運用程式分析於巨量資料處理分流暨系統資源動態配置以增進系統運作效能

溫演福，陳立祥

國立臺北大學資訊管理研究所

摘　要：由於網路科技的逐漸成熟與雲端運算技術的出現，在雲端運算的環境下進行大規模的運算工作變得更容易與快速，雲端運算背後概念主要為分散式架構，讓許多的電腦能夠透過網路的連結進行資源的結合與運用，資源能夠被使用、分享也可以進行動態的配置，透過分散式架構的優點，讓其應用能夠更佳的廣泛。但是同時有大量的連線輸入進來時會造成電腦運算上的極大的負擔，要如何有效的配置資源與運用，讓系統不會因為過載而導致當掉或延遲，是目前一個很重要的議題。本研究以購物網站為例，透過程式分析將多個不同程式進行分類，將程式中具有相同性質的資料歸類在一起，對同類性質和將執行同一程式流程的資料進行主機資源的優化配置，藉由了解該性質資料的所需運算時間、CPU 效能以及其他可能會導致運算速度上有所影響的因素，就能夠將預估的處理時間差縮小，減少主機在運算時所要分配給該資料的資源，進而減少主機的負擔量，本研究將透過模擬的方式，來將程式分析後所產生之類別序進行效能上的評估，來改善過去使用經驗的累積來配置適當的資源給該程式進行運算。

一、论述

1. 研究背景

隨著網際網路快速的發展，大數據的出現使的資料不只再侷限於文字與圖片，大數據(Big data)的 3V 特性包括：資料性(V)、多樣性(V)、和及時性(V)，意味著現在的時代將面臨的是資料量很龐大、種類複雜且要能夠及時處理，因應這樣的趨勢，雲端運算就顯得特別的重要，透過雲端運算的分散式架構概念，能夠將同性質的資料處理分隔到不同的主機運算來提升處理資料的速度。

根據 IDC 研究顯示,到了 2020 年的數位資料量將會成長到 40 ZB，隨著資料量的快速成長，在資料處理的方法上也變得相當不同，雲端運算是目前處理大數據的趨勢，雲端運算的關鍵技術為 MapReduce，它是一種解決問題的程式開發架構，也是拆解問題的方法,許多大企業像是 Google、Facebook、Twitter 都是透過 MapReduce 來處理巨量資料。

對於這麼多執行序的資料處理，要如何在有限的資源下提供符合需求的配置方法也成為重要的一個研究議題。例如：購物網站經常性地舉辦促銷活動，特別是阿里巴巴集團旗下的淘寶網、天貓商城於每年 11 月 11 日所推出的雙十一節折扣活動，這些活動使當日會湧入大量的流量，根據統計在當天尖峰時段的交易量每秒高達 1 萬 3 千張訂單量，這是一個非常龐大的交易量，依此系統運算需要能夠處理這樣龐大的流量，流量變大就會導致購物網站的伺服器負載增加，負載增加可能就會導致購物網站整個延遲甚至造成網站當掉，要如何有效的配置資源來降低系統的負載量去避免網站可能停擺或降低回應時間等現象，在雙十一節當天，淘寶網都並未出現上述的現象，淘寶網是如何做到負載平衡的呢?以下列出幾點淘寶網改善系統過載時的方法:

（1）彈性的快取機制:

若每筆查詢資料都要經過主機作處理，則有再多的 CPU 與頻寬都會不夠，所以透過在主機與使用者間架設內容傳遞網路(Content Delivery Network)來降低系統

承載量，以將流量降低至總量的十分之一。

CDN 是透過網路互相連接的電腦網路系統，利用最靠近每位使用者的伺服器，傳送資訊給使用者，以提供高效能、可擴展性及低成本的網路內容給使用者。

（2）讀寫分離的設計:

即時處理的資料，需要考慮資料寫入時的資料一致性問題，例如：使用者修改基本資料，則必須保證不會有兩種不同的版本，要付出的網路傳輸、運算、與檢查的成本都要比讀取資料來的多，透過將資料庫分割成讀寫分離的兩種資料庫，依不同需要制定各別的資料格式、資料演算法，就能夠使資料庫的效益發揮得更好。

（3）實施服務降級:

介紹了前面兩種解決過載的方法快取機制與讀寫分離，但在面對即時且大量的訂單量下，仍然可能出現系統過載的情況。若此時伺服器顯示「404 Not Found」這樣的回應訊息，可能會嚴重打擊到企業的形象。所以透過實施服務降級的方式，而不是停止服務。

所謂服務降級，是漸進式地停止邊緣性服務，確保核心流程的可用性，例如：流量過載時，應先停止使用者查詢購物過往的歷史或修改基本資料等功能，讓正在結帳階段的使用者優先執行結帳程序，不會等到按下確認匯款，才發現網站當機。

（4）最終一致性:

由於購物網站存在著龐大的使用者數量與訂單量，許多資訊的更新很難及時達成，常會出現分散式架構在計算裡，所謂最終一致性（Eventual Consistency）的問題。淘寶分析了使用者習慣的行為，來決定傳輸、運算資源配置與回應的優先順序，例如：使用者通常不會在下完單後，又立刻改單，因此，訂單資訊只要在 500 毫秒內達成資訊同步，是不影響使用者體驗的。

藉由了解阿里巴巴在處理購物網站流量過載的解決方法，本研究根據阿里巴巴解決過載方法中所的 CDN 概念提出了透過程式分析來將具有相同性質的資料進行分類以達到分流的效果，透過分流的方式以(i)改善資源配置和(ii)解決負載平衡的問題。

2. 研究動機

本論文以購物網站為例，購物網站特性包含了以下幾點：

（1）每天有大量的訂單出現，而訂單要能夠及時性進行回應。

（2）網站會包含許多的圖片、影片、音樂、音效、文字資料與流程步驟。

（3）網站每天都會有大量的連線進來進行瀏覽或交易的動作，會產生大量的資料。

根據以上幾點特性，符合大數據所提出的 3V 概念，資料量(Volume)、速度性(Velocity)與多樣性(Variety)，代表著購物網站在進行資料處理的方法上會採用分散式的架構來進行處理，當非常多的人同時使用購物網站時，著重於系統面會造成的影響：

（1）某些功能、資料記錄、以及資料欄位的處理形成一個瓶頸。

（2）各地區網路差異，跨境網站的可用性及性能問題。

（3）系統將因此變慢，以致影響到使用者使用系統的服務品質。

（4）客訴問題，以致於影響到整個公司運作。

為了解決大量連線導致的流量問題，以及如何有效地預估處理的流量，以動態地規劃資源配置是主要考量的議題之一，對於執行序所需的資源過去以來都是以經驗法則方式來預估、或是給定的預測公式，對於資源配置提供一個近似或是概估的演算法，對於實際應用時，可能會產生誤差，於是需要設置較高的資源門檻以降底預測不準時所造成的影響，因此，本研究著重於程式執行流程之分析，再依不同的輸入資料進行分類以更精準地配置所需執行的系統資源。

透過程式分析來了解不同程式流程(program flow)所需要的資源數目依分析

結果來決定功能與資料之切割的方式，以分散在不同的主機上運作。依功能與資料欄位來切割系統，使這些執行功能配置在不同的系統中執行，以及資料分群後，它需要多少主機來執行？依哪些資料被執行的頻率來決定執行主機配置區塊與大小？

3.研究目的

依上述改善目標，以達到有效率的分散式系統運作，本論文希望能夠達成的目的為:

（1）對系統負載的配置進行預先的準備。

（2）降低系統負載過量可能導致的問題。

（3）減少系統成本的花費、提升系統運作效能。

(4)改善使用者使用系統的服務品質、降低客訴問題量。

二、文獻探討

1.大數據

大數據(Big data)又稱巨量資料，指的是所涵蓋的資料量的規模大到無法透過人工或者計算機在合理的時間內做到擷取、管理、處理並整合為人類所能解讀的資訊。根據麥塔集團分析員道格・萊尼指出:「資料成長的挑戰和機遇有三個方向，分為資料大小(Volume)、資料傳輸速度(Velocity)與多樣性(Variety)」，意味著現在面對的是資訊量龐大、資料類型式多樣並且要能夠及時做處理的時代。

以淘寶網的購物網站為例，一個購物網站每天需要接收到的資料量是相當龐大且可觀的，購物網站也會提供多樣的服務，意味著資料的類型也會是多樣化，並且涉及到交易的問題要能夠及時的去做處理，有符合大數據所提出的 3V 概念，因此，根據現有處理巨量資量使用的架構為分散式架構，而分散式架構為雲端運算的一個概念，以下將會介紹。

2.雲端運算

（1）Hadoop

Hadoop 是用來處理與保存巨量資料的雲端運算平台，它是 Apache 底下的開放原始碼軟體，由 Java 編寫而成，提供大量資料的分散式運算環境專門用來架設叢集式電腦，其中 MapReduce 與 HDFS(Hadoop Distributed File System)為 Hadoop 的兩大核心，運算的部分以 Map Reduce 為架構，HDFS 則是 Hadoop 的檔案系統，並擁有高容錯率、高效性和高擴充性等優點，適合在巨量資料量下執行。

（2）MapReduce

MapReduce 為為 Google 的核心運算模型，是一種並行的程式設計模型，它可以將工作分發到多台電腦組成的叢集上進行運算處理，其主要概念是映射(Map)和化簡(Reduce)兩種，映射是指從主節點(master node)輸入一組 input，此 input 是一組 key/value，將這組輸入切分成好幾個小的子部分，分散到各個工作節點(worker nodes)去做運算；化簡指的是主節點(master node)收回處理完的子部分，將子部分重新組合產生輸出，以下是 MapReduce 的執行示意圖 1 所示，其步驟為：

①Split：把要執行的程式分割成多個子部分(File a、File b、File c)。

②Map：將分割的程式透過 Map 分配到 Worker a、Worker b、Worker c 機器中進行計算。

③Reduce:將計算完的結果傳入 Worker d 機器值中，進行計算結果的彙整與排序。

④Output:將運算結果輸出。

此概念雖然透過將一筆輸入的資料進行切割後到不同主機上計算，但是其切割的方式並沒有依照相同的性質進行切割，在分配給不同主機運算時可能會因為不同主機所擅長的資料處理類型而導致在計算上的速度可以會有偏差，且再進行合併資料時由於相同類型的資料被配置到不同主機上可能會受限於各主機距離，以致於回傳速度較慢，本研究透過程式分析來了解輸入的資料內容並找出相同性質的資料進

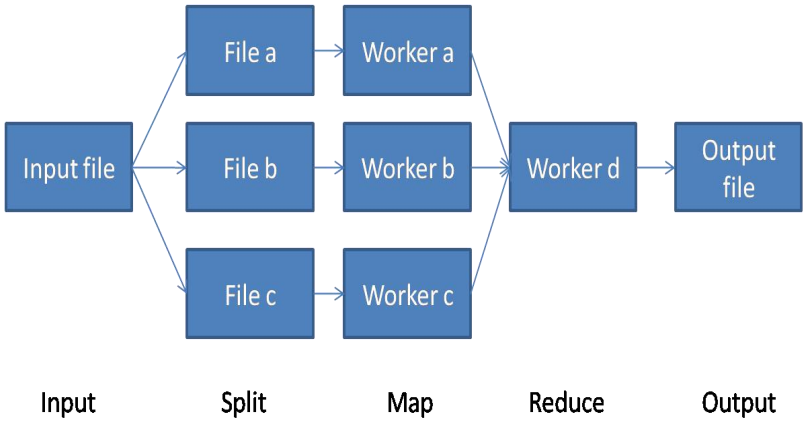

圖 1 MapReduce 的執行示意圖

行分類，再依據分類的類型選擇配置到所擅長處理該資料類型的主機上進行計算，透過這樣的方式能夠改善主機配置的資源與降低預估的時間差。

3. 負載平衡

負載平衡是指當利用多個運算資源來共同負擔計算所需要的資源時，基於不同的目的與需求，透過特定的方式，來控制在不同計算資源上的負載量，進而達到最佳化系統的目的。例如：透過雲端運算，系統可以利用多台主機，來同時負擔多人的連線需求，藉以提供更快速、更有品質的服務效能。

當有巨量流量進入購物網站，進行許多的交易行為，會產生許多的指令，指令會被分散到不同的伺服器上去進行運算再回傳給原主機，而當每台伺服器都為過載的情況時，要如何的善加分配後續的指令要到哪台伺服器去進行運算，為了能夠有效率提升處理的速度，常用的排程演算法為下列所列幾種:

（1）最短程序優先(Shortest Process First，SJF)

最短程序優先會估算在佇列中所有程序的執行時間，再選擇執行時間最短的程序，如果其中有兩個或以上的程序有相同的執行時間，就會採用先進先出法處理程序。

此方式的缺點為估算程序的執行時間誤差範圍可能會較大，由於要處理的程序有很多，若某個程序實際只需要 3 秒的時間就可處理完畢，但該方法可能會估算配置給該程序 5 秒的處理時間，會導致實際執行情況會較久。

（2）輪流法(Round-Robin，RR ）

輪流法的做法是先設定每一個程序一段時間量(quantum)，如果有程序在這段時間用完而還沒執行完時，處理器會執行下一個程序。而輪流法的效能完全取決於時間量的長短，如果時間量設定太長，那其效能就會跟先進先出法一樣；如果時間量設定的太短，內容轉換就會經常發生，系統的執行效率也會因此降低。

但是此方式的演算法有個缺點，若遇到需要執行的程序是相當龐大時，可能會導致該台主機要一直對該程序進行處理，所以會導致該系統一直處於負載的情況，因此在負載平衡上效果較差。

（3）CDN(內容傳遞網路，Content Delivery Network)

CDN 是透過網路互相連接的電腦網路系統，利用最靠近每位使用者的伺服器，更快、更可靠地傳送資訊給使用者，來提供高效能、可擴展性及低成本的網路內容傳遞給使用者。

藉由此方式能夠帶來分流的效果，靠近該使用者區域的伺服器就負責該區域的處理，本研究透過程式分析將相同類別的

程序進行分類，各個分類有其對應處理的伺服器區塊以專門負責處理，藉此來達到分流的效果，跟 CDN 的目的較為相似。

4. 程式分析

（1）程式流程(program flow)

要驗證一個程式的程式碼時會把主要程式拆成許多副程式，藉由驗證的過程可以看出一段程式碼是如何執行，而程式流程是指依特定資料而劃分所要執行的一組程式碼或是函式，依此於前端處理器收到的請求處理，即劃分不同的資料執行所需要的不同組的程式碼，進而安置不同的主機與資源(如：處理速度、傳輸頻寬、記憶體等)來負責處理特定範圍內的資料。

依圖 2 所示，Offline Process 的處理流程是配合程式在驗證時所產生，將變數值的範圍集合輸入至應用系統程式後，由於每個變數群執行完程式後的結果都會不同，再把所有可能處理的資料進行分群後，最後產生本研究所要用到的輸入對映執行序的表。例如：在圖 3 所示的範例程式中，如果選擇做加法，就會跳到加法的執行程式，作答完就會跳到檢查的程式，如果答對就顯示 Correct，反之則顯示 Wrong。但如果今天有非常龐大的使用者同時使用此程式，此程式的一開始就必須要估算有多少人要做加法、做減法，進而適當的進行後端主機資源配置，讓處理的時間縮到最少。

對於程式執行的迴圈數不同，則預估執行的時間將不同，針對這個問題可以運用下列幾種方法來進行預估：

①依長期資料執行之時間記錄，再進行統計分析以取得趨近的平均回應時間。

②當某一變數是某一迴圈的決策變數，則給定的變數值所需執行的迴圈數也將相同；當考量多個變數組成時，那特定範圍內的數值帶入程式流程也將相同，依此決定所給定的一組值所需執行的資源，進行推估在某一時間區間所需要的執行資源。

③依給定的條件與資料作為輸入，即可以產生特定範圍的程式流程。

2. 符號執行(Symbolic Execution)

是指一種程序分析的技術，透過分析程序得到讓特定代碼區域執行的輸入。使用符號執行來分析程序時，該程序會使用一組符號值作為輸入，而非一般執行程序時使用的具體值。在達到目標代碼時，分析器可以得到相應的路徑約束，然後通過約束求解器來得到可以觸發目標代碼的代表值。

五、時間預估

現有一般針對執行期的時間預估普遍採用經驗法則，公式包括：

1. PERT 計畫評核術

計畫評核術是一種規劃與控制專案的方法，它在時間預估方面採用的是三十估計法，也就是把期望時間分別定義為樂觀時間、悲觀時間與最可能時間，樂觀時間代表在最理想的狀態，且並不會有任何突發情況發生所預估的時間；悲觀時間則是相反，

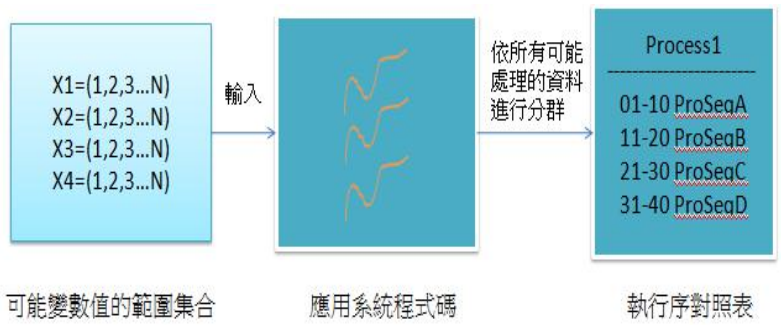

圖 2 Offline Process 的處理流程

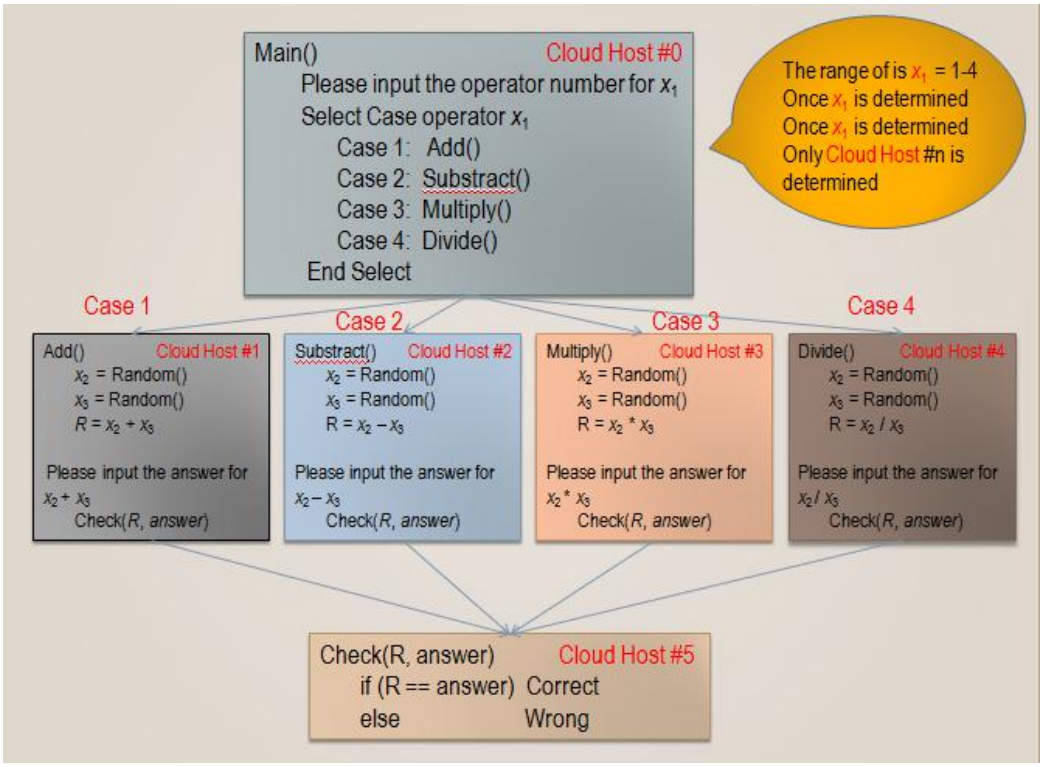

圖 3 程式流程執行分流範例

它代表在最極端惡劣的狀態下所預估的時間；而最可能時間代表介於樂觀時間與悲觀時間之間，比較符合在實際狀況下所預估的時間，三十估計法計算專案的期望時間的公式為：（樂觀時間 (Optimistic Performance Time)+4*最可能時間 (Most Likely Time) + 悲觀時間 (Pessimistic Time))/6。

2. 最壞執行時間(Worst-Case Execution Time)

最壞執行時間是用來計算工作或執行緒的「最差情況執行時間」，其計算方法可分為兩大類：靜態分析和測量式方法。靜態分析方法的做法為將程式碼與抽象的系統模型結合，藉此得到執行時間的上限。靜態分析法可分多階段進行，透過數值分析決定處理器的暫存器和區域變數值在每個程式點的範圍，控制流程分析計算可能的執行路徑，處理器行為分析預測記憶體、快取和管道等對於執行時間的影響，最後由估計計算決定 WCET 的估計值。

測量式方法則以給定的輸入集合在硬體或模擬器上執行程式來測量執行時間，可用於提供實際執行時間變化的概況，但無法保證執行時間的上界。此方法可透過額外的插樁程式碼來收集時戳或是 CPU 循環計數等方式進行測量。

3. 移動平均法

移動平均法是用來分析時間序列數據的方法之一，它是一種簡單平滑的預測技術，根據時間序列資料、逐項推移，依次計算包含一定項數的序時平均值，以反映長期趨勢的方法。適用於即期預測，其基本公式為 Ft=(At-1+At-2+At-3+ …+At-n)/n，藉由分析過去的資料而對未來進行預測。

4. 卸載(offloading)

亦有運用雲端資料卸載(offloading)的算式來計算所需執行的指令數，來將程式分段，並且決定這些處理是要在行動裝置、還是要在雲端主機上處理，資料卸載決策分為靜態(static)及動態(dynamic)

兩種模式，卸載執行之前需決定靜態及動態的影響變數以增進效能，靜態決策模式採用固定事先預測參數及決定程式的那些部分需要卸載，但是因為環境不同，以致於無法保證靜態決策為最好的模式；動態決策模式可以適應不同的環境及運行的時間條件，但是這種估算方法的缺點包括：(1)沒有考量到資料量；(2)需要動態配置運算主機，而無法事先運算；(3)只能估算指令集的數量，而無法準確預做之間迴圈與搭配資料量來進行估算。

三、解決方法

Offline 時的程式分析，給定一個應用系統程式碼和所有變數值的集合範圍，根據這兩方面的資料來做系統分析(利用程式驗證時所進行每段程式測試時所獲得的資料分佈來執行)，依不同變數值進行程式分析已產生對應的執行序，例如：輸入 10 的話會執行 a 執行序,輸入 20 的話會執行 b 執行序;再輸入一次 10 的時候還是會出現 a 執行序，不會跑出 b 執行序出來，這時就會產生出一個程式流程對照表(Table)，當 Online 執行時，只要依據這個 table 你就可以很迅速做出比對與對映。系統架構圖之線上處理說明：

1. 先將消費者的訂單依據某種分布分配到 Rn，例如：將鞋子類的訂單分配到 R1 群集主機，短褲類的訂單分配到 R2 群集主機。

2. 依輸入資料分出處理群，把所有訂單分類後放入佇列。

3. 將佇列與程式流程對照表比對與對應後，對訂單進行排程處理

4. 依據排程處理的資料量大小將後台處理主機做最適當的資源配置。

5. 將訂單資料處理完後，可在回饋給上一階段的排程處理以改善或增進排程方法。

四、結語

透過模擬的方式來將程序分析所分類出的程式流程對照表進行預跑的測試，由於剛開始並不知道各程式流程所需執行時間，所以我們會先假設每個執行緒都執行預跑 5 秒，等到預跑結果出來，再針對預跑的結果進行二次預跑，直到預跑的結果值達到較穩定狀態時，就可以作為往後被歸類到

表 1 系統執行期之時間預估比較表

預估方法		優點	缺點
依經驗法則	PERT 計畫評核術	得到的時間預估是最具代表性的	遇到比較極端會造成配置資源不夠或多餘。
	移動平均法	比較不複雜，又能反映現況	由於是用過去的資料為依據,在預測時還是可能會有落差
依實際量測	WCET	系統最不容易超載	系統資源分配過剩 必須事先量測所要執行的時間
	卸載(offloading)法	可讓行動端處理資料，減少處理時間	必須事先處理程式分割且量測所需花費處理的時間。

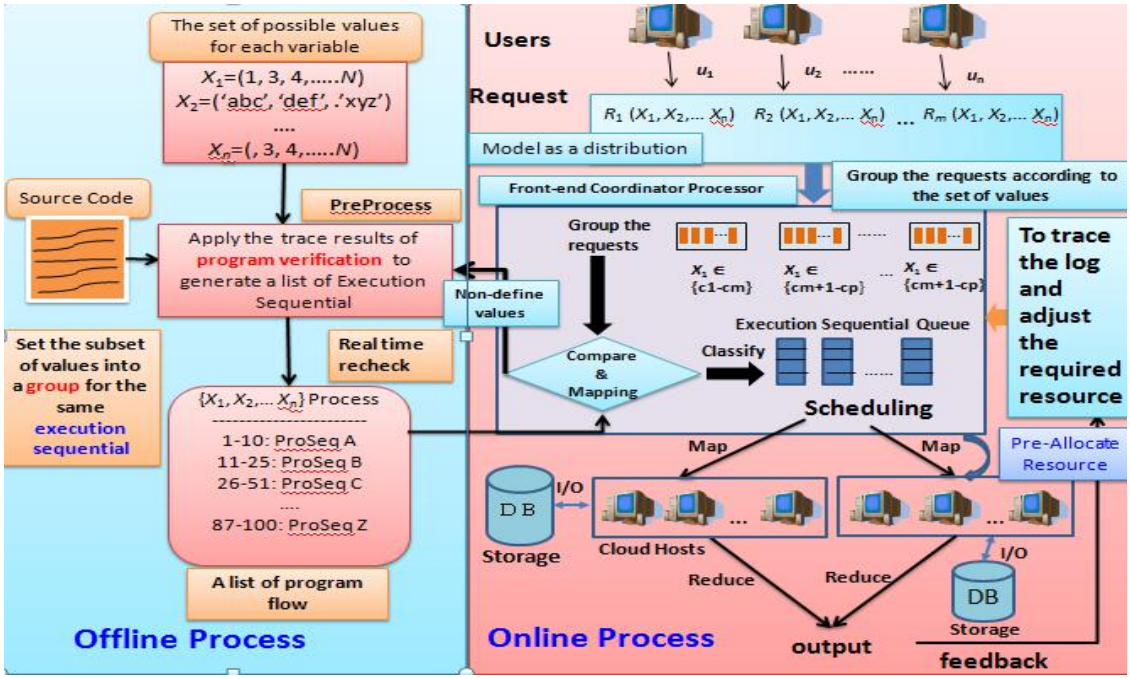

圖 4 程序分析的前置作業

該程式流程對照表的程序能夠使用更加準確的執行時間來配置給伺服器進行計算，以減少過去靠預估或是經驗法來配置的時間誤差值。

參考文獻

[1] http://www.ithome.com.tw/node/81790 負載平衡是分散式資源串連之鑰.

[2] http://www.ithome.com.tw/node/81962 淘寶雙 11 光棍節單日上億筆訂單零漏單的 IT 關鍵.

[3] 適用於雲端環境下有效率負載平衡演算法之研究與設計 靜宜大學資訊管理學系碩士論文 張鈞徨.

[4] Borthakur, D. (2008). HDFS architecture guide. Hadoop Apache Project, 53.

[5] Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters, OSDI '04, pages 137–150, 2004.

[6] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[7] Han, J., Ishii, M., & Makino, H. 2013, A Hadoop performance model for multi-rack clusters. In Computer Science and Information Technology (CSIT), 2013 5th International Conference on pp. 265-274.

[8] Heckmann, R., & Ferdinand, C. (2004, April). Worst-case execution time prediction by static program analysis. In In 18th International Parallel and Distributed Processing Symposium (IPDPS 2004, pages 26–30. IEEE Computer Society.

[9] K. Kumar, J. Liu, Y.H. Lu and B. Bhargava, "A survey of computation offloading for mobile systems," Mobile Netw. Appl., vo. 18, no. 1, pp. 129–140, Feb. 2013.

[10] S. Ou, K. Yang, and Q. Zhang, "An efficient runtime offloading approach for pervasive services," in Proc. of IEEE WCNC, Apr. 2006, pp. 2229–2234.

[11] Schwartz, Edward J., Thanassis Avgerinos, and David Brumley. "All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask)." Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 2010

[12] White, T. (2009). Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.".

[13] Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., ... & Stenström, P. (2008). The worst-case execution-time problem—overview of methods and survey of tools. ACM Transactions on Embedded Computing Systems (TECS), 7(3), 36.

[14] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, “HSim: A MapReduce simulator in enabling Cloud Computing,” Future Generation Computer Systems, 2011.