

東海大學資訊工程學系研究所
碩士論文

指導教授：林祝興 博士

Dr. Chu-Hsing Lin

協同過濾推薦演算法在雲端實現與效能評估

Implementation and Performance Evaluation of
a Collaborative Filtering Recommendation
Algorithm on the Cloud Using Hadoop

研究生：朱栢葦

中 華 民 國 一 〇 四 年 六 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 朱 栢 葦 所提之論文

協同過濾推薦演算法在雲端實現與效能評估

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



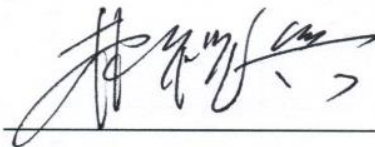
簽章

委

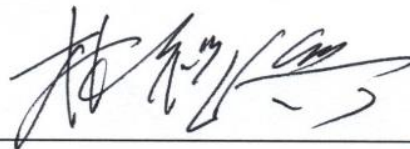
員







指導教授



簽章

中華民國 104 年 6 月 25 日

中文摘要

隨著雲端技術的發展和網路的快速普及，如何快速的從海量資料中獲取用戶想要的資訊逐步成為用戶關注焦點。通過獲取用戶在網路上的歷史日誌資訊，分析用戶的喜愛偏好，進而為用戶推薦其可能感興趣的資訊或商品。

但隨著網路的發展在資料庫中的用戶日誌越來越多，在推薦系統面臨儲存空間的擴展與分析計算效率之瓶頸本文針對該問題，以基於 Hadoop 分散式文件存儲 HDFS 和分散式計算框架 MapReduce 的工具與 Mahout 機器學習演算法資料庫為基礎，實現了一種基於分散式計算框架 Hadoop 之上的推薦系統架構，並且對基於項目協同過濾推薦演演算法，在雲端群集與單機上執行的效能和評估。

關鍵字：雲端計算、推薦系統、協同過濾、MapReduce、Mahout

ABSTRACT

With the rapid development of the cloud computing technology and the Internet, how to help users quickly gain valuable insights from the Big Data has gradually become the focus and challenge. From analysis of user log information on the Internet, preferences of the user are revealed, information or commodities most matched with user interests are recommended. However, with the fast development of the Internet, the amount of user log files explodes, causing bottlenecks in performance and storage spaces for recommendation systems. To solve this problem, in this thesis we implement a recommendation system architecture based on the Hadoop distributed computing framework, in which the Hadoop distributed file storage (HDFS), distributed computing frameworks MapReduce, and Mahout machine-learning algorithm based database are adopted. We also explore performance evaluation of the collaborative filtering recommendation algorithm on the cloud cluster and standalone personal computer.

Keywords: cloud computing, recommendation system, collaborative filtering, MapReduce, Mahout

致謝

兩年研究所的日子轉眼間就要結束了，這些日子以來有許許多多的人們需要感謝，感謝有你們的陪伴，幫忙與諒解。

首先需要感謝的是在研究所的指導教授林祝興老師，在論文與待人處世方面的教導與提醒。

另外也感謝撥空前來參加的口試委員：張隆池教授、蔡坤霖教授以及賴威伸教授，感謝你們的建議與認同，給了我在論文的改進上有了相當的幫助。

還有感謝家裡的父母，在最初鼓勵我繼續往上念研究所，提供我的日常所需與學費讓我無後顧之憂的完成學業，在這兩年的日子裡也不曾間斷的鼓勵與關心，由衷感謝你們這兩年間的支持。

最後要感謝實驗室中的學長、同學與學弟們，在我兩年的研究生生活過得很充實也很快樂，感謝實驗室唯一的女生思縈幫助我修改論文與許多實驗室的日常事務，也要感謝資工神人正傑與他的好朋友耕瑜讓我在實驗室有更多的樂趣與討論的夥伴，最後再次感謝我最敬愛的父母、師長與親朋好友們，在此致上最高的感謝。

朱栢葦 於東海大學研究所 資安實驗室 104年7月

目錄

中文摘要.....	3
ABSTRACT.....	4
致謝.....	5
目錄.....	6
圖目錄.....	7
第一章 簡介.....	9
第二章 相關文獻探討.....	10
2.1. Apache Hadoop.....	10
2.1.1. Hadoop Distributed File System.....	10
2.1.2. MapReduce	12
2.2. Apache Mahout.....	14
2.3. Collaborative Filtering Algorithm.....	15
2.3.1. 基於用戶的協同過濾演算法.....	16
2.3.2. 基於項目的協同過濾演算法.....	17
第三章 研究方法與實驗環境.....	19
3.1. 實驗環境介紹.....	19
3.2. 基於項目的協同過濾演算法.....	19
第四章 研究結果與分析.....	29
4.1. 實驗結果分析.....	29
4.2. 加速比.....	31
第五章 結論.....	33
參考文獻.....	34

圖目錄

圖 1	HDFS 架構圖.....	11
圖 2	MapReduce 流程圖.....	13
圖 3	Hadoop 架構角色分工圖.....	14
圖 4	Hadoop 與 Mahout 關係圖.....	15
圖 5	基於項目推薦關係圖.....	17
圖 6	基於項目推薦關係圖.....	18
圖 7	基於項目的協同過濾演算法 MapReduce 流程圖.....	21
圖 8	推薦結果圖.....	28
圖 9	資料量與執行時間的關係圖.....	30
圖 10	節點數量與執行時間的關係圖.....	31
圖 11	加速比圖.....	32

表目錄

表 1	測試資料表.....	20
表 2	項目索引值.....	22
表 3	用戶評價矩陣.....	22
表 4	項目評價矩陣.....	23
表 5	用戶項目評價數計數.....	24
表 6	項目的權重值.....	24
表 7	項目 102 對其他項目的相似度.....	26
表 8	用戶評價矩陣.....	26
表 9	User1 的項目相似度矩陣.....	27
表 10	各節點實驗完成時間表(節點數\資料量).....	30

第一章 簡介

隨著電腦網路與電子商務的蓬勃發展與規模不斷擴大，系統資料庫中的用戶資料與商品數量的增加，用戶想要從這些海量資料中找到自己需要的資訊與商品變得異常困難。推薦系統成為解決此問題的最好方案幾乎在所有的大型電子商務網站都已有應用了推薦系統，如全球最大的網路書店 Amazon.com 與 PChome 24H 購物等，根據用戶在網路上的瀏覽習慣與購物的歷史紀錄透過各種演算法和相似度的計算，在用戶瀏覽網頁或商品的時候顯示別人也看過這項商品相似的商品或推薦用戶需要的資訊或商品，減少尋找時間與增加商品購買率。但在隨著資料量和用戶增加的大型電子商務網站中需要更好擴展計算資源與儲存空間的方法，利用 Apache Hadoop 的開源軟體中的 Hadoop Distributed File System(HDFS) 與 MapReduce 的方法可以達到解決提升計算效能與儲存空間管理與增加容量。

在推薦系統中大致分為兩類進行推薦分別是商品與用戶，另外在與日俱增的用戶數量，在商品的數量與是相對較穩定，此篇研究主要方向利用項目的協同過濾演算法與雲端運算的結合，實驗在運算資源的增加與資料量的增加，在運算所需要時間與效能的探討與分析。

本論文共分為五章：本章為簡介，說明研究動機與目標；第二章中背景知識與現有技術的文獻探討；第三章主要是介紹實驗環境、測試資料集與使用的演算法；第四章為在雲端環境中運算的效能探討；第五章為結論與未來展望。

第二章 相關文獻探討

2.1. Apache Hadoop

2.1.1. Hadoop Distributed File System

HDFS[1]為 Hadoop[10]專案中是用來管理與操作分散式檔案系統，將分散的儲存資源整合成具容錯能力、高效率且超大容量的儲存環境，在 Hadoop 系統中大量的資料和運算時產生的暫存檔案，都是存放在這個分散式的檔案系統上。

Hadoop 是 Master/Slave 架構，HDFS 由兩種角色組成 NameNode 及 DataNodes，NameNode 負責檔案系統中各個檔案屬性權限等資訊(Metadata, NameSpace) 的管理及儲存；而 DataNode 通常由數以百計的節點擔任，一個資料檔會被切割成數個較小的區塊 (Block) 儲存在不同的 DataNode 上，每一個區塊還會有數份副本 (Replication) 存放在不同儲存節點(預設 3 個副本)。這樣當其中一個節點損壞時，檔案系統中的資料還能保存無缺，因此 NameNode 還需要紀錄每一份檔案存放的位置，當有存取檔案的需求時，協調 DataNode 負責回應；而有節點損壞時，NameNode 也會自動進行資料的搬遷和複製。

HDFS 雖然沒有整合進 Linux kernel，但能透過 Hadoop 的 dfs shell 進行檔案操作，但 Hadoop 下的系統都與 HDFS 整合，做為資料儲存備份及分享的媒介。

在 MapReduce 在系統分配運算工作時，會將運算工作分配到存放有運算資料的節點上進行，減少大量資料透過網路傳輸的時間。

如圖 1 當用戶寫入資料 3，在 HDFS 的 NameNode 會把寫入的資料 3 自動複製給其他 DataNode 以確保資料的正確性與完整性。假如用戶要讀取資料 1，若資料 1 毀損或資料遺失，NameNode 可以由其他 DataNode 把資料 1 還原。

(1,2,3,6 表示存在 HDFS 上的資料)

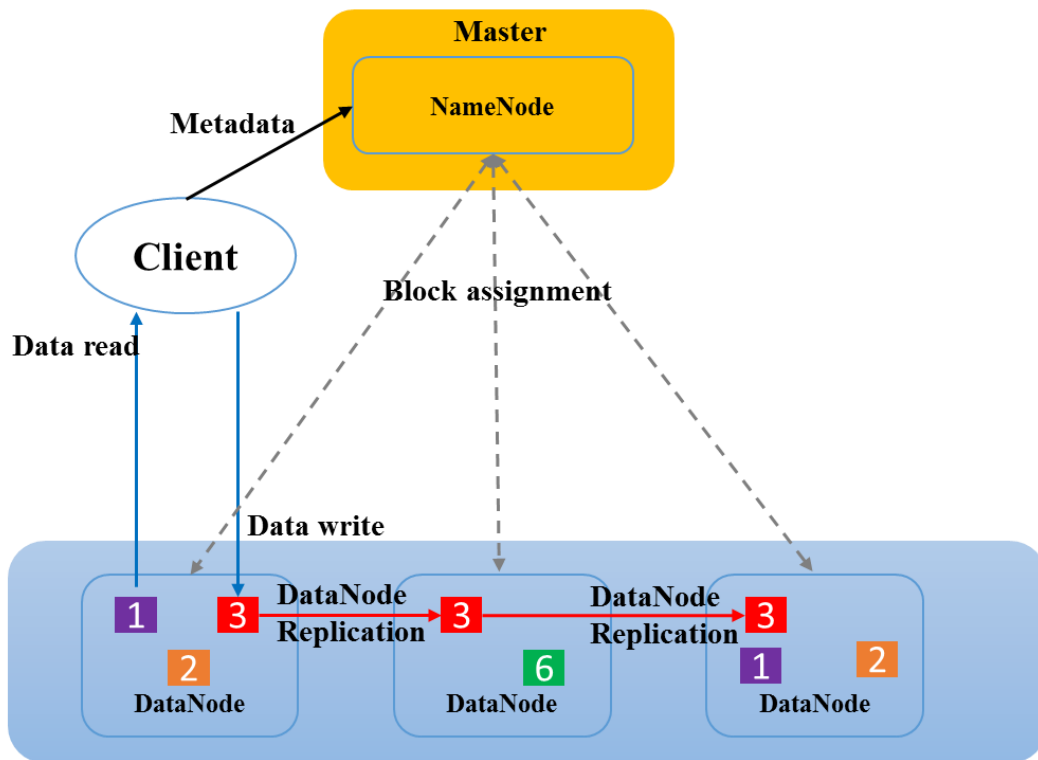


圖 1 HDFS 架構圖

2.1.2. MapReduce

MapReduce[2]是雲端運算的關鍵技術，將需要執行的運算，拆解成 Map 和 Reduce 的方式來執行，以達到分散運算的效果。開發者可以很簡單的撰寫程式，利用大量的運算資源，加速處理龐大的資料量。

一個 MapReduce 的運算工作可以分成兩個部份 Map 和 Reduce，大量的資料在運算開始的時候，會被系統轉換成多組的(key, value) 序對並自動切割成許多部份，分別傳給不同的 Mapper 來處理；Mapper 處理完成後也要將運算結果整理成多組的 (key, value) 序對，再傳給 Reducer 整合所有 Mapper 的結果，最後才能將整體的結果輸出，以下舉例說明。

如果此資料串(Pig,Cat,Dog,Pig,Bear,Dog,Pig,Cat,Bear)中想要知道 Pig 這個詞出現的次數將其統計且存入資料庫中 MapReduce 程式的執行過程如圖 2：

- 步驟 1 將要計算的資料存入 HDFS 並且在 Master 執行 MapReduce 程式。
- 步驟 2 Master 將 MapReduce 程式和經過切割的資料，傳入各個節點。
- 步驟 3 在各個節點上把資料轉換型態為<Key,Value>("pig",1)的型態。
- 步驟 4 在 Reduce 的步驟把各個<Key,Value>收集，把同 Key 裡面的值加總。
- 步驟 5 最後存回 HDFS 開發者可以利用 Hadoop 的 dfs shell 檢視結果。

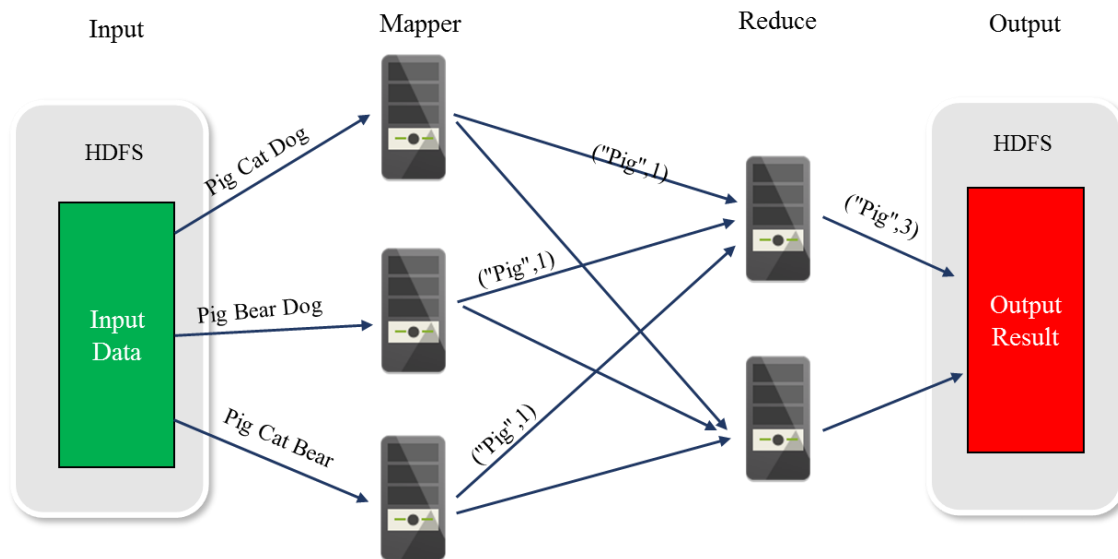


圖 2 MapReduce 流程圖

在系統的運作架構上，最簡單的 Hadoop 架構，可以分成上層的 MapReduce 運算層以及下層的 HDFS 資料層。

如圖 3， Master 節點的伺服器中會執行兩套程式，一個是負責安排 MapReduce 運算層任務的 JobTracker 程式，以及負責管理 HDFS 資料層的 NameNode 程式。而在 Node 中也有兩套程式，接受 JobTracker 指揮，負責執行運算層任務的是 TaskTracker 程式，而與 NameNode 對應的則是 DataNode 程式，負責執行資料讀寫動作，以及執行 NameNode 的副本。

在 MapReduce 運算層上，擔任 Master 節點的伺服器負責分配運算任務，Master 節點上的 JobTracker 程式會將 Map 和 Reduce 程式的執行工作，指派給 Node 上的 TaskTracker 程式，由 TaskTracker 負責執行 Map 和 Reduce 工作，並將運算結果回覆給 Master 節點上的 JobTracker。

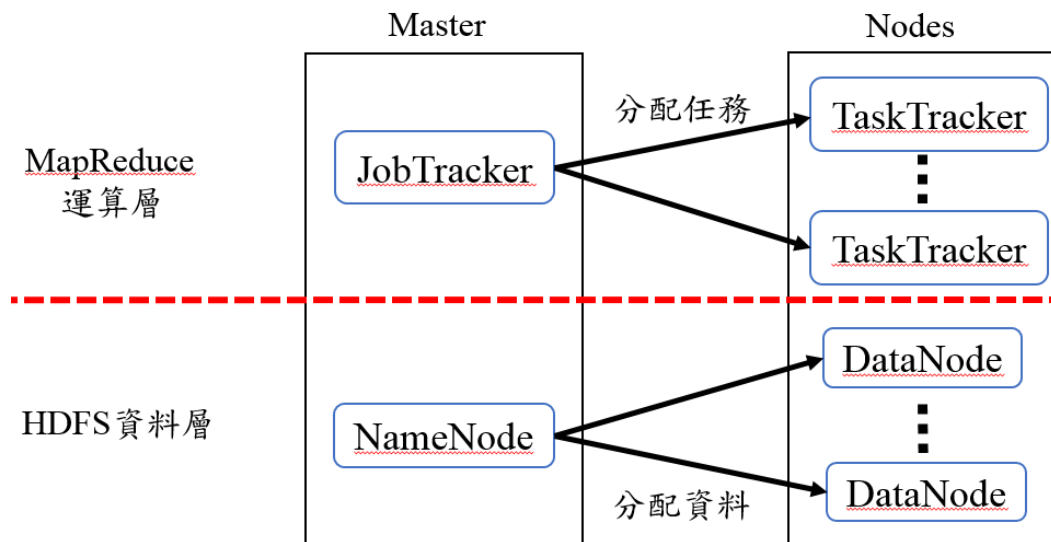


圖 3 Hadoop 架構角色分工圖

2.2. Apache Mahout

Apache Mahout[9]是 Apache Software Foundation 旗下的一個開放原始碼的專案，其專案的目的在於提供快速建構與高可擴展性的機器學習應用的環境。機器學習是一個很廣大的領域，其內容也包羅萬象，而 Mahout 的目標便是提供各種類型的經典演算法，以程式庫的方式呈現，並且具備高效及規模可擴充性的特質。Mahout 為了提供規模可擴充性，而基於 Apache Software Foundation 另一個名為 Hadoop 之專案的平台，以 MapReduce 的演算法，實作了像群集 (clustering)、分類 (classification)，以及協同過濾 (collaborative filtering) 的核心演算法。

除了運行在分散式多節點 Hadoop 平台上也提供單節點與其他版本上應用，這使得開發者，可以依據其對計算力及規模可擴充性的需要，來決定系統運行的

方式，甚至依需要隨時來改變配置方式。

此外，Mahout 的核心程式庫，即使是非分散式的版本，也都經過高度的最佳化，因而能提供極佳的執行效能，在此篇論文中主要會應用到協同過濾 (collaborative filtering) 的核心演算法，在分散式多節點 Hadoop 平台實作如圖 4。

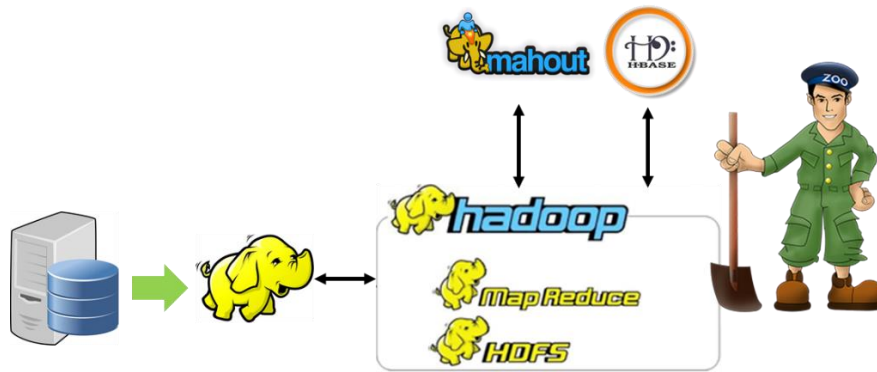


圖 4 Hadoop 與 Mahout 關係圖

2.3. Collaborative Filtering Algorithm

協同過濾演算法(Collaborative Filtering Algorithm) [3][4]是一種被廣泛使用在商業的個性化推薦技術，該演算法主要是利用興趣相投或擁有共同經驗的群體的喜好來推薦用戶感興趣的資訊。

用戶透過合作的機制給予資訊或商品相當程度的回應和評價並且以歷史紀錄儲存在資料庫，所給予的回應不一定侷限於特別感興趣的，特別不感興趣資訊的紀錄也相當重要。協同過濾演算法並不理會項目本身的內容或價格，不是從項目的內容來評估項目的相似性而是從被所有用戶偏好的情況來判斷項目間的相

似性，這個相似性並不是指它們內容相似，而是指喜歡他們的人相似，以及項目被同一用戶喜愛的相似。換言之，用戶的相像，是基於他們所喜愛的項目而相像；而項目之間的相像，則是因為喜歡它們的人相像。

省去內容了解究竟是什麼，是協同過濾式推薦方法的優點，但是當群體的數量不足、產生的資料不足以得到品質較好的計算結果時，協同過濾式推薦方法可能就比較難以做出準確的推薦。

在電子商務當中是很重要的一環，根據某顧客以往的購買行為以及從具有相似購買行為的顧客群的購買行為去推薦這個顧客其可能喜歡的項目，也就是藉由社群的喜好提供個人化的資訊、商品等的推薦服務，在對於市場評估上也有相當的幫助。

協同過濾演算法有兩種表現的形式：基於用戶的協同過濾演算法與基於項目的協同過濾演算法，研究中主要處理基於項目的協同過濾演算法。

2.3.1. 基於用戶的協同過濾演算法

基於用戶的協同過濾演算法(User-Based Collaborative Filtering Algorithm)[4]主要是利用一個假設：一個用戶會喜歡和他有相同愛好的用戶喜歡的東西；意指有相同興趣或愛好的人喜歡或感興趣的東西會一樣或相似即以人分群。

但用戶的興趣是廣泛的以電影為例喜歡恐怖片與科幻片的用戶也可能喜歡愛情片與卡通片，那如何把同樣喜好的用戶做分群並且計算用戶對其他用戶的相

似度，最後針對用戶的興趣進行預測，產生推薦結果。

利用圖 5 的例子，根據所有用戶的歷史紀錄，當用戶 1 想從 4 部電影中選部自己想看的電影，而目前他只看過電影 101 與電影 103，並且利用用戶 2 也看過電影 101、103、104 的關係，可以用來表示用戶 1 與用戶 2 兩個相似度較高，所以推薦用戶 1 觀賞電影 104。

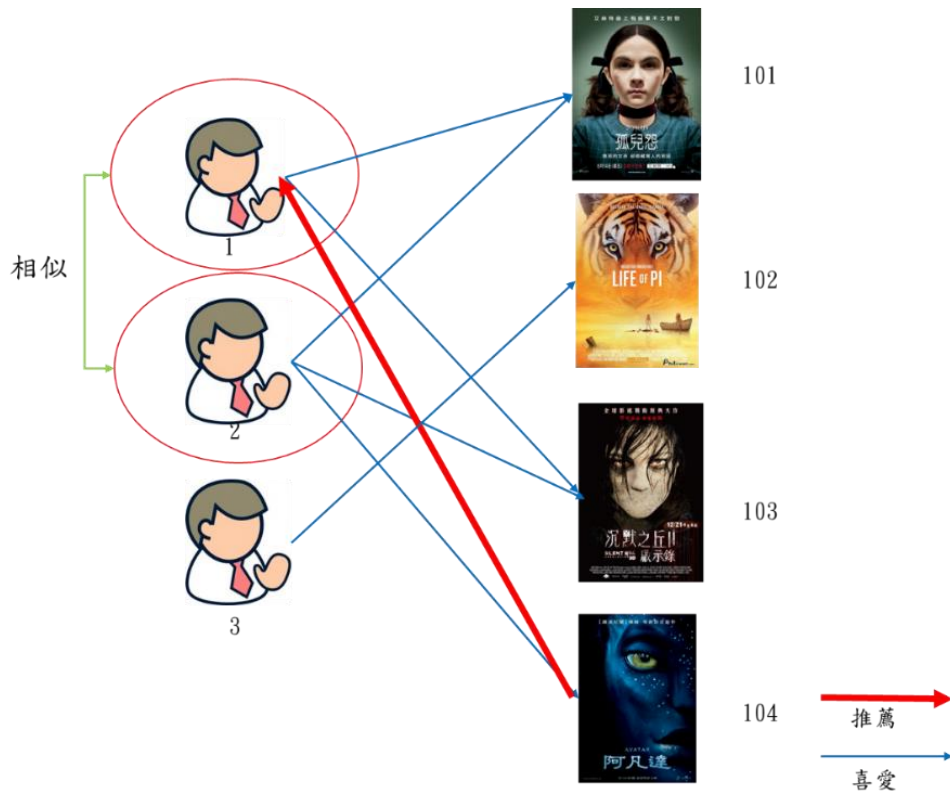


圖 5 基於項目推薦關係圖

2.3.2. 基於項目的協同過濾演算法

基於項目的協同過濾演算法(Item-Based Collaborative Filtering Algorithm)

[5][8][9]主要是依據於一個假設：透過計算項目之間的相似性，若一個項目能夠引起用戶興趣，那麼這個項目必定與之前評價高的項目相似。

利用項目的相似度來進行運算，並依照運算的結果去除用戶已評價過的項目，推薦用戶無接觸或無評價但卻與之前感興趣且評價的項目進行由高至低的排序，依照算出的值較高的顯示給用戶。

從計算的角度來看，就是將所有用戶評價過的電影定為一個向量，每個用戶對電影的評價作為向量的分量，通過向量值來計算電影間相似度，再利用根據歷史紀錄中已評價的電影來預測當前用戶尚未觀看和評價的電影，經計算得到一個按預測值由高至低排序並且依照所設定推薦個數顯示給用戶。

利用圖 6 的例子，根據所有用戶的歷史紀錄得知喜歡電影 101 的都喜歡電影 103，推測出電影 103 與電影 101 相似度高，而用戶 3 喜歡電影 101，那可以推測出用戶 3 也會喜歡電影 103。

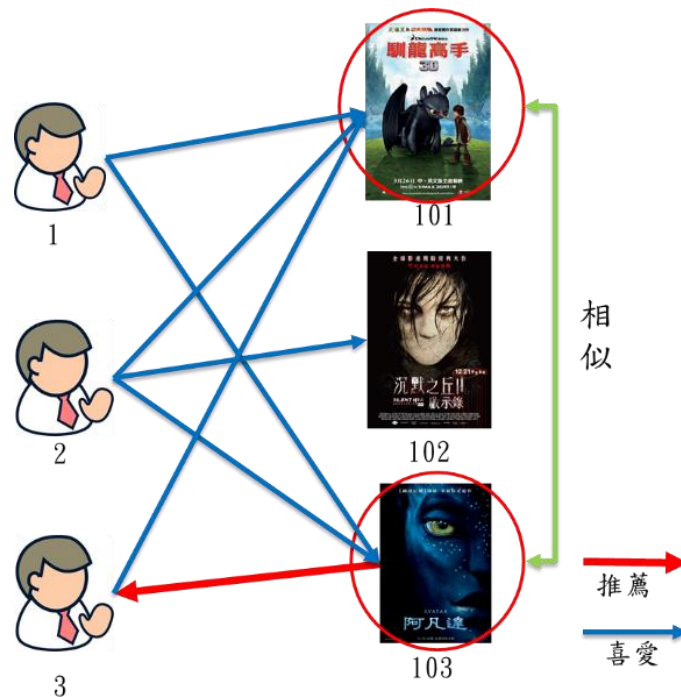


圖 6 基於項目推薦關係圖

第三章 研究方法與實驗環境

3.1. 實驗環境介紹

在實驗環境的部分如下：CPU Intel® Core™ i5-4690 Processor、記憶體 Transcend DDR3-1600 16G、硬碟 SEAGATE 1TB、作業系統 Window 8.1，並使用 Virtual Machine 的軟體技術建立 6 台虛擬機器在一台實體電腦上，每台的硬體規格分別是作業系統 CentOS 6.6、硬碟 20GB 記憶體 1GB 以及核心數 1 個，其中一台當 Master 其餘 5 台當 Slaves 建立起 Hadoop 群集；所有的機器都透過 Virtual Machine 軟體內的虛擬網路串聯起來。

使用 Hadoop 版本是 1.2.1、Mahout 版本為 0.9，實驗中使用的資料為 GroupLens [6][12]的資料集，GroupLens 資料集包括超過十萬筆收視資料，每個用戶至少評價過 20 部電影，資料集也包含用戶的簡單人口統計資訊，如姓名、年齡、職業及性別…等，該數據是通過 MovieLens 網站收集的。

3.2. 基於項目的協同過濾演算法

實驗中基於項目的協同過濾演算法在 MapReduce 主要分成四個步驟如下。
本文收集用戶與物品的資料，利用 MovieLens 所收集的歷史資料作為主要的資料

來源，資料格式的表示如下：

用戶編號，項目編號，項目評價值

項目評價值的是以[0.0 ~ 5.0]之間的實數作為代表，評價值越高代表此項目受喜愛程度越高，未評價之項目為 0，這邊用簡單的資料作為例子講解基於項目的協同過濾演算法使用的資料，如表 1。

輸入：用戶編號,此用戶已評價的 N 項目編號,項目評價值

輸出：用戶編號,推薦的 Top N 項目編號,推薦的 Top N 評價值

表 1 測試資料表

用戶編號	項目編號	評價值
1	101	5.0
1	102	3.0
1	103	2.5
2	101	2.0
2	102	2.5
2	103	5.0
2	104	2.0
3	101	2.5
3	104	4.0
3	105	4.5
3	107	5.0
4	101	5.0
4	103	3.0
4	104	4.5
4	106	4.0
5	101	4.0
5	102	3.0
5	103	2.0
5	104	4.0
5	105	3.5
5	106	4.0

MapReduce 的流程圖如圖 7 所示。

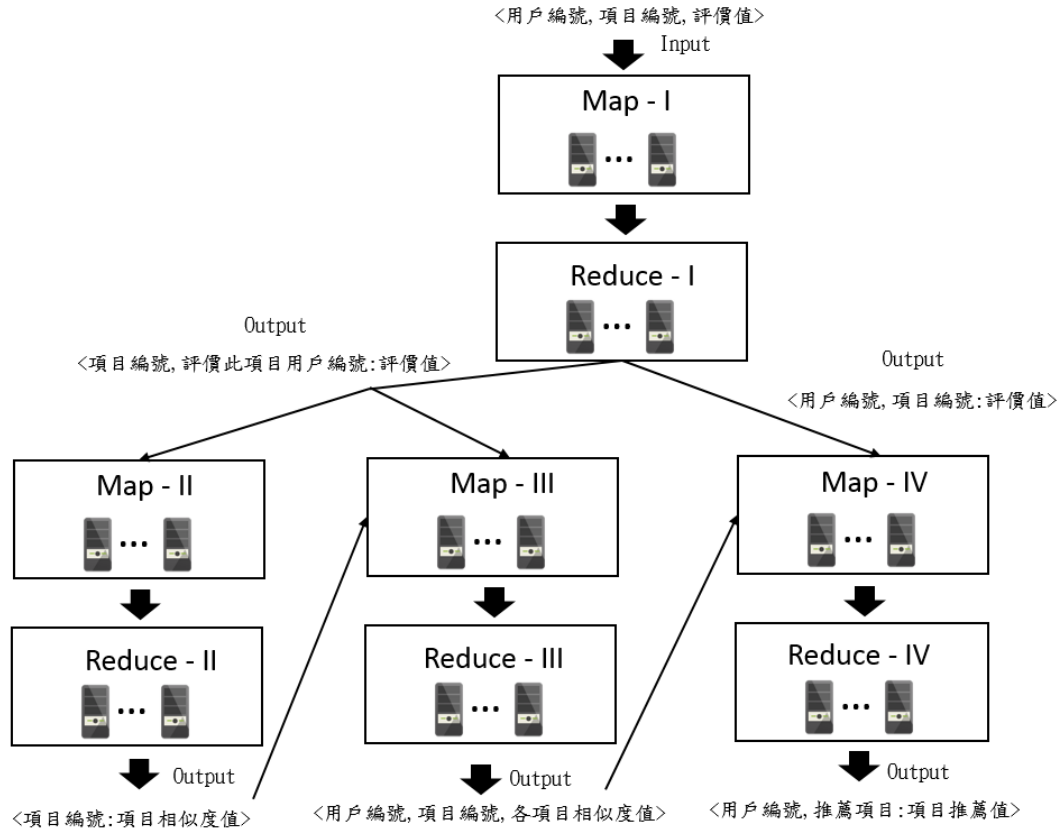


圖 7 基於項目的協同過濾演算法 MapReduce 流程圖

* 步驟 1 建立用戶向量與項目評價矩陣

基於項目為協同過濾演算法輸入的數據主要表示為 $u * i$ 的用戶評價矩陣 R ，其中 u 是用戶編號、 i 是項目編號、 $R_{u,i}$ 表示第 u 個用戶對第 i 個項目的推薦評價值，推薦的項目評價值越高代表此項目受喜愛程度越高，未評價之項目為 0。

此步驟主要分成三部分:

1-1 建立項目索引值

輸入資料格式為(用戶編號,項目編號,項目評價值)轉變成內部索引(索引值,項目編號),例如項目編號 102 轉變成索引值就是(102=102),如表 2。

表 2 項目索引值

項目索引值						
101=101	102=102	103=103	104=104	105=105	106=106	107=107

MapReduce 是把表 1 的資料去除用戶編號與項目評價值,只取出項目編號建立索引,<Key,Value>分別是<項目編號,項目索引編號>。

1-2 建立用戶評價商品矩陣

將輸入資料以行讀入,以用戶編號為索引組成用戶向量矩陣,內容包含各用戶所有評價的項目編號與評價值如表 3。

MapReduce 把表 1 的資料輸入且切割給不同主機,格式為用戶編號、項目編號與評價值,<Key,Value>分別是<用戶編號,項目編號:項目評價值>。

表 3 用戶評價矩陣

用戶編號	項目編號:評價值
1	101:5.0,102:3.0,103:2.5
2	101:2.0,102:2.5,103:5.0,104:2.0
3	101:2.5,104:4.0,105:4.5,107:5.0
4	101:5.0,103:3.0,104:4.5,106:4.0
5	101:4.0,102:3.0,103:2.0,104:4.0,105:3.5,106:4.0

1-3 建立項目與用戶關係矩陣

此步驟是以項目編號為索引把有評價過此項目的用戶來建立項目與用戶關係矩陣如表 4。

MapReduce 把表 1 的資料輸入且切割給不同主機，格式為用戶編號、項目編號與評價值， <Key,Value>分別是<項目編號,用戶編號:此項目評價值>。

表 4 項目評價矩陣

項目編號	用戶編號:評價值
101	5:4.0,4:5.0,3:2.5,2:2.0,1:5.0
102	5:3.0,2:2.5,1:3.0
103	5:2.0,4:3.0,2:5.0,1:2.5
104	5:4.0,4:4.5,3:4.0,2:2.0
105	5:3.5,3:4.5
106	5:4.0,4:4.0
107	3:5.0

*步驟 2 建立相似度矩陣

計算相似度矩陣是此運算中最重要步驟，輸入上一個步驟 1 的項目評價矩陣，輸出項目相似度矩陣。

2-1 計算用戶評價總數

把步驟 1 得出的表 4 輸入並且計數各用戶所評價的項目編號之個數如表 5，MapReduce 利用 for 迴圈把同樣的用戶編號判斷它有幾個評價項目作加總，<Key,Value>分別是<用戶編號,項目評價個數>。

表 5 用戶項目評價數計數

用戶編號	用戶評價的項目總數
1	3
2	4
3	4
4	4
5	5

2-2 計算項目間的權重

利用表 4 的項目評價矩陣來計算各項目的權重，輸入項目評價矩陣得到各項目的權重，此數值越高代表受人喜愛的程度越高，得到的各項目權重值如表 6 所示。

MapReduce 算法是把項目中的評價值平方後加總得到每個項目的權重值，可以見到的是越多人評價的項目權重值越高，也表示此項目受歡迎度較高，計算完後用陣列儲存這邊<Key, Value>分別是<項目編號, 項目權重值>。以項目 101 的權重值做計算如下：

項目編號	用戶編號:評價值
101	5:4.0,4:5.0,3:2.5,2:2.0,1:5.0

$$101=4^2 + 5^2 + 2.5^2 + 2^2 + 5^2 = 76.25$$

表 6 項目的權重值

項目編號	項目權重值
101	76.25
102	24.5
103	44.25
104	56.25
105	32.5
106	32.0
107	25.0

2-3 計算項目的相似度

此步驟是在推薦演算法中最重要的部分，計算各項目的距離相似度，使用歐基里德距離(Euclidean Distance)此公式(1)主要是用在計算兩點在 N 維空間內的距離，假設 x, y 是 N 維空間的兩點，兩點的歐式距離是：

$$d(x, y) = \sqrt{\sum(x_i - y_i)^2} \quad \text{公式(1)}$$

$$\text{simi}(x, y) = \frac{1}{1+d(x,y)} \quad \text{公式(2)}$$

相似度($\text{simi}(x,y)$)範圍是[0,1]值的距離越大代表 d 的值越小，表兩項目間距離越近，也就是說相似度越大計算，方式如公式(2)。使用項目評價矩陣裡面的值來計算項目之間的相似度，只要有一個共同評價項就能使用歐基里德距離計算相似度，若無共同評價項表示兩個項目根本不相似。

MapReduce 把每個項目 Map 出去開始計算項目和其他項目的相似度值，套入歐式距離的算法會得到該項目對其他項目相似度值陣列，最後 Reduce 以 $\langle \text{Key}, \text{Value} \rangle$ 儲存分別是 $\langle \text{項目編號}, \text{其他項目相似值} \rangle$ 。

利用一個例子解說項目 102 對項目 101 的相似度為例：

輸入項目 101 與項目 102 的用戶編號與用戶評分該項目的評價值

項目 101=[1:5,2:2,3:2.5,4:5,5:4]

項目 102=[1:3,2:2.5,5:3]

$$d(x,y)=\sqrt{(5-3)^2+(2-2.5)^2+(2.5-0)^2+(5-0)^2+(4-3)^2}=\sqrt{36.5}$$

$$\text{simi}(x,y)=\frac{1}{1+\sqrt{36.5}}=0.1420147320224587$$

得出項目 102 對項目 101 的相似度為 0.1420147320224587。

* 步驟 3 計算每個項目的推薦值

利用步驟 2 所得到的項目相似度矩陣與用戶評價矩陣，計算得到各項目的推薦值並且給步驟 4 運算，表 7 中的 NaN 表示用戶以評價過的項目。

表 7 項目 102 對其他項目的相似度

項目編號	項目相似度
106	0.1497250646352768
105	0.14328432083129883
104	0.12789210677146912
103	0.19754962623119354
102	NaN
101	0.14201472699642181

利用步驟 2-3 所得到的各項目的相似度進行推薦值的計算，這邊會使用到兩個矩陣計算分別是項目相似度矩陣與用戶評價矩陣，除掉以評價的項目然後對每個沒評價的項目做計算，把已評價過的項目評價值乘上要計算的項目相似度並且加總值得到矩陣 A，在把計算的項目對以評價過的項目的相似度值加總得到矩陣 B，最後把矩陣 A 總和與矩陣 B 總和相除的到該項目對用戶的推薦值。如果要計算表 9 中項目 106 的推薦值可利用以下步驟得出：

表 8 用戶評價矩陣

用戶 ID	項目與評價值
1	101:5.0,102:3.0,103:2.5

表 9 User 1 的項目相似度矩陣

項目編號	項目相似度
101	NaN
102	NaN
103	NaN
104	1.5353794321417809
105	1.2893039211630821
106	1.5174299776554108
107	0.5137624219059944

矩陣 A : $3 \times 106_{simi} + 2.5 \times 106_{simi} + 5 \times 106_{simi} \cong 1.5174$

矩陣 B : $101_{106simi} + 102_{106simi} + 103_{106simi} \cong 0.43459$

$A \div B \cong 3.49115$

得到 User 1 對項目 106 的推薦值為 3.49115。

MapReduce 各用戶的用戶評價與相似度矩陣 Map 給各節點開始做推薦值的計算最後 Reducec 收集結果，已<Key,Value>儲存分別是<用戶編號,所有項目的推薦值>。

* 步驟 4 取得推薦結果

這步驟會把推薦結果以<Key,Value>的方式收集並加總，另外把用戶已評價過得項目過濾掉，並且排序推薦項目由高至低顯示如圖 8，在圖 8 中所得到的推薦列表由上面的測試資料表 1 所提供的歷史評價紀錄預測當前用戶所預測的推薦列表。

MapReduce 得部分把步驟 3 的結果 Map 給各節點開始做推薦值的排序最後 Reduce 收集結果，以<Key,Value>儲存分別是<用戶編號,所有項目的推薦值(由高至低)。

```
1 [104:3.5838122,106:3.4916115,105:3.473163]
2 [106:2.8146582,105:2.7573717,107:2.0]
3 [106:3.694073,102:3.657834,103:3.5656683]
4 [107:4.716343,105:4.1627345,102:4.0136285]
5 [107:3.7592773]
```

圖 8 推薦結果圖

第四章 研究結果與分析

4.1. 實驗結果分析

在實驗中，Hadoop 的平台上每個節點最大可支持 5 個 Map 與 5 個 Reduce 任務，在群集中最多可以支持 25 個 MapReduce 任務，在資料集的部分，根據 GroupLens 的資料集，每個用戶至少評價過 20 部電影，也包含用戶的簡單人口統計資訊。

利用 GroupLens 的資料集得到的資料進行切割和過濾，使用的資料筆數從一萬筆到一千萬筆，不同節點數的計算完成時間如表 8 所示，可以見到在 1 Node 時，在較大量資料集計算推薦結果的時間較長，增加計算節點可以達到減少計算時間，也就是說只要不斷增加計算節點就能夠達到減少推薦結果計算的時間，另外在實際的情形下計算節點與儲存節點的可以同時增加，證明說 Hadoop 的架構能有更大的儲存空間與更短計算時間。

為了測試 Hadoop 平台的擴展可能性，實驗分兩組不同實驗第一組實驗為固定節點數資料集大小可變，第二組實驗為資料集固定節點可變，將結果量化顯示如圖 9 與圖 10，可見到在節點的越多，可以將計算時間減少從圖中看到在 10M 的部分原本在 1Node 上執行的時間需要 37m17s，在 5 Nodes 只需 12m07s 代表在計算節點增加可以有效的提升運算效能，並且在處理大量的資料可以減少等待

結果的時間，MapReduce 的節點配置最好是以奇數為單位這樣可以提高

MapReduce 在 Reduce 的速度。

表 10 各節點實驗完成時間表(節點數\資料量)

	1 Node	2 Nodes	3 Nodes	4 Nodes	5 Nodes
10K	7m16s	5m23s	3m41s	3m23s	3m16s
1M	12m56s	8m38s	6m10s	5m56s	4m58s
2M	15m23s	10m03s	7m23s	7m01s	5m49s
3M	18m04s	11m43s	8m28s	8m11s	6m37s
4M	20m41s	13m33s	9m42s	9m28s	7m32s
5M	24m19s	15m28s	10m58s	10m33s	8m21s
6M	25m42s	16m46s	12m15s	11m31s	9m27s
7M	28m13s	17m59s	13m04s	12m23s	9m42s
8M	30m10s	19m19s	15m11s	13m53s	10m21s
9M	33m04s	21m36s	17m15s	15m51s	11m00s
10M	37m17s	24m02s	18m47s	18m20s	12m07s

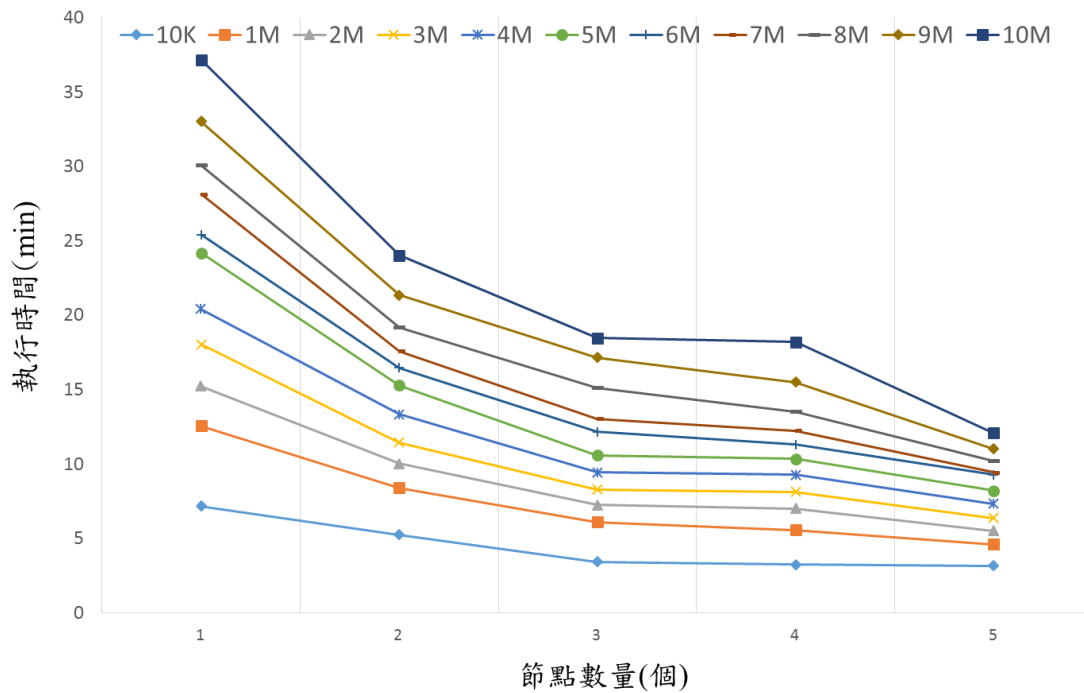


圖 9 資料量與執行時間的關係圖

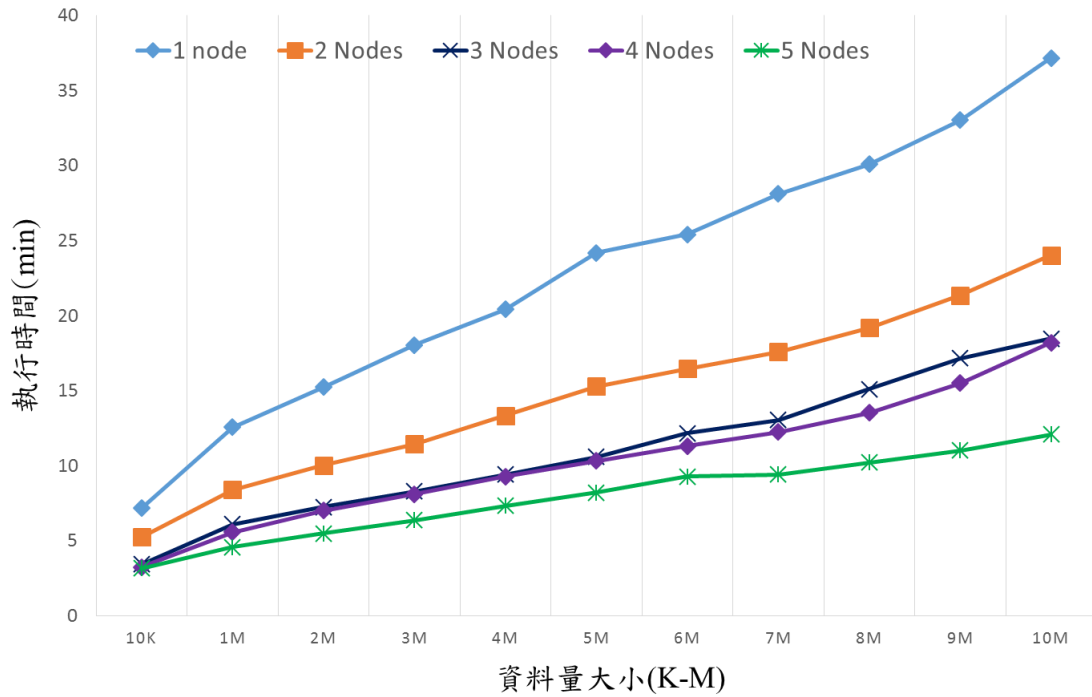


圖 10 節點數量與執行時間的關係圖

4.2. 加速比

利用 Amdahl's law[7]粗略模型的性能加速比(Speedup)的公式 3，在公式中 T_1 表示在單節點的計算執行時間， T_p 代表多節點計算的執行時間。

此演算法若是具可伸縮性的，加速比會隨著資料集的增加呈線性成長如圖 10 所示，另外在一千萬筆資料的部分在奇數節點會有更好的效能，但在一萬筆資料的部分加速比數值並無明顯提高，原因是在 1 萬筆的資料在現今硬體設備的運算能力完全能由一台處理且迅速完成計算，但在各別主機上運算需要透過網路回傳計算結果，網路的傳輸速度會影響運算時間導致結果輸出的時間增加進而影響加速比的數值。

$$Speedup = \frac{T_1}{T_p} \quad \text{公式(3)}$$

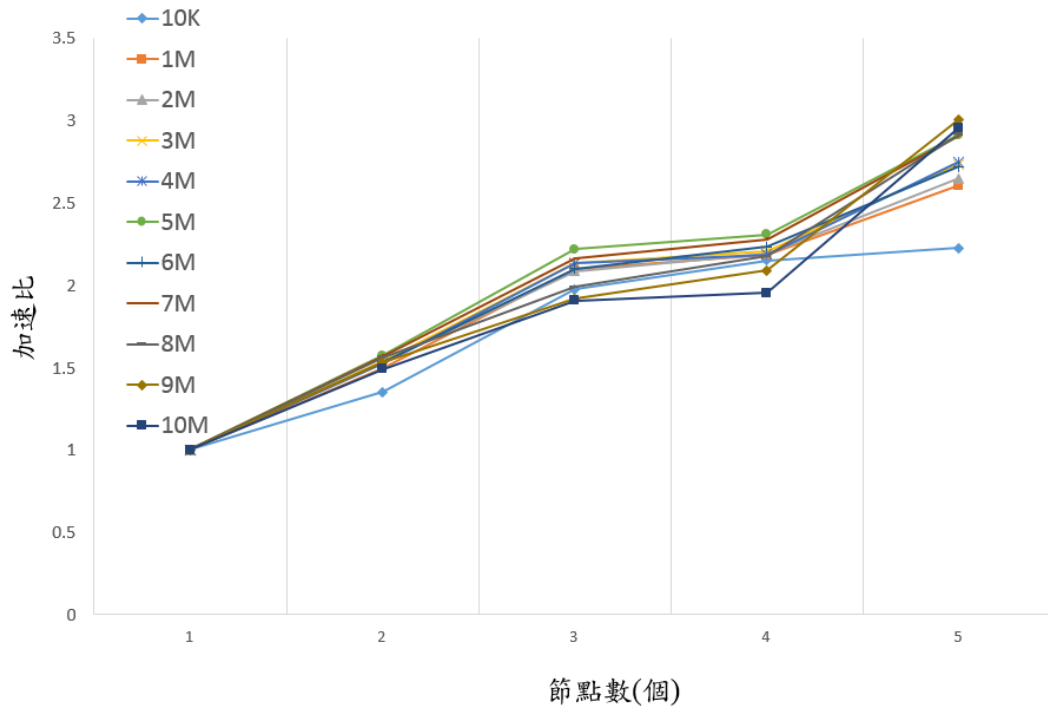


圖 11 加速比圖

第五章 結論

在本篇研究中，通過把 MapReduce 的方法將基於項目的協同過濾演算法改寫成分散式運算的程式，因為實際環境在資料量增加情形下，單台的運算主機需要較多的計算時間，儲存空間也會因資料過大造成空間不足無法計算出結果，透過 Hadoop 平台上面的 MapReduce 方法將運算與資料分散給不同的運算主機，最後收集各主機運算結果集成最後結果，把這些運算與資料分給其他台主機去運算可以節省運算時間也能節省儲存空間，另外 HDFS 解決了資料正確性與完整性的問題可以保證資料的正確性，實驗結果上可以看見單節點在資料增加後，計算時間呈直線成長，再增加節點後計算時間有明顯的減少，另外可以見得在奇數節點的計算效能較佳，證明 Hadoop 平台的擴展性利用增加節點提高處理速度與增加儲存空間，實驗使用了虛擬化技術在實體資源上會有限制，所以與實體主機會有效能上的差別。

在未來可以結合混合推薦的技術，依照不同情況做適當的推薦提升推薦的準確度，在硬體的部分能提升硬體設備的效能，達到提高基本運算與儲存的能力，在不同領域的資料，可以作收集資料並且抓取出所需要運算的資料格式與型態，來做不同領域的推薦。

參考文獻

- [1] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, “The Hadoop Distributed File System,” IEEE 26th Symposium on Mass Storage Systems and Technologies , 2010
- [2] J. Dean, S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, 2004.
- [3] Heng-Song Tan, Hong-Wu Ye, “A Collaborative Filtering Recommendation Algorithm Based on Item Classification,” Pacific-Asia Conference on Circuits Communications and System, pp 694- 697,2009.
- [4] Zhi-Dan Zhao, Ming-Sheng Shang, ”User-based Collaborative-Filtering Recommendation Algorithms on Hadoop,” Third International Conference on Knowledge Discovery and Data Mining, pp 478- 481,2010.
- [5] Badrul Sarwar,George Karypis, Joseph Konstan, and John Riedl, ”Item-based Collaborative Filtering Recommendation Algorithms,” Proceedings of the 10th International Conference on World Wide Web, pp 285- 295,2001.
- [6] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl, “MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System,” Proceedings of the 8th International Conference on Intelligent User Interfaces, pp 263-66,2004.

- [7] G. Amdahl, “ Validity of the Sigle-Processor Approach to Achieving Large Scale Computing Capabilities,” American Federation of Information Processing Societies Conference, pp 483-485,1967.
- [8] Apache Mahout.<http://mahout.apache.org/>.
- [9] GroupLens Research Data Set. <http://www.grouplens.org/node/73>.
- [10] Apache Hadoop. <https://hadoop.apache.org/>.
- [11] J. Ben Schafer, Joseph Konstan, and John Riedl, “Recommender Systems in E-Commerce,” Proceedings of ACM E-Commerce Conference,1999
- [12] Lyle Ungar, and Dean Foster, “Clustering Methods for Collaborative Filtering,” Recommendation Systems at the 15th National Conference on Artificial Intelligence,1998
- [13] Mukund Deshpande, and George Karypis, “Item-based top-N Recommendation Algorithms,” ACM Transactions on Information Systems, pp 143-177,2004.
- [14] Loren Terveen, Will Hill, Brian Amento, David McDonald, Josh Creter, “A System for Sharing Recommendations,” Communications of the ACM, pp 59-62,1997.
- [15] Badrul Sarwar, George Karypis, Konstan, Joseph Konstan, and John Rield , “Analysis of Recommendation Algorithms for E-Commerce,” ACM EC'00 Conference. Minneapolis, MN. pp. 158-167,2000.

- [16] Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl, "Combining Collaborative Filtering with Personal Agents for Better Recommendations," Association for the Advancement of Artificial Intelligence Conference, pp. 439-446, 1999.
- [17] Chumki Basu, Haym Hirsh, and William Cohen, "Recommendation as Classification :Using Social and Content-Based Information in Recommendation," Association for the Advancement of Artificial Intelligence Conference, pp. 11-15, 1998.
- [18] 奉國和、黃家興，『基於 Hadoop 與 Mahout 的協同過濾圖書推薦研究』，圖書情報工作，第五十七卷·第 18 期：116~121 頁，2013。
- [19] 張永霞、王洪波、程時端，『一種基於 Hadoop 的個性化推薦系統架構』，新型工業化，第二卷·第 8 期：7~12 頁，2012。
- [20] 王耀聰、辜文元、魏綸毅譯，Hadoop: The Definitive Guide, Second Edition. Tom White 著，Hadoop 技術手冊 第二版，台北，碁峯資訊股份有限公司，2010。