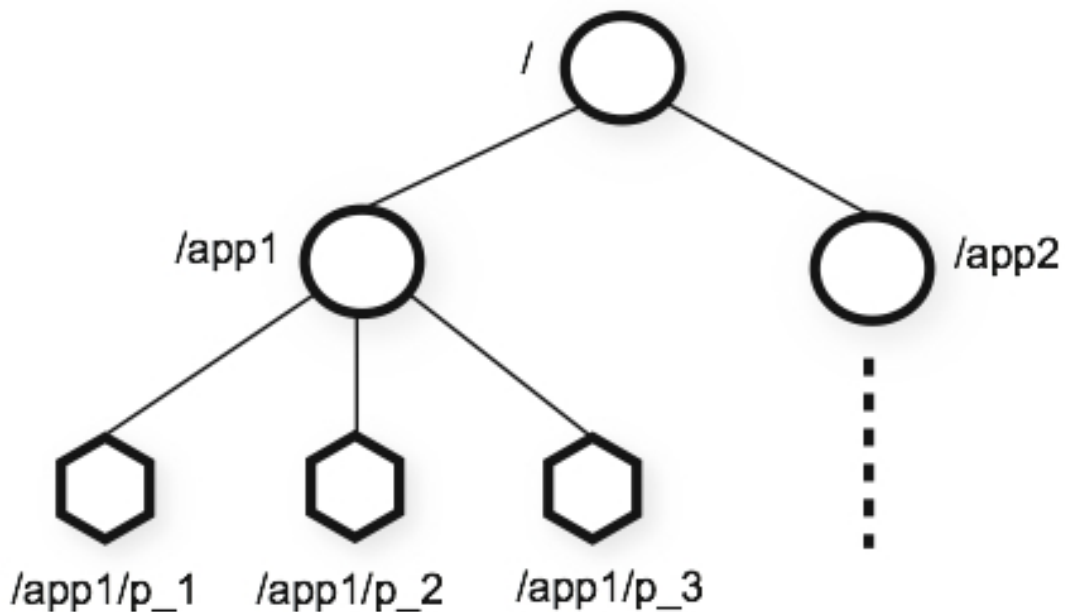


FIGURE 2.10: Zookeeper 數據結構

2.8 Flicker

Flicker [5] 簡單來說，即液晶螢幕畫面閃爍現象，LCD 在更新頻率 60hz 下，人眼目視可以直接觀察到 LCD Flicker 現象，則代表 LCD Flicker 是沒有調整到 V-COM 電壓值最小值，即不閃爍的狀況。LCD Flicker 目前均使用光學 Sensor 來做量測，Sensor 是將光轉換成相對應之電壓值，再根據電壓起伏高低，作為閃爍程度的比對，當 Flicker 閃爍程度高時，相對於電壓起伏大，Flicker 閃爍程度小時，則電壓起伏小。

目前中小尺寸 LCD Flicker 調整方式，目前均使用調整 LCD V-COM 電壓值 [6]，V-COM 電壓值是一個區間範圍可做電壓調整，此區間內一定會有一組 Flicker 最小值，Flicker 最小值呈現微笑曲線，如圖 Figure 2.11 範例所顯示 8 片 LCD Flicker 微笑曲線。

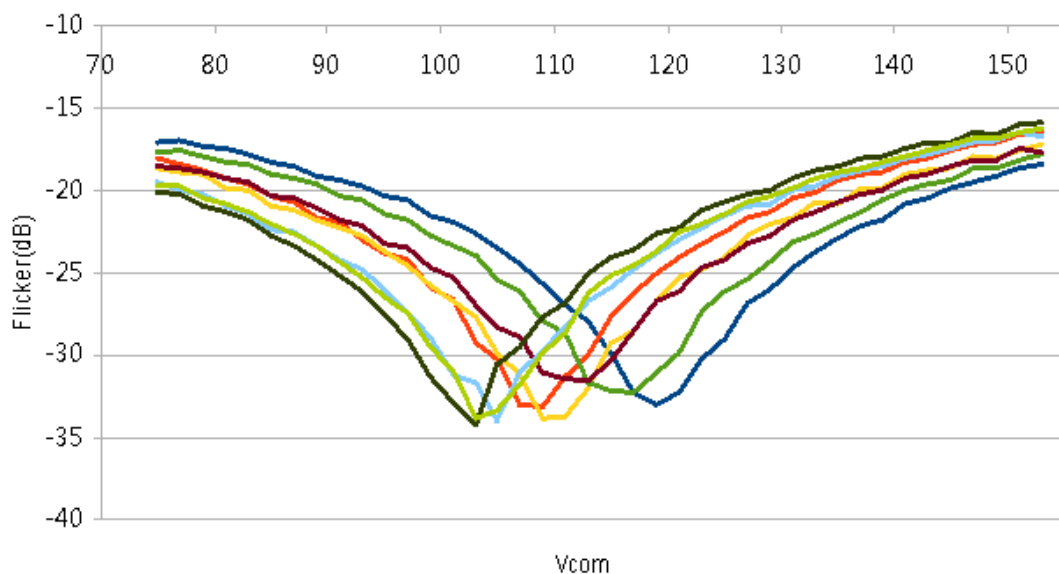


FIGURE 2.11: LCD Flicker 微笑曲線圖

Chapter 3

系統設計與實作

3.1 系統架構

本研究提出一套結合 Hadoop 與 HBase 的集群系統來傳送與儲存 Flicker 數據，且考量底層虛擬化技術與上層 Hadoop 檔案系統與 HBase 資料庫之跨層次效能評估，效能評估是本論文著重的部分，故效能評估有三個階段分別是效能模型 (Performance Model)、效能測試 (Performance Test)、效能分析 (Performance Analyzer)。

- 第一階段：效能模型主要是由四種 Open Source(Hadoop、HBase、Zookeeper、Hbase Client) 來建構雲端儲存系統。
- 第二階段：效能測試是將 Flicker 數據傳送到雲端儲存系統，並確認傳送之時間。
- 第三階段：效能分析是將傳送與寫入效能數據進行比較分析，產出評估報告。

如圖 Figure 3.1 顯示，如何使用 Open Source 來建構雲端儲存系統架構：

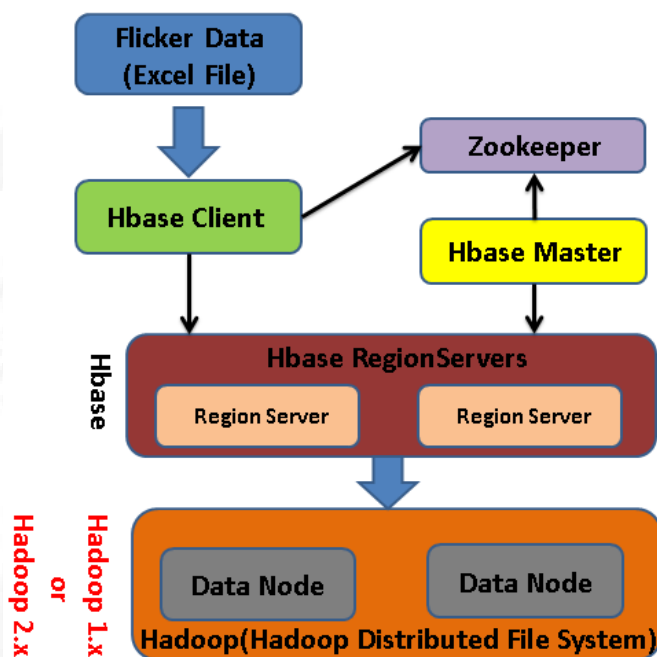


FIGURE 3.1: 系統架構圖

本系統包含了 Hadoop、HBase、Zookeeper、Hbase Client、Excel File，五大模組；系統架構圖主要在說明將 Excel 工作表中之 Flicker 數據轉換成字串經由 Hbase client 端將資料傳送到 HBase，再過 HBase 內部演算法將 Flicker 數據寫入 HDFS 檔案系統中儲存。

- Excel File：所有的 Flicker 數據均在 Excel 工作表中。
- ZooKeeper：管理 Hadoop 集群中的 NameNode，HBase 中 HBaseMaster 的選取，Servers 之間狀態同步等。
- Hmaster：啟動時候會將 HBase 系統表 -ROOT- 加載到 Zookeeper Cluster，通過 Zookeeper Cluster 可以獲取當下之系統表.META. 所存儲對應的 Regionserver 資訊。而 HMaster 主要作用在於，通過 HMaster 維護系統表 -ROOTtable，.META.table，並記錄 RegionServer 所對應 Region 變化資訊。此外還負責監控處理當前 HBase Cluster 中的 RegionServer 狀態與變化資訊。

- HBase RegionServer：則用於多個或是單個的 Region 維護。Region 則對應 HBase 之數據區域與數據維護。
- HBase Client：使用 Zookeeper 查找 Hbase RegionServer 用來訪問 HBase 並利用 put/get/scan 指令操作 HBase。
- Hadoop HDFS：HBase 中的所有數據 (Hlog 與 Hfile) 均儲存在 HDFS 上。

由於 Flickr 數據最終將會儲存到 HDFS 中，故詳細說明系統如何將 Flickr 數據寫入 HBase 再存入 Hadoop 中的 HDFS。

HBase 之 Hlog 運作方式：

Hlog 在 HBase 寫入數據時相對應的工作階段如圖 Figure 3.2 顯示。

HBase 將 Flickr 數據經由 HBaseClient -> 連接的 ZooKeeper -> -ROOT- -> META.->RegionServer->Region。

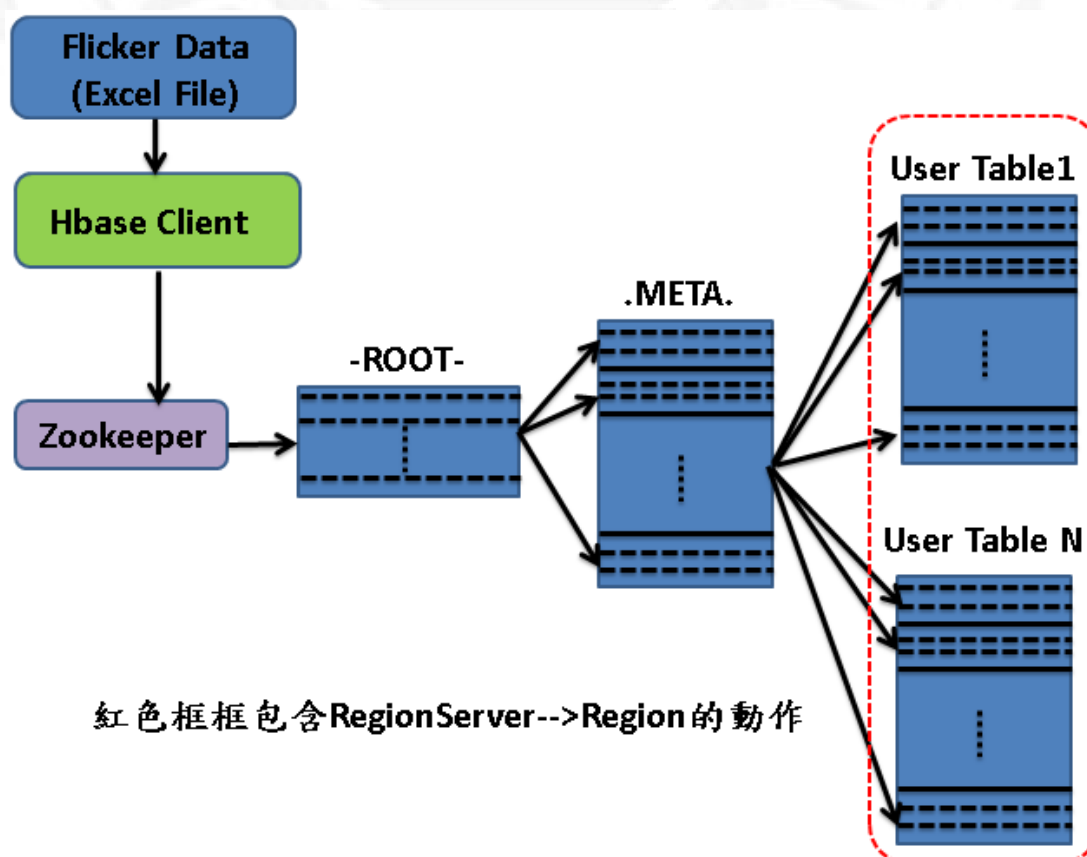


FIGURE 3.2: HBase 寫入工作階段圖

因本實驗為了提高 Flicker 資料寫入性能，將 hbase 的預寫日誌功能 (WAL) 設定關閉，禁止使用。Region 寫數據之前會先檢查 MemStore：

- 如果 Region 的 MemStore 已經有暫存的寫入的數據，則直接返回。
- 如果沒有暫存，則寫入 MemStore. 成功後再返回。

HBase 之 Hfile 運作方式：

RegionServer 管理一系列 HRegion 項目，每個 HRegion 又對應了 Table 中的一個 Region，HRegion 中由多個 HStore 組成。每個 HStore 對應了 Table 中的一個 Column Family 的存儲，所以可以看出每個 Column Family 其實就是一個集中的儲存單元，因此最好將具備共同 IO 特性的 column 放在一個 Column Family 中，是效率最高的。

HStore 是 HBase 儲存的核⼼，是由兩部分組成，分別為 memStore 與 StoreFiles。MemStore 是 Memory Buffer，將要寫入的 Flicker 數據首先會放入 MemStore，當 MemStore 滿了以後會 Flush 成一個 StoreFile (最後為 HFile)，當 StoreFile 文件大小到增加一定容量後 (預設大小為 64MB)，會觸發 Compact 合併的操作，將多個 StoreFiles 合併成一個 StoreFile，合併過程中會進行版本合併和數據刪除，由此可以看出 HBase 其實只有增加數據，所有的更新和刪除操作都是在後續的 compact 過程中進行的，這樣可以讓寫入的操作只進入記憶體中就可以立即返回，確保 HBase I/O 的高性能。當 StoreFiles Compact 後，會逐步形成越來越大的 StoreFile，當單個 StoreFile 大小超過一定大小後，會觸發 Split 操作，同時把當前 Region Split 成 2 個 Region，主 Region 會往下 Split 出的子 Region 並會被 HMaster 分配到相應的 HRegionServer 上，使得原先 1 個 Region 的壓力得以分流到 2 個 Region 上，這樣可以保持各個 HRegionServer 間之負載平衡。

本實驗在 6、10、15、20 萬筆 Flicker Data 的部分由於未超過 256MB 大小之預設值，故無使用到 Split 功能，如 Figure 3.3 描述了 Compaction 和 Split 的過程 (由於無使用到 Split 功能故將其遮住)：

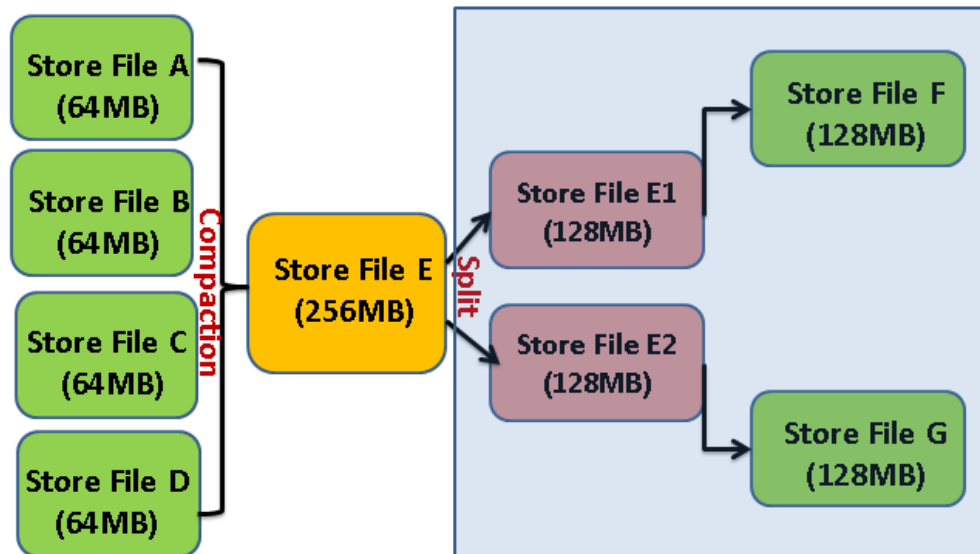


FIGURE 3.3: HStore 之 Compaction 和 Split 的過程圖

本實驗使用 Java 來撰寫 Client API，此 Client API 用來傳送 Flickr 數據到 Hadoop 與 Hbase 平台中，HBase Client API [7] 主要是利用 Zookeeper 主機 IP Address 與 ClientPort 的設定，來作為訪問雲端儲存平台使用，並可以透過 Java 程式，將 Excel 檔案內容轉換為字串，將字串傳送到 HBase 中。

HBase 提供了 Write Buffer 的方式，讓資料可以批量寫入資料到 HBase 中。由於 Flickr 數據每一筆數據均低於 KB 等級的小數據，故利用開啟 Write Buffer 的方式來增加寫入之效率。在預設配置下，Write Buffer 大小為 2MB，可以根據應用實際情況，通過 Java 程式來修改 Write Buffer 大小，而這些資料將會暫時寫入至緩衝區 (write-buffer)，直到緩衝區滿了，才會將資料傳送至 HBase，另外 WriteBufferSize() 寫入緩衝區以 bytes 為單位，越大的緩衝區，client 端和 server 端 memory 的需求就越大。

- `table.setWriteBufferSize(X * 1024 * 1024);` (X 為 Buffer 值，Buffer 大小是以 MB 為單位)
- `table.setAutoFlush(false);` (此指定為關閉 AutoFlush 達到 batch 寫入)

如 Figure 3.4 說明整個系統將 Flickr 數據寫入 HBase 操作流程，最後經由 Hase 中的 HRegionServer 將 Hfile 儲存到 HDFS 檔案系統中，因實驗上有將

Flicker 資料寫入 Buffer 與無寫入 Buffer，故在流程上寫入 HRegionServer 會有兩種流程，另外由於無使用到 Split 功能故將流程圖畫上一個 X 符號。

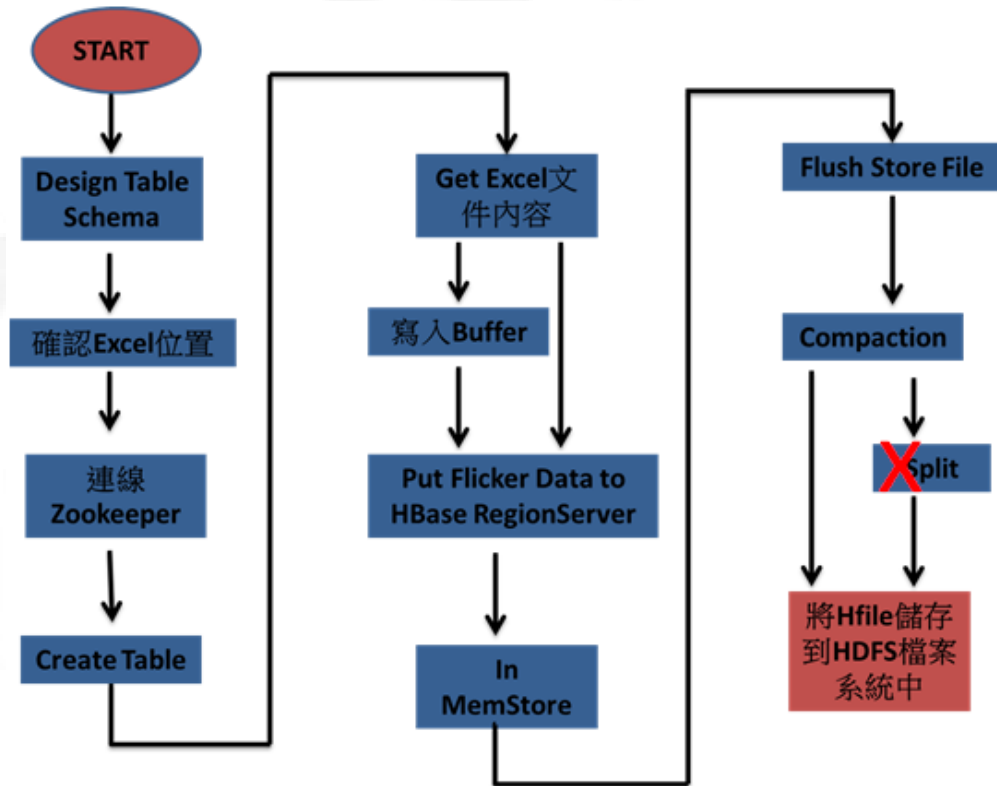


FIGURE 3.4: Flicker 數據寫入到 HBase 流程圖

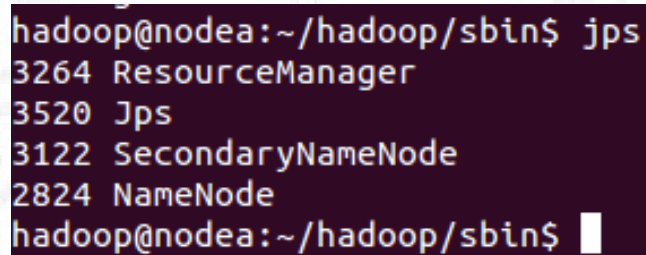
3.2 系統安裝

本實驗研究在使用 VMware Workstation10 與 VMware vSphere 來架設虛擬機環境平台，並在此平台安裝 Hadoop 與 HBase，虛擬機包含一台 Name Node 虛擬機與一台 Data Node 虛擬機，虛擬環境均使用 Ubuntu 12.04.01 Desktop 為作業系統，並在作業系統上安裝不同版本之 Hadoop 與 Hbase，作為雲端 Flicker 儲存系統 [8-10,13,14]。

系統建置步驟：

- 安裝 Ubuntu 作業系統時以 hadoop 為使用者帳戶。
- 更新及升級作業系統套件，並安裝 SSH 服務。

- 更改電腦名稱 (建議使用 PieTTY 工作操作)
- 更改 hosts 設定檔。
- 安裝 jdk。
- 設定 SSH 免密碼登入。
- 安裝與設定 Hadoop。
- 設定 Hadoop 組態檔。
- 將設定好的 Hadoop 組態檔複製到 Nodeb(Data Node)。
- 格式化即啟動 Hadoop。
- 輸入 JPS 指令並確認是否正常啟動 Hadoop。

A terminal window showing the output of the 'jps' command on a Hadoop node. The output lists several processes: ResourceManager (PID 3264), Jps (PID 3520), SecondaryNameNode (PID 3122), and NameNode (PID 2824). The prompt is 'hadoop@nodea:~/hadoop/sbin\$' and the cursor is at the end of the last line.

```
hadoop@nodea:~/hadoop/sbin$ jps
3264 ResourceManager
3520 Jps
3122 SecondaryNameNode
2824 NameNode
hadoop@nodea:~/hadoop/sbin$
```

FIGURE 3.5: Hadoop JPS 結果

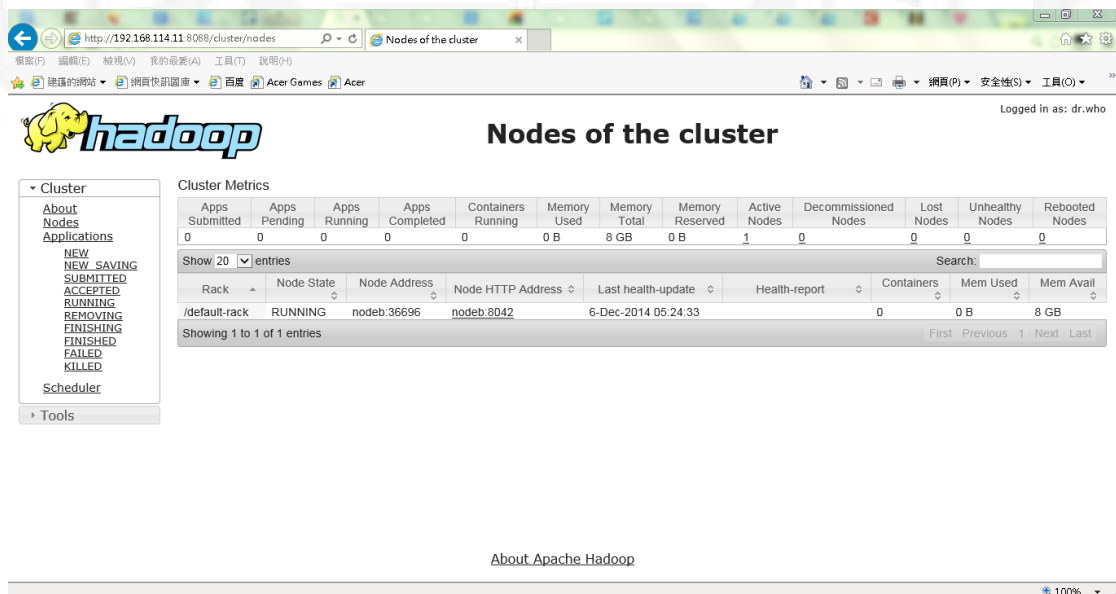
- 關閉 Hadoop。
- 下載與安裝 Hbase。
- 設定 Hbase 組態檔。
- 將設定好的 Hbase 組態檔複製到 Nodeb(Data Node)。
- 啟動 Hadoop。
- 啟動 Hbase。

- 輸入 JPS 指令並確認是否正常啟動 Hadoop 與 Hbase。

```
hadoop@nodea:~/opt/hbase-0.96.1.1-hadoop2/bin$ jps
4210 HRegionServer
3264 ResourceManager
4597 Jps
3122 SecondaryNameNode
3955 HMaster
3888 HQuorumPeer
2824 NameNode
hadoop@nodea:~/opt/hbase-0.96.1.1-hadoop2/bin$
```

FIGURE 3.6: Hadoop 與 Hbase JPS 結果

- 輸入指令 Hbase shell。
- 確認可正常建立 Table 與查詢 Table。
- 輸入指令 exit 結束 Hbase shell。
- 確認 Hadoop 網頁是否正常顯示。



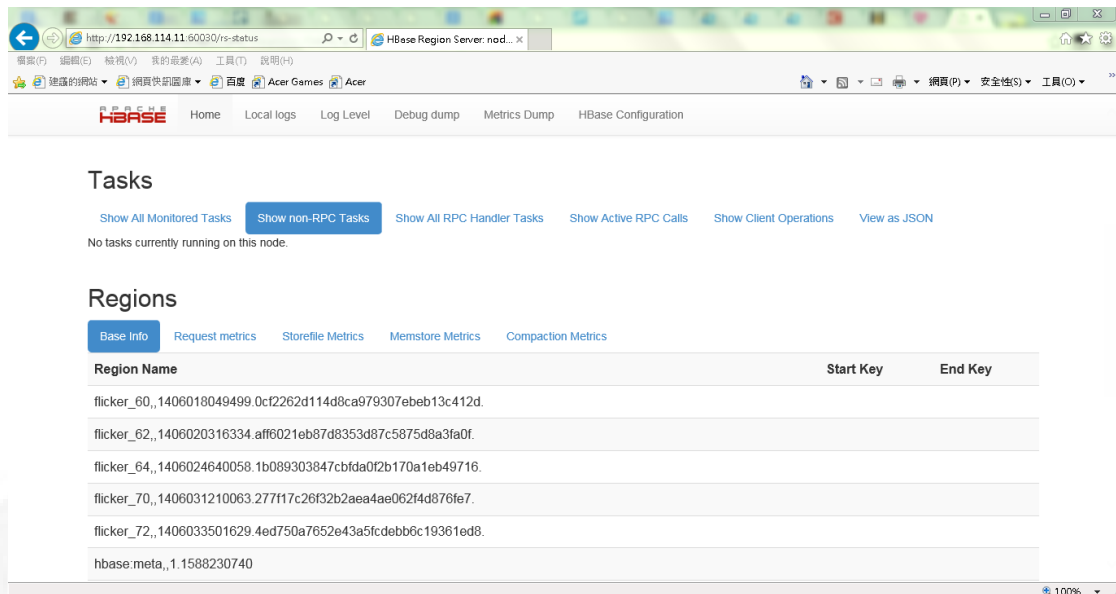
The screenshot shows the Hadoop web interface for 'Nodes of the cluster'. The page includes a navigation menu on the left with options like 'Cluster', 'About Nodes', and 'Applications'. The main content area displays 'Cluster Metrics' and a table of nodes.

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	8 GB	0 B	1	0	0	0	0

Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail
/default-rack	RUNNING	nodeb:36696	nodeb:8042	6-Dec-2014 05:24:33		0	0 B	8 GB

FIGURE 3.7: Hadoop 網頁

- 確認 HBase 網頁是否正常顯示。



The screenshot shows the HBase web interface. The browser address bar indicates the URL is `http://192.168.114.11:60030/rs-status`. The page title is "HBase Region Server: nod...". The navigation menu includes "Home", "Local logs", "Log Level", "Debug dump", "Metrics Dump", and "HBase Configuration".

Tasks

Buttons: Show All Monitored Tasks, Show non-RPC Tasks (selected), Show All RPC Handler Tasks, Show Active RPC Calls, Show Client Operations, View as JSON

No tasks currently running on this node.

Regions

Buttons: Base Info (selected), Request metrics, Storefile Metrics, Memstore Metrics, Compaction Metrics

Region Name	Start Key	End Key
flicker_60,,1406018049499.0cf2262d114d8ca979307eb13c412d.		
flicker_62,,1406020316334.aff6021eb87d8353d87c5875d8a3fa0f.		
flicker_64,,1406024640058.1b089303847cbfda0f2b170a1eb49716.		
flicker_70,,1406031210063.277f17c28f32b2aea4ae062f4d876fe7.		
flicker_72,,1406033501629.4ed750a7652e43a5fcdebb6c19361ed8.		
hbase.meta,,1.1588230740		

FIGURE 3.8: HBase 網頁

Chapter 4

實驗環境與結果

4.1 實驗環境

本實驗研究在使用 VMware vSphere5.1.0 與 VMware Workstation10 作為雲端基礎架構即服務 (IaaS) 平台，兩種平台上均使用 Ubuntu 12.04.01 Desktop 為作業系統，在安裝 Hadoop 與 Hbase 系統，並測試其效能。

NB 實驗環境：使用 VMware Workstation10 建置虛擬機環境平台，並安裝 Hadoop(一台 Name Node，一台 Data Node) 與 HBase，為求實驗的公平性，所有硬體使用一樣的規格，虛擬機設定均使用 1 顆 2 核的 CPU，2GB 的記憶體，20GB 的磁碟空間，作業系統則全部採用 Ubuntu 12.04 LTS 64 位元 Desktop，Hadoop 版本為 1.0.4 搭配 Hbase0.94 與 Hadoop2.2.0 搭配 Hbase0.96。如 Figure 4.1所顯示：

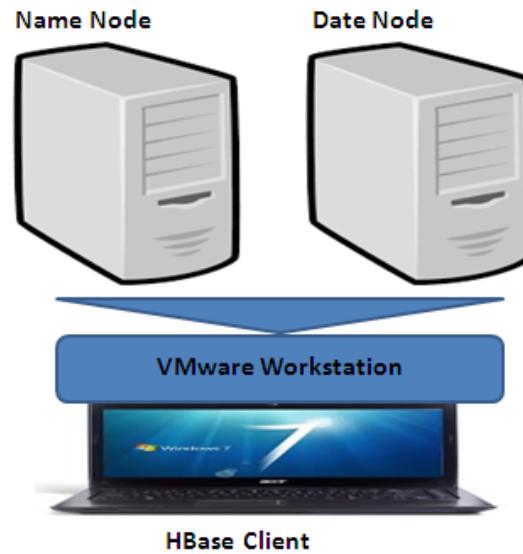


FIGURE 4.1: VMware Workstation 之實驗系統環境圖

一台 Server 實驗環境：使用 VMware vSphere5.1.0 建置虛擬機環境平台，並安裝 Hadoop(一台 Name Node，一台 Data Node) 與 HBase，虛擬機設定均使用 1 顆 2 核的 CPU，8GB 的記憶體，100GB 的磁碟空間，作業系統則採用 Ubuntu 12.04 LTS 64 位元 Desktop 版本，與安裝 Hadoop2.2.0 搭配 Hbase0.96。如 Figure 4.2 所顯示：

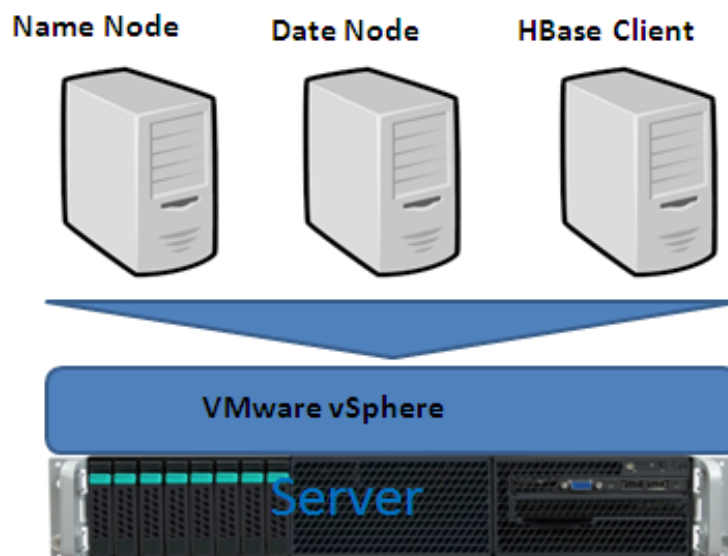


FIGURE 4.2: 一台 Host Server 之實驗系統環境圖

三台 Host Server 實驗環境：使用 VMware vSphere5.1.0 建置虛擬機環境平台，並安裝 Hadoop(一台 Name Node，一台 Data Node) 與 HBase，虛擬機設定均使用 1 顆 2 核的 CPU，8GB 的記憶體，100GB 的磁碟空間，3Gbps 的網路頻寬，作業系統則採用 Ubuntu 12.04 LTS 64 位元 Desktop 版本，與安裝 Hadoop2.2.0 搭配 Hbase0.96。如 Figure 4.2 所顯示：

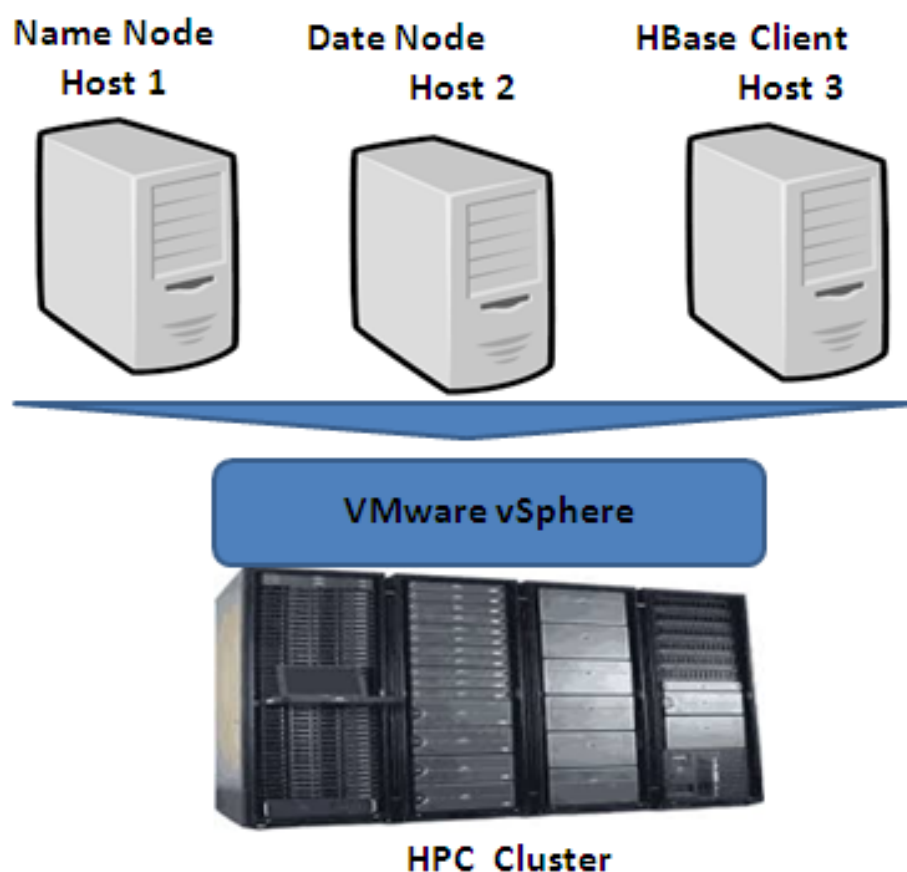


FIGURE 4.3: 三台 Host Servers 之實驗系統環境圖

實驗環境配置如 Table 4.1 至 Table 4.4，兩種虛擬化環境平台差異在於，VMware vSphere 是建立一個伺服器虛擬化環境，本身就包含一個 Linux 內核的作業系統，故不需要建置於其他作業系統上，而 VMware Workstation 需要建置於其他作業系統上才能運行，所以十分依賴主作業系統，若主作業系統出問題時，虛擬機也會受到影響，故較不穩定，而 VMware vSphere 是直接控制及支援實體硬體，則穩定性會比 VMware Workstation 好。

TABLE 4.1: VMware Workstation 與 VMware vSphere 系統配置表

VMware Workstation 配置	VMware vSphere 配置
實體機 WIN 7 Home 版	Linux 內核
Ubuntu12.04	Ubuntu12.04
Hadoop1.0.4 與 2.2.0	Hadoop2.2.0
HBase0.94 與 0.96	HBase0.96

TABLE 4.2: Client 端硬體規格表

VMware Workstation 配置	VMware vSphere 配置
WIN 7 Home 版	WIN 7 Home 版
1 顆 2 核 CPU	1 顆 2 核 CPU
8GB 記憶體	8GB 記憶體
750GB 硬碟	100GB 硬碟

TABLE 4.3: Hadoop Name Node 虛擬機硬體規格表

VMware Workstation 配置	VMware vSphere 配置
Ubuntu LTS 64bit Desktop	Ubuntu LTS 64bit Desktop
1 顆 2 核 CPU	1 顆 2 核 CPU
2GB 記憶體	8GB 記憶體
20GB 硬碟	100GB 硬碟

TABLE 4.4: Hadoop Data Node 虛擬機硬體規格表

VMware Workstation 配置	VMware vSphere 配置
Ubuntu LTS 64bit Desktop	Ubuntu LTS 64bit Desktop
1 顆 2 核 CPU	1 顆 2 核 CPU
2GB 記憶體	8GB 記憶體
20GB 硬碟	100GB 硬碟

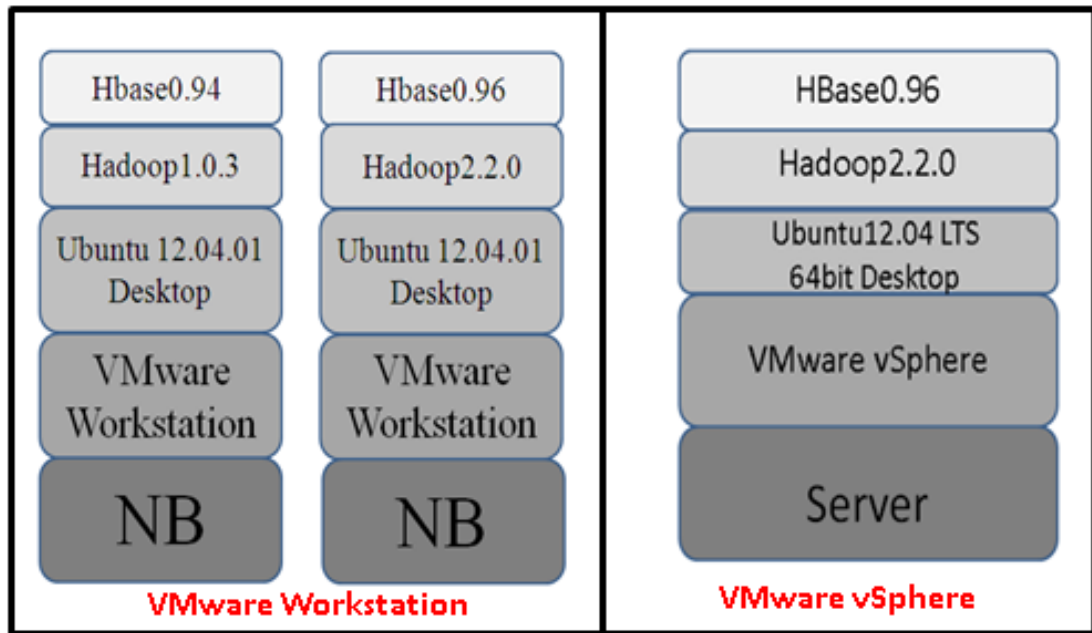


FIGURE 4.4: VMware Workstation 與 VMware vSphere 實驗系統架構圖

4.2 實驗架構

如 Figure 4.2 連線環境圖所示，Excel File 透過 HBase Client API 來傳送與寫入 Flickr 數據到 Hadoop 與 HBase 中儲存，所有的 Flickr 數據是存放在 Excel File 裡面，使用 HBase Client API 來訪問 HBase 並將 Excel File 裡的數據傳送到 Hadoop 與 Hbase 系統中，並紀錄所需時間：



FIGURE 4.5: 連線環境示意圖

實驗過程說明：

- 實驗開始：先在 VMware Workstation10 與 VMware vSphere5.1.0 上，將虛擬機安裝 Ubuntu 12.10 Desktop 作業系統，作業系統安裝完後，要將系統更新。
- 系統環境建置：在 Ubuntu 12.10 Desktop 作業系統上安裝 Hadoop 包含 Name Node 與 Data Node，並確認正常啟動無誤，在 NB 實驗環境之 Hadoop 會建立兩種版本，一種為 Hadoop1.0.4 版本另一種為 Hadoop2.2.0 版本；而 Server 環境中則只建立 Hadoop2.2.0 版本。
- 建立資料庫：安裝 HBase 資料庫，HBase 在版本上有所區別，舊版本的 Hadoop1.0.4 搭上 HBase0.94；新版本 Hadoop2.2.0 搭上 HBase0.96 版本，確認兩種版本均啟動無誤後，建立相關 HBase Table。
- 連線測試：使用 Hbase Client 端管理功能 API，作為訪問 Hbase 之連線工具，此 API 使用 NetBeans 軟體開發工具並 Java 來撰寫 Hbase 客戶端管理功能 API。
- 確認 Flickr Data 在個實驗環境中的傳送與寫入時間，並確認是否正確傳送到 HBase 中，並分析效能。

- 實驗結束。

4.3 實驗方法

由本實驗，使用虛擬機作為雲端基礎架構即服務 (IaaS) 平台，並以在虛擬機上建立 Hadoop 與 HBase，並使用 HBase Client 端 API 來傳送 Flickr Data，並記錄在有無開啟 Buffer 的條件下來比較不同環境下之效能，此效能差異是以傳送 6、10、15、20 萬筆 Flickr Data 所需的傳送與寫入時間來做為比較效能依據。Table 4.5 為 Flickr 資料內容，A 為 family，B 為 column，A 與 B 需先在系統建立表格，C 為 Flickr 資料內容，由客戶端傳送至 Hadoop 與 HBase 雲端儲存系統。

TABLE 4.5: Flickr 資料內容表

family	colum	flicker data
1Project	English	XXXXXX
2Project No	No	1234567
3Program	Code	XXXXXX
4Owner	Name	Danny Hsieh
5StartTime	StartTime	2012-10-17
6EndTime	EndTIME	下午 01:53:17
7CycleTime	CycleTime	下午 01:57:10
Flick0	Flicker0	4.63E+00
Flick1	Flicker1	4.63E+00
Flick2	Flicker2	4.12E+00
Flick3	Flicker3	3.64E+00
Flick4	Flicker4	3.21E+00
Flick5	Flicker5	2.76E+00
Flick6	Flicker6	2.21E+00
Flick7	Flicker7	1.85E+00
Flick8	Flicker8	2.43E+00
Flick9	Flicker9	3.06E+00
FlickA	FlickerA	4.86E+00
MinFlick	MinFlicker	1.85E+00
SensorCurvePass	SensorCurvePass	1
VcomValue	VcomValue	Pass

本實驗主要是比較不同建置環境下之 Flicker Data 所需的傳送與寫入時間，每一項測試實驗均測試五次，並在五次中以最快的時間作為時間紀錄。

NB 實驗環境：在 VMware Workstation10 虛擬環境中比較 Hadoop 新舊版本效能，此環境均在 NB 中建立 2 台虛擬機並使用 NB 當作 Client 端測試其傳送與寫入效能。

- 實驗一、不開啟 Buffer 狀況下，比較 6、10、15、20 萬筆 Flicker Data 傳送與寫入時間。
- 實驗二、開啟 Buffer 狀況下，Buffer Size 5MB、10MB、15MB、20MB 之 6、10、15、20 萬筆 Flicker Data 所需的傳送與寫入時間。

Server 實驗環境：在 VMware vSphere5.1.0 虛擬環境中測試 Hadoop2.2.0 效能，測試使用一台 Server 建立三台虛擬機之傳送與寫入效能。

- 實驗一、不開啟 Buffer 狀況下，比較 6、10、15、20 萬筆 Flicker Data 傳送與寫入時間。
- 實驗二、開啟 Buffer 狀況下，Buffer Size 5MB、10MB、15MB、20MB 之 6、10、15、20 萬筆 Flicker Data 所需的傳送與寫入時間。

3 Host Server 實驗環境：在 VMware vSphere5.1.0 虛擬環境中測試 Hadoop2.2.0 效能，此環境經由網路傳送 Flicker Data，網路頻寬 3Gbps。

- 實驗一、不開啟 Buffer 狀況下，比較 6、10、15、20 萬筆 Flicker Data 傳送與寫入時間。
- 實驗二、開啟 Buffer 狀況下，Buffer Size 5MB、10MB、15MB、20MB 之 6、10、15、20 萬筆 Flicker Data 所需的傳送與寫入時間。

4.4 實驗結果

NB 實驗環境：實驗一不開啟 Buffer 時，在 6、10、15、20 萬筆 Flicker 資料下，顯示所得到之結果如 Table 4.6、Figure 4.6：

- 傳送與寫入時間，Hadoop2.2.0 優於 Hadoop1.0.4。
- 效能上 Hadoop2.2.0 高於 Hadoop1.0.4 約 19 至 23 個百分點。

TABLE 4.6: NB 環境無 Buffer 之效能差異表

	Hadoop1.0.4	Hadoop2.2.0	效能差異
6 萬筆資料	1328 秒	1037 秒	21.91265 百分點
10 萬筆資料	2043 秒	1841 秒	23.3874 百分點
15 萬筆資料	3282 秒	2597 秒	20.87142 百分點
20 筆資料	4841 秒	3880 秒	19.85127 百分點

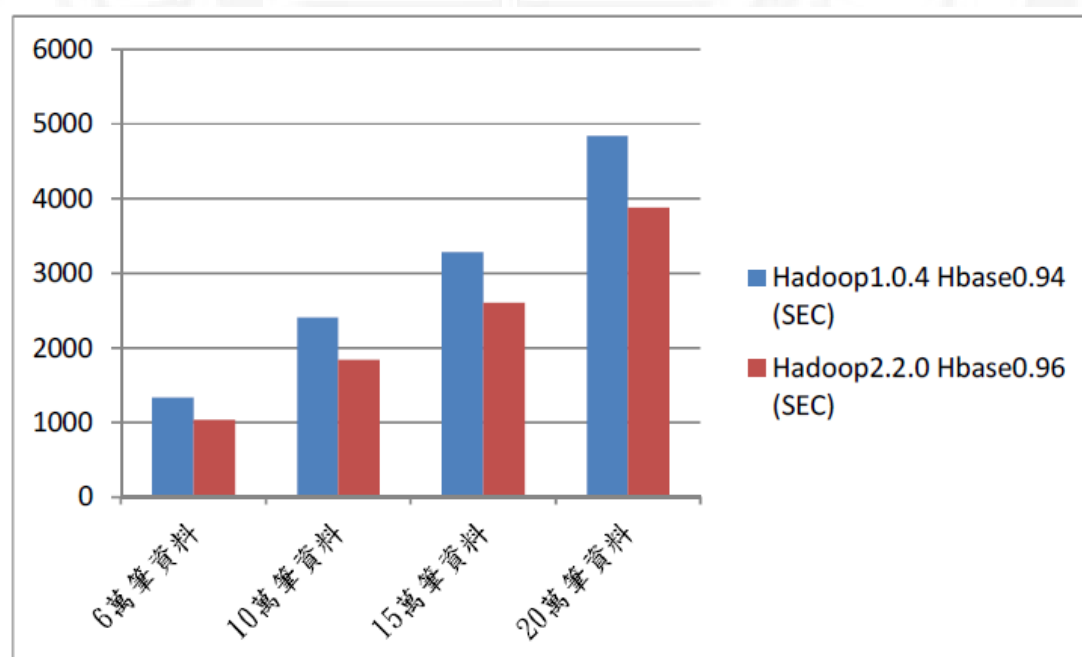


FIGURE 4.6: NB 環境無 Buffer 之比較圖

比較 Hadoop2.2.0 在三種實驗環境中不開啟 Buffer 時，在 6、10、15、20 萬筆 Flicker 資料下，顯示所得到之結果如 Table 4.7、Figure 4.7：

- NB 環境：NB 內部 Bus 頻寬有限，並使用同一顆硬碟，故效能最差。

- Server 環境：使用單一 Server 建置三台虛擬機，並共用同一顆硬碟，但因 Server 內部 Bus 頻寬大，故效能最佳。
- 3 Host Servers 環境：獨立三台 Server 分別建立一台虛擬機，非共用硬碟狀況下，理論上效能應該最佳，受限於當時網路狀態的影響，故效能在於 NB 環境與 Server 之間。

TABLE 4.7: 三種實驗環境中無 Buffer 之效能差異表

	NB	Server	3 Host Servers
6 萬筆資料	1037 秒	376 秒	532 秒
10 萬筆資料	1841 秒	599 秒	729 秒
15 萬筆資料	2597 秒	882 秒	1090 秒
20 筆資料	3880 秒	1200 秒	1684 秒



FIGURE 4.7: 三種實驗環境中無 Buffer 之比較圖

三種實驗環境開啟 Buffer 後，Buffer Size 不同大小狀況下傳送與寫入 Flicker Data 的時間記錄如 Table 4.8至 Table 4.15與 Figure 4.8至 Figure 4.17所示。

NB 實驗環境下，Hadoop 1.0.4 與 Hadoop 2.2.0 效能比較，如 Table 4.8與 Figure 4.8所顯示：

TABLE 4.8: NB 實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 時間表

Buffer Size	Hadoop1.0.4	Hadoop2.2.0
5MB	312 秒	446 秒
10MB	216 秒	487 秒
15MB	166 秒	141 秒
20MB	200 秒	487 秒

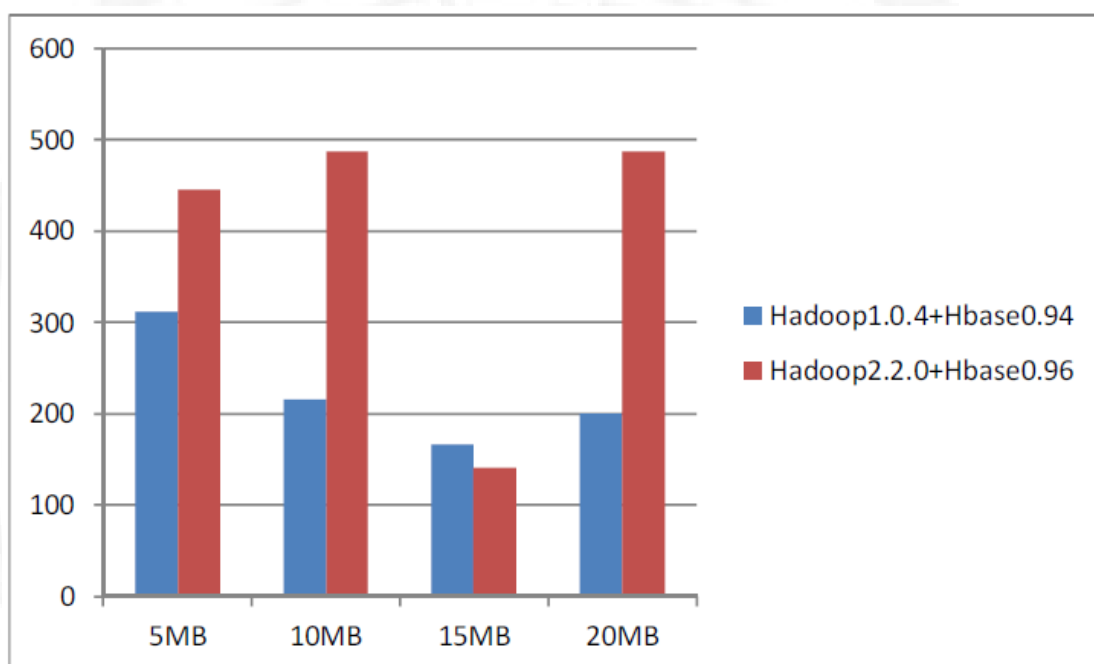


FIGURE 4.8: NB 實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 比較圖

Figure 4.8的實驗在六萬筆 Flicker Data 條件下，Buffer Size 在 15MB 時，Hadoop2.2.0 效能優於 Hadoop1.0.4。

比較 Hadoop2.2.0 在三種實驗環境中開啟 Buffer 時，在 6 萬筆 Flicker 資料下，顯示所得到之結果如 Table 4.9、Figure 4.9：

TABLE 4.9: 三種實驗環境開啟 Buffer 之 6 萬筆 Flicker Data 時間表

Buffer Size	NB	Server	3 Host Servers
5MB	446 秒	169 秒	148 秒
10MB	487 秒	475 秒	446 秒
15MB	141 秒	478 秒	444 秒
20MB	487 秒	482 秒	457 秒

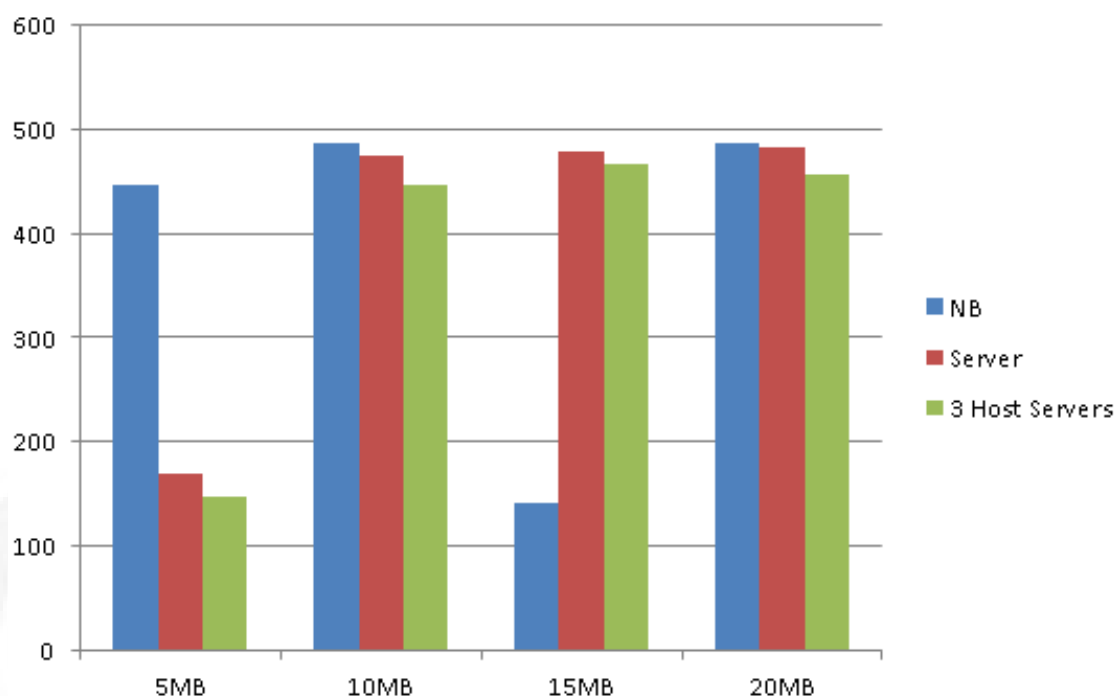


FIGURE 4.9: 三種實驗環境開啟 Buffer 之 6 萬筆 Flickr Data 比較圖

針對一台 Server 與 3 台 Host Servers 環境下做比較，兩者實驗數據類似，3 台 Host Servers 環境下優於單一 Server 環境，在 6 萬筆資料下均在 5MB 有較好的效能，而 NB 環境必須在 15MB 才有較好的效能。

NB 實驗環境下，Hadoop 1.0.4 與 Hadoop 2.2.0 開啟 Buffer 之 10 萬筆 Flickr Data 效能比較，如 Table 4.10 與 Figure 4.10

TABLE 4.10: 開啟 Buffer 之 10 萬筆 Flickr Data 時間表

Buffer Size	Hadoop1.0.4	Hadoop2.2.0
5MB	258 秒	245 秒
10MB	340 秒	250 秒
15MB	288 秒	284 秒
20MB	343 秒	320 秒

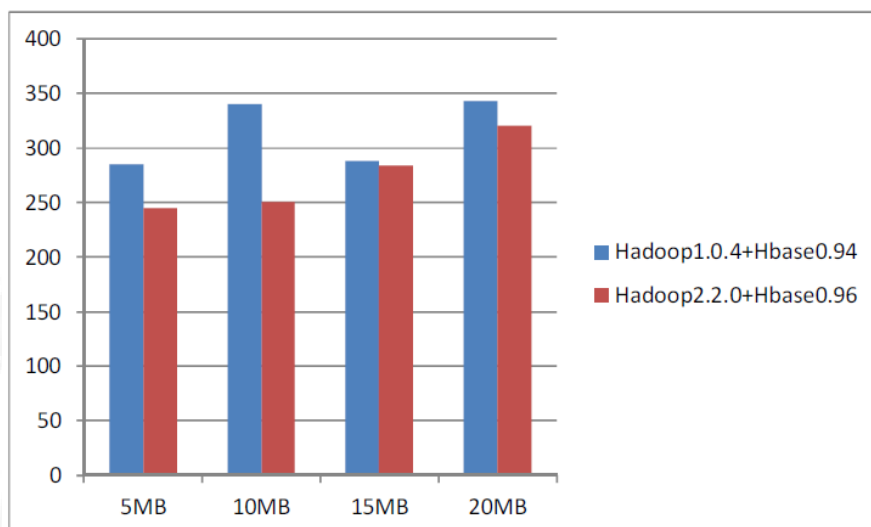


FIGURE 4.10: 開啟 Buffer 之 10 萬筆 Flickr Data 比較圖

Figure 4.10的實驗在 10 萬筆 Flickr Data 條件下，Buffer Size 在 5MB、10MB、15MB、20MB 時，Hadoop2.2.0 效能均優於 Hadoop1.0.4。

比較 Hadoop2.2.0 在三種實驗環境中開啟 Buffer 時，在 10 萬筆 Flickr 資料下，顯示所得到之結果如 Table 4.11、Figure 4.11：

TABLE 4.11: 三種實驗環境之開啟 Buffer 之 10 萬筆 Flickr Data 時間表

Buffer Size	NB	Server	3 Host Servers
5MB	245 秒	271 秒	233 秒
10MB	250 秒	568 秒	543 秒
15MB	284 秒	269 秒	225 秒
20MB	320 秒	581 秒	549 秒

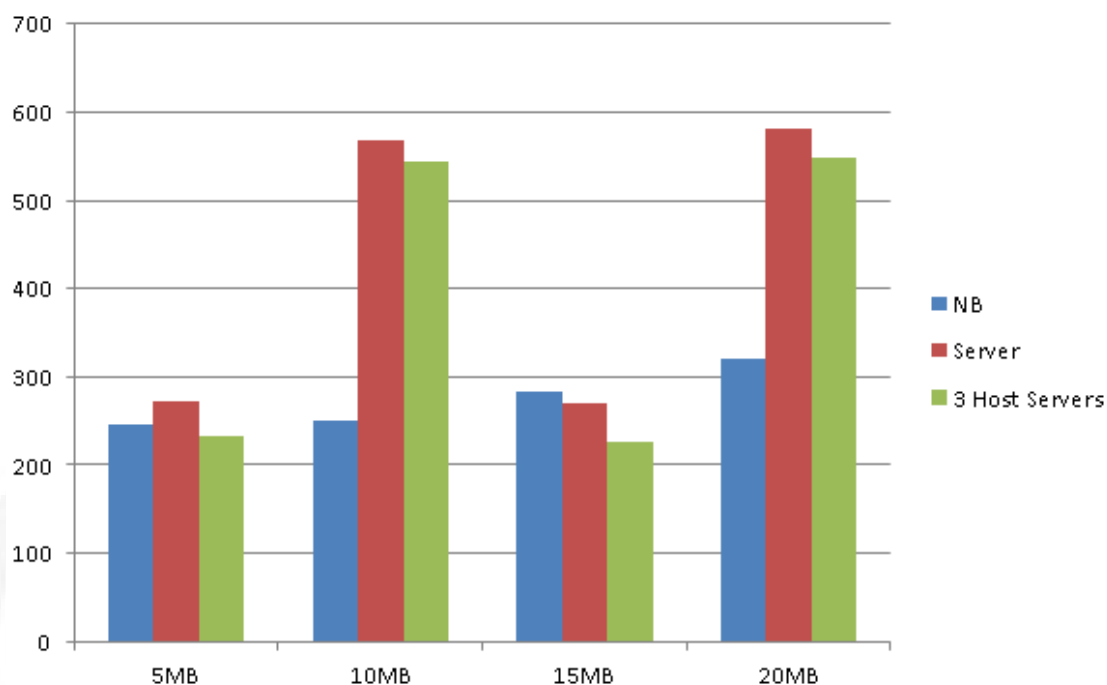


FIGURE 4.11: 三種實驗環境開啟 Buffer 之 10 萬筆 Flickr Data 比較圖

傳送 10 萬筆資料狀況下，針對一台 Server 與 3 台 Host Servers 環境下做比較，兩者實驗數據類似，3 台 Host Servers 環境下優於單一 Server 環境，在 10 萬筆資料下在 5MB 與 15MB 有較好的效能，而 NB 環境在 5MB 才有較好的效能。

NB 實驗環境下，Hadoop 1.0.4 與 Hadoop 2.2.0 開啟 Buffer 之 15 萬筆 Flickr Data 效能比較，如 Table 4.12 與 Figure 4.12

TABLE 4.12: 開啟 Buffer 之十五萬筆 Flickr Data 時間表

Buffer Size	Hadoop1.0.4	Hadoop2.2.0
5MB	514 秒	406 秒
10MB	506 秒	727 秒
15MB	512 秒	711 秒
20MB	495 秒	740 秒

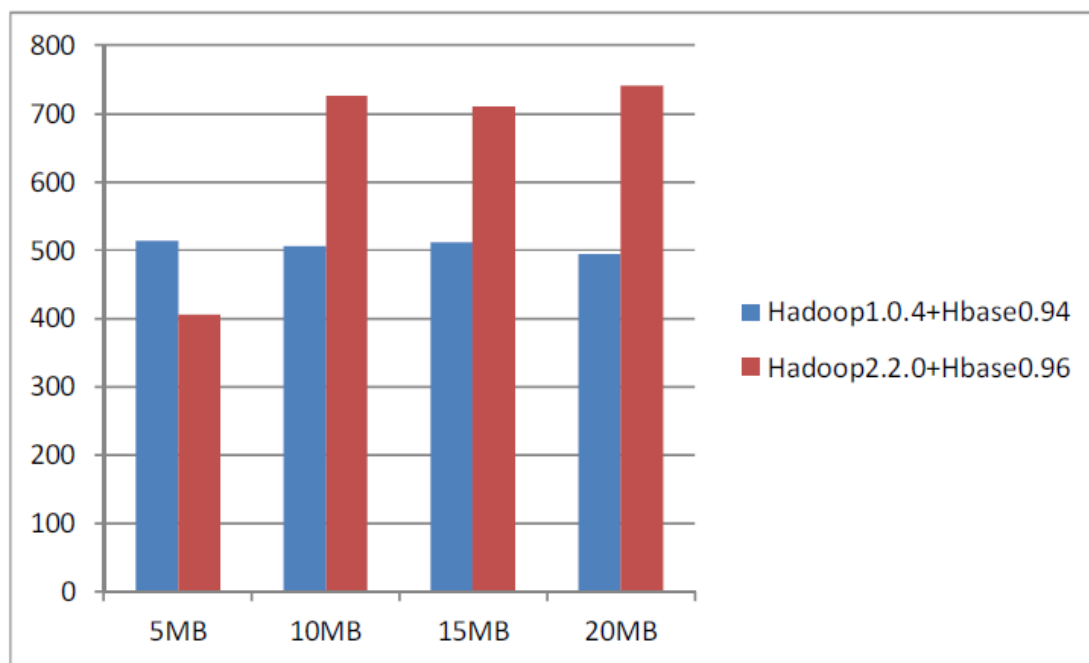


FIGURE 4.12: 開啟 Buffer 之十五萬筆 Flickr Data 比較圖

Figure 4.12的實驗在十五萬筆 Flickr Data 條件下，Buffer Size 在 5MB 時，Hadoop2.2.0 效能優於 Hadoop1.0.4，另外獨立來看其餘 Buffer Size 在 Hadoop1.0.4 與 Hadoop2.2.0 時間上只有些微差距，效能沒有明顯提升。

比較 Hadoop2.2.0 在三種實驗環境中開啟 Buffer 時，在 15 萬筆 Flickr 資料下，顯示所得到之結果如 Table 4.13、Figure 4.13：

TABLE 4.13: 三種實驗環境開啟 Buffer 之 15 萬筆 Flickr Data 時間表

Buffer Size	NB	Server	3 Host Servers
5MB	406 秒	416 秒	350 秒
10MB	727 秒	714 秒	657 秒
15MB	711 秒	426 秒	367 秒
20MB	740 秒	746 秒	669 秒

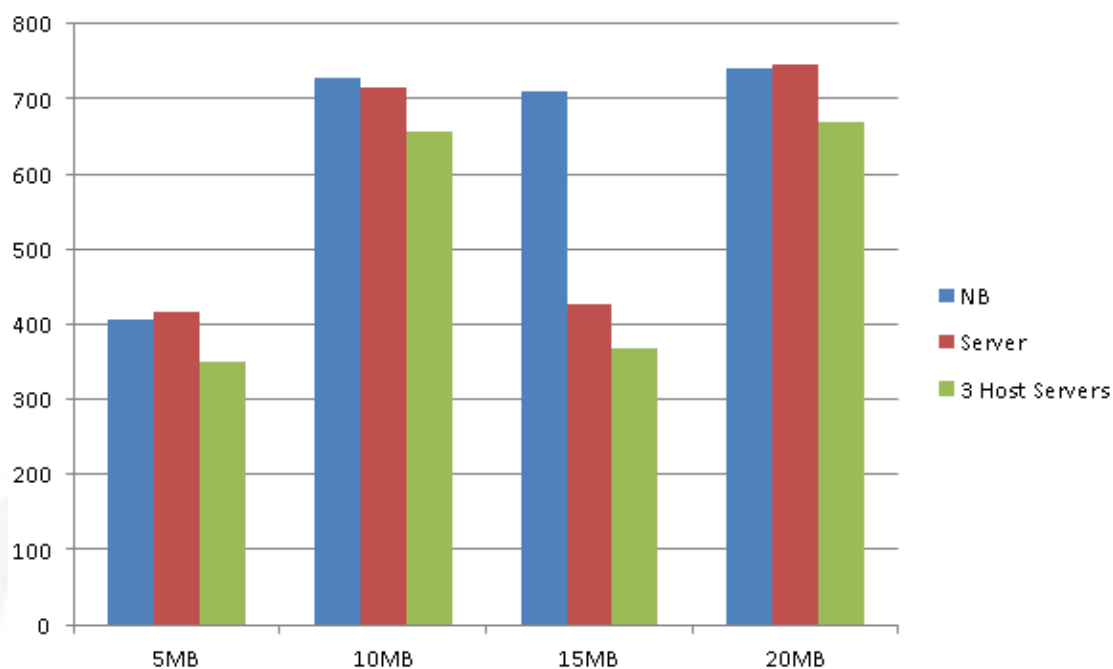


FIGURE 4.13: 三種實驗環境開啟 Buffer 之 15 萬筆 Flickr Data 比較圖

傳送 15 萬筆資料狀況下，針對一台 Server 與 3 台 Host Servers 環境下做比較，兩者實驗數據類似，3 台 Host Servers 環境下優於單一 Server 環境，在 15 萬筆資料下在 5MB 與 15MB 有較好的效能，而 NB 環境在 5MB 才有較好的效能。

NB 實驗環境下，Hadoop 1.0.4 與 Hadoop 2.2.0 開啟 Buffer 之 20 萬筆 Flickr Data 效能比較，如 Table 4.14 與 Figure 4.14

TABLE 4.14: 開啟 Buffer 之 20 萬筆 Flickr Data 時間表

Buffer Size	Hadoop1.0.4	Hadoop2.2.0
5MB	809 秒	664 秒
10MB	730 秒	923 秒
15MB	738 秒	912 秒
20MB	793 秒	930 秒

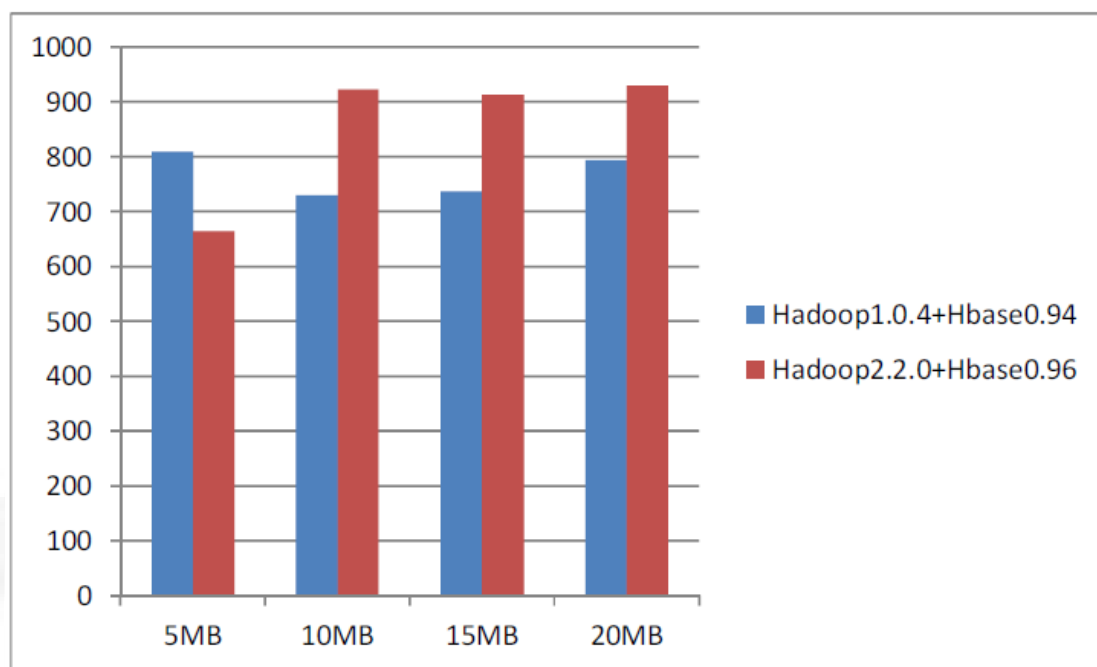


FIGURE 4.14: 開啟 Buffer 之 20 萬筆 Flickr Data 直條圖

Figure 4.14實驗在二十萬筆 Flickr Data 條件下，Buffer Size 在 5MB 時，Hadoop2.2.0 效能優於 Hadoop1.0.4，獨立來看其餘 Buffer Size 在 Hadoop1.0.4 與 Hadoop2.2.0 時間上只有些微差距，效能沒有明顯提升，與十五萬筆 Flickr Data 比較圖一致，不因為 Buffer Size 增加而效能有明顯之差異。

比較 Hadoop2.2.0 在三種實驗環境中開啟 Buffer 時，在 20 萬筆 Flickr 資料下，顯示所得到之結果如 Table 4.15、Figure 4.15：

TABLE 4.15: 三種實驗環境之開啟 Buffer 之 20 萬筆 Flickr Data 時間表

Buffer Size	NB	Server	3 Host Servers
5MB	664 秒	592 秒	503 秒
10MB	923 秒	590 秒	476 秒
15MB	912 秒	902 秒	793 秒
20MB	930 秒	924 秒	820 秒

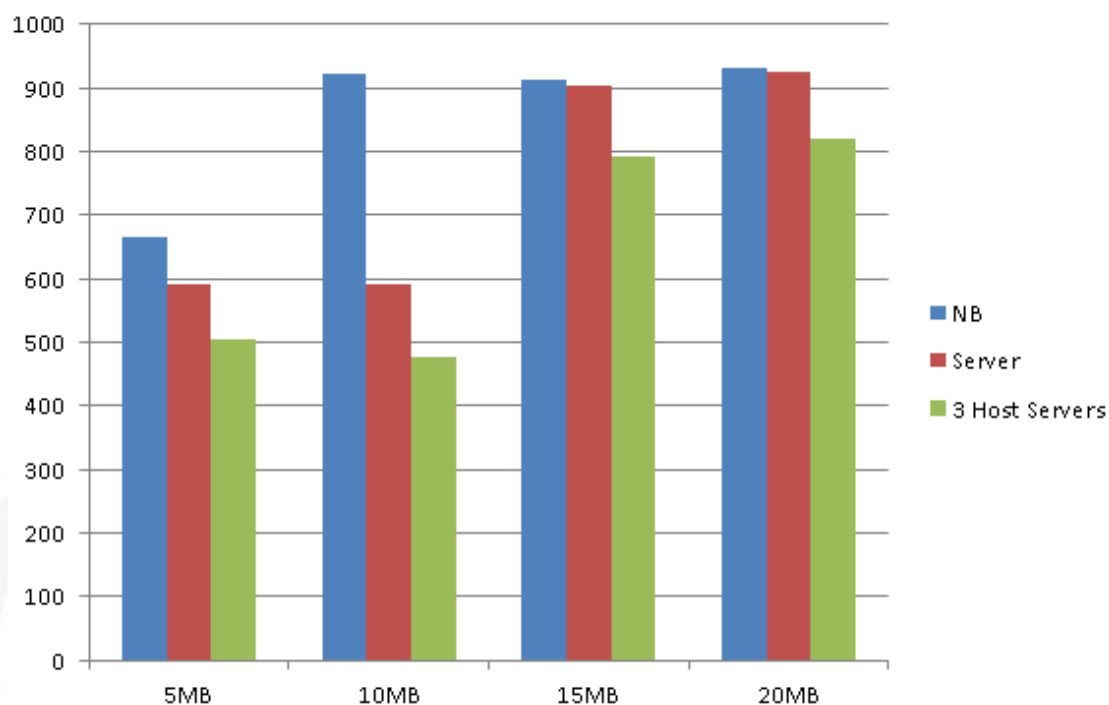


FIGURE 4.15: 三種實驗環境開啟 Buffer 之 20 萬筆 Flickr Data 比較圖

傳送 20 萬筆資料狀況下，針對一台 Server 與 3 台 Host Servers 環境下做比較，兩者實驗數據類似，3 台 Host Servers 環境下優於單一 Server 環境，在 20 萬筆資料下在 10MB 有較好的效能，而 NB 環境在 5MB 才有較好的效能。

由 Figure 4.16 NB 環境下將所有數據彙整後顯示在 Hadoop1.0.4 與 Hadoop2.2.0，數據量越大所需之傳送與寫入時間長，開啟 Buffer 後 Hadoop2.2.0 效能必須要在最佳 Buffer Size 才能夠優於 Hadoop1.0.4。

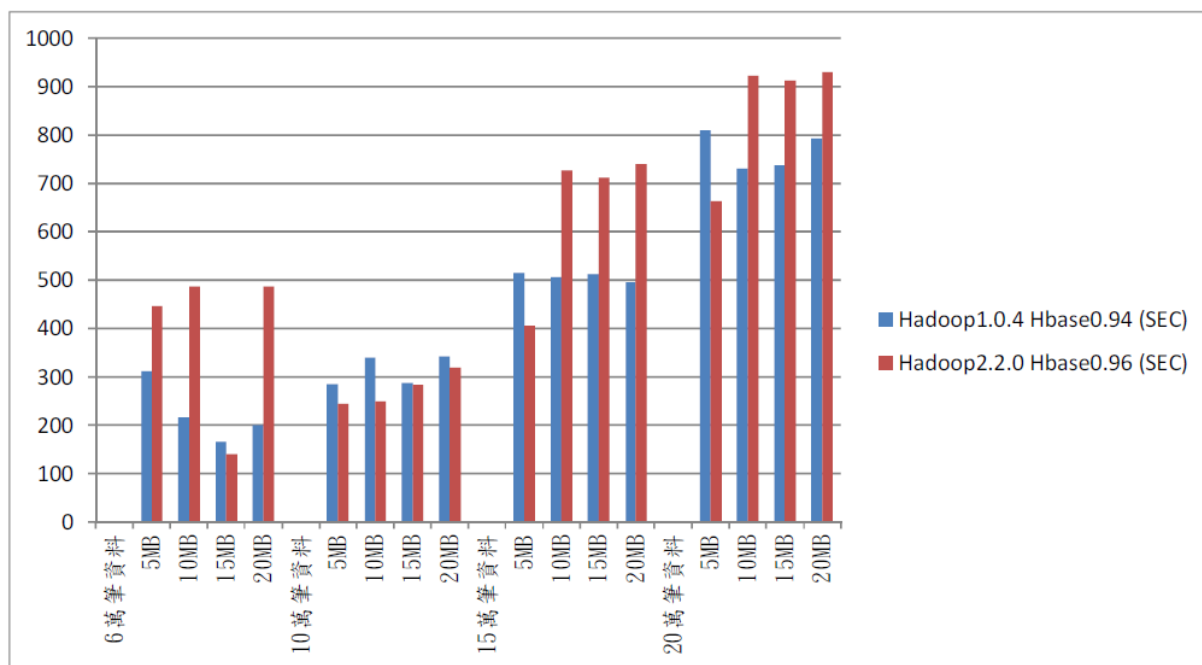


FIGURE 4.16: NB 環境下開啟 Buffer 各筆數 Flicker Data 比較圖

經由上述實驗結果得知在 NB 環境中以 VMware Workstation 建立虛擬機，並安裝 Hadoop 與 HBase 之雲端儲存系統，對於執行資料傳送與寫入效能差異如下：

- 在無 Buffer 環境之傳送與寫入所需時間遠高於有 Buffer 之傳送與寫入所需時間，這個部份說明了開啟 Buffer 比無 Buffer 效能佳。
- 開啟 Buffer 後，在 10 萬筆資料內，Buffer Size 大小對於傳送與寫入時間會有影響，大於 10 萬筆資料後，Buffer Size 大小影響程度就明顯變小。
- Hadoop 新版與舊版的效能比較上，在不同的資料量，不同 Buffer Size 會有所差異，在非最佳 Buffer Size 時，Hadoop1.0.3 效能優於 Hadoop2.2.0；在最佳 Buffer Size 下，Hadoop2.2.0 效能優於 Hadoop1.0.3。
- 當資料大於 10 萬筆，架構在 1 個 Name Node 與 1 個 Data Node 下，最佳 Buffer Size 會是 5MB，小於 10 萬筆資料 Buffer Size 則會是在 15MB。

如 Figure 4.17彙整三個實驗環境下 Hadoop2.2.0 實驗數據，3 台 Host Servers 環境在開啟 Buffer 下效能最佳，一台 Server 裝三台虛擬機在開啟 Buffer 狀態下效能次佳，NB 環境下需要再最佳 Buffer Size 下才会有較好的效能。

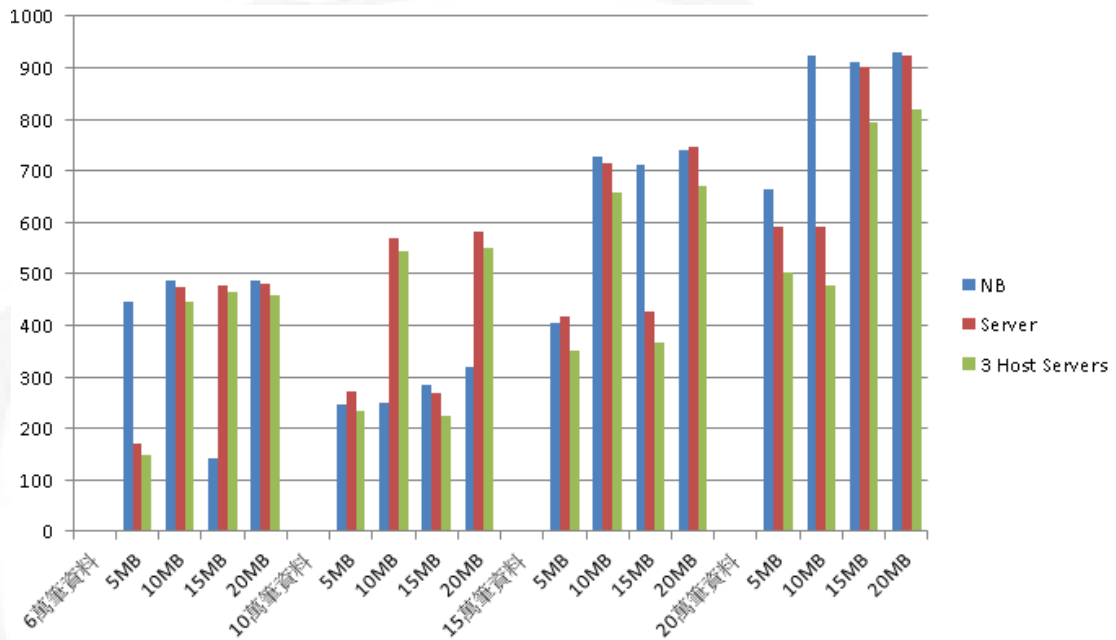


FIGURE 4.17: 三個實驗環境下開啟 Buffer 各筆數 Flicker Data 比較圖

上述之實驗結果，說明在三種環境下建立虛擬機（一個 Name Node 與 Data Node），其實驗結果會有所差異：

- NB 環境下，因為 NB 內部 Bus 頻寬限制，與共用同一顆硬碟，使得效能落差較大，傳送與寫入時間也比較長。
- 在 Server 環境下，因為內部 Bus 效能較好，則效能比 NB 環境好，傳送與寫入時間也比較短。
- 三台 Host Servers 環境下，硬碟非共用與使用網路傳送資料，實驗結果效能最好，也與單一 Server 實數據趨勢相同。

因本研究在三種實驗環境，建立一個 Name Node 與一個 Data Node 架構下所得到的實驗結果，會因為在不同的環境架構，不同的電腦硬體效能與不同連線架構下所得到的 Flicker 資料傳送與寫入結果，定會與目前的實驗結果有所差異。

Chapter 5

結論與未來方向

5.1 結論

在大數據的時代裡，資料數據就是一切，不只會影響到企業的競爭力，也影響產業的提升能力，而雲端儲存系統可以不限時間，不限地點，只要有網路的地方，即可以使用企業內部的資源，做最有效率的分析與決策，對於國際化公司而言，這是一個趨勢。使用者可以直接連上公司內部網路查詢目前生產線的生產數據並立即處理與分析，這樣可以快速又有效率的解決工廠生產報廢的風險與成本的控管。資料數據對於面板產業也是提升公司競爭力的關鍵要素，如果沒有將資料數據妥善儲存，那麼就無法利用資料數據加以分析並找出產品不良率的真因，也無法將資料數據做有效分析，使得良率無法提升，導致公司競爭力下降，

故本論文建構三種雲端儲存系統環境，並測試其效能，如果只需要建置小型 LCD Flicker 雲端儲存系統，可以利用現有 NB 或 PC 主機來建置此系統，如果需要跨廠區或是跨國之大型 LCD Flicker 雲端儲存系統，考慮到系統穩定性與安全性，則建議使用 Cluster Server 環境來建置系統，不管建置環境為何，資料的儲存對於一間公司是競爭力的根本，也是公司穩定發展的基礎。

5.2 未來方向

面板下線測試數據不單單只有 Flicker 數據，未來可以將其他面板數據一並存入雲端儲存系統，並測試其效能。未來可以實驗之方向如下：

- 未來可以增加 Data Node 主機數目，來測試其傳送與儲存時間，效能是否更加快速就完成儲存。
- 加入其他面板數據，如：光學數據（色座標、亮度、Gamma、穿透率...等）、電壓、電流。
- 嘗試使用商業套裝軟體，如:Couldera，來做資料儲存的效能分析。
- 針對資料探勘及資料分析之類的相關知識及技術研究。

參考文獻

- [1] Catherine P. Jayapandian, Chien-Hung Chen, Alireza Bozorgi, Samden D. Lhatoo, Guo-Qiang Zhang, and Satya Sanket Sahoo. Cloudwave: Distributed processing of "big data" from electrophysiological recordings for epilepsy clinical research using hadoop. In *AMIA 2013, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 16-20, 2013*.
- [2] Chao-Tung Yang, Jung-Chun Liu, Ching-Hsien Hsu, and Wei-Li Chou. On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. *The Journal of Supercomputing*, 69(3):1103–1122, 2014.
- [3] D Carstoiu, A Cernian, and A Olteanu. Hadoop hbase-0.20. 2 performance evaluation. In *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, pages 84–87. IEEE, 2010.
- [4] Toshimori Honjo and Kazuki Oikawa. Hardware acceleration of hadoop mapreduce. In *Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA*, pages 118–124, 2013.
- [5] 楊士明. 液晶顯示器畫面產生抖動閃爍成因探討及解決方法之研究. Master's thesis, 國立成功大學, 2003.
- [6] 江志祥. 液晶顯示器畫面產生閃爍現象之自動調整改善研究. Master's thesis, 逢甲大學, 2011.

- [7] 呂欣汶. 雲端醫療紀錄之巨量資料存取與處理平台建置. Master's thesis, 東海大學, 2014.
- [8] 黃元鴻. 基於 hadoop mapreduce 與 hbase 之醫療資訊快速分析平台. Master's thesis, 台灣大學, 2011.
- [9] 廖本加. 於雲端計算環境上高可用性儲存系統之實作. Master's thesis, 東海大學, July 2011.
- [10] 黃智霖. 實作一個雲端計算上具資源監控的分散式資料儲存系統. Master's thesis, 東海大學, July 2011.
- [11] Attila Kovári and P. Dukan. KVM & openvz virtualization based iaas open source cloud virtualization platforms: Opennode, proxmox VE. In *10th IEEE Jubilee International Symposium on Intelligent Systems and Informatics, SISY 2012, Subotica, Serbia, September 20-22, 2012*, pages 335–339, 2012.
- [12] Vinod Kumar Vavilapalli, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, Bikas Saha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin Reed, and Eric Baldeschwieler. Apache hadoop YARN: yet another resource negotiator. In *ACM Symposium on Cloud Computing, SOCC '13, Santa Clara, CA, USA, October 1-3, 2013*, page 5, 2013.
- [13] George Kousiouris, George Vafiadis, and Theodora A. Varvarigou. A front-end, hadoop-based data management service for efficient federated clouds. In *IEEE 3rd International Conference on Cloud Computing Technology and Science, CloudCom 2011, Athens, Greece, November 29 - December 1, 2011*, pages 511–516, 2011.
- [14] Hieu Hanh Le, Satoshi Hikida, and Haruo Yokota. An evaluation of power-proportional data placement for hadoop distributed file systems. In *IEEE*

- Ninth International Conference on Dependable, Autonomic and Secure Computing, DASC 2011, 12-14 December 2011, Sydney, Australia*, pages 752–759, 2011.
- [15] Zhiwei Xu, Bo Yan, and Yongqiang Zou. Beyond hadoop: Recent directions in data computing for internet services. *IJCAC*, 1(1):45–61, 2011.
- [16] Ronald C. Taylor. An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. *BMC Bioinformatics*, 11(S-12):S1, 2010.
- [17] Lizhi Cai, Shidong Huang, Leilei Chen, and Yang Zheng. Performance analysis and testing of hbase based on its architecture. In *2013 IEEE/ACIS 12th International Conference on Computer and Information Science, ICIS 2013, Niigata, Japan, June 16-20, 2013*, pages 353–358, 2013.
- [18] Rong Shean Lee, Ko Jen Mei, Peng Xu, and Chang Ming Wu. Open architecture of virtual machine tool for cloud computing. In *2014 IEEE International Conference on Automation Science and Engineering, CASE 2014, New Taipei, Taiwan, August 18-22, 2014*, pages 905–909, 2014.
- [19] 專題報導: 林欣柔. 雲端儲存 (一): 帶我上雲端! 迎接網路雲端儲存新時代 <http://scitechvista.most.gov.tw/zh-tw/Feature/C/0/13/10/1/543.htm>, 2013.
- [20] 林文彬. 機房革命虛擬化起飛 <http://www.ithome.com.tw/node/41169>, 2006.
- [21] Kai Hwang、Jack Dongarra、Geoffrey Fox. 雲端運算: 虛擬化類別 <http://technet.microsoft.com/zh-tw/magazine/hh802393.aspx>, 2012.
- [22] Welcome to apache™ hadoop®! <http://hadoop.apache.org/>, 2014.
- [23] Welcome to apache hbase™ <http://hbase.apache.org/>, 2014.
- [24] 邱莉燕、鄭婷方. Big data 大數據正在改變生活·創造新生意 <http://mag.nownews.com/article.php?mag=4-68-15984>, 2013.

- [25] 吳其勳. Big data 是怎麼一回事 <http://www.ithome.com.tw/node/76529>, 2012.
- [26] AD Partner. 廣告數據協助企業建立核心價值 <http://www.vmware.com/virtualization/virtualization-basics/how-virtualization-works>, 2014.
- [27] Apache hadoop nextgen mapreduce (yarn) <http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html>, 2014.
- [28] Dan Sullivan. Getting started with hadoop 2.0 <http://www.tomsitpro.com/articles/hadoop-2-vs-1,2-718.html>, 2014.
- [29] Zookeeper <http://zookeeper.apache.org/doc/r3.4.6/zookeeperOver.html>, 2014.
- [30] Chapter 2. apache hbase configuration <http://hbase.apache.org/book/configuration.html#Hadoop>, 2014.
- [31] 王宏仁. 徹底解讀 it 明日之星—雲端運算 <http://www.ithome.com.tw/node/49410>, 2008.

附錄 A

安裝前環境準備

1. 硬體規格與軟體版本

CPU: Intel Core i5-3210M 2.5GHz

RAM: 8 GB

HD: 750GB

Network: 100M/1000M bps Ethernet

OS: Windows 7 64-bit Home

Hadoop 版本：1.0.4 與 2.2.0

Hbase 版本：0.94 與 0.96

VM Platform: VMware Workstation 10.0.0

VM Guest OS: Ubuntu 12.04.3 Desktop 64-bit

VM RAM: 2GB

VM HD: 20GB

2. 本實驗須建立兩台虛擬伺服器 (VM):

一台 Namenode(Master): 主機名稱 nodea: 192.168.114.11

一台 Datanode(Slave): 主機名稱 nodeb: 192.168.114.12

3. 在 VMware 建置及設定虛擬機器

1. 開啟 VMware Workstation，新增虛擬機器 [File]->

[New Virtual Machine] 。

2. 選擇 Installer disc image file(iso) , 指定 ubuntu-12.04.1-desktop-amd64.iso 。
3. 輸入 Full name 、Username 和 Password 。
4. 輸入 Virtual machine name 和 LocationVM 所在位置 。
5. 設定 Maximum disk size(GB) 為 20GB , 選擇 Store virtual disk as single file 。
6. 按 [Customize Hardware...] 進入自定 Hardware 。
7. 設定 Memory 為 2048MB 。
8. 勾選 「Virtualize Intel VT-x/EPT or AMD-V/RVI」 和 「Virtualize CPU performance counters」 。
9. 網路連結設定為 NAT , 按 [Close] 結束設定。
Nodea IP Address=> 192.168.114.11:2201
Nodeb IP Address=> 192.168.114.12:2202
10. 準備安裝 , 按 [Finish] 繼續開始安裝 。

4. 安裝 VMware Tools

VM > Guest > Install/Upgrade VMware Tools

```
sudo apt-get -y install gcc
```

```
sudo su -
```

```
mount /dev/cdrom /mnt
```

```
cd /mnt
```

```
cp VMwareTools-9.0.5-1065307.tar.gz ~
```

```
cd ~
```

```
tar xzvf VMwareTools-9.0.5-1065307.tar.gz
```

```
cd vmware-tools-distrib
```

```
perl ./vmware-install.pl
```

附錄 B

Hadoop1.0.4 與 Hbase0.94 安裝步驟

1. 必要安裝的軟體套件及說明：

安裝 Hadoop 前需先安裝兩個套件，分別是 Java 及 ssh。

Hadoop 以 Java 撰寫而成，所以必需在 Java 的環境 (JRE) 下運作。

系統需安裝 Java 第六版 (或更新的版本)。

本次架設 nodea(Name node) 與 nodeb(Data node)。

2. 套件更新、升級以及安裝 SSH

```
sudo apt-get update – 套件更新
```

```
sudo apt-get -y upgrade – 套件升級
```

```
sudo apt-get -y install ssh – 安裝 SSH
```

3. 更改預設電腦名稱 (以下步驟建議使用 PieTTY 工具操作)

若安裝時, 未設定 nodea, nodeb, 電腦名稱為 ubuntu , 故須修改電腦名稱
指令如下:

```
sudo vi /etc/hostname
```


nodea, nodeb , 修改後輸入:wq 離開。

4. 更改 hosts

指令:sudo vi /etc/hosts

加入 192.168.114.11 nodea ,192.168.114.12 nodeb

輸入:wq 後離開, 並輸入 sudo reboot

重新開機

5. 安裝 jdk(安裝前請先重新開機 sudo reboot)

apt-cache search jdk – 尋找 jdk 套件

sudo apt-get -y install openjdk-6-jdk –開始安裝 jdk

6. 檢查 jdk 安裝

hadoop@nodea: *which java*

/usr/bin/java

hadoop@nodea : java -version

java version "1.6.0_24"

OpenJDK Runtime Environment (IcedTea6 1.11.5) (6b24-1.11.5-0ubuntu1 12.04.1)

OpenJDK 64-Bit Server VM (build 20.0-b12, mixed mode)

7. 設定 SSH 免密碼登入

指令:ssh-keygen -t dsa -P "" -f ~/.ssh/id_dsa – 產生 ssh key, 建議用 dsa ,

複製 key 檔案, 輸入命令: cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

PS: 至此已經完成 nodea 基本 vm 設定, 要先將 nodea 關機並複製出 nodeb 關機後進行複製, 分別複製出 nodeb, 並重新修改步驟 2 與步驟 3。

authorized_keys – 複製檔案

指令:

```
cat ~/.ssh/id_dsa.pub » ~/.ssh/authorized_keys - 複製檔案  
複製到 nodea 與 nodeb
```

```
ssh-copy-id -i ~/.ssh/id_dsa.pub hadoop@nodea
```

```
ssh-copy-id -i ~/.ssh/id_dsa.pub hadoop@nodeb
```

8. 檢查 SSH 免密碼登入設定結果

```
指令:ssh localhost
```

```
指令:ssh nodea/nodeb
```

9. 安裝與設定 Hadoop HDFS(做 nodea 就好，其他節點使用檔案複製方式)

```
wget http://ftp.twaren.net/Unix/Web/apache/hadoop/common/stable/  
hadoop-1.0.4.tar.gz
```

```
解壓縮指令:tar zxvf hadoop-1.0.4.tar.gz
```

10. 以下設定環境及組態檔案：hadoop-env.sh 文件底下增加內容

```
指令: vi ~/hadoop-1.0.4/conf/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk-amd64
```

```
export HADOOP_PID_DIR=/home/hadoop/hadoop-1.0.4/pids
```

11. 設定 core-site.xml 檔案

```
指令: vi /hadoop-1.0.4/conf/core-site.xml
```

```
<configuration>
```

```
  <property>
```

```
    <name>fs.default.name</name>
```

```
    <value>hdfs://nodea:9000</value>
```

```
  </property>
```

```
</configuration>
```

12. 設定 hdfs-site.xml 檔案

指令: vi /hadoop-1.0.4/conf/hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.name.dir</name>
    <value>/home/hadoop/data/namenode</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/data/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

13. 設定 mapred-site.xml 檔案

指令: vi /hadoop-1.0.4/conf/mapred-site.xml

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>nodea:9001</value>
  </property>
</configuration>
```

14. 設定 masters 檔案

指令: vi /hadoop-1.0.4/conf/masters

nodea

15. 設定 slaves 檔案

指令: vi /hadoop-1.0.4/conf/slaves
nodeb

16. 將設定好的 Hadoop 檔案複製到其他節點

指令: scp -r ~/hadoop-1.0.4 hadoop@nodeb: ~
指令: scp -r ~/hadoop-1.0.4 hadoop@nodec: ~

17. 格式化

指令: ~/hadoop-1.0.4/bin/hadoop namenode -format

18. 啟動 Hadoop

指令: ~/hadoop-1.0.4/bin/start-all.sh

19. 檢查 Hadoop

hadoop@nodea: ~ \$ jps
hadoop@nodeb:/\$ jps

20. 關閉 Hadoop

指令: ~/hadoop-1.1.2/bin/stop-all.sh

21. 觀看 Hadoop 狀態

http://192.168.114.11:50070/ – NameNode
http://192.168.114.12:50030/ – JobTracker

22. 安裝 Hbase

下載 HBase

首先到 HBase 的官網 (<http://hbase.apache.org/>) 下載 HBase 壓縮檔
目前可下載最新版 hbase-0. 0.94.7.tar.gz，解壓縮後放到目錄/opt/內並
更名為 hbase，最後進入 HBase 的目錄中

23. 將下列內容加入到 conf/hbase-env.sh 裡

```
export JAVA_HOME=/usr/lib/jvm/java-6-sun-1.6.0.06
export HBASE_MANAGES_ZK=true
export HBASE_LOG_DIR=/tmp/hadoop/hbase-logs
```

24. 編輯 conf/hbase-site.xml 內容，並設定 HBase 的一些相關參數

指令: vi conf/hbase-site.xml

並將下列內容加入到 conf/hbase-site.xml 裡：

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://nodea:9000/hbase</value>
  </property>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
  <property>
    <name>hbase.zookeeper.property.clientPort</name>
    <value>2222</value>
  </property>
  <property>
    <name>hbase.zookeeper.quorum</name>
```

```
        <value>nodna,nodeb</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
    </property>
    <property>
        <name>hbase.tmp.dir</name>
        <value>/var/hadoop/hbase-$user.name</value>
    </property>
    <property>
        <name>hbase.master </name>
        <value>nodea:60000</value>
    </property>
</configuration>
```

25. 接著利用 vi 編輯 conf/regionservers

指令: vi conf/regionservers

nodea

此範例只有一台 Slave，所以在 slaves 檔中只有一行該 Slave 的主機名稱。

nodeb

26. 複製 Hadoop 的設定檔至 HBase 的目錄 conf/ 中

```
cp /opt/hadoop/conf/core-site.xml conf/
```

```
cp /opt/hadoop/conf/mapred-site.xml conf/
```

```
cp /opt/hadoop/conf/hdfs-site.xml conf/
```

複製 hadoop 目錄中的 hadoop-1.0.4-core.jar 到 hbase 的目錄 lib 內作為替換。

27. 將目錄 hbase 複製到其它 Slave 節點

指令: `scp -r /opt/hbase hadoop@nodeb:/opt/hbase`

28. 啟動 Hbase(需先啟動 Hadoop...)

指令: `bin/start-hbase.sh`

檢查其他主機是否有正常啟動

nodea 之 JPS(正常啟動後的 Master)

nodeb 之 JPS(正常啟動後的 slave)

29 Shell Mode

指令: `./bin/hbase shell`

透過執行 `hbase shell` 指令，進入 HBase 控制台後輸入 `list`，若可正常執行，則表示 HBase 安裝完成。

`hbase(main):001:0> list` ← 輸入 list 並按下 Enter

TABLE

0 row(s) in 0.3950 seconds

`hbase(main):002:0>`

30. 檢查網頁介面

HBase 也提供了網頁介面供管理者監控

查看 Master 狀態

在 Master 節點上的瀏覽器網址輸入 `http://localhost:60010/`

查看 Region Server 狀態

在 Slave 節點上的瀏覽器網址輸入 `http://localhost:60030/`

查看 ZooKeeper

在 Master 節點上的瀏覽器網址輸入 `http://localhost:60010/zk.jsp`

31. 停止服務

指令:./bin/stop-hbase.sh



附錄 C

Hadoop2.2.0 與 Hbase0.96 安裝步驟

1. 必要安裝的軟體套件及說明：

- 安裝 Hadoop 前需先安裝兩個套件，分別是 Java 及 ssh。
- Hadoop 以 Java 撰寫而成，所以必需在 Java 的環境 (JRE) 下運作。
- 系統需安裝 Java 第六版 (或更新的版本)。
- 本次架設 nodea(Name node) 與 nodeb(Data node)。

2. 套件更新、升級以及安裝 SSH

```
sudo apt-get update – 套件更新
```

```
sudo apt-get -y upgrade – 套件升級
```

```
sudo apt-get -y install ssh – 安裝 SSH
```

3. 更改預設電腦名稱 (以下步驟建議使用 PieTTY 工具操作)

若安裝時, 未設定 nodea, nodeb, 電腦名稱為 ubuntu , 故須修改電腦名稱
指令如下:

```
sudo vi /etc/hostname
```

nodea, nodeb , 修改後輸入:wq 離開。

4. 更改 hosts

指令:sudo vi /etc/hosts

加入 192.168.114.11 nodea ,192.168.114.12 nodeb

輸入:wq 後離開, 並輸入 sudo reboot

重新開機

5. 安裝 jdk(安裝前請先重新開機 sudo reboot)

apt-cache search jdk – 尋找 jdk 套件

sudo apt-get -y install openjdk-6-jdk –開始安裝 jdk

6. 檢查 jdk 安裝

hadoop@nodea: *which java*

/usr/bin/java

hadoop@nodea : java -version

java version "1.6.0_24"

OpenJDK Runtime Environment (IcedTea6 1.11.5) (6b24-1.11.5-0ubuntu1 12.04.1)

OpenJDK 64-Bit Server VM (build 20.0-b12, mixed mode)

7. 設定 SSH 免密碼登入

指令:ssh-keygen -t dsa -P "" -f ~/.ssh/id_dsa – 產生 ssh key, 建議用 dsa ,

複製 key 檔案, 輸入命令: cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

PS: 至此已經完成 nodea 基本 vm 設定, 要先將 nodea 關機並複製出 nodeb

關機後進行複製, 分別複製出 nodeb, 並重新修改步驟 2 與步驟 3。

authorized_keys – 複製檔案

指令:

```
cat ~/.ssh/id_dsa.pub » ~/.ssh/authorized_keys - 複製檔案  
複製到 nodea 與 nodeb  
ssh-copy-id -i ~/.ssh/id_dsa.pub hadoop@nodea  
ssh-copy-id -i ~/.ssh/id_dsa.pub hadoop@nodeb
```

8. 檢查 SSH 免密碼登入設定結果

```
指令:ssh localhost  
指令:ssh nodea/nodeb
```

9. 安裝與設定 Hadoop HDFS(做 nodea 就好，其他節點使用檔案複製方式)

```
wget http://ftp.twaren.net/Unix/Web/apache/hadoop/common/  
hadoop-2.2.0.tar.gz  
解壓縮指令:tar zxvf hadoop-2.2.0.tar.gz  
輸入命令: mv hadoop-2.2.0 hadoop  
(視窗介面可右鍵 rename:hadoop-2.2.0 改名為 hadoop)
```

10. 新增環境變數

```
輸入命令:sudo vi .bashrc  
export JAVA_HOME=/usr/lib/jvm/jdk/  
export HADOOP_INSTALL=/home/hadoop/hadoop  
export PATH=$PATH:HADOOP_INSTALL/bin  
export PATH = $PATH :HADOOP_INSTALL/sbin  
export HADOOP_MAPRED_HOME=HADOOP_INSTALL  
export HADOOP_COMMON_HOME =HADOOP_INSTALL  
export HADOOP_HDFS_HOME=HADOOP_INSTALL  
export YARN_HOME =HADOOP_INSTALL
```

11. 設定 `hadoop-env.sh`，註解 `JAVA_HOME`。

輸入命令:`sudo vi /hadoop/etc/hadoop/hadoop-env.sh`

`export JAVA_HOME=/usr/lib/jvm/jdk`

視窗環境可點選檔案，右鍵使用 `Open With Text Editor` 來做修改並 `save`。

12. 設定 `core-site.xml` 檔案

輸入命令:`sudo vi /hadoop/etc/hadoop/core-site.xml`

```
< configuration>
  < property>
    < name>fs.default.name</ name>
    < value>hdfs://nodea:9000</ value>
  </ property>
</ configuration>
```

視窗環境可點選檔案右鍵，使用 `Open With Text Editor` 來做修改並 `save`。

13. 設定 `hdfs-site.xml` 檔案

輸入命令: `mkdir -p /hadoop/mydata/hdfs/namenode`

輸入命令: `mkdir -p /hadoop/mydata/hdfs/datanode`

輸入命令: `sudo vi /hadoop/etc/hadoop/hdfs-site.xml`

```
< configuration>
  < property>
    < name>dfs.name.dir</ name>
    < value>/home/hadoop/mydata/hdfs/namenode</ value>
  </ property>
  < property>
    < name>dfs.data.dir</ name>
    < value>/home/hadoop/mydata/hdfs/datanode</ value>
  </ property>
  < property>
```

```
<name>dfs.replication</name>
<value>1</value>
</property>
< /configuration>
```

視窗環境可點選檔案右鍵, 使用 Open With Text Editor 來做修改並 save。

14. 設定 yarn-site.xml

輸入命令: `sudo vi /hadoop/etc/hadoop/yarn-site.xml`

```
< configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>nodea</value>
  </property>
< /configuration>
```

視窗環境可點選檔案右鍵, 使用 Open With Text Editor 來做修改並 save。

15. 設定 mapred-site.xml 檔案

輸入命令: `cp /hadoop/etc/hadoop/mapred-site.xml.template /hadoop/etc/hadoop/mapred-site.xml`

輸入命令: `sudo vi /hadoop/etc/hadoop/mapred-site.xml`

```
< configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
```

```
<property>
    <name>mapred.job.tracker</name>
    <value>nodea:9001</value>
</property>
< /configuration>
```

視窗環境可點選檔案右鍵，使用 Open With Text Editor 來做修改並 save。

16. 設定 masters 檔案

輸入命令: `sudo vi /hadoop/etc/hadoop/masters`

內容輸入: nodea

視窗環境可點選檔案右鍵，使用 Open With Text Editor 來做修改並 save。

17. 設定 slaves 檔案

輸入命令: `sudo vi /hadoop/etc/hadoop/slaves`

內容輸入: nodeb

或是如圖所示: 點選檔案, 右鍵使用 Open With Text Editor 來做修改並 save。

18. 將設定好的 Hadoop 複製到 nodeb

輸入命令: `scp -r /hadoop nodeb:/home/hadoop`

19. hdfs 格式化

輸入命令: `/hadoop/bin/hdfs namenode -format`

(重要: 若再重做此項動作, 必須先把各端的 `/home/hadoop/mydata` 刪除始可格式化。)

20. 啟動 Hadoop (在 nodea 端)。

輸入命令: `/hadoop/sbin/start-all.sh`

21. 檢查 Hadoop

nodea 輸入命令: `jps`

nodeb 輸入命令: `jps`

22. 停止 Hadoop (在 nodea 端)。

輸入命令: `/hadoop/sbin/stop-all.sh`

23. 觀看 Hadoop 狀態

`http://192.168.114.11:50070/` – NameNode

24. 安裝 Hbase

下載 HBase

首先到 HBase 的官網 (<http://hbase.apache.org/>) 下載 HBase 壓縮檔
目前可下載最新版 `hbase-0.9.4.7.tar.gz`，解壓縮後放到目錄 `/opt/` 內並
更名為 `hbase`，最後進入 HBase 的目錄中

25. 將下列內容加入到 `conf/hbase-env.sh` 裡

```
export JAVA_HOME=/usr/lib/jvm/java-1.6.0-openjdk-amd64
```

```
export HBASE_MANAGES_ZK=true
```

26. 編輯 `conf/hbase-site.xml` 內容，並設定 HBase 的一些相關參數

指令: `vi conf/hbase-site.xml`

並將下列內容加入到 `conf/hbase-site.xml` 裡：

```
<configuration>
```

```
  <property>
```

```
        <name>hbase.master</name>
        <value>192.168.114.11:60000</value>
    </property>
    <property>
        <name>hbase.rootdir</name>
        <value>hdfs://192.168.114.11:9000/hbase</value>
    </property>
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.clientPort</name>
        <value>2181</value>
    </property>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>192.168.114.11,192.168.114.12</value>
    </property>
</configuration>
```

27. 接著利用 vi 編輯 conf/regionservers

指令: vi conf/regionservers

nodea

此範例只有一台 Slave，所以在 slaves 檔中只有一行該 Slave 的主機名稱。

nodeb

28. 替換 jar 檔案

將 hadoop 下方三個檔案

hadoop-common-2.1.0-beta.jar

hadoop-hdfs-2.1.0-beta.jar

hadoop-auth-2.1.0-beta.jar

copy 到 Hbase 的 lib 資料夾中

hadoop-common-2.2.0.jar

hadoop-hdfs-2.2.0.jar

hadoop-auth-2.2.0.jar

29. 將目錄 hbase 複製到其它 Slave 節點

指令: `scp -r /opt/hbase hadoop@nodeb:/opt/hbase`

30. 啟動 Hbase(需先啟動 Hadoop...)

指令: `bin/start-hbase.sh`

檢查其他主機是否有正常啟動

nodea 之 JPS(正常啟動後的 Master)

nodeb 之 JPS(正常啟動後的 slave)

31 Shell Mode

指令: `./bin/hbase shell`

透過執行 `hbase shell` 指令，進入 HBase 控制台後輸入 `list`，若可正常執行，則表示 HBase 安裝完成。

hbase(main):001:0> list ← 輸入 list 並按下 Enter

TABLE

0 row(s) in 0.3950 seconds

hbase(main):002:0>

32. 檢查網頁介面

HBase 也提供了網頁介面供管理者監控

查看 Master 狀態

在 Master 節點上的瀏覽器網址輸入 <http://localhost:60010/>

查看 Region Server 狀態

在 Slave 節點上的瀏覽器網址輸入 <http://localhost:60030/>

查看 ZooKeeper

在 Master 節點上的瀏覽器網址輸入 <http://localhost:60010/zk.jsp>

33. 停止服務

指令: `./bin/stop-hbase.sh`