

銘傳大學

資訊傳播工程學系
專題研究總審報告

動態節能省電的雲端系統研究

指導教授：葉生正 博士

專研學生：邱盛宇 徐一中 劉宇捷 黃紹維

二 ○ 一 五 年 一 月 十 六 日

動態節能省電的雲端系統研究

摘要

隨著日新月異的科技更迭，硬體發展上日益進步。對於各大公司上來說，除需要大量的電腦運算支援工作外，還須考慮到硬體的汰舊換新。此時，為了解決這方面的問題，日前 Apache 已針對這方面的問題建立了一套雲端運算軟體——Hadoop。在現有的硬體在還能運算的狀況下還要不浪費的善加利用，使用 Hadoop 的技術平行化處理問題，化繁為簡，是最重要的目的。

相對的為了這方面的需求，企業們也逐步接納並採用此策略。相對而言，必須考慮的層面，就已經深入化。在一系列的平行運算中，總是不可能都需要用到所有的硬體做運算以解決問題。這時，節能省電與動態節能成為了重要的議題。本研究就這方面深入解決。

研究結果希望達到，在不需人為控制的環境下，由本系統能自動判斷測出新進與舊有的工作所需使用的運算量。自動控制並開啟所需使用的電腦數量已達到有效運算與節能省電，並可直接於本系統管理網頁上看到相關節能的報表資訊。另外，針對覺得需要自行控制的使用者，也在網頁中提供手動開關與控制的功能以便使用。

關鍵字：Hadoop、雲端、省電、動態節能

目錄

摘要.....	i
目錄.....	ii
表目錄.....	vi
圖目錄.....	vii
第一章 研究動機與目的	1
1.1 概述.....	1
1.2 研究動機	2
1.3 研究目的	3
第二章 相關文獻與技術探討	4
2.1 雲端運算	4
2.1.1 雲端運算概述	4
2.1.2 雲端運算的服務模式	5
2.1.3 雲端運算的部署模型	6
2.1.4 雲端運算的基本特徵	7
2.2 虛擬化技術	9
2.2.1 虛擬化技術概述	9

2.2.2	常見虛擬化軟體	9
2.3	Linux 簡介	13
2.3.1	Debian 系列	13
2.3.2	Red Hat 系列	14
2.3.3	Gentoo 系列	14
2.3.4	Slackware 系列	14
2.3.5	Arch Linux 系列	14
2.3.6	其它	14
2.4	Hadoop 簡介	15
2.4.1	Hadoop Distributed File System (HDFS)	16
2.4.2	MapReduce	19
2.4.3	蒙地卡羅方法	22
2.4.4	Thrift	23
2.4.5	HBase	23
2.5	LAMP 簡介	24
2.5.1	Apache	24
2.5.2	MySQL	24
2.5.3	PHP	24
2.6	雲端節能	25

第三章	雲端系統設計	26
3.1	系統架構	26
3.2	研究方法與步驟	27
3.2.1	研究方法	27
3.2.2	研究步驟	32
第四章	研究成果	45
4.1	基礎研究成果	45
4.2	動態之決策	48
4.3	網頁控制系統	50
4.3.1	Hadoop 基本控制層面	51
4.3.2	節點控制層面	53
4.3.3	節點狀態檢查層面	54
4.3.4	使用者介面層面	56
4.4	動態配置與比較	58
4.4.1	開關機與待機的差異	58
4.4.2	預設與動態配置計算之差異	59
4.4.3	動態配置之展示	68

第五章	結論與未來工作	70
附錄.....		75
工作分配表.....		75
工作時程表.....		76

表目錄

表 二.1 NAMENODE 介紹	16
表 二.2 DATANODE 介紹	17
表 二.3 MAPREDUCE 運作的情形	20
表 三.1 HADOOP 完整安裝與架設	36
表 四.1 雲端運算時的消耗功率。(單位：W)	46
表 四.2 雲端運算時的耗時。(單位：秒)	47
表 四.3 雲端運算的耗能。(單位：J)	47
表 四.4 運算節點比較表(CPU 使用率%)	48
表 四.5 DATANODE-1 之省電幅度測試(CPU 使用率%)	49
表 四.6 待機與開關機比較表(W)	58
表 四.7 低負載運算之省電比較表	59
表 四.8 高負載運算之省電比較表	61
表 四.9 高負載運算之省電比較表 2	63
表 四.10 平衡運算之省電比較表	65
表 四.11 高負載模擬之動態釋義表	68

圖目錄

圖 二.1 雲端的架構(圖片來源: NIST[11])	8
圖 二.2 VMWARE 軟體畫面(圖片來源:VMWARE).....	10
圖 二.3 VIRTUALBOX 軟體畫面(圖片來源:ORACLE)	11
圖 二.4 HYPER-V 軟體畫面(圖片來源:MICROSOFT).....	12
圖 二.5 待命中的 HADOOP 示意圖	15
圖 二.6 檔案備份分配圖	18
圖 二.7 MAPREDUCE 運行架構.....	21
圖 二.8 蒙地卡羅架構	22
圖 二.9 負載平衡 VS 合併運算	25
圖 三.1 系統架構圖	26
圖 三.2 待命中的 HADOOP 工作示意圖	28
圖 三.3 集中主機運算	29
圖 三.4 分配主機運算	30
圖 三.5 集中分配主機運算	31
圖 三.6 LINUX UBUNTU(圖片來源:UBUNTU).....	32
圖 三.7 安裝畫面一	33
圖 三.8 安裝畫面二	33

圖 三.9 安裝 TASKSEL	34
圖 三.10 設定套件	34
圖 三.11 MySQL 根密碼設定	35
圖 三.12 SSH 金鑰建立	37
圖 三.13 安裝 JAVA.....	37
圖 三.14 安裝 HADOOP	38
圖 三.15 HADOOP-ENV.SH.....	38
圖 三.16 MAPRED-SITE.XML	39
圖 三.17 HDFS-SITE.XML	39
圖 三.18 CORE-SITE.XML.....	40
圖 三.19 刪除 HADOOP 暫存檔	40
圖 三.20 格式化開始	41
圖 三.21 啟動 HADOOP	41
圖 三.22 檢視狀態	41
圖 三.23 執行範例檔	42
圖 三.24 運算過程與結果	42
圖 三.25 結束 HADOOP	43
圖 三.26 HADOOP-50030	43

圖 三.27 HADOOP-50070	44
圖 四.1 控制系統首頁	50
圖 四.2 系統功能選項	50
圖 四.3 啟動 HADOOP 運算	51
圖 四.4 PI 10 10 運算	52
圖 四.5 PI 100 100 運算	52
圖 四.6 PI 1000 1000 運算	53
圖 四.7 單一電腦 CPU 使用率	54
圖 四.8 所有電腦 CPU 使用率	55
圖 四.9 狀態檢查-1	55
圖 四.10 狀態檢查-2	56
圖 四.11 電腦端使用介面	57
圖 四.12 平板端使用介面	57
圖 四.13 手機端使用介面	57
圖 四.14 低負載運算之省電比較	60
圖 四.15 高負載之省電比較	62
圖 四.16 高負載之省電比較	64
圖 四.17 平衡運算之省電比較	66

圖 四.18 高負載模擬之動態釋義圖	69
--------------------------	----

第一章 研究動機與目的

1.1 概述

本研究的目的是在於減少雲端系統的電力消耗。研究中將利用 Hadoop 建構一個雲端系統來模擬企業所使用的雲端環境，以不同規格、CPU 運算能力不同的電腦模擬企業中不同時期購進的主機，並藉由 Hadoop 內建程式擬蒙特卡羅法計算 π 來模擬企業的雲端運算 [1] [2] [3]。

本研究擬在雲端架構之外，用 PHP 建設一個使用者介面來管理雲端系統，包括使用者手動調節和系統動態調節 [4]。在系統動態調節的部分，當系統負載較低時，因為運算能力高的主機足以負擔整個雲端運算，因此系統會關閉閒置的主機，只保留運算能力較高的主機進行運算，而當系統負載較高時，運算能力高的主機對於整體的運算效率提升有限，因此關閉運算能力較高的主機，只保留一般的主機，以達到節省電力的效果 [5]。

1.2 研究動機

在現今的世代中，雲端作為一種新技術，因為其便利、低成本、易維護的特點，逐漸地發展壯大，無論是個人或企業，都不缺乏雲端的愛用者。

雲端的使用上，最常見的應用不外乎雲端運算、雲端硬碟與雲端掃描 [6] [7]。這些雲端上的運用最大的共通點在於，無論何時、何地，只要有網路就可執行，然而隨著雲端的發展，讓我們發現了一個問題。在這便利性的背後，付出了一些代價，如：整個雲端環境除了維護時的伺服器關機外，基本上是不關閉的，也唯有如此，才能保證整個服務的即時性，而其造成的便是大量的電力資源損耗。

如何解決電力資源的過度損耗同時又能夠持續保證服務的即時性便促成了我們研究的動機。

1.3 研究目的

為了解決電力資源的過度損耗，在研究中找出了一些突破點。誠然，以 Google 公司為例，雲端產業是其企業的重要收入來源之一，也因此雲端系統是每時每刻在使用，基本上除了整體架構配備的發展更新，難以有介入優化的餘地 [8]。然而由於此次研究的目標在於一般企業，雲端對他們而言不是對外的收入來源，而是增進其工作效率的工具；相對而言，雲端就不會時常處於運算階段，會有其待機的時間，而這待機時間便是本次研究中所找出減少電力資源消耗的突破點。

在計畫中，伺服器待機時，各主機將進入睡眠狀態，並在獲得工作指令時才快速開機，除此之外，在運算進行時，系統會對整個運算進行評估，當系統負載較低時，只保留運算能力高的主機，其餘主機一律進入睡眠狀態；當系統負載較高時，因運算能力高的主機對整體的運算效率幫助不大，然而耗能卻會增加不少，因此讓運算能力高的主機進入睡眠狀態，保留其餘主機。以達到節省電力資源的耗損，而本研究會把本文中的主機從關閉狀態改為睡眠狀態以達成電力資源消耗的最小化。

第二章 相關文獻與技術探討

2.1 雲端運算

2.1.1 雲端運算概述

雲端運算是一種新興的網路服務，無論在何時何地，只要擁有電腦及網路即可使用。而其也有許多的應用型態，平行運算處理、雲端硬碟以及雲端掃毒等，都是常見的雲端運算應用型態。

其特點在於，使用者端基本上不用安裝龐大的應用程式。只需要安裝小小的使用者介面，甚至只要有一個瀏覽器，就可以方便的使用安裝在雲端上的應用程式。

為了其方便性，對於使用者的設備要求並不高，只是作為一個使用媒介，所以將漸漸地取代現有的設備，甚至出現只擁有最基本的應用程式，其他一切仰賴雲端的特殊電腦產品，如：Chrome Book [9]。

2.1.2 雲端運算的服務模式

以下為 NIST 對雲端運算的定義中確立的三種服務模式 [10] [11]。

(1) 軟體即服務(SaaS)：

消費者只使用應用程式，但不掌控網路、硬體、系統等基礎架構，軟體商以租賃而非賣出的方式進行服務。通常是藉由給予一組 帳號密碼進行服務。

(2) 平台即服務(PaaS)：

消費者以主機使用應用程式，掌控應用程式執行環境與部分主機，但不掌控網路、硬體、系統等基礎架構。

(3) 基礎架構即服務(IaaS)：

消費者使用基礎運算資源，即處理能力、儲存空間等，能掌控系統、網路、儲存空間，但不能掌控雲端基礎架構。

2.1.3 雲端運算的部署模型

以下為 NIST 對雲端運算的定義中確立的四種部署模型 [10] [11]。

(1) 公用雲

公用雲可以透過網路及第三方服務供應者提供客戶使用，通常會對使用者實施使用存取控制機制。

(2) 私有雲

私有雲除了具備公用雲的諸多優點，如彈性、適合提供服務外，因為資料與程式都在組織裡管理，較不會有安全上的疑慮以及頻寬、法規上的限制，讓使用者與服務提供者更能掌控雲端基礎架構。

另外，近年來掀起討論的自建雲(BYOC, Build You Own Cloud) 亦是屬於私有雲的部分 [12]。概念上，因認定公有雲有安全危險的使用者，在有能力支援後，自行建立的私人雲端平台。

(3) 社群雲

社群雲為眾多利益相同的組織聯合控制及使用，社群成員共用雲端程式及資料。

(4) 混合雲

混合雲結合共用雲與私有雲，使用者通常將非企業關鍵資訊外包，並在公用雲上處理，在此同時又可以掌控企業關鍵服務與資訊。

2.1.4 雲端運算的基本特徵

以下為 NIST 對雲端運算的定義中確立的五種基本特徵。

(1) 自助式隨需服務(On-demand self-service)：

使用者可以依照個人需求提取計算資源(例如:伺服器、儲存空間…等)，整個過程是單方面自動化的，不須與資源提供者互動。

(2) 廣泛網路接取(Broad network access)：

服務本質上經由網路提供，有標準機制讓不同的使用者平台(如平板電腦、智慧型手機或筆記型電腦等)都能使用。

(3) 共享資源池(Resource pooling)：

服務商提供的計算資源(例如：儲存空間、網路頻寬、計算能力、虛擬機器數量…等)可比擬為一個大水池，隨時依照需要來(重新)分配給不同平台的多個使用者。而使用者不需了解資源的實體位置，只要知道大概位置即可(好比資源是在某國家或是某資料中心)。

(4) 快速的彈性(Rapid elasticity)：

計算資源可以快速且有彈性地被提供或釋放，相對於使用者而言，資源是取之不盡且可以依想要的量購買。

(5) 可量測的服務(Measured service)：

資源的計算依其提供的服務特性被自動控管與最佳化。提供與使用

者雙方都可透明地監控資源使用與變動的情形。

由下圖 二.1 可明確表現出上述雲端運算的服務模式、部署模型與基本特徵的架構。

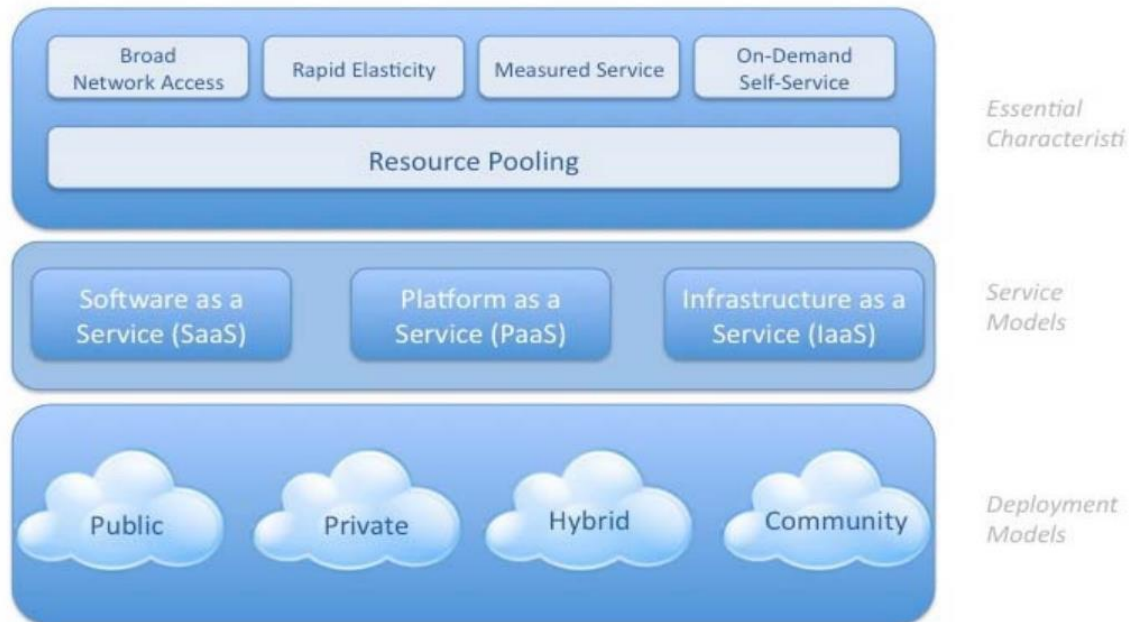


圖 二.1 雲端的架構(圖片來源: NIST [11])

2.2 虛擬化技術

2.2.1 虛擬化技術概述

指將電腦的物理資源，如記憶體、伺服器、網路等做一個抽象的轉換後呈現出來，能提供給使用者比原本更好的組態運用這些資源。而這些新的虛擬化資源不會受到原本的電腦物理組態、地域的限制 [13]。

2.2.2 常見虛擬化軟體

(1) VMware

VMware 公司是全球著名的虛擬機軟體公司，目前為 EMC 公司的全資子公司 [14]。其允許使用者同時建立和執行多個 x86 或 x64 的虛擬機器。個別虛擬機器可以執行自己的客戶端作業系統，目前支援系統以 Windows 與 Linux 系統為主 [15] [16]，如下圖 二.2。

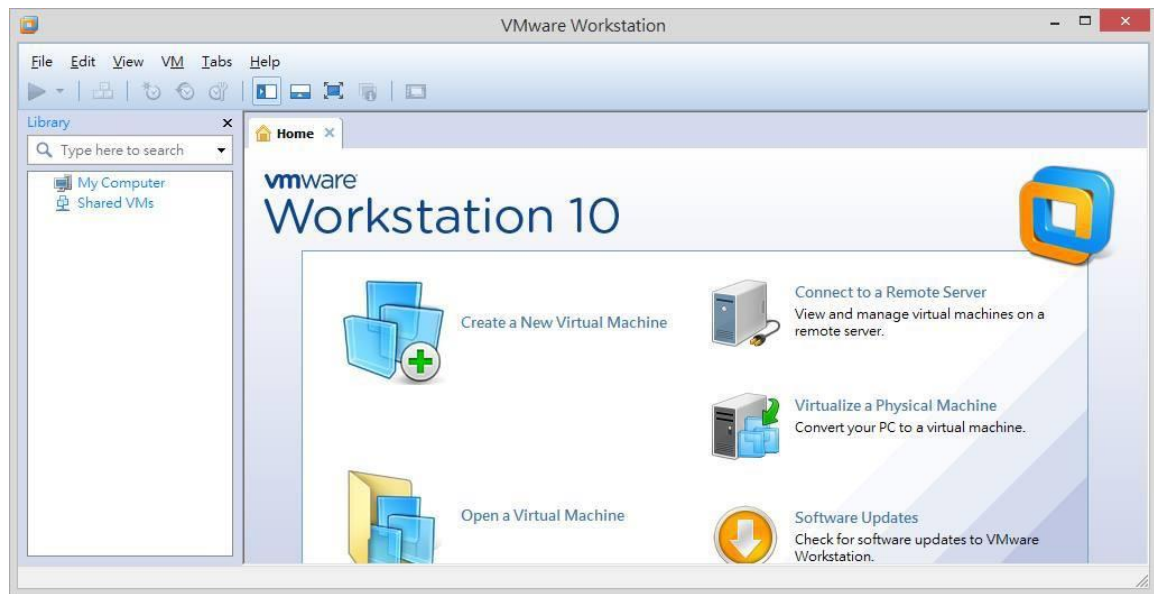


圖 二.2 VMware 軟體畫面(圖片來源:VMware)

(2) VirtualBox

Oracle VirtualBox 是由德國 InnoTek [17]軟體公司出品的虛擬機器軟體，現在則由甲骨文公司進行開發，是甲骨文公司 xVM 虛擬化平臺技術的一部份 [17]。它提供使用者在 32 位元或 64 位元的 Windows 、 Solaris 及 Linux 作業系統上虛擬其它 x86 的作業系統。使用者可以在 VirtualBox 上安裝並且執行 Solaris 、 Windows 、 DOS 、 Linux 、 OS/2 Warp 、 OpenBSD 及 FreeBSD 等系統作為客戶端作業系統，如下圖 二.3 。



圖 二.3 VirtualBox 軟體畫面(圖片來源:Oracle)

(3) Hyper-V

微軟所開發的 Hyper-V 提供了基礎結構可虛擬化應用程式和工作負載，有以下功能：建立或擴充私人雲端環境、提升硬體使用率、提升業務連續性、建立或擴充虛擬桌面基礎結構(VDI)與提升開發和測試活動的效率 [18]。

其中，提升業務連續性是將排程工作的新增與解除所導致的負載停機時間影響降至最低。而提升開發和測試活動的效率是指能使用其虛擬機器重現不同的運算環境，且不用取得或維護您原先可能需要的所有硬體，如圖 二.4 。

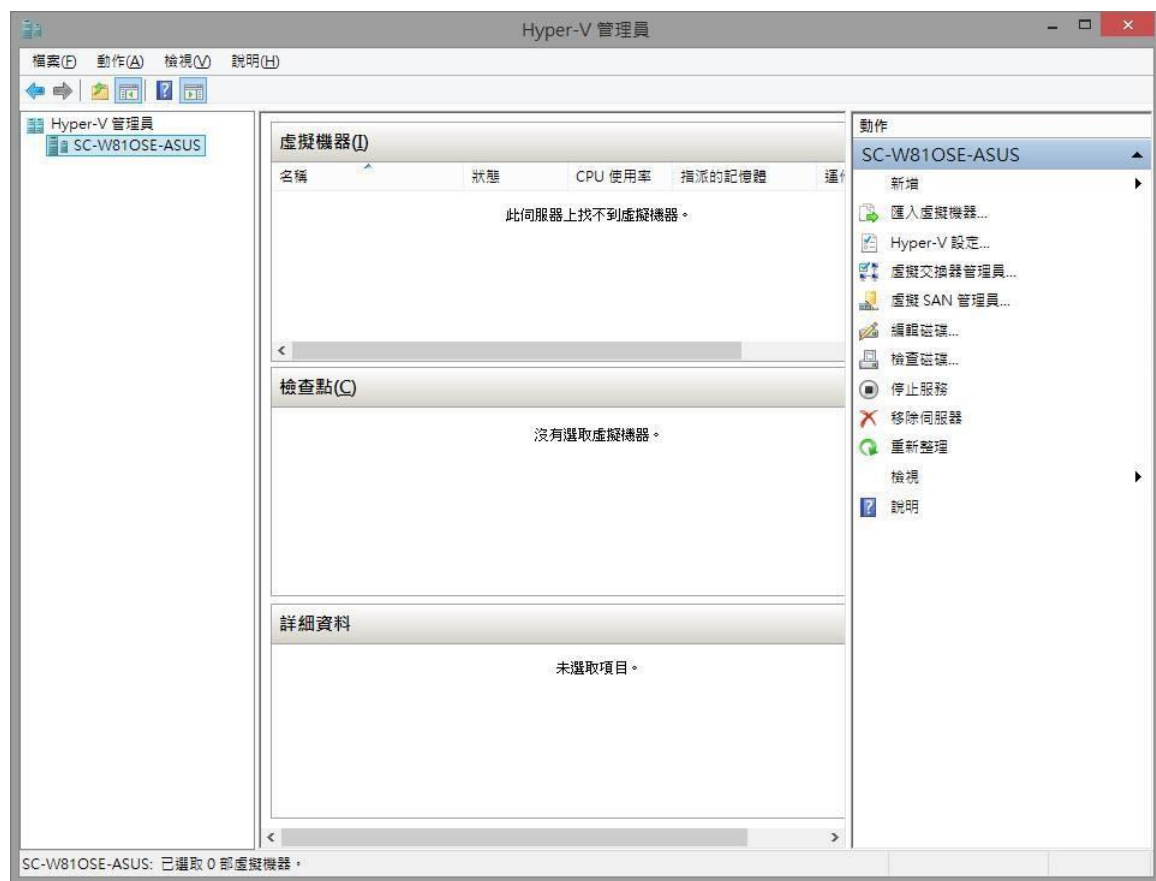


圖 二.4 Hyper-V 軟體畫面(圖片來源:Microsoft)

2.3 Linux 簡介

Linux 是一種著名的自由和開放原始碼類 Unix 操作系統，最初該作業系統的核心是由林納斯·托瓦茲開發 [19][20]。基本上使用者只要遵循 GNU 通用公共許可證，任何使用者或機構都可以自由的使用、修改、再發行。

相對而言，如今的 Linux 系統已有眾多分支與演繹。大多為以核心相同的架構，對於不同的需求與不同的套件使用的想法打包出不同版本與風格的 Linux，細項的說明如下：

2.3.1 Debian 系列

有 Debian、Knoppix、MEPIS、Xandros、CrunchBang Linux 與其中還分支出來的，基於 Ubuntu。此也是本研究的重點。此細分支中，有 Ubuntu、Kubuntu、Elementary OS、Edubuntu、Lubuntu、Xubuntu、Ubuntu GNOME、Linux Mint、Linux Deepin、PUD_GNU/Linux、StartOS、Trisquel 與 ackTrack。而本研究所使用的平台為 Ubuntu 系統。

2.3.2 Red Hat 系列

有 Red Hat Enterprise Linux、Fedora、CentOS、Scientific Linux、Fermi Linux、Oracle Linux、Qomo Linux、Red Flag Linux 與其中還分支出來的，基於 Mandriva 的分支。有 Mandriva Linux、PCLinuxOS、Unity Linux 與 Mageia。

2.3.3 Gentoo 系列

Gentoo Linux、Sabayon Linux、Calculate Linux、Funtoo Linux、Chromium OS 與 Google Chrome OS。

2.3.4 Slackware 系列

Slackware、Zenwalk、VectorLinux 與 SLAX。

2.3.5 Arch Linux 系列

Arch Linux、Chakra GNU/Linux、ArchBang、Manjaro Linux 與 Parabola GNU/Linux-libre。

2.3.6 其它

SUSE/openSUSE、Puppy Linux、Damn Small Linux 與 Slitaz。

2.4 Hadoop 簡介

Hadoop 是一種基於 Google 概念而衍生出的產品，目的是為了減少單一機器的運算負擔以及提高運算的效率。在這個環境下，整個系統會有分配工作的節點(Master)，以及執行工作的節點(Slaves)，當系統在執行運算時，若該運算是可切割的運算，根據系統的機制，擔任 Master 的 Namenode 可將該運算分布至各個 Datanode(Slaves)進行分布式運算，以達到提升整個系統運算效率以及減少單一機器進行運算的負擔的效果，如下圖 二.5。

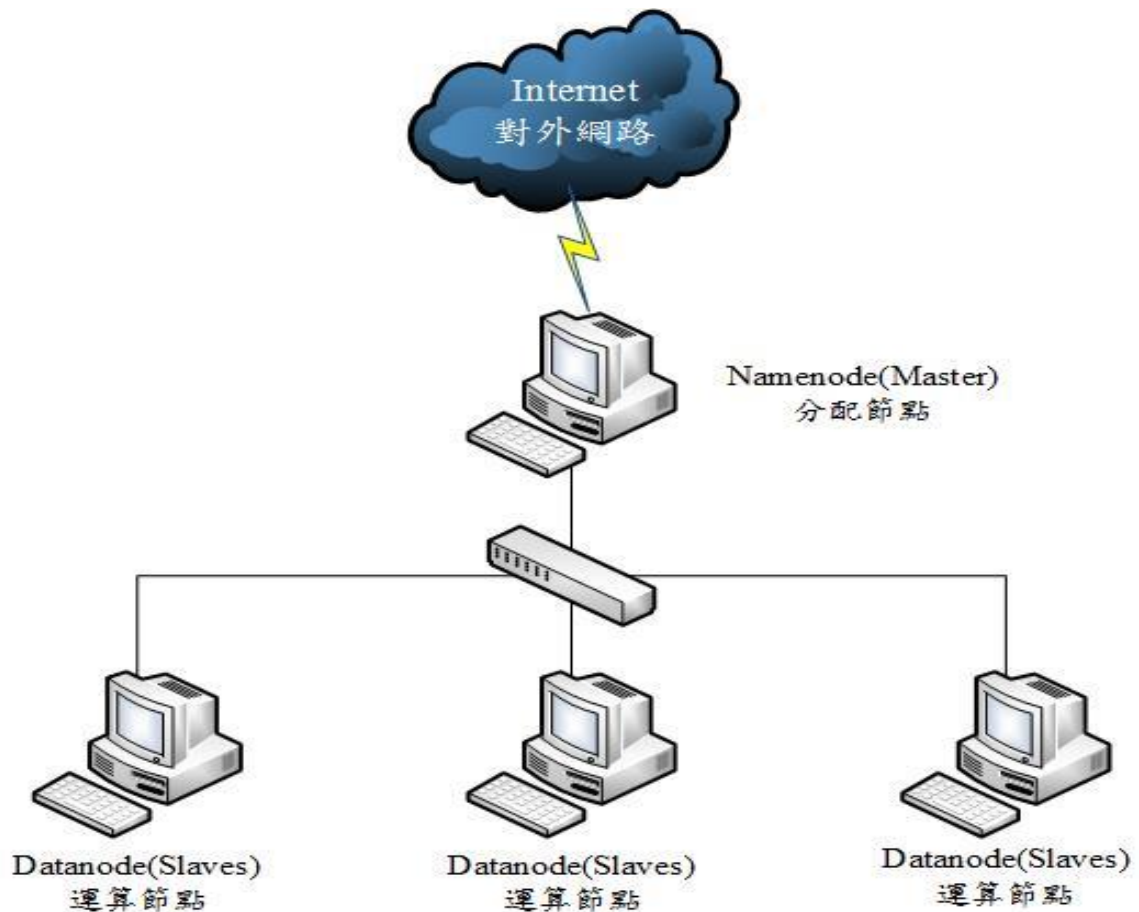


圖 二.5 待命中的 Hadoop 示意圖

以下為 Hadoop 最基礎的兩項功能：

2.4.1 Hadoop Distributed File System (HDFS)

(1) HDFS 介紹：

HDFS 是 Hadoop 幾大功能之一，其功用在於整個系統的儲存空間管理，將節點分成兩種，如表 二.1 講述 Namenode 節點。

表 二.1 Namenode 介紹

節點名稱	說明	特性	特性說明
Namenode	儲存資料屬性、儲存位置。	具備唯一性。	只會有一個，且為主控節點。

而表 二.2 講述 Datanode 節點。

表 二.2 Datanode 介紹

節點名稱	說明	特性	特性說明
Datanode	儲存資料並定期回傳給 Namenode 所儲存資料的列表。	容錯機制。	防備資料毀損、容許節點遺失、Namenode 錯誤備份、HeartBeat、Secondary-namenode 之 FSImage(備份 Namenode)。
		大規模資料集。	PetaBytes 等級之磁碟空間、大區塊(64MB, 大檔
		一致性模型。	一次寫入, 可多次讀取。
		異質平台。	可跨 Windows、Linux Ubuntu、CentOS、FreeBSD、MAC OS X 等

(2) HDFS 運作：

Namenode 會根據設定檔裡設定的區塊大小將欲儲存的檔案分割成許多的區塊，並按照副本數的設定，將每個區塊複製至該設定數量後，分別儲存至各個 Datanode，以避免當資料毀損或是節點遺失導致檔案出錯，藉此達成備份的效果。假設區塊大小設定為 64MB，備份數為 3，分配到六台 Datanode，如下圖 二.6。

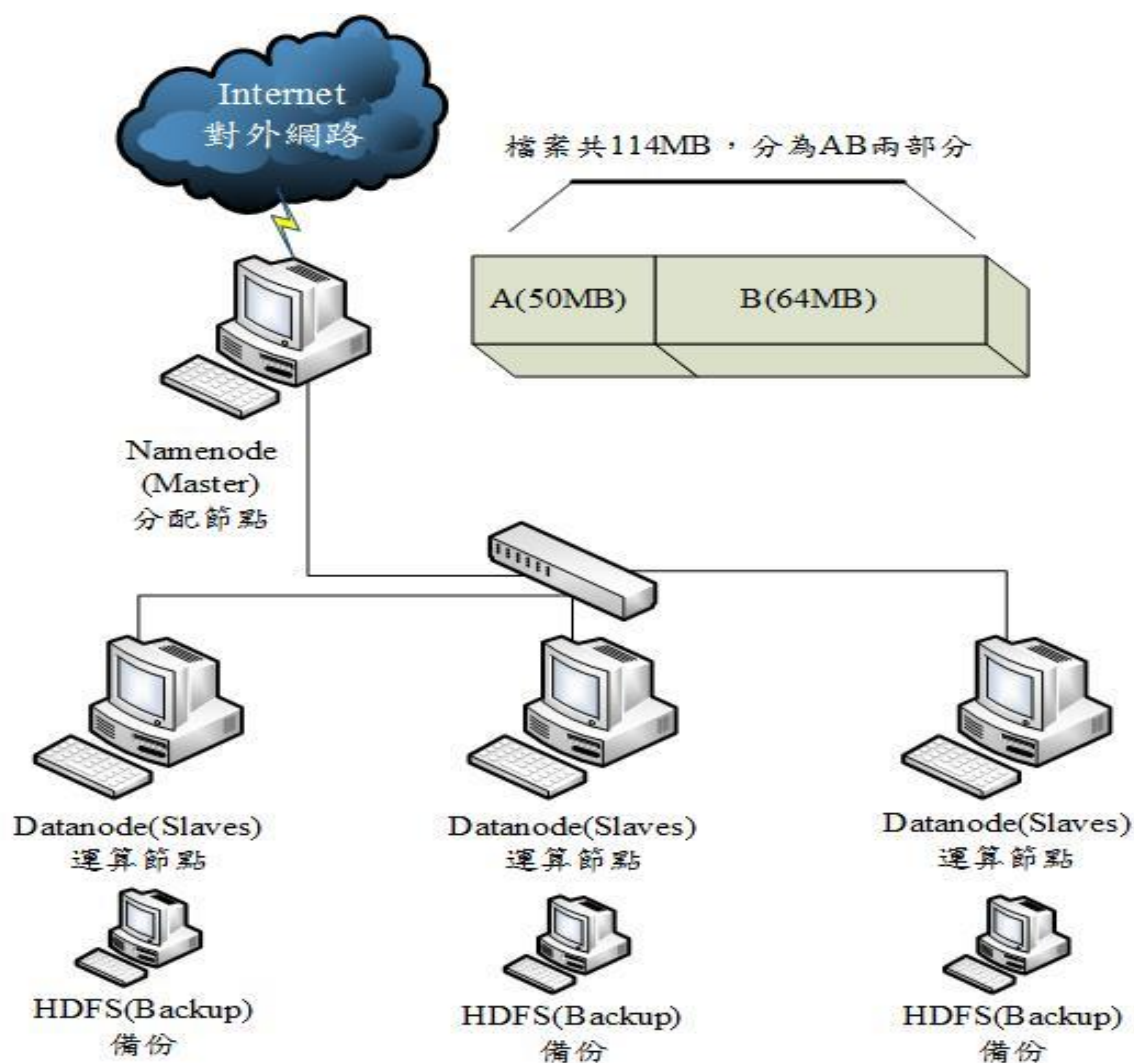


圖 二.6 檔案備份分配圖

2.4.2 MapReduce

MapReduce 是另一個 Hadoop 常用到的功能，他主要是用來處理大資料排程及系統管理，他會將 Hadoop 上的節點分為以下兩種：

(1) JobTracker

接收工作回報、指派工作、工作排程、錯誤處理。JobTracker 與 Namenode 一樣，具有唯一性，而通常 JobTracker 與 Namenode 會使用同一個節點。

(2) TaskTracker

執行工作、儲存工作結果。通常會有多個 TaskTracker，並與 Datanode 使用同一個節點。

以下表 二.3 為 MapReduce 運作的情形：

表 二.3 MapReduce 運作的情形

步驟	作法
一	JobTracker 跟NameNode 取得需要運算的blocks。
二	JobTracker 選數個TaskTracker 來作Map 運算，產生一些中間檔案。
三	JobTracker 將中間檔案整合排序（Shuffle）後，複製到需要的TaskTracker。
四	JobTracker 派遣TaskTracker 作reduce。
五	reduce 完後通知JobTracker 與Namenode 以產生Output。

(Shuffle-第一個字元依照特定的順序排序， Mapper 傳送前會先經過 Shuffle 洗牌)。

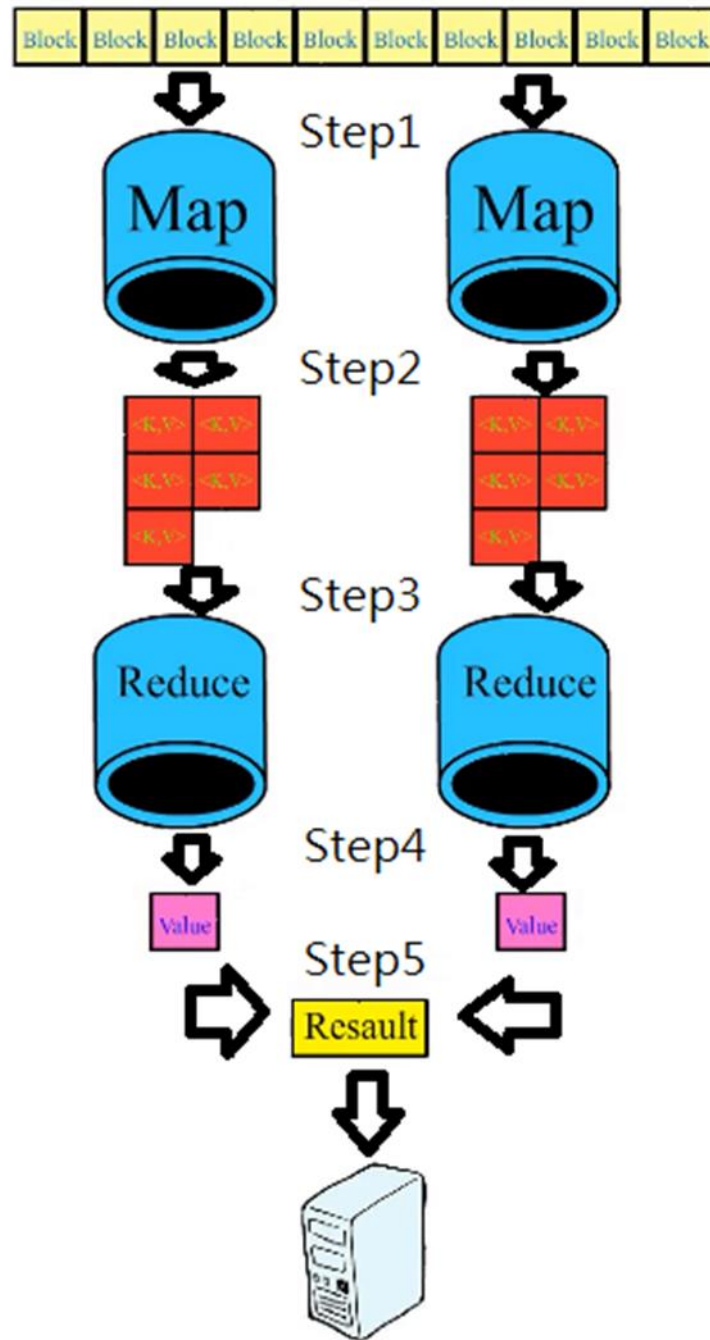


圖 二.7 MapReduce 運行架構

對照表 二.3 ，所提供之步驟與方法可明確理解與對應於上圖之圖

二.7 。繼續討論與 Hadoop 相關的功能：

2.4.3 蒙地卡羅方法

蒙地卡羅方法 (Monte Carlo method)，也稱統計模擬方法，是二十世紀四十年代中期由於科學技術的發展和電子計算機的發明，而被提出的一種以機率統計理論為指導的一類非常重要的數值計算方法。是指使用隨機數（或更常見的偽隨機數）來解決很多計算問題的方法。

在本研究中，使用的 Hadoop 範例檔即是使用蒙地卡羅方法來計算 π 的值。他是在一個邊長為一的正方形裡，隨機產生無數個點，統計落在扇形裡的點與扇形外的點，便可得到四分之一的圓周率，再乘以四便可得到圓周率，如圖 二.8。

其唯一的缺點在於主機容易產生重複的點或是分布不均勻，因此加上哈爾頓序列(Halton sequence)來修正這兩個問題，使點的分布更符合實際狀況。

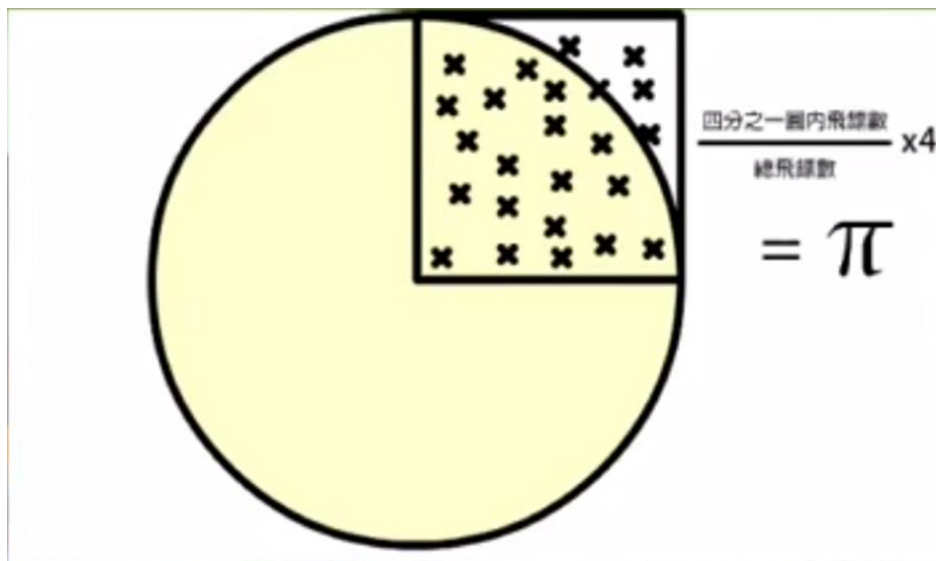


圖 二.8 蒙地卡羅架構

2.4.4 Thrift

Thrift 最初是由 Facebook 提出，這是一個服務端與客戶端之間的架構體系，目的是為了解決 Facebook 系統中各個系統間的大數據量的傳輸通訊以及系統之間不同語言環境，需要能夠跨平台，也因此 Thrift 可以支援多種程式語言(諸如 C++、C#、JAVA、PHP、Ruby) [21]。

2.4.5 HBase

HBase 是一個開源的分布式資料庫，具有以下特性：高可靠性、高性能、可伸縮、並非建立在關係模型基礎上，用以存儲大規模結構化數據。由於 Hadoop 的運作方式是分部式運算，因此使用 HBase 資料庫才能夠連結 PHP 與 Hadoop 之間。而 HBase 是用 Java 語言編寫的，本身沒有 C 語言的 API，但是可以通過 Thrift 實現 [22]。

2.5 LAMP 簡介

對於 LAMP，為一組常使用來執行動態網站或者伺服器的自由軟體名稱首字母縮寫。而這些軟體分別指的是：L- Linux 作業系統，A- Apache 網頁伺服器，M-MariaDB 或 MySQL 資料庫管理系統/伺服器，P- PHP、Perl 或 Python 電腦程序語言 [23] [24] [25] [26]。

由於前面章節以敘述並解釋過 Linux 系統，因此在此章節只針對尚未介紹的 Apache、MySQL 與 PHP 做相關解釋與概述，如下：

2.5.1 Apache

Apache 是開放原始碼的架設網頁伺服器軟體，可在大多數電腦作業系統平台中運行。因其跨平台與安全性，而能被廣泛使用。是當今最流行的 Web 伺服器建立軟體。

2.5.2 MySQL

為一免費資料庫建置與編寫系統，同樣也是開放源始碼的程式。其效能高、成本低、可靠性好的特性，讓其成為日前最流行的開源資料庫。可由 PHP 網頁撰寫存取其資料庫。

2.5.3 PHP

PHP 是一種開源的通用電腦指令碼語言，尤其適用於網路開發，並可與 HTML 結合使用。他將藉由 Thrift 這個中介函式庫對 HBase 資料庫做出存取的動作。

2.6 雲端節能

雲端節能有兩種方式：一則以負載平衡(Load Balncing)，其作法是將所有工作平均分配給每台主機處理。另一則以合併運算(Consolidation)，其作法是將所有工作集中於一台主機滿載處理。如圖 二.9 所示。

提高工作效率及達到有效節能的方式為資源監控再執行負載平衡，如資源無法監控，負載就不能平衡。一般監控的部分為 CPU 使用率和記憶體儲存設備的使用率；相對於網路的監控則是網路流量大小、路由狀態以及輸入、輸出。負載平衡(Load Balancing)為雲端基礎設施資源管理的重要一環，優於合併運算(Consolidation)的主因為：負載平衡能達到節能和有效運算的平衡點優於合併運算。因此我們選用負載平衡作為雲端節能系統的研究主軸。

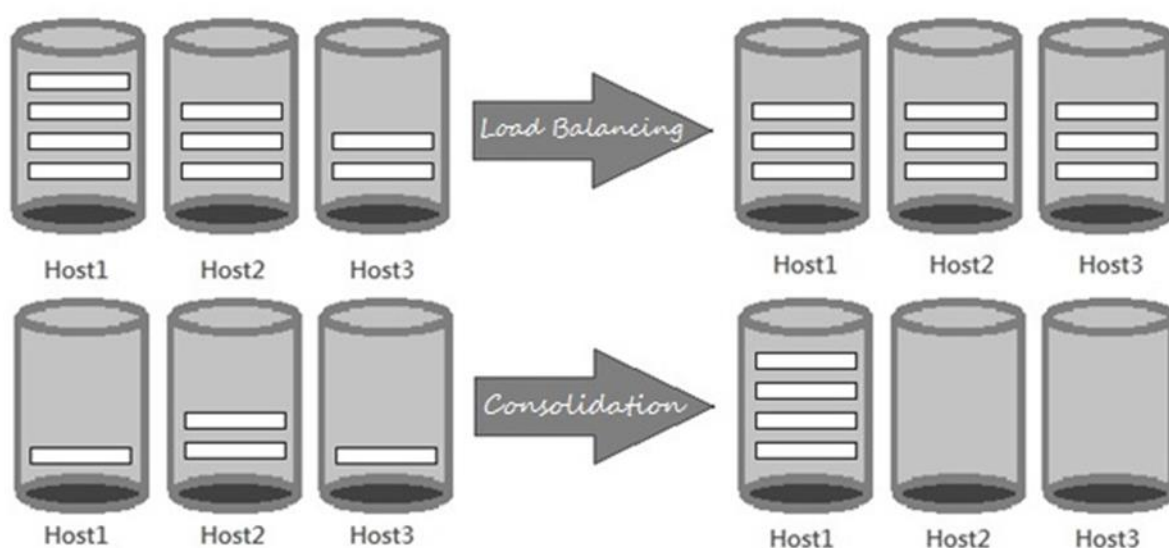


圖 二.9 負載平衡 VS 合併運算

第三章 雲端系統設計

3.1 系統架構

本研究中，以 PHP 架構一個使用者介面，並以 Hadoop 架構一個雲端環境。

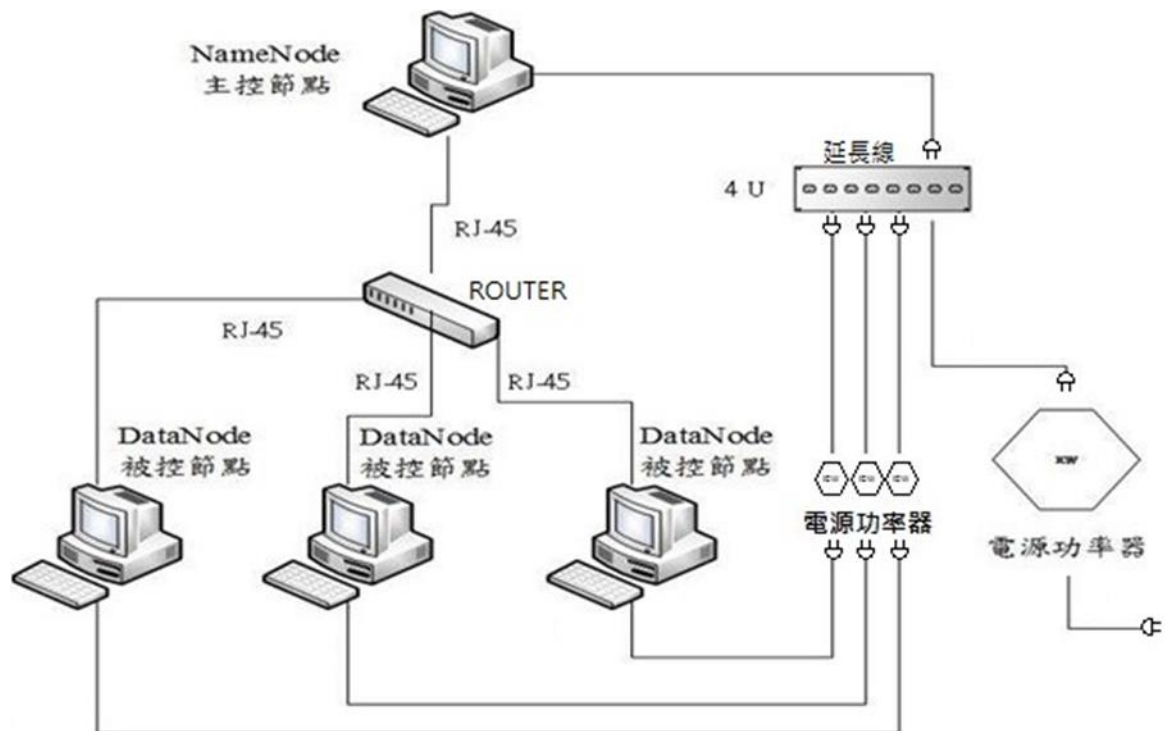


圖 三.1 系統架構圖

系統架構如上圖 三.1，內容包括一台 Namenode 與三台 Datanode，皆以 Ubuntu Linux 作為作業系統，彼此之間以一台 Switch 作為連接。

Namenode 主要用來接受指令並將工作分配給三台 Datanode，而 Datanode 將處理那些分配下來的工作，儲存並回傳結果給 Namenode。在利用 Hadoop 內建範例程式進行雲端運算模擬的同時，使用電源功率監測器來觀察四台主機的電能消耗量。

3.2 研究方法與步驟

3.2.1 研究方法

該研究方法將會被設計在使用者介面裡。工作內容是判斷當一項工作量少時由「單一節點執行運算」或是工作量很多時「分布到各節點進行運算」；甚或是工作量不致較多，「將工作只集中於某些節點運算」。在這三種模式中，那一種模式會是最適合該工作，在「節能」與「效率」中取得平衡 [27]。除此之外，當 Hadoop 運作時，Slaves 需要待機以等待工作，而待機的過程需要耗費電力等能源，藉由使用者介面，可以讓待機中的 Datanode 進入睡眠模式，當有工作需要分配時，再喚醒，如此便可以減少不必要的能源消耗。以下為各種模式說明：

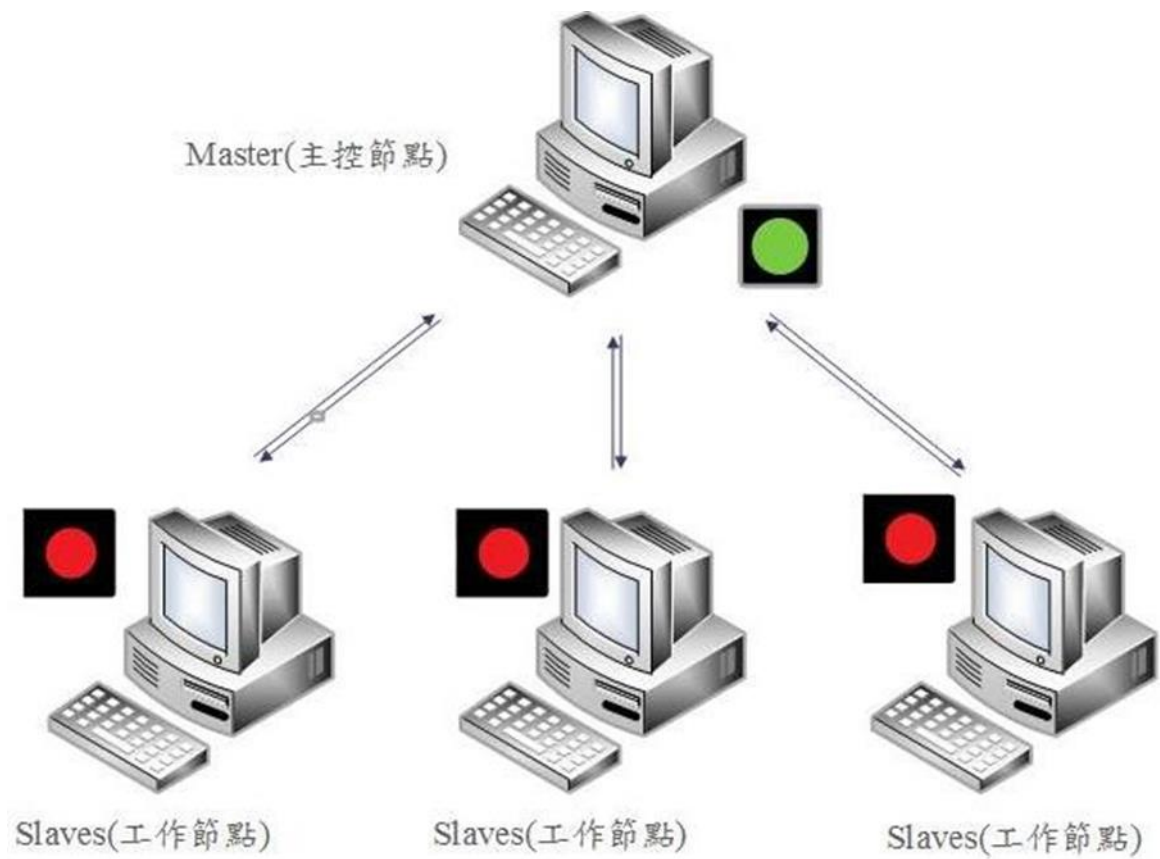


圖 三.2 待命中的 HADOOP 工作示意圖

(1) 一般待命等待工作

平時只需等待接應工作，所以只須將 Namenode(主控電腦)開啟即可，如圖 三.2。

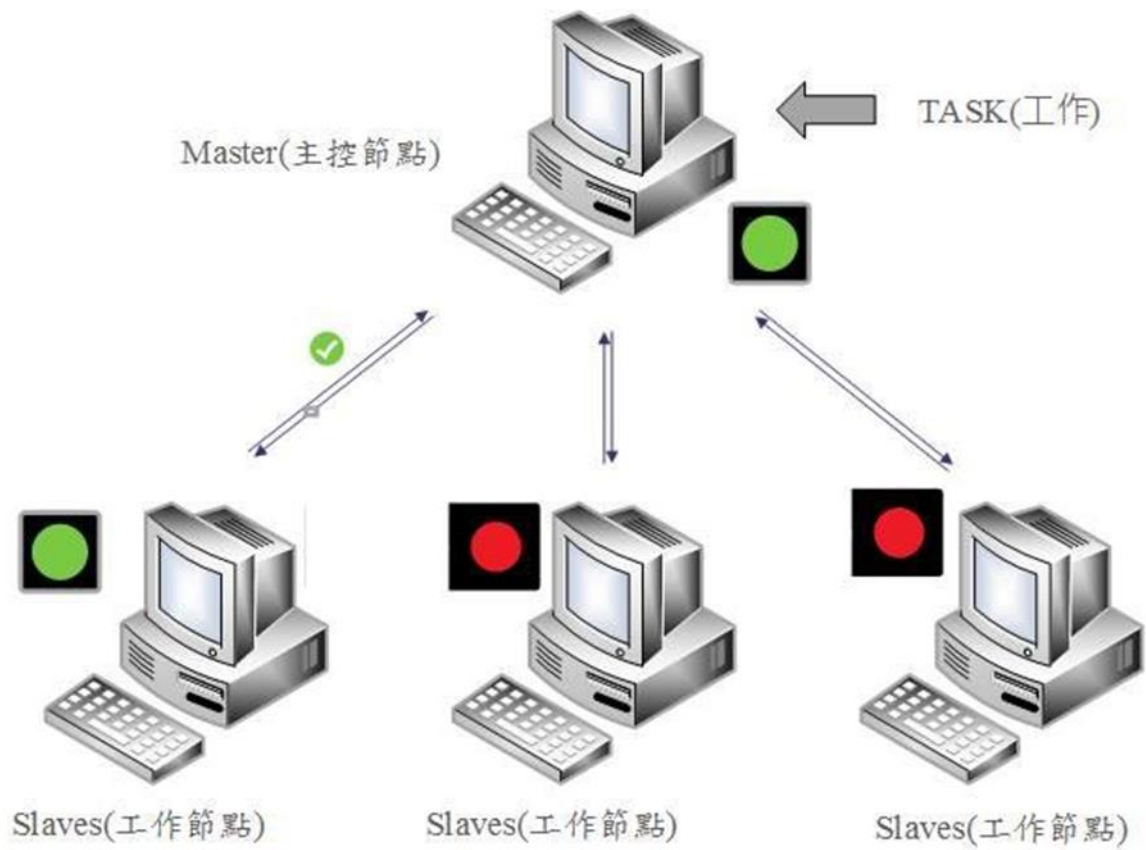


圖 三.3 集中主機運算

(2) 單一主機執行運算

考慮到有時運算可以由單一主機獨力完成，且耗能較所有節點

一同運算低，因而使用此一模式，如圖 三.3。

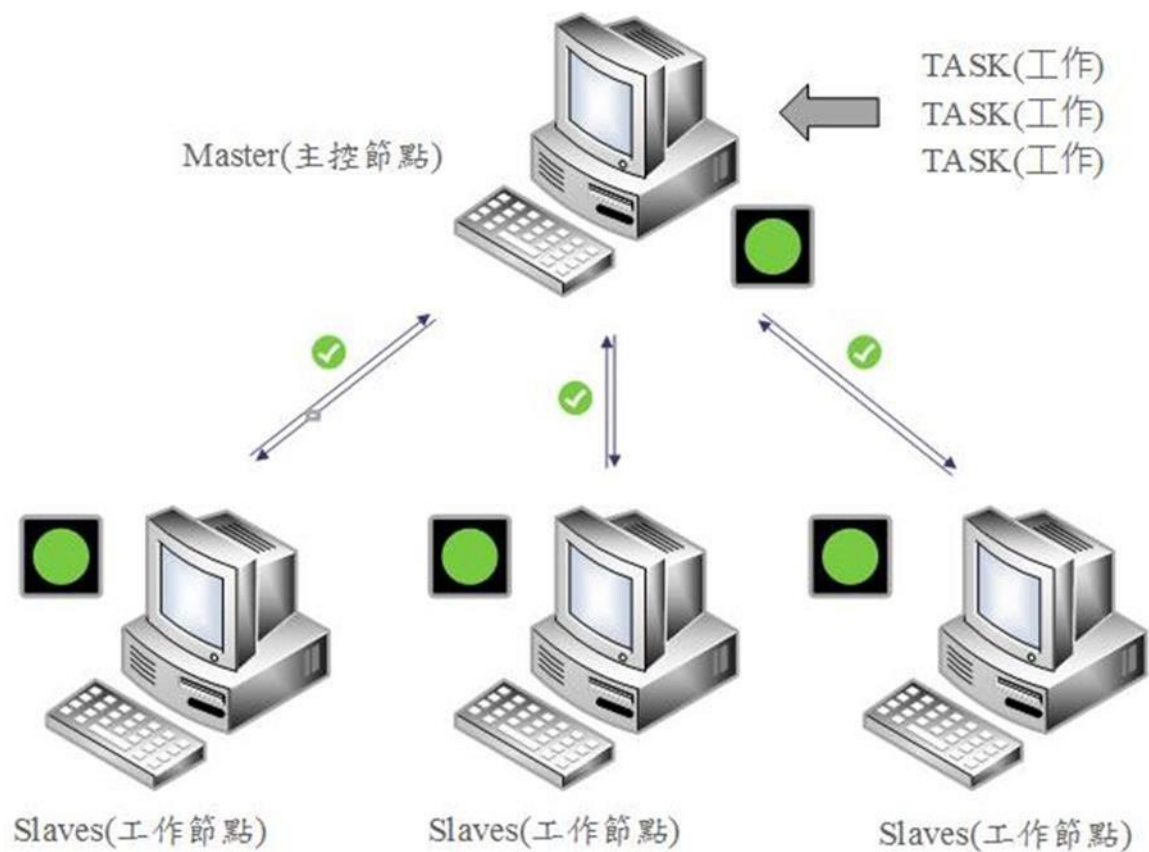


圖 三.4 分配主機運算

(3) 分布到各主機進行運算

考慮到有時運算可以由單一主機獨力完成，且耗能較所有節點一同運算低，因而使用此一模式，這是預想中最為最普遍的模式，因為其將會是運算最具效率的模式，如圖 三.4。

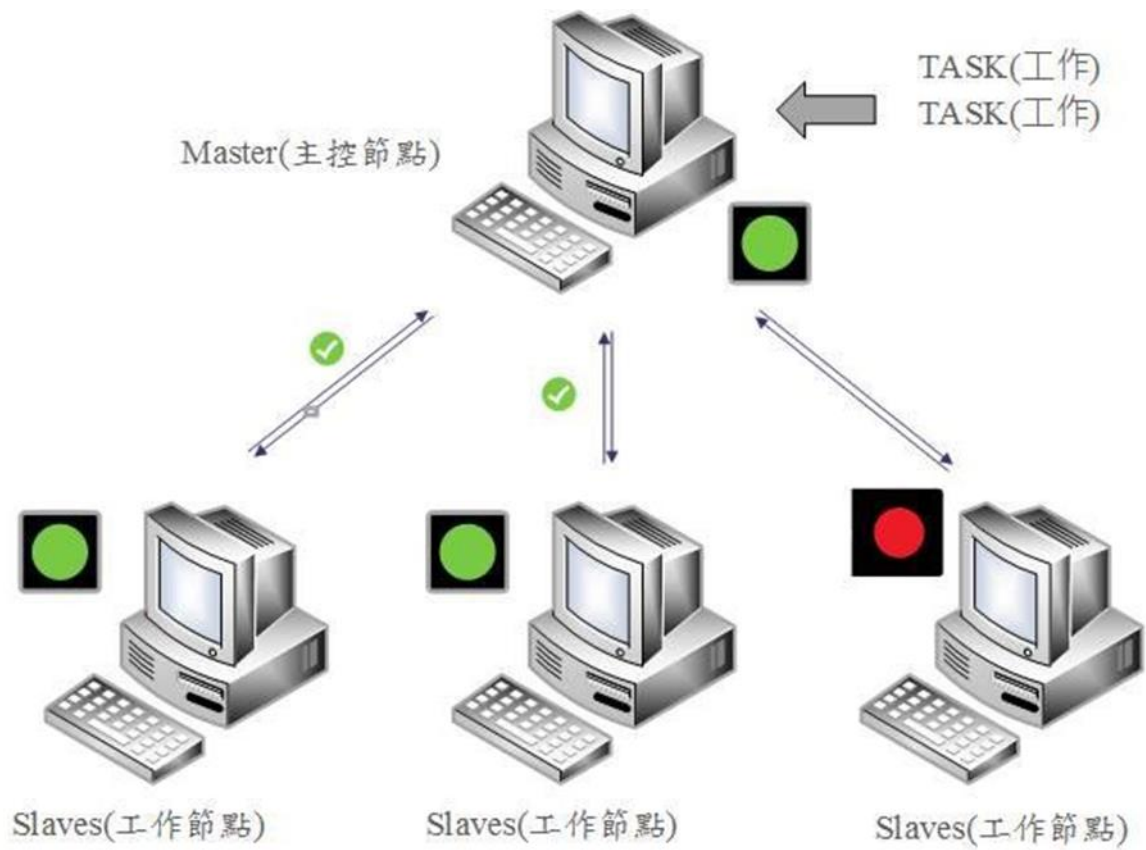


圖 三.5 集中分配主機運算

(4) 將工作只集中於某些主機運算

因考量所需運算工作偏大，但啟用所有節點工作時較耗費資源，故只開啟特定數量節點進行運算，如圖 三.5。

以上的各個模式便是本計畫中最開始假定的最佳分配模型，而我們將先利用此些模型來進行實驗，確認是否確實是各種分配中效果最好的幾種。

3.2.2 研究步驟

(1) 安裝系統

本研究使用的是 Ubuntu Linux 系統 [27]，因其自由且免費授權使用的特性，較利於本研究的開發。



圖 三.6 Linux Ubuntu(圖片來源:Ubuntu)

主要下載點如圖 三.6。



圖 三.7 安裝畫面一



圖 三.8 安裝畫面二

以下為其安裝過程，圖 三.7 中，必須注意電腦使用名稱的設定。此處與往後再設定 Hadoop 連線上，修改 Hosts 檔案時會用到。而圖 三.8 為完成設定等待安裝的畫面。



圖 三.9 安裝 tasksel

另外，安裝完系統後，因後續架設管控網頁與 HADOOP 連線之需求，需再做額外套件的安裝與設定。圖 三.9 為安裝 tasksel，輸入 `sudo apt-get install tasksel`，用以安裝其他所需套件，如 OpenSSH 與 LAMP 套件 [28]。

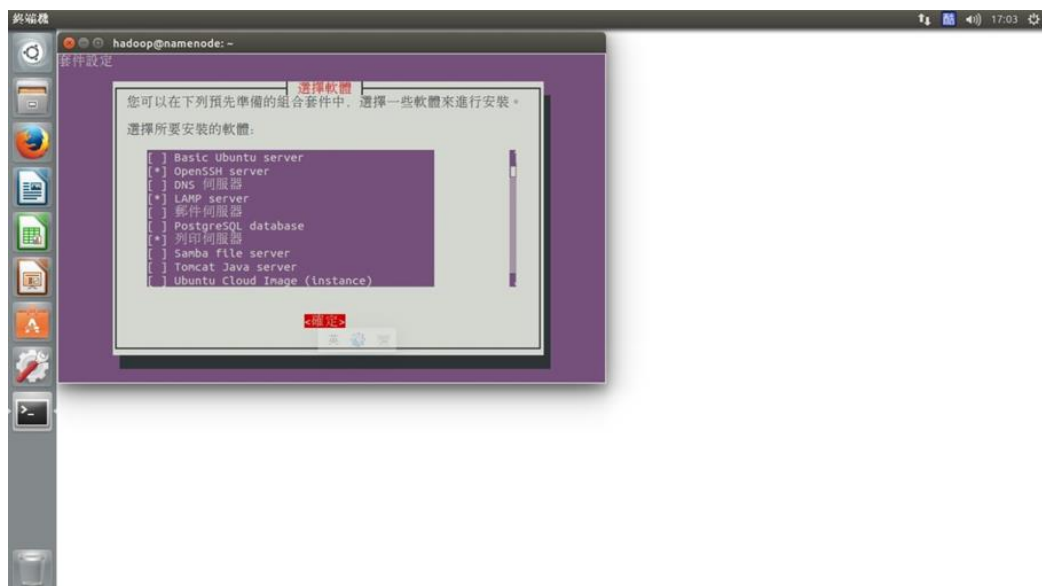


圖 三.10 設定套件

圖 三.10 為勾選欲安裝套件之畫面。

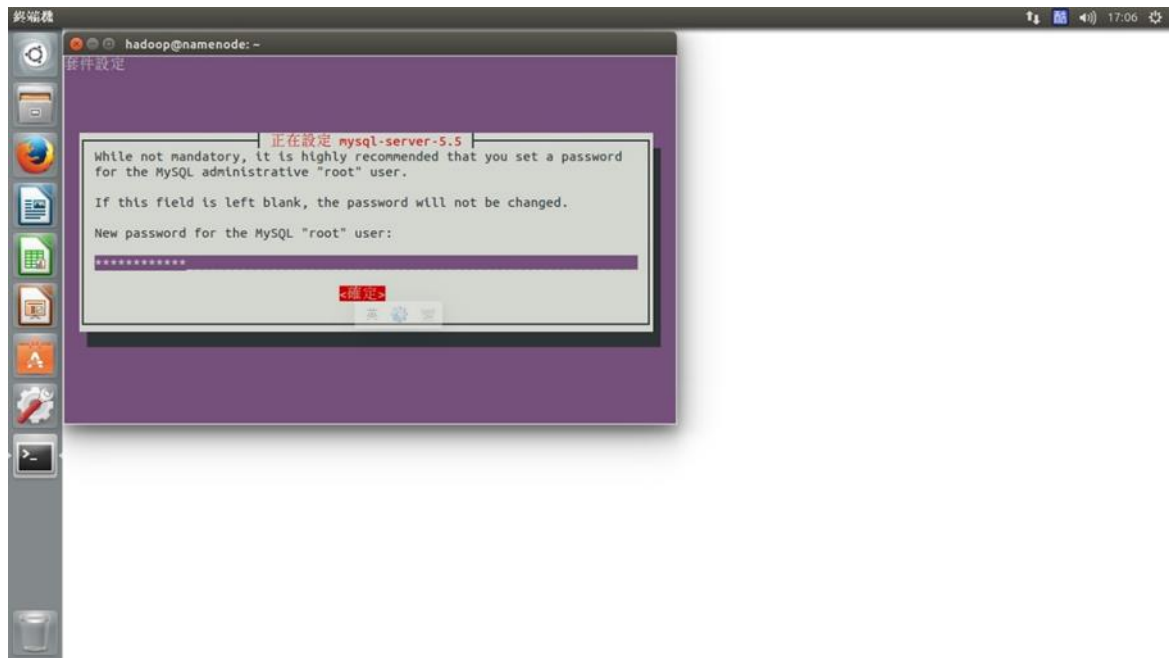


圖 三.11 MySQL 根密碼設定

圖 三.11 為設定資料庫 MySQL 根目錄密碼之畫面。

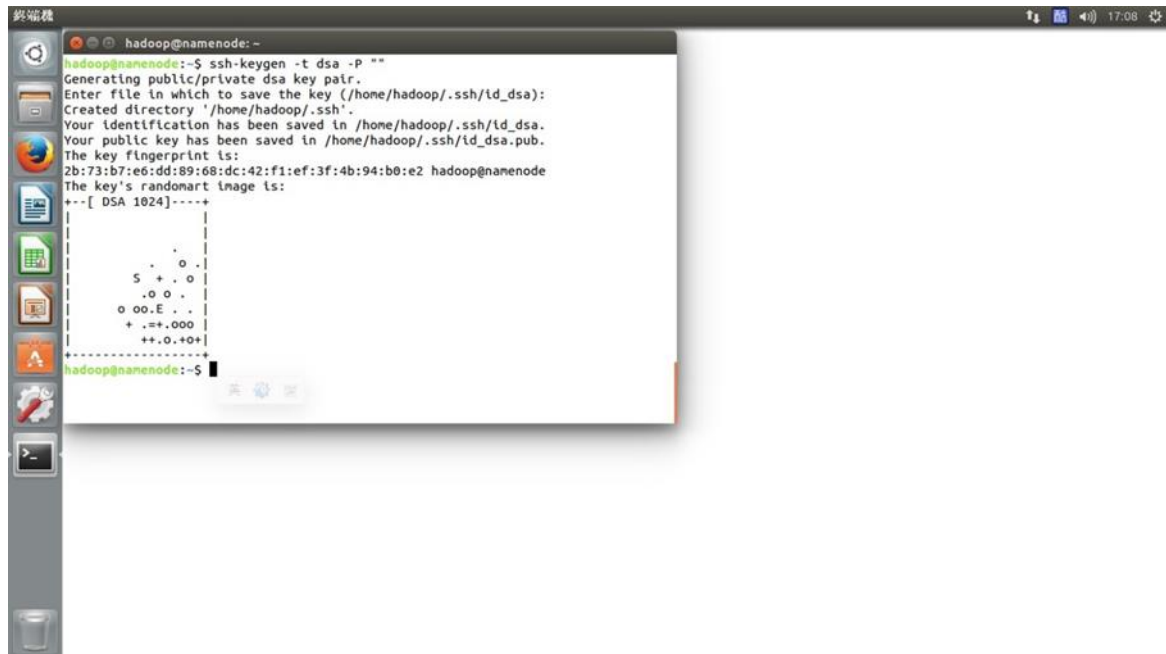
(2) 建構 Hadoop 系統

建構 Hadoop 本身除了 Hadoop 之設置外，仍需安裝相關軟體套件，如：OpenSSH, Java。本研究由下表 三.1 表列出所需安裝之軟體與步驟等詳細資訊。

表 三.1 Hadoop 完整安裝與架設

步驟	作法	敘述
一	SSH 公私鑰	Hadoop 平台的連線是建立於 SSH 的機制之上，因每次連線都需要經過 SSH 的加密驗證，故使用公私鑰認證免除每次連線都需要輸入密碼可增加研究的便利性 [29]。
二	安裝 JAVA	Hadoop 平台是建立在 JAVA 之上，因此要使用 Hadoop 平台需先安裝 JAVA [30]。
三	安裝與設定 Hadoop	安裝、設定 Hadoop 平台的過程。
四	Hadoop 雲端運算模	研究中以 Hadoop 內建程式作為雲端運算的模擬。
五	Hadoop 內建狀態網頁	可以觀察目前存活和死亡的節點個數，Hadoop 磁碟空間使用量和剩餘量。

以下為 SSH 的建立金鑰過程：

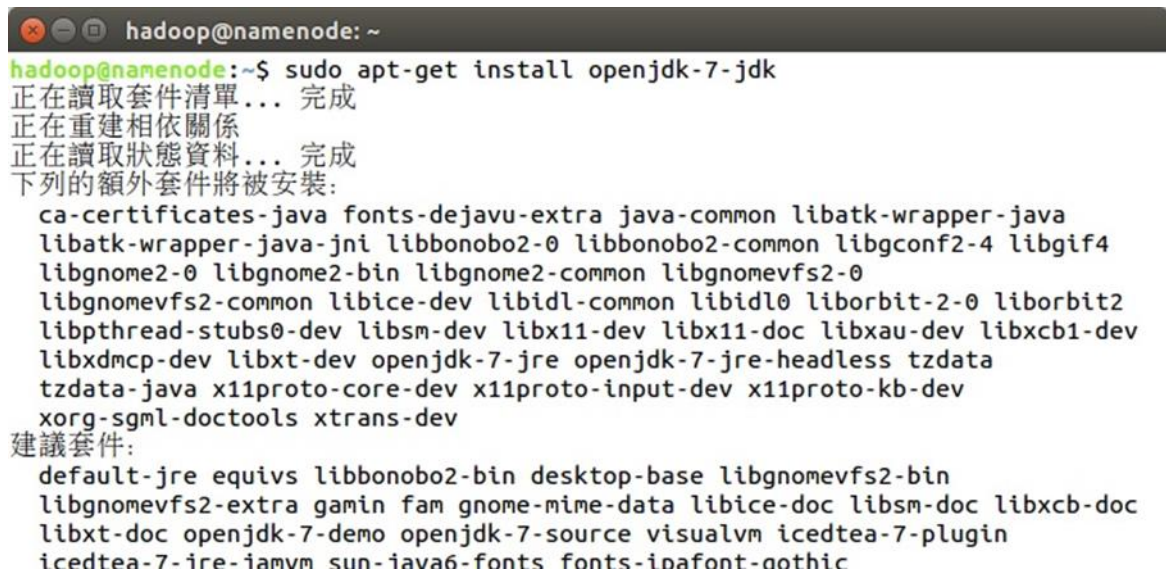


```
hadoop@namenode: ~  
hadoop@namenode:~$ ssh-keygen -t dsa -p ""  
Generating public/private dsa key pair.  
Enter file in which to save the key (/home/hadoop/.ssh/id_dsa):  
Created directory '/home/hadoop/.ssh'.  
Your identification has been saved in /home/hadoop/.ssh/id_dsa.  
Your public key has been saved in /home/hadoop/.ssh/id_dsa.pub.  
The key fingerprint is:  
2b:73:b7:e6:dd:89:68:dc:42:f1:ef:3f:4b:94:b0:e2 hadoop@namenode  
The key's randomart image is:  
+--[ DSA 1024]-----+  
| . o |  
| S + . o |  
| .o o . |  
| o oo.E . |  
| + .+=.ooo |  
| ++.O.+0+ |  
+-----+  
hadoop@namenode:~$
```

圖 三.12 SSH 金鑰建立

輸入指令：`ssh-keygen -t dsa -p ""` 產生金鑰，如圖 三.12。

以下為 JAVA 安裝過程：



```
hadoop@namenode: ~  
hadoop@namenode:~$ sudo apt-get install openjdk-7-jdk  
正在讀取套件清單... 完成  
正在重建相依關係  
正在讀取狀態資料... 完成  
下列的額外套件將被安裝：  
ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java  
libatk-wrapper-java-jni libbonobo2-0 libbonobo2-common libgconf2-4 libgif4  
libgnome2-0 libgnome2-bin libgnome2-common libgnomevfs2-0  
libgnomevfs2-common libice-dev libidl-common libidl0 liborbit-2-0 liborbit2  
libpthread-stubs0-dev libsm-dev libx11-dev libx11-doc libxau-dev libxcb1-dev  
libxdmcp-dev libxt-dev openjdk-7-jre openjdk-7-jre-headless tzdata  
tzdata-java x11proto-core-dev x11proto-input-dev x11proto-kb-dev  
xorg-sgml-doctools xtrans-dev  
建議套件：  
default-jre equivs libbonobo2-bin desktop-base libgnomevfs2-bin  
libgnomevfs2-extra gamin fam gnome-mime-data libice-doc libsm-doc libxcb-doc  
libxt-doc openjdk-7-demo openjdk-7-source visualvm icedtea-7-plugin  
icedtea-7-ire-iamvm sun-java6-fonts fonts-inafont-notthic
```

圖 三.13 安裝 JAVA

輸入指令：`sudo apt-get install openjdk-7-jdk` 安裝 JAVA 如圖 三.13。

以下為安裝、設定 Hadoop：



圖 三.14 安裝 Hadoop

輸入指令：`wget hadoop-1.2.1.tar.gz` 壓縮檔網址下載並取得 `hadoop-1.2.1.tar.gz` 壓縮檔，如圖 三.14。

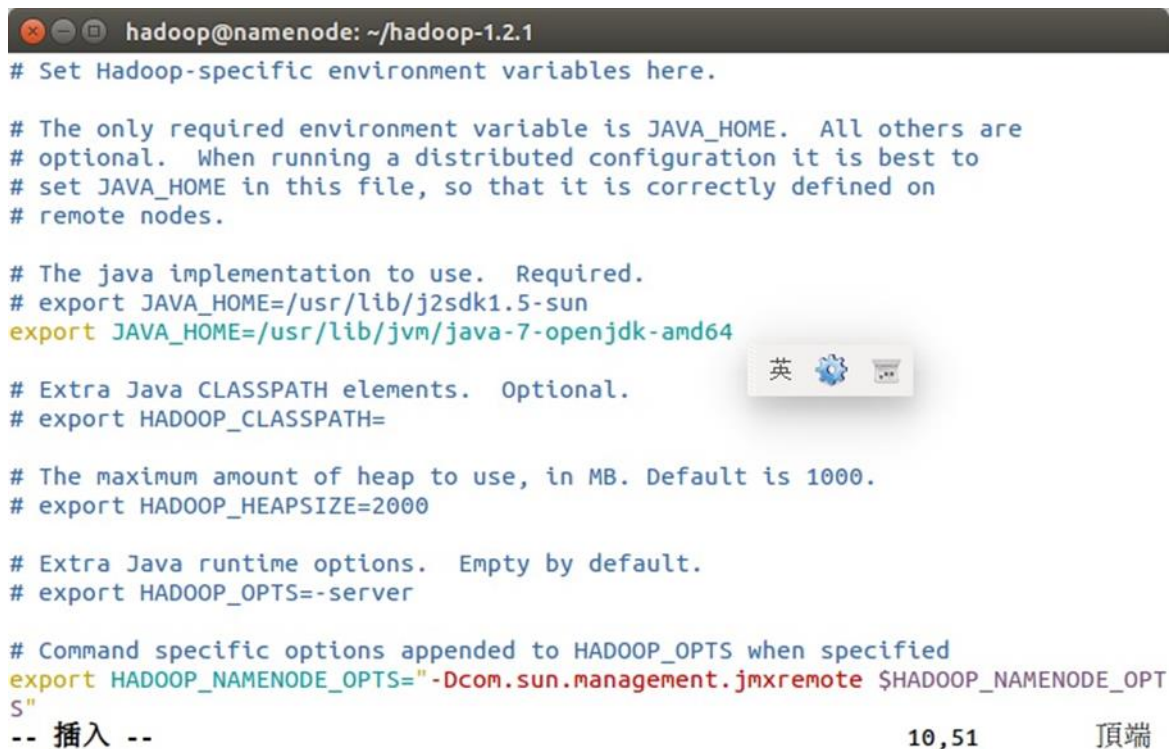


圖 三.15 hadoop-env.sh

在 `hadoop-env.sh` 檔案內加入：`export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386` 設定 JAVA 目標路徑，如圖 三.15。

```
hadoop@namenode: ~/hadoop-1.2.1
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

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

圖 三.16 mapred-site.xml

修改 mapred-site.xml 檔案，設定 namenode 主機指派 map 工作，
如圖 三.16。

```
hadoop@namenode: ~/hadoop-1.2.1
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.block.size</name>
    <value>10240</value>
  </property>
</configuration>
~
~
~
~
~
```

圖 三.17 hdfs-site.xml

修改 hdfs-site.xml，將資料分割為三部分；大小為 1024，如圖 三.17。

```
hadoop@namenode: ~/hadoop-1.2.1
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://namenode:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/hadoop-1.2.1/hadoop_tmp</value>
  </property>
</configuration>
~
~
~
~
~
~
~
```

14,19 全部

圖 三.18 core-site.xml

修改 core-site.xml 設定，決定分派主機位置與設定 hadoop_tmp 系統暫存檔路徑，如圖 三.18。

雲端運算模擬如下：

```
hadoop@namenode: ~/hadoop-1.2.1
hadoop@namenode:~/hadoop-1.2.1$ rm -rf hadoop_tmp/
```

圖 三.19 刪除 Hadoop 暫存檔

輸入指令：rm -rf hadoop_tmp/，刪除 hadoop 暫存資料夾，如圖 三.19。

```

hadoop@namenode: ~/hadoop-1.2.1
hadoop@namenode:~/hadoop-1.2.1$ bin/hadoop namenode -format
14/11/13 17:35:54 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:  host = java.net.UnknownHostException: namenode: namenode: 未知的
名稱或服務
STARTUP_MSG:  args = [-format]
STARTUP_MSG:  version = 1.2.1
STARTUP_MSG:  build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.2 -r 1503152; compiled by 'mattf' on Mon Jul 22 15:23:09 PDT 2013
STARTUP_MSG:  java = 1.7.0_65
*****/

```

圖 三.20 格式化開始

輸入指令：bin/hadoop namenode -format，格式化各節點，如圖 三.20。

```

hadoop@namenode: ~/hadoop-1.2.1
hadoop@namenode:~/hadoop-1.2.1$ bin/start-all.sh
starting namenode, logging to /home/hadoop/hadoop-1.2.1/libexec/./logs/hadoop-h
adoop-namenode-namenode.out
localhost: starting datanode, logging to /home/hadoop/hadoop-1.2.1/libexec/./lo
gs/hadoop-hadoop-datanode-namenode.out
localhost: starting secondarynamenode, logging to /home/hadoop/hadoop-1.2.1/libe
xec/./logs/hadoop-hadoop-secondarynamenode-namenode.out
starting jobtracker, logging to /home/hadoop/hadoop-1.2.1/libexec/./logs/hadoop
-hadoop-jobtracker-namenode.out
localhost: starting tasktracker, logging to /home/hadoop/hadoop-1.2.1/libexec/./
logs/hadoop-hadoop-tasktracker-namenode.out
hadoop@namenode:~/hadoop-1.2.1$ █

```

圖 三.21 啟動 Hadoop

輸入指令：bin/start-all.sh，啟動 hadoop，如圖 三.21。

```

hadoop@namenode: ~/hadoop-1.2.1
hadoop@namenode:~/hadoop-1.2.1$ jps
20878 JobTracker
20648 DataNode
21105 Jps
20504 NameNode
20792 SecondaryNameNode
21023 TaskTracker
hadoop@namenode:~/hadoop-1.2.1$ █

```

圖 三.22 檢視狀態

輸入指令：jps，查看 hadoop 運行狀態，如圖 三.22。


```

hadoop@namenode: ~/hadoop-1.2.1
hadoop@namenode:~/hadoop-1.2.1$ bin/hadoop jar hadoop-examples-1.2.1.jar pi 100
100
Number of Maps = 100
Samples per Map = 100
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Wrote input for Map #10
Wrote input for Map #11
Wrote input for Map #12
Wrote input for Map #13
Wrote input for Map #14
Wrote input for Map #15
Wrote input for Map #16
Wrote input for Map #17

```

圖 三.23 執行範例檔

輸入指令：bin/hadoop jar hadoop-examples-1.2.1.jar pi 100 100，執行

範例檔，如圖 三.23。

```

hadoop@namenode: ~/hadoop-1.2.1
14/11/13 18:04:20 INFO mapred.JobClient: Map-Reduce Framework
14/11/13 18:04:20 INFO mapred.JobClient: Map ooutput materialized bytes=2800
14/11/13 18:04:20 INFO mapred.JobClient: Map input records=100
14/11/13 18:04:20 INFO mapred.JobClient: Reduce shuffle bytes=1800
14/11/13 18:04:20 INFO mapred.JobClient: Total committed heap usage (bytes)=
16305356800
14/11/13 18:04:20 INFO mapred.JobClient: CPU time spent (ms)=31420
14/11/13 18:04:20 INFO mapred.JobClient: Map input bytes=2400
14/11/13 18:04:20 INFO mapred.JobClient: SPLIT_RAW_BYTES=13590
14/11/13 18:04:20 INFO mapred.JobClient: Combine input records=0
14/11/13 18:04:20 INFO mapred.JobClient: Reduce input records=200
14/11/13 18:04:20 INFO mapred.JobClient: Reduce input groups=200
14/11/13 18:04:20 INFO mapred.JobClient: Combine output records=0
14/11/13 18:04:20 INFO mapred.JobClient: Physical memory (bytes) snapshot=16
992567296
14/11/13 18:04:20 INFO mapred.JobClient: Reduce output records=0
14/11/13 18:04:20 INFO mapred.JobClient: Virtual memory (bytes) snapshot=473
28104448
14/11/13 18:04:20 INFO mapred.JobClient: Map output records=200
Job Finished in 133.628 seconds
Estimated value of Pi is 3.140800000000000000000000
hadoop@namenode:~/hadoop-1.2.1$ █

```

圖 三.24 運算過程與結果

運算出的結果為 3.1408，如圖 三.24，包含過程與結果。

```
hadoop@namenode: ~/hadoop-1.2.1
hadoop@namenode:~/hadoop-1.2.1$ bin/stop-all.sh
stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: stopping datanode
localhost: stopping secondarynamenode
hadoop@namenode:~/hadoop-1.2.1$
```

圖 三.25 結束 Hadoop

輸入指令：bin/stop-all.sh，停止 Hadoop 運作，如圖 三.25。狀態

網頁如下：

hadoop-Aspire-V3-571G Hadoop Map/Reduce Administration

State: RUNNING
Started: Tue Apr 29 20:51:33 CST 2014
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf
Identifier: 201404292051
SafeMode: OFF

Cluster Summary (Heap Size is 62.25 MB/889 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	0	1	0	0	0	0	2	2	4.00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	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

圖 三.26 Hadoop-50030

由網頁連結 <http://localhost:50030/> 可進入如圖 三.26 的網頁。主要列出了現在階段主控端所見之存活節點個數、各節點的分配比例與各節點的運算百分比…等。

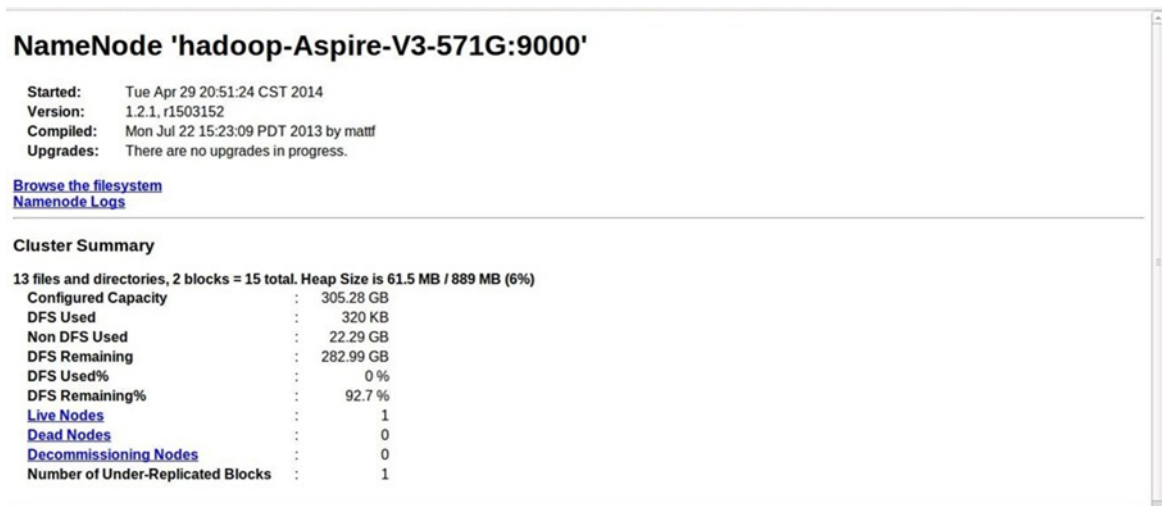


圖 三.27 HADOOP-50070

由網頁連結 <http://localhost:50070/> 可進入如圖 三.27 的網頁。主要列出了現在階段總的運算狀況百分比與各基本硬體使用量…等。

(3) 使用者介面

關於使用者介面，它的功用在於觀察、偵測 Hadoop 系統上各個節點的狀況(諸如耗能狀況、CPU 溫度、記憶體載量)，並藉由系統機制分析狀況來介入 Namenode 分配工作的情形，做適當的動態調節與自動遠端操控 Datanode 之開關機。甚至在特殊情況時，可由使用者來手動分配工作。

除此之外，將利用與 HBase 資料庫來記錄進行雲端運算時各個節點的資源分配、耗損的情況，可讓使用者於下次進行雲端運算時參考使用。

第四章 研究成果

4.1 基礎研究成果

在此次的計畫中，進行了幾項先期實驗，實驗是使用 Hadoop 內建的程式計算 π ，而該程式是使用蒙特卡羅方法計算。並由程式本身自動計算總體消耗時間。本次實驗所用的參數有兩種，第一種是對 10000 個靶各投 10000 次，所做的數據有三台 Datanode(3Datanode)與兩台 Datanode(2Datanode)。第二種是對 100 個靶各投 100 次，所做的數據有 2Datanode。表 四.1 中 3Datanode 表示同時啟動三台電腦進行運算，而 2Datanode 代表以兩台電腦進行運算，1Datanode 代表以一台電腦進行運算。而其中紀錄了十次的電腦消耗功率，用以作為節能後消耗功率之對照組。平均消耗功率之計算用以校準精確度，可作更準確的比較。其中，第一次的消耗功率偏低乃 Hadoop 本身在運算前須做分配工作，此時由於僅有 Namenode 在運算及評估如何分派，而 Datanode 本身並未接受到任何須執行之工作指派。

表 四.1 雲端運算時的消耗功率。(單位：W)

	第一次	第二次	第三次	第四次	第五次	第六次
3Datanode	80	137	132	134	135	132
2Datanode	92	90	90	91	89	90
1Datanode	30	48	49	50	49	49
	第七次	第八次	第九次	第十次	Average	
3Datanode	131	133	130	130	127.4	
2Datanode	91	91	91	94	90.9	
1Datanode	49	50	51	49	47.4	

資料來源：本專題研究整理

表 四.2 中 3Datanode 表示同時啟動三台電腦進行運算，而 2Datanode 代表以兩台電腦進行運算。一共做了四次紀錄，由於資料密集集中於某個小區間，故沒有再取平均。此紀錄用以之後在決定工作量變更時，開機時間的排班調整。

表 四.2 雲端運算時的耗時。(單位：秒)

	第一次	第二次	第三次	第四次
3Datanode	6353.305	6458.661	6540	6547.058
2Datanode	8867.301	8565.311	8649.222	8827.981
1Datanode	118	117	117	122

資料來源：本專題研究整理

根據以上二表，可推估出雲端運算時的耗能。可得出當使用 2Datanode 時相較 3Datanode 可減少大約 5%的能源，然而卻會增加大約 34%的耗時。見下表 四.3。

表 四.3 雲端運算的耗能。(單位：J)

3Datanode	2Datanode	1Datanode
826241.6162	792312.2154	5498.4

資料來源：本專題研究整理

4.2 動態之決策

在本次研究中，共有三台運算節點提供計算。由固定使用平衡模擬運算之工作狀況來決定主要第一台啟動運算之電腦。如下表 四.4 中所示，三台運算節點中只有 Datanode-1 可提供較穩定的決策數據，因此本研究將動態決策主以此電腦進行數據量測。

表 四.4 運算節點比較表(CPU 使用率%)

節點	跑 pi 前	跑 pi 期間				跑 pi 後
Datanode-1	0.8	90.4	90.2	90.3	20.6	0.7
Datanode-2	0.3	99.7	100	100	62.9	1
Datanode-3	1	99.3	100	100	72.6	0.7

資料來源：本專題研究整理

接著，必須判斷出開啟其他節點之時機，因此我們另外針對 Datanode-1 進行低負載至高負載的模擬運算工作。如下表 四.5 所示，我們針對低負載、平衡與高負載之模擬運算各實施三次。如此得知，選擇使用率穩定介於 90% 至 100% 的期間將啟動其他台運算節點以支援運算並達到動態節能之目的。

表 四.5 Datanode-1 之省電幅度測試(CPU 使用率%)

模擬運算	跑 pi 前		跑 pi 期間		跑 pi 後	
低負載-1	0.4	5.9	29.6	45.3	1.5	0.5
低負載-2	0.7	9	44.6	41	1.3	0.3
低負載-3	0.6	1.1	33.6	76.9	27.6	0.8
平衡-1	0.8	90.4	90.2	90.3	20.6	0.7
平衡-2	0.4	91.1	90.1	89.9	30.9	0.8
平衡-3	0.8	93.1	88.1	92.1	25.7	0.7
高負載-1	0.3	91.3	90.6	89.5	26.5	0.8
高負載-2	0.5	90.2	89.3	89.5	28.1	0.7
高負載-3	0.4	90.2	89.1	92.9	24.6	0.4

資料來源：本專題研究整理

4.3 網頁控制系統

基於使用者為考量，為了增加便捷性並減少記憶指令的需求，我們建立了網頁控制系統，並以所需使用動態調節之概念加入相關選項之功能，如下圖 四.1 與圖 四.2。



圖 四.1 控制系統首頁

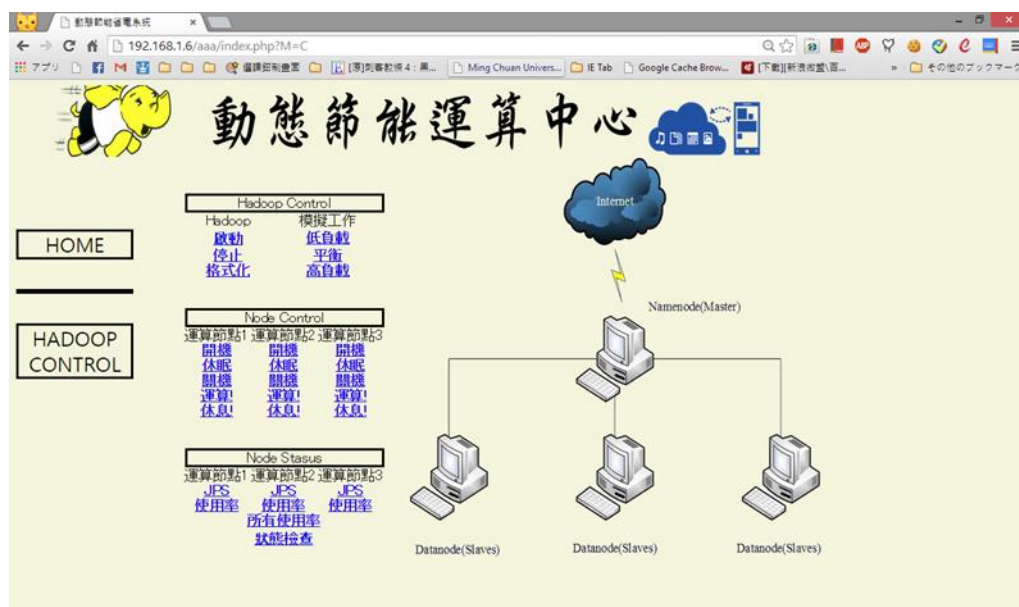


圖 四.2 系統功能選項

在製作網頁控制系統期間，我們針對以下幾個層面做討論並加入功能：Hadoop 基本控制層面、節點控制層面、節點狀態檢查層面與使用者介面層面。

4.3.1 Hadoop 基本控制層面

主要用以處理基本運作。如啟動、結束運算，還有針對每次開始運算前必須要做的格式化動作，如下圖 四.3。另外，由於本專題研究致力於動態節能層面，需要提供至少高負載與低負載兩種以上的模擬運算，以模擬企業在真實情況下接收到不同分量的工作並實施運算。因此，經討論與探查前一節實驗結果後在網頁介面上建立了低負載、平衡與高負載三種工作選項。分別對應到的是計算 pi 10 10、pi 100 100 與 pi 1000 100 此三種運算，如下圖 四.4、圖 四.5 與圖 四.6。

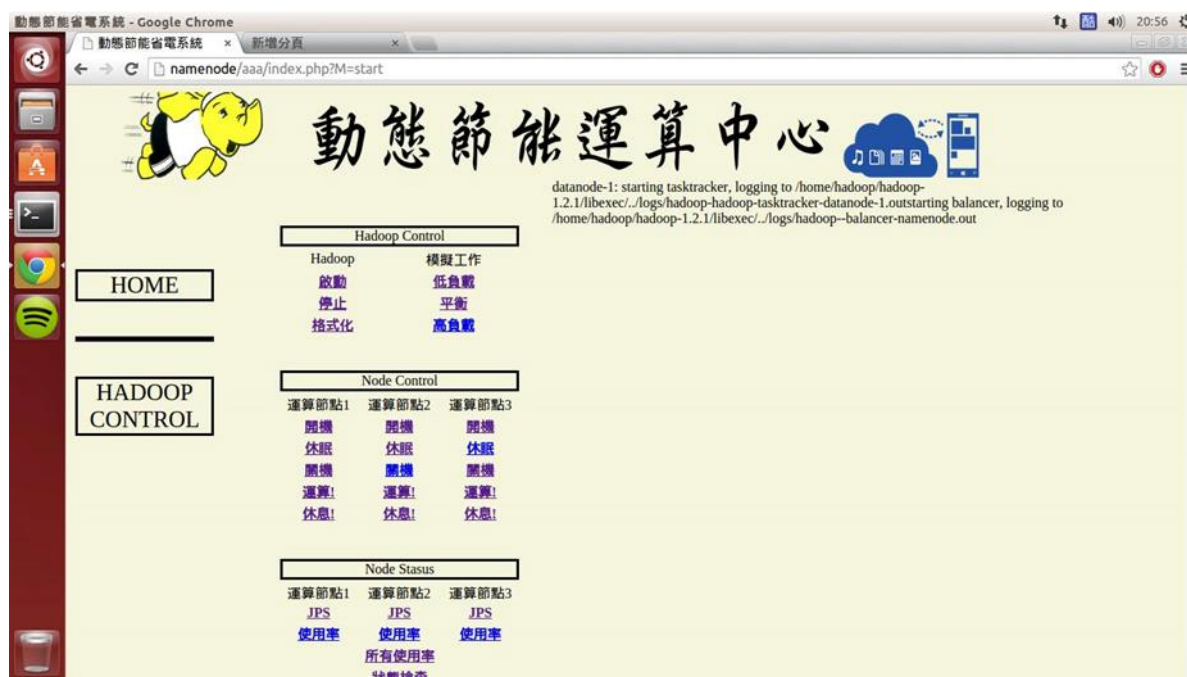


圖 四.3 啟動 Hadoop 運算

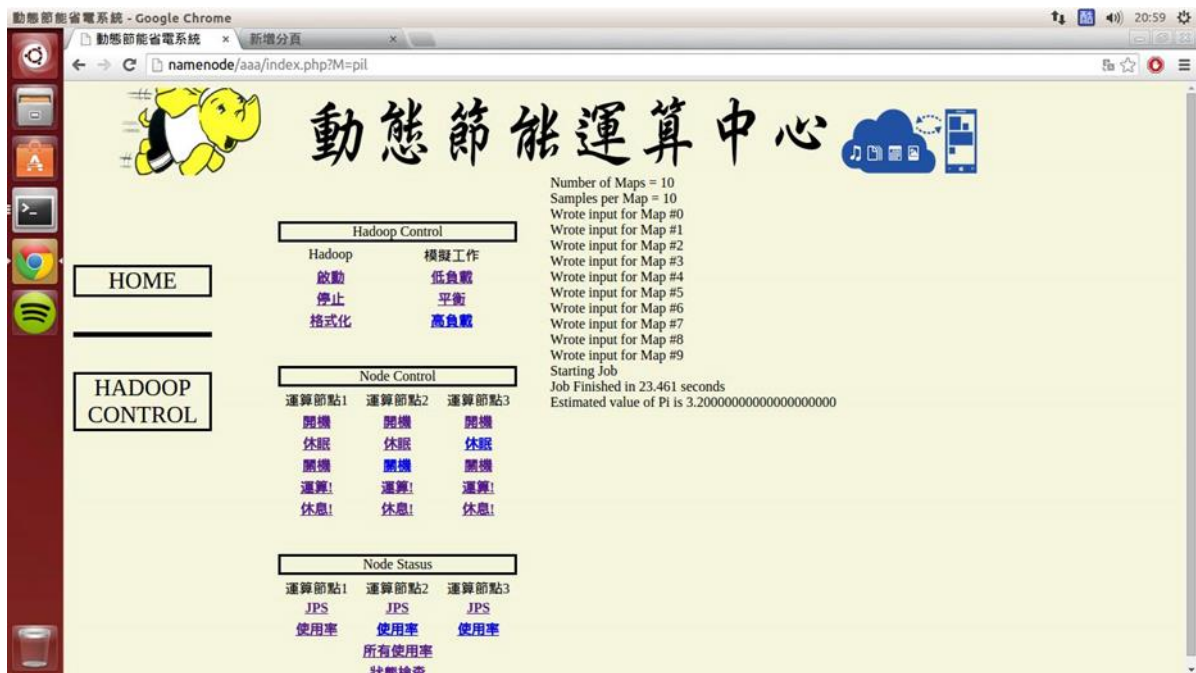


圖 四.4 pi 10 10 運算



圖 四.5 pi 100 100 運算



圖 四.6 pi 1000 1000 運算

4.3.2 節點控制層面

針對控制各節點之需求提供項目，並且以使用者慣用之瀏覽方式呈現。在動態期間，將會關閉不需要之節點，因此提供了關機與休眠兩種選項，一方面在使用者上可自行決策要使用哪種關閉節點模式，另外一方面經我們實際測試後，發現各有優點與缺點。針對關機的選項，開機較為迅速，但必須重新配置節點跟佈署。而休眠選項因開機時需回復狀態而較慢，但可立即上工，不須重新佈署。另外，針對使用開機與關機的部分，將會影響到 Hadoop 本身節點之運作，因而提供運算與休息之選項。

4.3.3 節點狀態檢查層面

針對系統狀態與各節點在 Hadoop 之運作等狀態提供查詢服務。其中，JPS 選項為監測各節點是否有正常開啟與運算。而使用率選項為檢視不同電腦之即時性 CPU 使用率，也是本次專題研究判斷動態之標準，如圖 四.7 與圖 四.8。狀態檢查選項則為清楚掌握各節點電腦之開機與否，如圖 四.9。如此一來，管理者只需要靠此網頁就能清楚並精準掌握各種狀態。



圖 四.7 單一電腦 CPU 使用率



圖 四.8 所有電腦 CPU 使用率

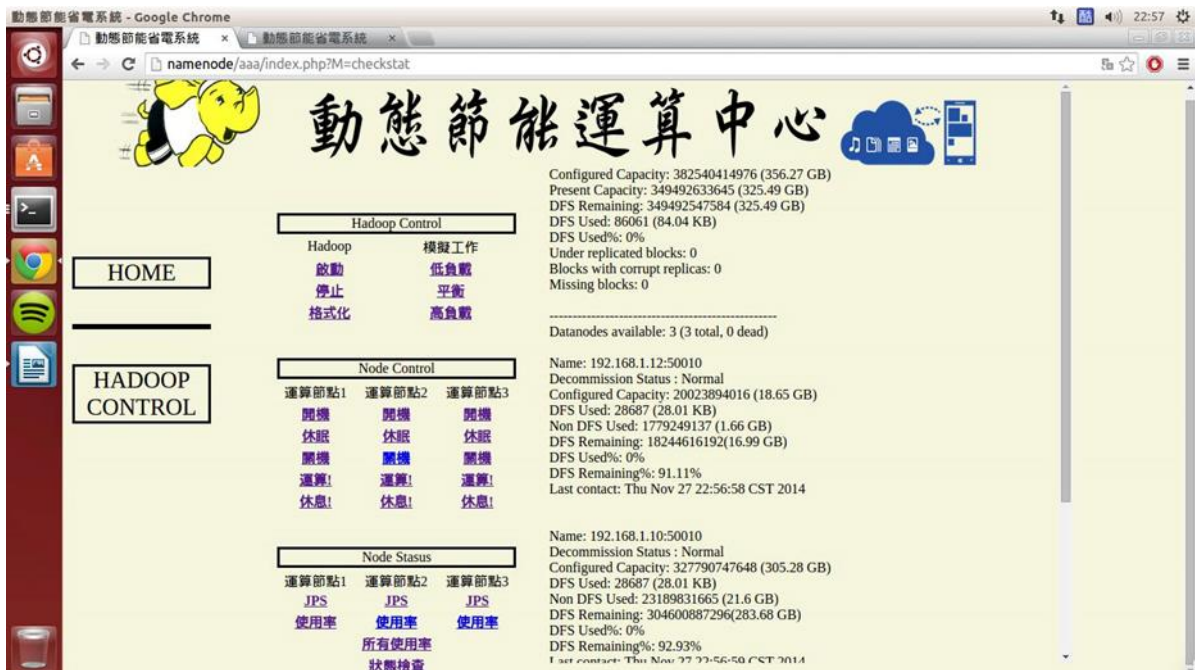


圖 四.9 狀態檢查-1

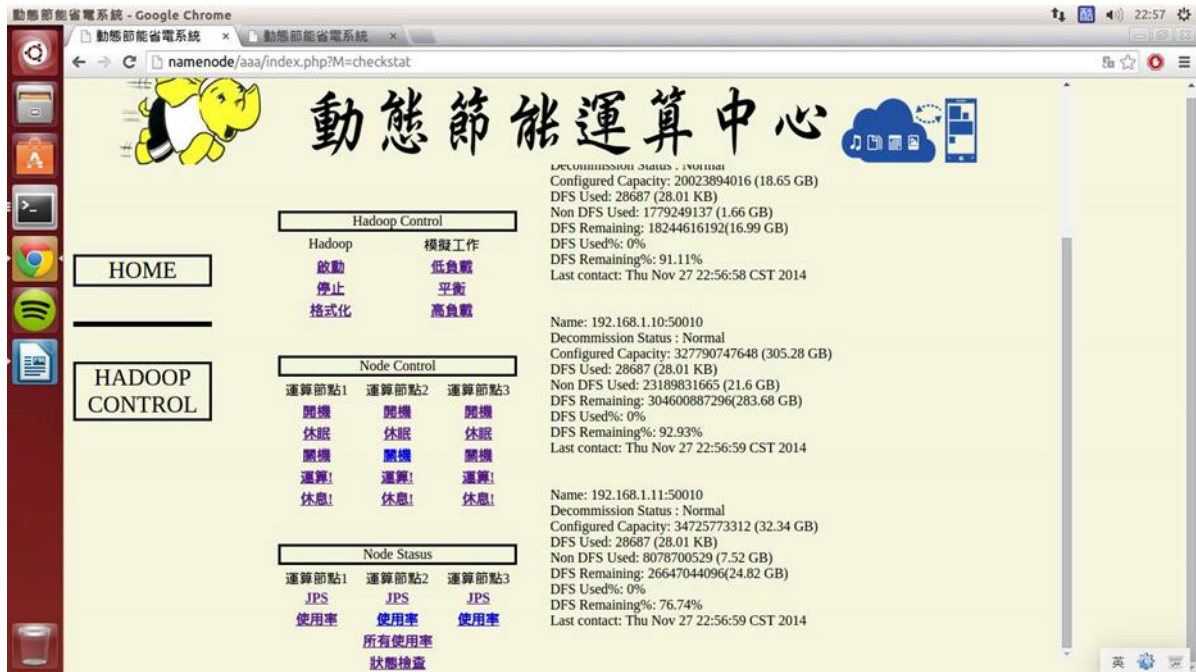


圖 四.10 狀態檢查-2

4.3.4 使用者介面層面

在討論與設計期間，考慮到未來此系統的使用者特性。使用者可能忙碌於工作而不一定可以隨時待在電腦身邊，建構了除電腦網路瀏覽顯示之介面外，另提供平板與手機之介面。藉由不同硬體裝置之螢幕大小，提供最適切與方便的介面瀏覽，如圖 四.11、圖 四.12 與圖 四.13。



圖 四.11 電腦端使用介面

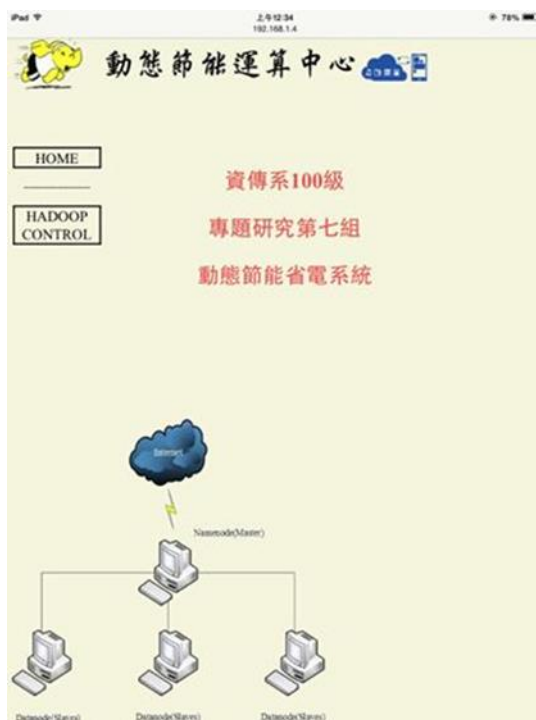


圖 四.12 平板端使用介面



圖 四.13 手機端使用介面

4.4 動態配置與比較

4.4.1 開關機與待機的差異

針對此部分我們在開關機與待機中做了嚴密的測試。在開機的部分取開機期間之最高值，在待機與關機中取平均值，如下表 四.6。其中，針對本次研究的極端情況，持續性的開關機，將以關機與開機最高直取平均做為決策值。比起穩定待機等待工作來說，Namenode 與 Datanode-1 以待機為佳，而 Datanode-2 與 Datanode-3 則是有開關機的設置為佳。同時間，本系統是以 Namenode 與 Datanode-1 皆開機的基礎狀態，因此確實會達到最好的省電狀態與效果。

表 四.6 待機與開關機比較表(W)

	開機	關機	待機	瞬間開關
Namenode	34	0.55	12.5	17.25
Datanode-1	37	0.75	15.5	18.87
Datanode-2	36	0.55	24	18.27
Datanode-3	26	0.75	17	13.37

資料來源：本專題研究整理

4.4.2 預設與動態配置計算之差異

此部分分成低負載、高負載與平衡之量測，低負載之判斷如圖 四.14 與表 四.7 可知，使用系統(動態配置)計算明顯優於原始(預設)計算，較為節電。

表 四.7 低負載運算之省電比較表

時間軸(秒)										開始
低負載-系統	30.5	30.6	30.5	30.8	30.5	31.1	30.5	44.4	49.6	41.2
低負載-原始	75.5	74.9	75	75.2	75.4	76.5	74.4	74.3	74.5	73.6
時間軸(秒)	—————>									
低負載-系統	48.3	46.2	51.8	51.9	52.6	61.7	54.1	52.1	51.7	52.8
低負載-原始	74.3	75.3	74.7	87.4	92.8	86.1	93.4	92.4	101	117
時間軸(秒)	—————>							原始結束		>
低負載-系統	51.6	52.6	36.3	32.4	32.6	32.4	40.2	32.1	47.5	44.3
低負載-原始	120	120	119	119	92.5	85.8	79.7	81.7	77.2	89.4
時間軸(秒)	—————>		系統結束							
低負載-系統	31.3	31.4	31.1	31.4	31.1	31	31.1	31.3	30.8	31.2
低負載-原始	90.3	77.6	75.5	75.8	74.7	74.6	75.8	74.5	74.9	74.6
	平均耗電量(W)					省電比				
低負載-系統	44.175									
低負載-原始	93.98421053									

資料來源：本專題研究整理

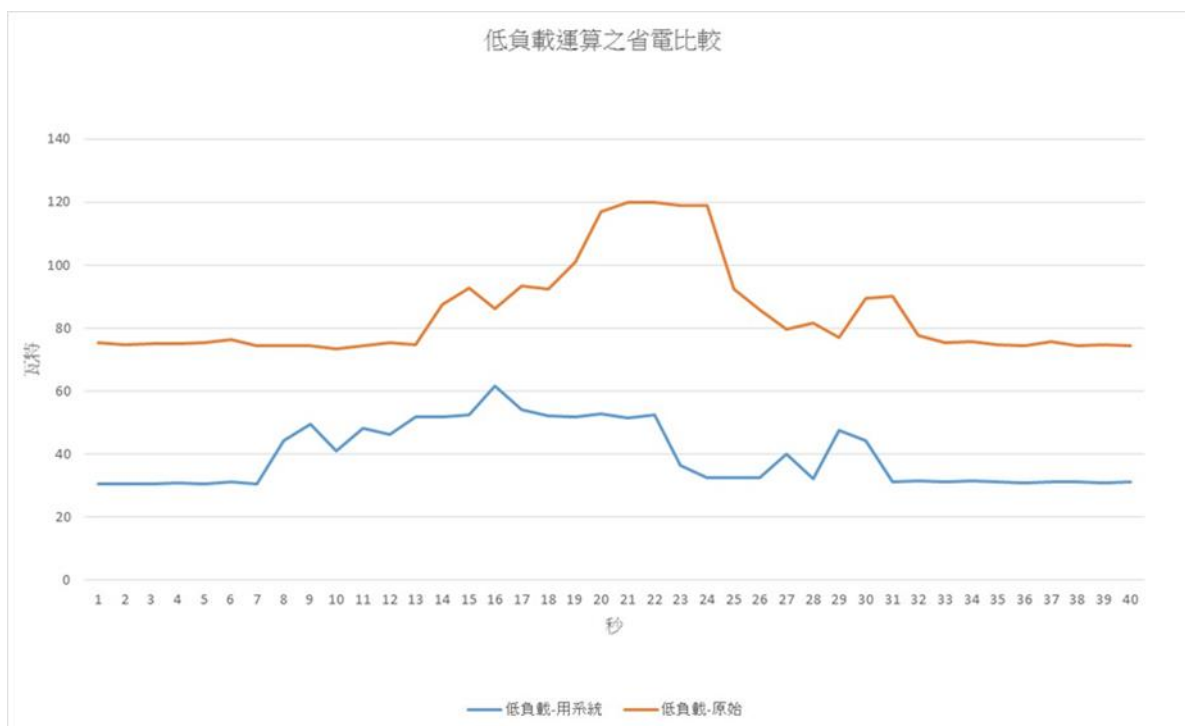


圖 四.14 低負載運算之省電比較

高負載如圖 四.15、圖 四.16、表 四.8 與表 四.9 所示，使用系統(動態配置)計算明顯優於原始(預設)計算，因計算需求大，呈現略為省電。

表 四.8 高負載運算之省電比較表

時間						開始	—————>			
高負載-系統	31.9	30.7	30.9	31	31.8	38.2	52.8	51.8	52	108
高負載-原始	74.5	73.2	73.7	75.5	74.9	84.4	84.2	119	118	120
時間	—————>									
高負載-系統	104	106	108	98	110	103	109	104	118	123
高負載-原始	118	118	120	120	120	119	119	120	121	120
時間	—————>									
高負載-系統	119	120	120	119	119	120	120	121	122	121
高負載-原始	121	120	120	120	125	121	120	120	121	120
時間	—————>					原始結束	—————>			系統結束
高負載-系統	120	121	121	120	121	120	120	121	122	121
高負載-原始	121	121	121	121	120	120	78	75.4	73	76
時間				平均耗電量(W)			省電比			
高負載-系統	94.2	38.9	31.3	108.3942857						
高負載-原始	72	75	74	117.8258065						
							8%			

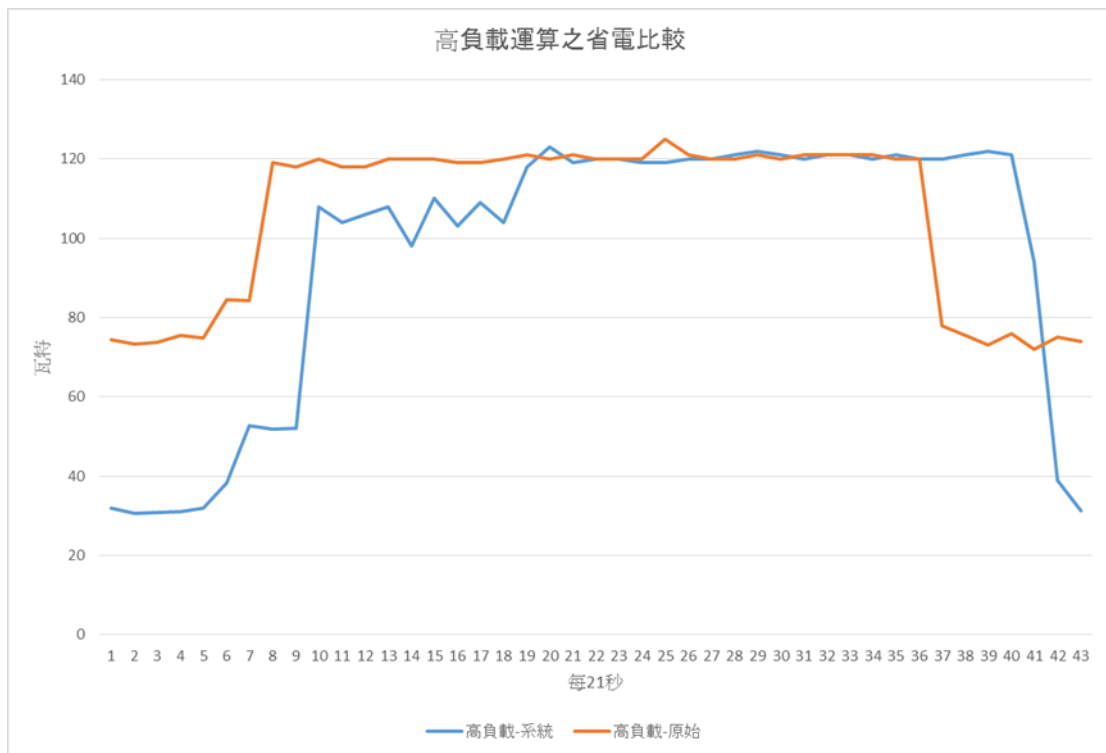


圖 四.15 高負載之省電比較

表 四.9 高負載運算之省電比較表 2

時間							開始	—————>		
高負載-動態	36.4	34.2	35.2	34.3	39.7	55.4	110	107	95.4	127
高負載-手動	33.3	32	32.7	34.6	33.6	33.2	41.8	46.2	54.9	53
高負載-原始	67.1	66.6	66.4	66.9	66.3	68.2	93.1	67	85.4	125
時間	—————>									
高負載-動態	73	98.5	67.3	104	106	131	124	132	131	130
高負載-手動	102	112	113	101	102	95.5	125	107	96.5	124
高負載-原始	127	128	128	129	127	127	125	128	129	125
時間	—————>					原始結束	—————>			
高負載-動態	124	130	129	130	129	129	130	130	132	78.4
高負載-手動	129	128	130	123	129	130	128	128	121	130
高負載-原始	126	130	129	106	133	131	75.2	68.6	67.1	66.7
時間	—————>		手動結束	—————>				動態結束		
高負載-動態	46.8	46	47	47.4	47	34.6	32.7	32.9	32.4	32.3
高負載-手動	129	130	70.3	72.8	71.7	70	73.6	69.8	70.2	71
高負載-原始	66.6	68.6	66.5	68.3	64.8	69.6	66.7	68.6	66.6	66.7
時間			平均耗電量(W)				省電比			
高負載-動態	33.1	33.8	97.25				19%			
高負載-手動	70	69.7	106.64				11%			
高負載-原始	64	66.3	119.93							

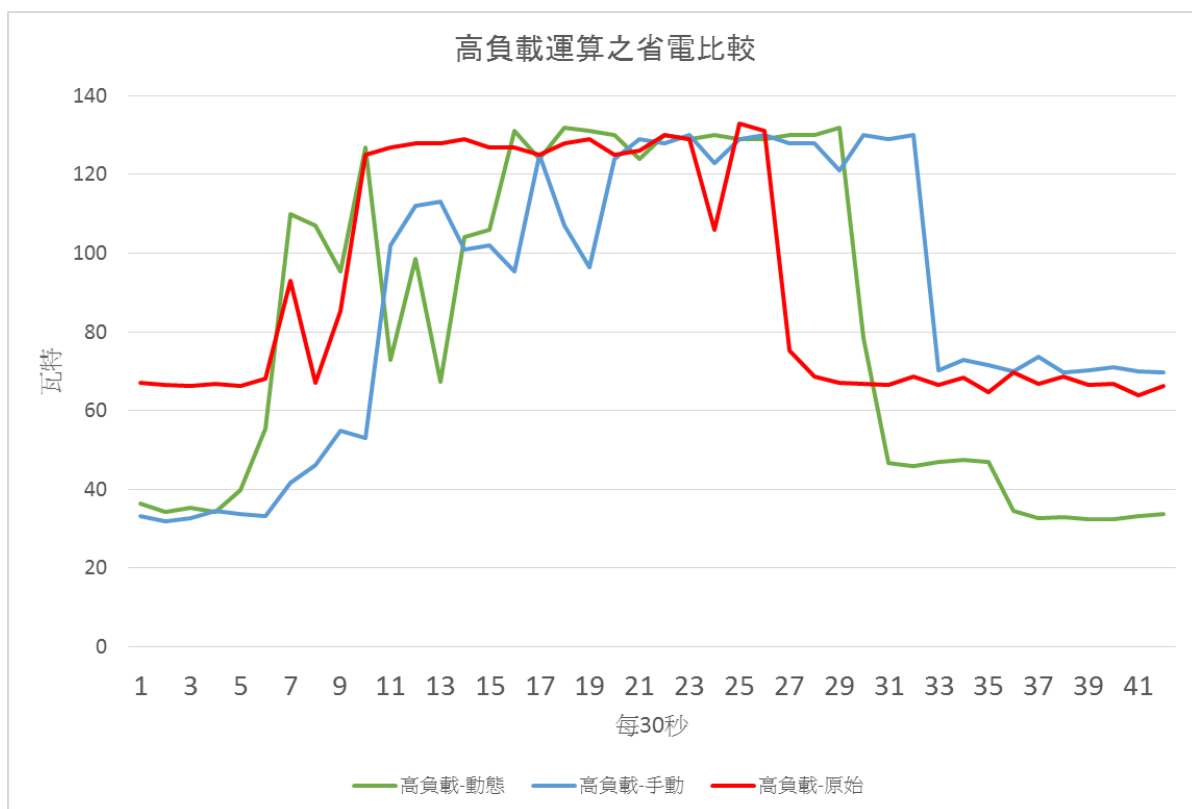


圖 四.16 高負載之省電比較

平衡如圖 四.17 與表 四.10 所示，使用系統(動態配置)計算明顯優於原始(預設)計算，較為節電。

表 四.10 平衡運算之省電比較表

時間											
平衡-系統	33.3	31.2	32.3	31.1	31.2	31.1	30.8	31.1	31.2	31.5	
平衡-原始	77.7	74.6	75	73.5	73.4	72.9	73.1	72.3	75.3	75	
時間	開始									➤	
平衡-系統	31.1	37.6	48	51.8	51.7	50.8	52.9	52.1	51.6	52	
平衡-原始	81	82.2	85.3	116	118	118	119	118	118	117	
時間											➤
平衡-系統	52.7	52.4	52	51.8	52.1	52	53	53.9	54	53	
平衡-原始	118	120	118	120	120	120	119	119	118	92.8	
時間	——➤	原始結束							➤	系統結束	
平衡-系統	53.6	52.4	53.1	51.8	51.8	53.4	51.9	52.1	53.3	52.5	
平衡-原始	76.7	79.5	78.3	77.1	74.2	75.3	74.6	74.3	74.5	75.8	
時間			平均耗電量(W)				省電比				
平衡-系統	34.9	47.2	51.08								
平衡-原始	76.6	74.3	108.7954545								

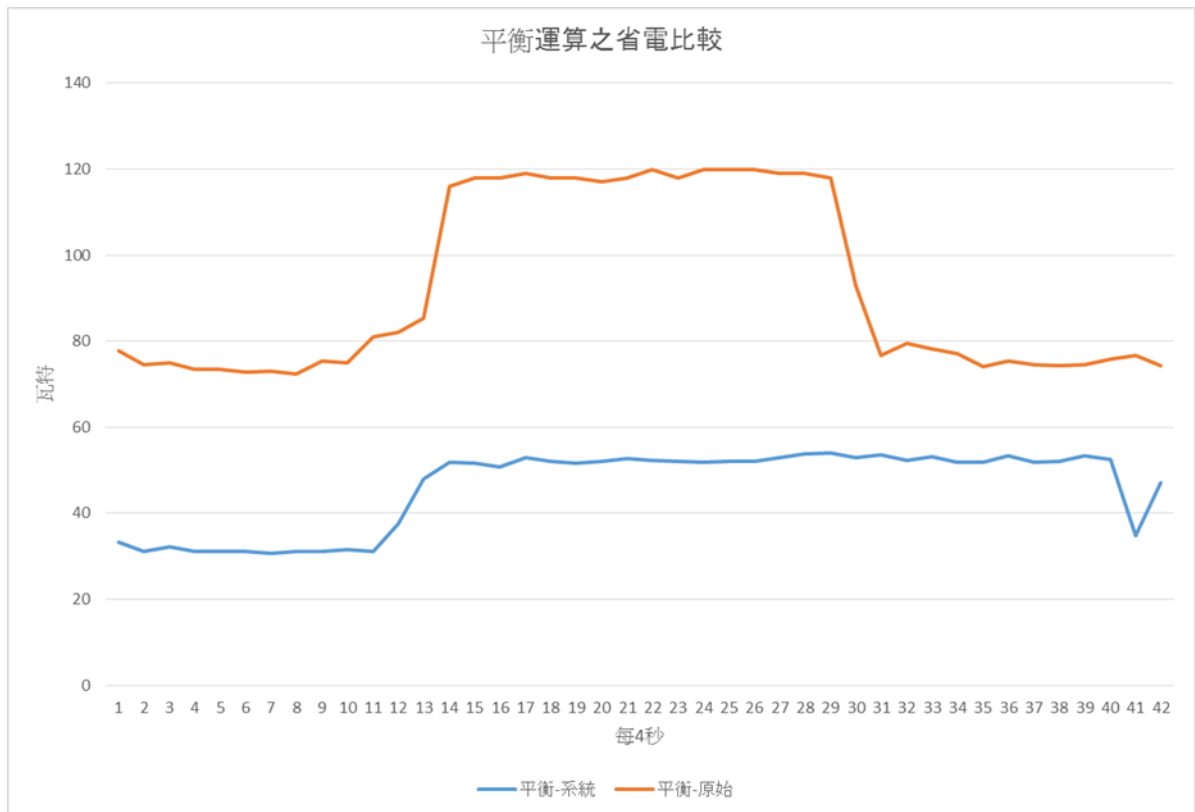


圖 四.17 平衡運算之省電比較

經由上面三項測試與研究後可知，當高負載與低負載時期的省電各節省了 8% 與 53% 的功率耗損，但相對來說，每次的運算時間需比以往的運算多出一兩秒，在這個資訊化的年代中，網路本身依然存在著線路傳輸等等的延滯時間。我們覺得，在每次運算的差異皆小於一分鐘的情況下，對於使用者是屬於可以容忍的範圍，而真正又能達到不減少運算下能減少耗能的狀態。

另外，除了比較極端的高負載與低負載之外，我們針對平衡運算也做了比較，事實證明，使用我們的系統在日常較常見的工作量下，依然擁有 53% 的良好功率節省效果，所以即使高負載的節省情況看似不如預期，由這個部分即可彌補省電效果較差的狀況。

4.4.3 動態配置之展示

由下表 四.11 與圖 四.18 可知，顯示各節點之每五秒 CPU 使用率，展現出動態配置之成果。其中，表格資料空白處表示該節點尚未開機運作，相對節能。

表 四.11 高負載模擬之動態釋義表

CPU-1	0.7	0.6	0.8	0.7	0.4	1.1	0.7	0.8	0.7	0.8	5.1	5	2.6	87.5	89.9
CPU-2															
CPU-3															
CPU-1	91	93	92	88	88	88	88	62	79	90	91	92.1	75	91.5	90.1
CPU-2						100	100	72	30						
CPU-3					100	100	100	78	100	100	98	100	100	100	100
CPU-1	86	91	89	90	90	92	88	90	90	0.8	8.6	90.8	41	0.7	87.6
CPU-2								74	84	100	100	84	75	75.1	100
CPU-3	100	100	100	100	100	100	100	99	100	48	2.7				
CPU-1	62	91	87	90	41	68	88	0.9	91	80	90	88.3	90	90.7	90.2
CPU-2	95	100	84	64	100	95	82	85	92	53	95	80.9	85	86.4	77.4
CPU-3			100	100	100	100	88	98	100	100	100	96.4	100	100	100
CPU-1	91	90	88	14	1	1.2	0.8	2	1.2	0.8	0.9	1.2	0.9	1.4	1.2
CPU-2	93	85	96	55	3.3										
CPU-3	100	100	100	3.3											
CPU-1	2.5	1	91	92	90	91	92	90	86	30	1.1	3.2	0.6	0.7	0.8
CPU-2									96	60	75	73.4	31		
CPU-3							100	100	100	100	100	1.3			
CPU-1	0.9	1.6	1.2	0.7	0.4	0.8	0.7	0.6	平均		47.48		總 CPU 平均耗能		
CPU-2									平均		79.32				
CPU-3									平均		91.37				

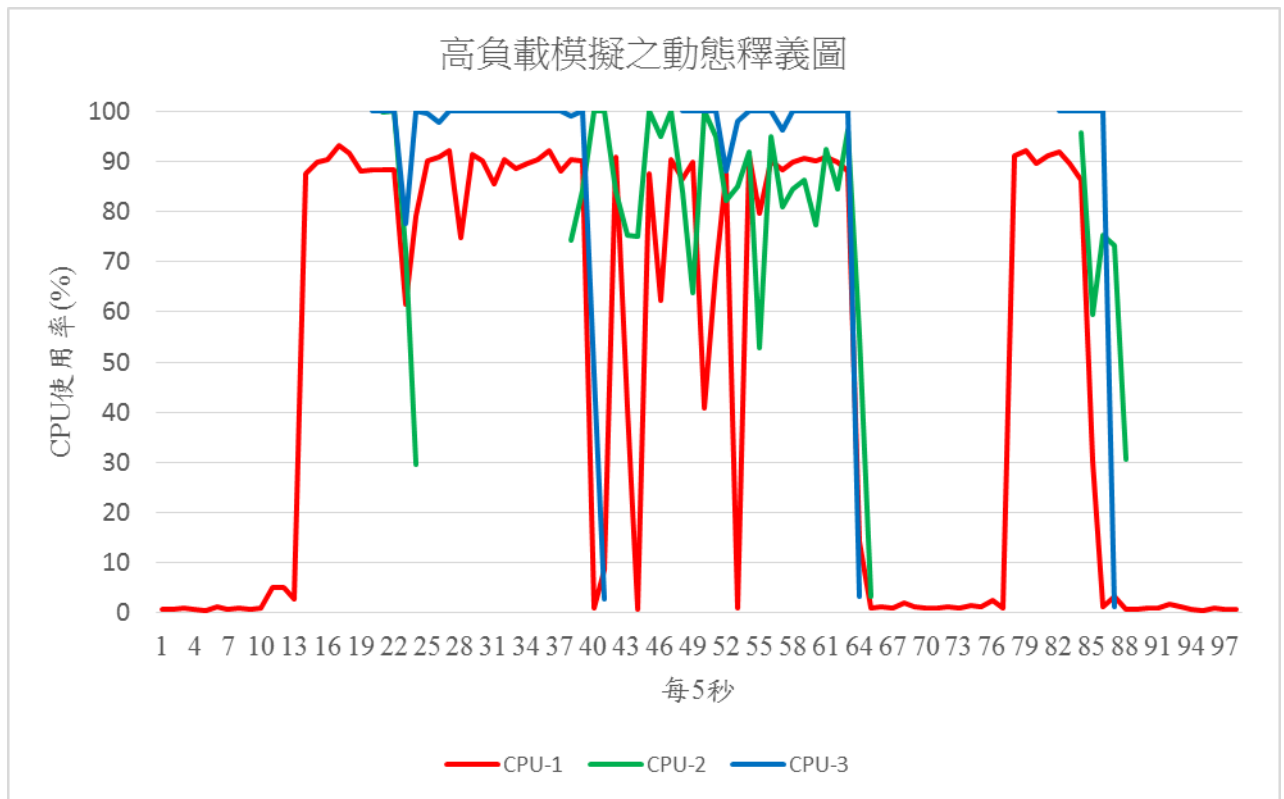


圖 四.18 高負載模擬之動態釋義圖

第五章 結論與未來工作

現今雲端系統的出現及發展帶給社會極大的變化，雲端服務已經融入我們日常生活及各個企業中，而將淘汰的設備與雲端系統融合正創造新的環境與龐大商機。另外，以個人家庭來說，結合現有的電腦設施建立出與本研究架構相仿的個人雲。用在處理日常生活所需，達到善加利用雲端運算提升個人生活品質，同時也能夠達到市井小民對現階段的高物價獲得節能省電的措施。

故本研究畫以 Hadoop 雲端架構為基礎，針對小型至中大型的雲端環境，利用 PHP 來讀取雲端架構中各運算節點的主機狀態，再透過網頁控制調整雲端架構中運算分配的模式，在功率消耗和時間效率中取得最大平衡，減少不必要的資源消耗。在本次研究中，我們發現在低負載與平衡負載工作中，使用合併運算(Consolidation)較能達到動態節能的目的。另外針對高負載工作中使用近似負載平衡(Load Balance)亦可達到相同的目的與功效。

實際應用層面，我們希望本系統除個人與家庭為使用之對象外，更注重於中小企業之私人雲建置與維護。藉由我們的系統套用於私人雲上，不僅不會影響到與以往相同的運作效率，更能幫助他們節省不必要的資源浪費，真是如虎添翼。

如今只達到手動動態節能，基礎上讓主控與被控電腦之間有達到基本的手動控制與溝通，配合已知的法則作為人工判斷工作量之大小並動態調節系統。同時在運作時期能即時顯示各台節點的運作狀況與 CPU 使用率。期望未來能將手動的部分保留，額外增加一件自動動態偵測並自動調控所需啟用節點與節點數之功能。

參考文獻

- [1] “Hadoop,” [線上]. Available: <http://hadoop.apache.org/>. [存取日期: 2014].
- [2] “CPU,” [線上]. Available: <http://zh.wikipedia.org/wiki/%E4%B8%AD%E5%A4%AE%E5%A4%84%E7%90%86%E5%99%A8>. [存取日期: 2014].
- [3] WIKI, “蒙地卡羅法,” [線上]. Available: <http://zh.wikipedia.org/wiki/%E8%92%99%E5%9C%B0%E5%8D%A1%E7%BE%85%E6%96%B9%E6%B3%952>. [存取日期: 2014].
- [4] “PHP,” [線上]. Available: <http://www.php.net/>. [存取日期: 2014].
- [5] 陳冠宏, “利用 Hadoop 建構具備資源管控之雲端系統,” 銘傳大學, 2003.
- [6] WIKI, “雲端運算,” [線上]. Available: <http://zh.wikipedia.org/wiki/%E9%9B%B2%E7%AB%AF%E9%81%8B%E7%AE%97>. [存取日期: 2014].
- [7] Google, “Google 雲端硬碟,” Google, [線上]. Available: <https://support.google.com/drive/?hl=zh-Hant#topic=14940>. [存取日期: 2014].
- [8] Google, “企業資訊,” Google, [線上]. Available: <http://www.google.com/about/company/>. [存取日期: 2014].

- [9] Google, “Chrome Book,” Google, [線上]. Available: <http://www.google.com/intl/en/chrome/devices/chromebooks.html>. [存取日期: 2014].
- [10] 江政哲、張迺貞, “初探雲端運算,” 於 第十屆海峽兩岸圖書資訊學學術研討會, 2012.
- [11] “NIST,” NIST, [線上]. Available: <http://www.nist.gov/>. [存取日期: 2014].
- [12] “BYOC,” [線上]. Available: <http://www.ithome.com.tw/node/847522>. [存取日期: 2014].
- [13] 陳滢, 雲端策略：雲端運算與虛擬化技術, 天下雜誌, 2012.
- [14] VMware, “VMware Workstation,” VMware, [線上]. Available: <http://www.vmware.com/tw/products/workstation/>. [存取日期: 2014].
- [15] Microsoft, “Windows,” Microsoft, [線上]. Available: <http://windows.microsoft.com/zh-tw/windows/home>. [存取日期: 2014].
- [16] GNU, “Linux,” [線上]. Available: <http://www.linux.org/>. [存取日期: 2014].
- [17] Oracle, “VirtualBox,” Oracle, [線上]. Available: <https://www.virtualbox.org/>. [存取日期: 2014].
- [18] Microsoft, “Hyper-V,” Microsoft, [線上]. Available:

- <http://technet.microsoft.com/zh-tw/library/hh831531.aspx>. [存取日期: 2014].
- [19] Open Group, “UNIX,” [線上]. Available: <http://www.unix.org/>. [存取日期: 2014].
- [20] WIKI, “Linux 分支,” [線上]. Available: <http://zh.wikipedia.org/wiki/Linux%E5%8F%91%E8%A1%8C%E7%89%88%E5%88%97%E8%A1%A8>. [存取日期: 2014].
- [21] Apache, “Thrift,” Apache, [線上]. Available: <http://thrift.apache.org/>. [存取日期: 2014].
- [22] Apache, “HBase,” Apache, [線上]. Available: <http://hbase.apache.org/>. [存取日期: 2014].
- [23] WIKI, “LAMP,” [線上]. Available: <http://zh.wikipedia.org/wiki/LAMP>. [存取日期: 2014].
- [24] Apache, “Apache,” Apache, [線上]. Available: <http://httpd.apache.org/>. [存取日期: 2014].
- [25] M. Foundation, “MariaDB,” MariaDB Foundation, [線上]. Available: <https://mariadb.org/>. [存取日期: 2014].
- [26] Oracle, “MySQL,” Oracle, [線上]. Available: <http://www.mysql.com/>. [存取日期: 2014].
- [27] 鄭健忠, “雲端運算之主機資源技術及比較,” 銘傳大學, 2011.
- [28] Linux Community, “tasksel,” [線上]. Available:

<http://www.linuxidc.com/Linux/2008-01/10292.htm>. [存取日期: 2014].

[29] the OpenBSD Project, “OpenSSH,” [線上]. Available: <http://www.openssh.com/>. [存取日期: 2014].

[30] Oracle, “JAVA,” Oracle, [線上]. Available: http://java.com/zh_TW/download/help/. [存取日期: 2014].

[31] Ubuntu Community, “Ubuntu Linux,” [線上]. Available: <http://www.ubuntu-tw.org/>. [存取日期: 2014].

附錄

工作分配表

	邱盛宇	徐一中	劉宇捷	黃紹維
資料尋找	●	●	●	●
文件撰寫	●	●	●	
文件校對	●	●		●
文件排版				●
程式撰寫	●		●	
研究架構	●	●	●	●
動態製作	●	●		●

工作時程表

工作項目	民國102年 3月	民國102年 4月	民國102年 5月	民國102年 6月	民國102年 7月	民國102年 8月	民國102年 9月	民國102年 10月	民國102年 11月	民國102年 12月	民國103年 1月	民國103年 2月	民國103年 3月	民國103年 4月	民國103年 5月	民國103年 6月	民國103年 7月	民國103年 8月	民國103年 9月	民國103年 10月	民國103年 11月
專題主題																					
討論																					
導師																					
確切指導																					
老師																					
訂立專題 主題																					
環境建置																					
環境建置																					
問題解決																					
安裝																					
hadoop																					
臨p測資- 基礎																					
臨p測資- 變換																					
臨p測資- 多機																					
計算耗電 量																					
建置hdp 網頁																					
問題解決																					
撰寫初稿 文件																					
繪製專題 影片																					
閱讀參考 文獻																					
撰寫摘要 文件																					
測試外觀																					
功能 測試確認																					
照像																					
查詢功能 資料																					
與老師討 論																					
與李進學 長討論																					
與謝乾學 長討論																					