研究以 Linux Container 為基礎之資料倉儲叢集的實現與不同網路架構下之效能分析

[1]林俊宏 [2]吳冠賢
國立中山大學資訊工程學系
[1]lin@cse.nsysu.edu.tw, [2]juliansifgno@gmail.com

摘要

雲端運算是目前最熱門的話題之一，時至今日，資料成長的速度越趨快速，大資料的議題更是成為資料探勘中的顯學，如何能加快資料處理的速度便成為一個重要的議題。

本論文以資料的結構化程度、網路對資料運算時間上的影響、如何在相同的成本之下建構出可以發揮硬體成本的雲端運算叢集做探討與研究。

最後，本論文以 Linux Container 為建立資料倉儲環境的基礎，探討此類型虛擬環境是否可以取得實體機器的所有運算資源，除此之外，在硬體資源容許的情況下，增加 Linux Container 虛擬機器的數量，探討在這樣的情況下是否更能發揮出硬體應有的運算效能。

關鍵字：資料倉儲、Hadoop、Big Data、Linux Container、網路卡對接。

Abstract

Cloud computing is currently one of the popular topics. Nowadays, the pace of data growth is much faster than before. How to accelerate the speed of data processing has become an important issue.

In this paper, we will discuss and research the degree of structure in the data, the impact of network transmission on computing, and how to construct the most efficient cloud computing cluster under the same hardware costs.

Finally, we build Linux-Container-based data warehouse to discuss whether Linux Container can get whole the real host hardware resources, in addition, under the case of hardware resources allowance, we increase the number of Linux Container virtual machines to examine under this situation whether the hardware should play better computing performance.

Keywords: Hadoop、Data Warehouse、Linux Container

1. 前言

目前市面上有許多利用 Hadoop 為基礎建構的資料倉儲軟體，而 Hadoop 的特性就是大量利用成本低廉的機器組成運算叢集處理大資料的工作，再加上最近興起的 Linux Container 虛擬環境技術，利用虛擬環境的方式模擬全新的作業系統，可以分享硬體資源，讓單一機器內可以有多個虛擬環境，與過往的將硬體資源獨立切割出來的虛擬機器相比更具有彈性，更可以將硬體資源分配給需要的虛擬環境使用，以加快運算的速度；另外一點則是網路傳輸的瓶頸，當網路交換設備因為需要服務的機器越多，就越有可能達到其可處理的效能瓶頸，為了解決這一問題，更新設備會是一種方式，但是基於成本的考量，如果機器在硬體上有尚未使用到的網路卡，則利用機器的網路卡把彼此對接，是否能達到降低網路交換設備的負擔，並以此方式降低成本，也是本論文要探討與驗證的目標。

2. 相關研究

自從 MapReduce[1]與 HDFS[2]的分散式架構發表後，Hadoop 便開始蓬勃發展，目前在大數據分析方面已有不少論文、網站在探討如何利用 Hadoop 建構運算叢集，或是分析如何加快 Hadoop 的速度，目前也有不少資料倉儲[3]軟體也是建構在 Hadoop 的基礎之上，本論文則是採用 Cloudera[4]公司推出的 Open Source 資料倉儲軟體 CDH(Cloudera's Distribution including Apache Hadoop)作為各種實驗環境的軟體測試平台。

LXC[5][6][7]全名為 Linux Container，是一套建立虛擬環境的軟體套件，有別於需要透過 Hypervisor 的架構管理的虛擬化技術，Linux Container 透過 chroot 的概念，把作業系統切割開來，所以從虛擬機器的角度來看，Linux Container 並非完全擁有獨立的資源，而是利用隔離的方式建立出虛擬機器，需要硬體資源時，實體機器上的資源可以動態的配置資源給有所需求的 Linux Container，也由於可以直接使用到實體機器上的所有軟硬體資源，再加上可以共享實體機器上的所有函式庫，所以效能在理論上可以有更好的表現。

3. 系統環境規格

所有測試環境都在表 3-1 與表 3-2 所描述的硬體規格的機器下建立；所有的測試環境使用的軟體如表 3-3 所示；所有的測試環境佈署的 Hadoop 服務如表 3-4 所示。

表 3-1 實驗機器硬體規格表

	Hitachi CR210H	Dell OptiPlex9020
處理器型號	Intel® Xeon® E5-2620 2.00GHz	Intel® Core™ i7-4770 3.4GHz
處理器數量	4	1
核心數	6	8
總核心數	24	8
記憶體	16 GB	4 / 16 GB
硬碟規格	300 GB Raid-0 10000 RPM	1 TB 7200 RPM
網路界面	2 Dual port GbE	1 port GbE

表 3-2 網路交換設備規格表

	HP ProCurve Switch 3500yl-48G	D-Link DE-809TC
連接埠	44 10/100/1000 ports	8 10/100 ports

表 3-3Hadoop 與 LXC 相關軟體版本

套件名稱	版本資訊
CDH	5.0.1
Apache Hadoop	2.3.0
Apache Hive	0.12.0
Apache Pig	0.12.0
Apache Zookeeper	3.4.5
Cloudera Impala	1.3.1
Hue	3.5.0
Linux Container	1.0.4
Bridge-utils	1.5

表 3-4Hadoop 服務配置表

節點類型	佈署服務
管理節點	HDFS NameNode HDFS SecondaryNameNode Hive Gateway Hive Metastore Server HiveServer2 Hue Server Impala Catalog Server Impala StateStore Oozie Server Sqoop 2 Server Cloudera Management Service Alert Publisher Cloudera Management Service Event Server Cloudera Management Service Host Monitor Cloudera Management Service Service Monitor YARN(MR2 Included) JobHistory Server YARN(MR2 Included) ResourceManager
計算節點	HDFS DataNode Hive Gateway Impala Daemon YARN(MR2 Included) NodeManager Zookeeper Server

4. 實驗設計與數據說明

4.1 資料型態

測試資料分為結構化資料如圖 4-1 與非結構化資料如圖 4-2 兩種；結構化資料就是在尚未處理，或是在蒐集的階段，會考慮到未來想取得的資料內容而定義要蒐集的資料型態，並逐一且有一定格式蒐集的資料，然而可能在蒐集的途中，資料內容可能過於龐大，且有可能同一份資料要達到多種用途的目的，所以需要在另外對原始的資料做處理，以達成所要取得的資訊；非結構化資料指的是欄位長度不固定，且每格欄位又可以由重複或不重複、長度不一定的子欄位構成，像是文字、圖片、影像、PDF、電子郵件、網頁等檔案，通常會有這樣類型的資料是因為在資料蒐集的初期因為還無法明確的定義出哪些資料是需要被保留，哪些則是可以刪除，所以會將所有的資料一律保留下來，並等到未來一旦發現需要取得某些資訊時，則在一併從中挖取有用的資料。

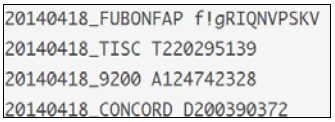

圖 4-1 結構化資料範例

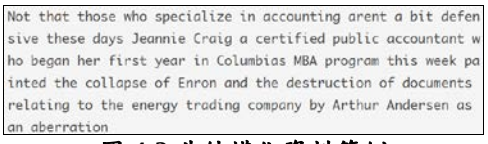

圖 4-2 非結構化資料範例

4.2 測試程式說明

測試程式將針對上述兩種測試資料由四種語言撰寫測試程式，分別為 Java、Pig、Hive 與 Impala，每組測試數次並求其執行時間的平均值，處理的目標分為，結構化資料是計算檔案中各公司當日各有多少不同的使用者上線，而非結構化資料則求其各單字在檔案中出現的次數。

Java 結構化資料測試程式說明

結構化資料如圖 4-1 中所示，考慮到同一位使用者可能在同一日上線許多次，所以要計算當日有多少不同使用者上線需要最少做兩次 MapReduce，以下將分述每次 MapReduce 要達到的目的。

- 第一次 MapReduce：將重複的使用者刪除，其運算流程如圖 4-3 所示。

讀檔案的時候以行為單位依序讀取資料，如圖 4-3 的第一階段；由於 MapReduce 的核心觀念就是利用大量的運算單位進行相同的事情，所以資料會被許多不同的 Mapper 讀取，如圖 4-3 的第二階段；接著為了達到將相同的使用者去除掉的目的，所以在第三階段我利用讀取到的資料設為 Key 值，也就是第三階段粗體字的部份，Value 值的部份則是將 Key 值以空白切割，並取其前段的部份，前段資料就是整行資料代表公司的部份，以其代表 Value 值；建立出 Key-Value 組合後，將有相同 Key 值的資料合併在一起，如圖 4-3 的第四階段所示，由於 Key 值的意義是公司與使用者的組合，這也就代表擁有相同 Key 值就會是同一位使用者，所以在這樣的情況下，不管同 Key 值的群組裡有多少筆資料，Reducer 都可以將其視為僅有一筆資料，如圖 4-3 的第五階

段所示，最後，再將所有的結果集合起來成為下一次 MapReduce 階段的輸入資料，因為使用者的名稱在這裡已經不會再被使用到，所以只要保留公司名稱即可，即同一位使用者利用其公司代號代表。

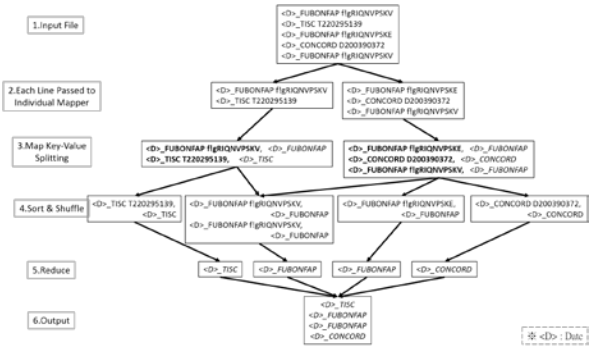

圖 4-3 第一次 MapReduce 流程圖

- 第二次 MapReduce：計算同一公司當日有多少使用者上線，其運算流程如圖 4-4 所示。

由圖 4-3 的第六階段取得的輸出結果當作這次的輸入來源，如圖 4-4 的第一階段所示；如圖 4-4 的第二階段，所有的 Mapper 取得資料後，開始進入第三階段，由於在此次 MapReduce 的目的是要計算出各有多少位使用者在同一天上線，且由第一次 MapReduce 取到的結果是每位使用者所代表的公司代號（圖 1-5 的第一階段為例，其中有兩項 <D>_FUBONFAP，此為兩位不同的使用者），每項賦予其 Value 值為 1；第四階段則是將相同 Key 值的資料集合在一起；第五階段則開始計算此集合中共有多少個項目，由於在第四階段每一項資料的 Value 值都是 1，所以只要將此集合中的 Value 值全部相加即可得到結果，如圖 4-4 的第五階段所示，最後將所有的資料整合在一起如圖 4-4 的第六階段所示，就可得到結果。

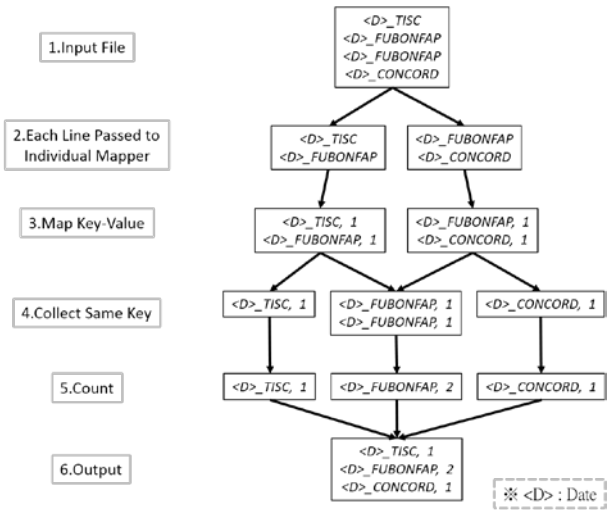

圖 4-4 第二次 MapReduce 流程圖

Java 非結構化資料測試程式說明

非結構化資料測試程式以 Cloudera 官方提供的程式碼第一版[8]為例執行。

4.3 測試環境架構

第一組與第二組測試環境各有四台 Hitachi 實體機器，其架構如圖 4-5 所示；兩組測試環境都是由 Hitachi-1 負責執行/安裝所有管理節點相關的服務，其餘三台實體機器負責執行/安裝所有計算節點相關的服務。

兩組測試環境不一樣的地方在於網路交換設備的型號

- 測試環境一網路交換設備型號：D-link DE-809TC
- 測試環境二網路交換設備型號：HP ProCurve Switch 3500yl-48G

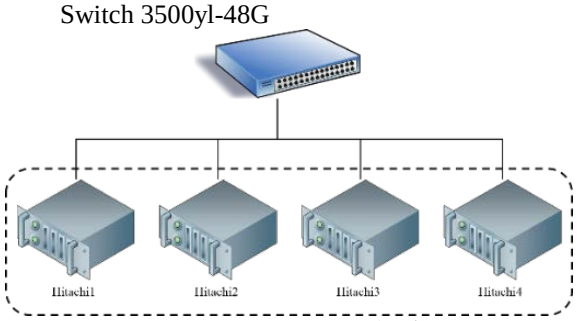

圖 4-5 測試環境一/二架構圖

第三組測試環境有四台 Hitachi 實體機器，其架構如圖 4-6 所示；此環境由 Hitachi-1 負責執行/安裝所有管理節點相關的服務，其餘三台實體機器負責執行/安裝所有計算節點相關的服務，此組測試環境除了使用 HP 的網路交換設備建立對外網路，內部四台機器更以剩下尚未用到的網路孔彼此直接對接，如圖 4-6 中的黑色粗線所示。

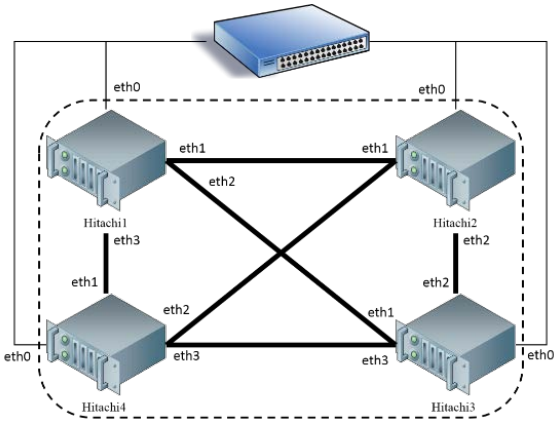

圖 4-6 測試環境三架構圖

第四組測試環境是由四台虛擬機器所組成，此四台虛擬機器分別建立在所有的 Hitachi 機器中，其架構如圖 4-7 所示，此環境由 Hitachi1-Container1 負責執行/安裝所有管理節點相關的服務，其餘三台虛擬機器負責執行/安裝所有計算節點相關的服務，在此測試環境中所使用的網路交換設備為 HP ProCurve Switch 3500yl-48G。

在 Container 網路的部份，由於 Hadoop 要求的是固定 IP，雖然 Container 一旦被執行後，會被賦予一組 IP，但是此 IP 位置會隨著 Container 重新啟動，而 IP 也會跟著被重新賦予一組新的 IP，所以為了讓 IP 是固定的，另外建立一組命名為 br0 的 Bridge，利用此 Bridge 達到除了自己可以有固定的 IP，也可以直接控制底層網路卡而有對外的功能，其 Bridge 架構圖如圖 4-8 所示。

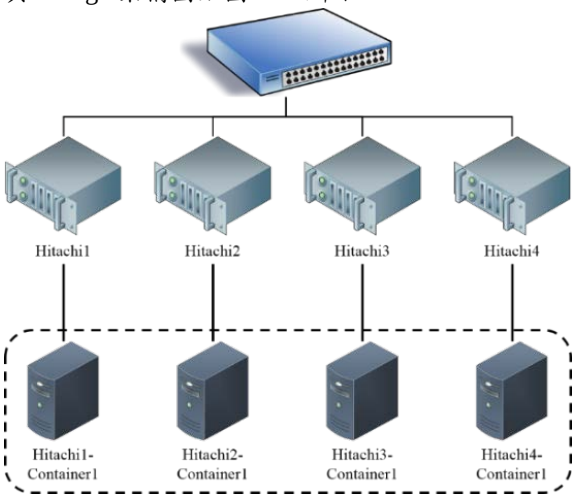

圖 4-7 測試環境四架構圖

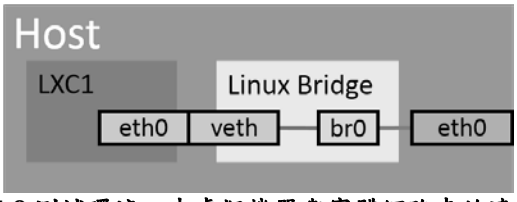

圖 4-8 測試環境四中虛擬機器與實體網路卡的連結示意圖

第五組測試環境與第四組相似，不同的地方在於實體機器彼此還利用網路卡直接對接。

由於此測試環境需要作到對接的功能，讓各實體機器內的 Container 可以直接互相溝通，所以 Container 內部需要設置路由表。

測試環境六的架構與測試環境四一樣，僅在測試機器不同，在測試環境六中所使用的實體機器為 Dell[1-4]，虛擬機器為 Dell[1-4]-Container1。

測試環境七的實體機器是由 Dell[1-4]、虛擬機器由 Dell[1-4]-Container[1-2]組成，測試環境八的實體機器是由 Hitachi[1-4]、虛擬機器由 Hitachi[1-4]-Container[1-2]組成，這兩組測試環境分別由 Dell1-Container1 與 Hitachi1-Container1 負責執行/安裝所有控制節點相關的服務，其餘七台虛擬機器負責執行/安裝所有計算節點相關的服務，在此測試環境中所使用的網路交換設備為 HP ProCurve Switch 3500yl-48G。

最後兩組測試環境則是在各實體機器中安裝四台 Linux Container 虛擬機器，測試環境九是由建立在 Dell[1-4]下的 Dell[1-4]-Container[1-4]所組成的群集，測試環境十則是由建立在 Hitachi[1-4]下的 Hitachi[1-4]-Container[1-4]所組成的群集，兩組測試環境分別由 Dell1-Container1 與 Hitachi1-Container1 負責執行/安裝所有控制節點相關的服務，其餘十五台虛擬機器負責執行/安裝所有計算節點相關的服務，在此測試環境中所使用的網路交換設備為 HP ProCurve Switch 3500yl-48G。

5. 實驗結果

5.1 實驗一：環境一 / 二 / 三

5.1.1 實驗目的

實驗一的實驗目的因為在測試環境一、測試環境二與測試環境三這三組測試環境中，所使用的網路交換設備或是網路傳輸的方式不同，比較這三種不同的網路傳輸方式對運算時間的影響。

從測試環境一與測試環境二比較兩種不同等級的網路交換設備對於運算執行時間的影響情況；從測試環境二與測試環境三比較利用網路交換設備傳輸資料的方式與利用網路對接傳輸資料的方式對於運算執行時間的影響情況。

5.1.2 實驗結果說明

實驗一各項測試程式執行時間數據圖如圖 5-1 所示，詳細的執行時間如表 5-1 所示。

在此實驗中，結構化資料的執行目標是必須把相同使用者多次登入的情況刪除，因此各個計算節點必須互相溝通告知哪些使用者有多次登入的情況，所以網路的傳輸便顯現出其重要性，由於測試環境一中的網路交換設備與測試環境二的網路交換設備在相比之下，效能有明顯的差距，也因此執行時間也有明顯的差別，甚至在 Impala 語言的情況，由於其運算速度比其他語言更快，然而其結果卻與 Hive 或是 Java 這兩種語言的執行時間相當，更證明了整體執行時間的運算瓶頸是在網路傳輸這部份；另外，我們也發現，在測試環境三中的網路對接情況下，其執行時間也與測試環境二的網路交換設備相當，可以顯示出網路對接除了降低網路交換設備的硬體成本外，傳輸速度上也有不錯的結果，假如未來在測試環境二的網路交換設備中，其所有的網路孔都完全被使用，可能發生網路傳輸的瓶頸在網路交換設備，當此情況發生時，或許網路對接這就會是可以解決這樣問題的一種方式。

在實驗的程式中，由於非結構化資料執行目標是計算各單詞出現的次數，所以各個計算節點僅需要計算自己被分配到的資料哪些單詞出現過幾次的，各節點的運算速度反而才佔據整個執行時間的最大比重，換句話說，非結構化資料的測試程式佔用到的網路頻寬並不大，佔用的網路資源並不多，至於在 Hive 方面會有與其他語言不同的結果則是因為

程式撰寫方式不同所導致，原因在於 Hive 在讀取完資料後會先後建立兩次表以處理資料，這樣的情況導致會需要網路傳輸資料以建立資料表，所以才有這樣的結果。

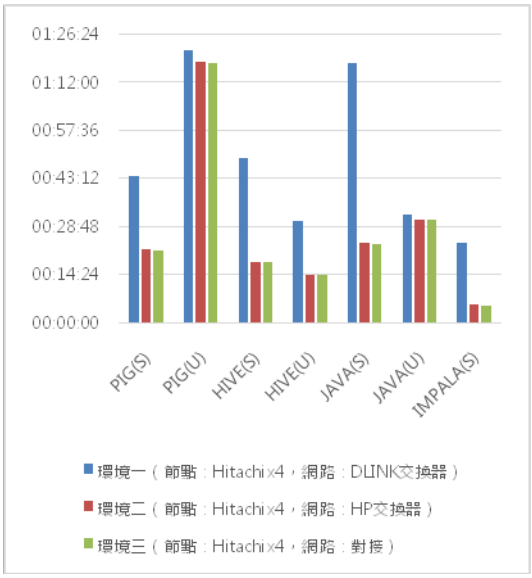

圖 5-1 實驗一：各環境不同語言之平均執行時間數據圖

5.2 實驗二：環境二/ 三/ 四/ 五

5.2.1 實驗目的

實驗二的目的在於測試 Linux Container 的效能，測試環境二與測試環境四這兩組測試環境可以比較在利用網路交換設備的狀況下，有無使用 Linux Container 對運算執行時間的影響；測試環境三與測試環境五這兩組測試環境可以比較在利用網路對接的情況下，除了可以檢驗有無使用 Linux Container 對運算執行時間的影響外，也可以測試 Linux Container 是否可以使用到底層網路硬體設備，以達到即使是分散各實體機器下的 Linux Container 也可以有與實體機器網路對接一樣不需要透過網路交換設備傳輸資訊。

5.2.2 實驗結果說明

在此次實驗結果中比較有無使用 Linux Container 虛擬機器對計算的影響，也比較 Linux Container 對於硬體層面，例如網路操控等，直接控制的效果，而實驗二各項測試程式執行時間數據圖如圖 5-2 所示，詳細的執行時間如表 5-1 所示。

在測試環境二與測試環境四中，可以看出 CDH 相關套件無論是佈署在實體機器上或是佈署在 Linux Container 虛擬機器上，其執行測試程式的運算時間都相當，由這樣的情況可以說明，Linux Container 可以取得底層硬體層面的資源，並無像是 Virtual Box 類型虛擬機器需要透過中介者模擬硬體資源花費額外的時間而造成運算時間延遲的狀況，另外，這也顯示出 Linux Container 可以使用到實體機器的所有運算資源，這也是需要透過中介者類型的虛擬機器無法作到的一項缺點。

在測試環境三與測試環境五中一樣也可以比較出測試環境二與測試環境四的結果，除此之外，測試環境三與測試環境五是利用網路對接的方式傳輸運算所需交換的資料，從實驗結果發現，Linux Container 也一樣可以作到網路對接提供的網路傳輸速度，也再次證明 Linux Container 可以取得所有底層硬體資源。

從這四組測試環境可以總結歸納出，Linux Container 可以完全取得硬體資源，這是需要中介者類型的虛擬機器所無法達到的事情，另外，也再度證明網路對接的資料傳輸方式可以達到與 HP 網路交換設備相同等級的網路傳輸速度外，換句話說，我們可以利用網路對接的方式，降低硬體成本，或是減輕同等級網路交換設備在傳輸資料方面的負擔。

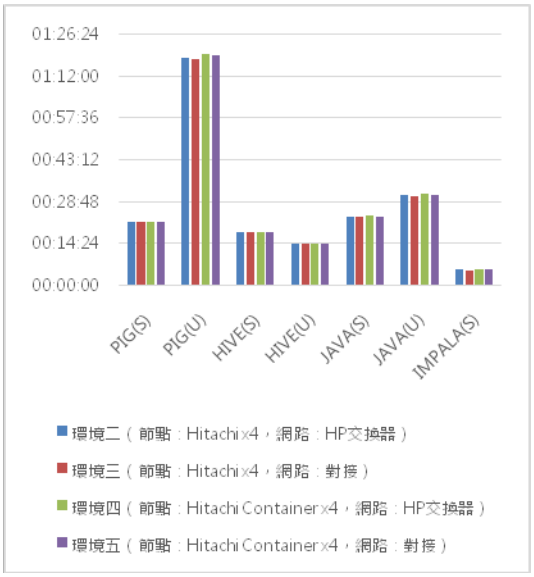

圖 5-2 實驗二：各環境不同語言之平均執行時間數據圖

5.3 實驗三：環境四 /六 /七 /八 /九 /十

5.3.1 實驗目的

實驗三的目的在於測試單一機器上不同數量的 Linux Container 虛擬機器對於實體機器在執行時間上的影響，而且比較在 Hitachi 伺服器等級機器與 Dell 個人電腦等級機器這兩種不同類型的機器，在硬體資源許可之下，在單一機器上不同等級的運算機器各可以容許多少數量的 Linux Container 虛擬機器。

5.3.2 實驗結果說明

實驗三各項測試程式執行時間數據圖如圖 5-3 所示，詳細的執行時間如表 5-1 所示。

此次實驗綜合比較了 Dell 與 Hitachi 這兩種機器在不同數量的 Linux Container 虛擬機器下的運算情況。

從程式語言的角度來看，Impala 由於在架構上有別於其他程式語言，所以不管在每台機器中虛擬機器的數量是一台、二台或四台這三種情況，其執行時間雖然相較於其他的語言都有明顯的優勢，但是卻也顯示出執行時間並沒有因為機器數量的倍數提昇，而有倍數的減少；相較之下，其他基於 MapReduce 架構的程式語言在虛擬機器數量上有所提昇時卻有明顯不同的結果，尤其在 Hitachi 機器中，當虛擬機器的數量由一台提昇至二台時，很明顯的可以看出其執行時間有明顯的減半，而當再由二台增加至四台時，其運算速度提昇的效果雖然沒

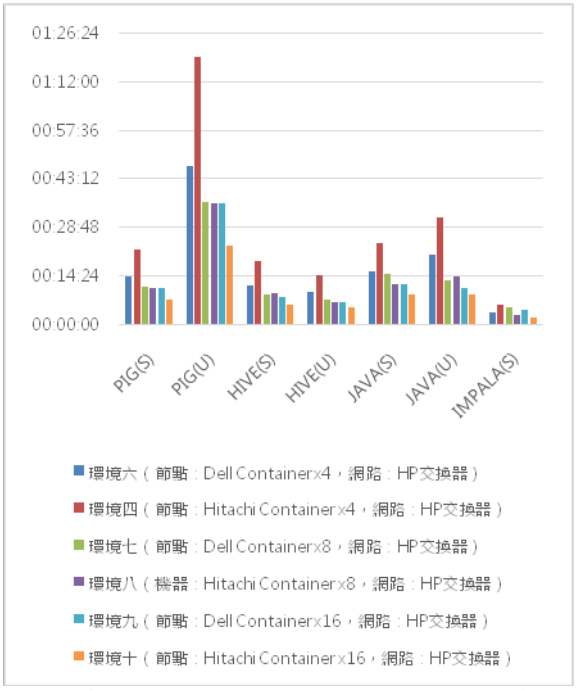

圖 5-3 實驗三：各環境不同語言之平均執行時間數據圖

6. 結論

從實驗結果可以得到幾個結論，第一，網路對執行時間上有著明顯的影響，在不同的網路交換設備時，其明顯的反映出執行時間會因此而有所改變，在更進一步，GbE 網路卡的對接可以達到與有 GbE 等級的網路交換設備如本論文所採用的 HP 交換器一樣的效果之下，一旦網路交換設備因為需要服務的機器越多，越有機會達到其能服務的效能瓶頸，為了能讓所有的機器不再被網路交換設備所限，更換設備便會是一種解法，然而這就又回到成本考量的問題上，而本論文提供另外一種方式，可以利用軟體套件輔助，或是利用作業系統的環境設定，讓在同一叢集內的機器彼此對接，除了減輕網路交換設備的負擔，達到更佳的網路傳輸效果，也有像先前一樣明顯地提昇兩倍，可是卻也有執行時間減少三分之一的成效；由上述結果更再度證明，MapReduce 架構需要大量的運算節點，其原因在於其設計的目標是建立在大量廉價的機器上，不管機器硬體勝敗優劣為何，當越多機器參與運算，就越能分散運算負擔，所以當運算節點數量越多，就越能發揮其分散式運算的效果。

從硬體的角度來看，Dell 機器從單機二台 Container 到單機四台 Container，其執行時間並無太大的差別，這可以表示出 Dell 機器的硬體資源為了應付工作已經呈現飽和的狀態，相同條件下，Hitachi 機器仍有小幅度的成長，代表 Hitachi 機器仍有可以在增加 Container 的空間，這也說明了，在硬體資源可以應付範圍內，Linux Container 的數量越多，越能榨取出硬體的所有資源，換句話說，越可以榨取出更多資源也代表越能節省硬體成本。

更能降低在硬體上的成本，容錯機制方面，也由於運算設備除了對接之外，仍與網路交換設備連接，一旦在某網路線路出現問題時，仍可透過另外一條路線到達目的地，或是透過其他節點以到達目的地；第二，Linux Container 可以有效的直接利用硬體層面的資源，在整個實體機器裡只裝有一組 Linux Container 的實驗中，可以看出 Linux Container 直接使用整台實體機器的完整資源，在其他相同類型的實驗下，其執行結果也近乎相同，此點更顯示出現有的虛擬機器如 Virtual Box 等不能完全使用到整個實體機器所有資源的問題；第三，在 Hitachi 與 Dell 兩種機器下可以發現，直接使用實體機器當作叢集的一份子時，並不能完全發揮其完整或是應有的運算資源，所以在硬體資源許可的情況下，越增加 Hitachi 與 Dell 兩種機器的 Container 數量，越能榨取其運算資源，發揮出硬體應該有的效能；最後，增加 Container 虛擬環境與增加 Virtual Box 類型的虛擬機器相比，由於 Virtual Box 類虛擬機器即使在閒置的狀況下仍會佔據硬體資源，因此 Container 類型等會分享硬體資源的虛擬環境更佳適合此類雲端叢集的運算。

參考文獻

[1] HDFS Architecture。
http://hadoop.apache.org/docs/r2.2.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html。

[2] 淺談資料倉儲。
http://webmail.hwai.edu.tw/~linys/DSS/DW.pdf。

[3] Cloudera。http://www.cloudera.com/。

[4] Linux Container。https://linuxcontainers.org/。

[5] Configuring LXC - Linux Containers。
http://kaivanov.blogspot.tw/2012/07/configuring-lxc-linux-containers.html。

[6] Containers—Not Virtual Machines—Are the Future Cloud。
http://www.linuxjournal.com/content/containers%E2%80%94not-virtual-machines%E2%80%94are-future-cloud。

[7] Stephen J. Vaughan-Nichols. 2006. New Approach to Virtualization Is a Lightweight. Computer 39, 11 (November 2006), 12-14.

[8] Java MapReduce WordCount Example。
http://www.cloudera.com/content/cloudera-content/cloudera-docs/HadoopTutorial/CDH4/Hadoop-Tutorial.html。