基於 OpenStack 的雲端測試即服務平台
An OpenStack Based Testing as a Service Platform

葉書維、鄭永斌
國立中央大學資訊工程學系
Shu-Wei Yeh, Yung-Pin Cheng
Email: 103525007@cc.ncu.edu.tw, ypcheng@csie.ncu.edu.tw

摘　要

軟體開發過程中，自動測試雖然是確保軟體品質重要的一環，然而學習、維護、佈署自動測試環境與工具所帶來的成本卻仍不容忽視。本論文提出一個測試即服務(Testing as a Service, TaaS)雲端平台，結合 OpenStack 以及自行開發的 TaaS Web Potal 介面，將測試工具虛擬化，大幅降低維護及佈署系統所需的成本。開發團隊能夠在容易操作的網頁介面下上傳測試工具的測試腳本並調整測試相關參數，即可利用自動測試工具進行測試，無須面對測試環境的建立、相關硬體準備與設定等種種問題，減少執行測試所需的人力成本。同時，由於 TaaS 是使用 OpenStack 作為雲端運算平台，只需擴充 OpenStack 的運算節點數量即可快速的擴充。

關鍵字：Testing as a Service、TaaS、OpenStack

一、 簡介

軟體測試無論在開發中、開發完成後的交付、後續維護及擴充等，都是相當重要的一環。對軟體使用者而言，持續並正常通過軟體測試是確保使用中的軟體品質最重要的依據。對開發團隊而言，測試能夠讓開發者確認自己撰寫的程式模組有無如自己預期的正確執行，效能表現是否足以應付需求，有沒有潛在的資訊安全風險等，並且在後續維護及新功能開發時能夠確保新加入的更動沒有影響到舊有功能的正常運行。

對軟體的測試主要可以分為兩類：功能性測試(Functional Testing)及非功能性測試(Non-functional Testing)。功能性測試的目的在於測試軟體中的各項功能是否正常如預期的運作。例如測試網頁時，當使用者點選超連結，網頁是否如規格所規定的跳轉到正確目的地網頁。此種測試的主要困難點，其一是如何撰寫足夠數量，且能夠覆蓋到錯誤所在位置的測試案例。另外一點是此種測試儘管只需照著軟體的規格書或使用手冊照著步驟做一遍即可，但若以人工的方式進行此種測試，不但曠日廢時，更會浪費許多人力資源，且難保因人為失誤而產生錯誤的測試結果。

非功能性測試則是在測試系統的運作情形及效能等特性，例如負載測試、壓力測試及安全性測試等。負載測試及壓力測試的目的在於測試軟體在面對大量使用者同時上線時，能否維持正常運作而不會遇到資料庫存取或網路頻寬等瓶頸。更進一步，透過壓力測試我們可以測試系統的極限在哪裡，確認系統在現有的瓶頸狀況下，能夠容納多少同時上線使用者，以評估改進這些瓶頸是否符合效益，抑或是這些瓶頸在現有的網頁服務使用者數量下並不會造成影響。安全性測試則是對軟體進行全面性的掃瞄，檢查軟體是否有潛在的資訊安全風險。相較於功能性測試，由於不同的軟體對於流量、安全性等指標會有不同的可接受程度，非功能性測試較難有一個量化的指標明確界定測試通過與不通過。另外，由於非功能性測試時常需要模擬大量同時上線使用者，此種測試所需的運算資源相較於功能測試會多上數倍，同時也不可能由真人進行大量同時上線使用者的模擬。

隨著科技的進步，軟體規模日益擴大，使用人力來進行測試所需耗費的人力成本也越來越高。為了降低對軟體進行測試所需耗費的成本，我們可以使用自動測試工具來協助開發團隊對軟體進行測試。透過使用測試工具，執行軟體測試時可以避免因人為失誤造成錯誤的測試結果，同時減少測試所需耗費的人力及時間。由於測試案例能夠被自動執行，當開發團隊對軟體進行更動時，可以馬上對新版本進行軟體測試，當新更動對舊有功能造成影響時能夠及早被發現。

然而在減少人工進行測試所需耗費的人力成本的同時，使用自動測試工具也會為開發團隊帶來額外的成本。首先，開發團隊除了原本的開發工具外，需要花額外的時間尋找符合需求的測試工具並學習如何使用該工具撰寫測試腳本。另外，測試腳本的程式碼也是程式碼，開發團隊仍需要額外花人力資源對其進行後續維護及修正。再者，針對壓力及負載測試等需要大量運算資源，雖然部分工具有提供分散模式能夠串連多部電腦同時進行測試，測試人員仍須熟知要如何佈署該測試工具於多台電腦上，並對每一台電腦調整其設定，才能確保分散測試能夠正常運作。基於以上原因，許多開發團隊雖然了解自動測試的重要性，但仍然不會對開發中的專案進行測試案例的開發及維護。

為了降低開發團隊使用自動測試工具的門檻，本實驗室過去曾開發一套 TaaS Web Portal 第三方測試即服務平台。該平台整合兩套網頁自動測試工具、持續整合工具，以及本實驗室所開發，包含專案管理及測試工具設定的網頁操作介面。開發人員只需上傳測試腳本並在網頁介面上調整測試相關

參數即可進行測試，無須顧慮測試工具的相關設定及佈署細節。然而該系統在雲端平台的架設上不甚方便，平台維護人員在擴充運算資源及相關設定的時候，仍需手動對各台機器進行設定調整，因此雖然減少了開發人員使用測試工具的成本，對於測試服務的維護人員，其維護成本仍然沒有降低。

本研究基於 TaaS Web Portal 對其進行改良，將其與 OpenStack[1]的架構即服務(Infrastructure as a Service, IaaS)雲端平台進行整合。測試人員在使用改良過後的 TaaS Web Portal 時，仍然只需將測試腳本上傳並調整相關測試參數即可直接使用測試工具，測試工具的設定及佈署等會由 TaaS Web Portal 自動在 OpenStack 上完成設定。對 TaaS 服務的維護者而言，只需擴增 OpenStack 的運算節點數量，即可讓所有測試工具使用新增加的運算資源進行測試，且無須額外對各台機器重新進行測試工具的設定。

二、 背景與相關研究

2-1 自動化測試工具

面對測試自動化，目前已經有許多商業化的自動測試工具或開放原始碼的測試工具可供使用。透過使用自動測試工具，開發人員可以將測試案例製作成對應於自動測試工具的測試腳本，之後再利用自動測試工具重播測試腳本，即可完成測試案例的自動測試。透過使用自動化測試工具，可以避免測試時的人為失誤，減少測試所需耗費的人力及時間。同時，因為有可以自動測試的測試案例，開發團隊進行程式碼開發及整合時，只需將測試案例全部執行過一次，便可知道新修改的程式碼是否有影響到過去正常運行的程式，使軟體開發上能夠更及時的發現錯誤。

在TaaS Web Portal中，我們選擇Selenium[2]作為我們對網頁進行自動測試所使用的測試工具。Selenium是一套錄製及重播(Capture and Replay)的網頁功能性測試工具。測試人員除了自行撰寫測試腳本外，在錄製腳本階段時測試人員可以實際操作一次瀏覽器，透過Selenium官方提供的Selenium IDE瀏覽器套件將其透過網頁瀏覽器所做的點擊、輸入等動作錄製成測試腳本，同時記錄用以判斷測試是否成功的測試準則(Test Oracle)。重播時，Selenium會開啟瀏覽器，並將測試腳本中記錄的動作重新撥放一遍，並比對測試腳本中所記錄的測試準則(Test Oracle)是否與重播時產生的結果相同，藉以判斷測試是否成功。

非功能性測試的部分，TaaS Web Portal整合JMeter[3]作為壓力與負載測試的自動測試工具。JMeter藉由模擬使用者透過瀏覽器送出及接受的HTTP Request來模擬使用者操作瀏覽器的行為。由於省略了瀏覽器，所以相較於人工操作瀏覽器進行測試，JMeter能夠以較為節省運算資源的方式模擬大量的同時上線使用者。在使用上，開發人員可以透過JMeter提供的GUI介面，在測試腳本中輸入欲測試的網頁頁面，GET及POST方法中所需使用到的變數值等，以及欲模擬的使用者數量、測試循環次數等參數，即完成一個測試腳本。進行自動測試時，JMeter會以測試腳本中記錄的目標網址、Request中所需一併送出的變數及變數值等相關資訊，包裝成HTTP Request送出至待測系統，同時記錄待測系統對於該Request的反應結果、反應時間、延遲時間、回傳資料大小等數據，最後產生記錄檔供開發團隊觀看每一個HTTP Request的測試結果。

JMeter在進行壓力及負載測試時，可以選擇使用單機模式或分散模式(Distributed Mode)。使用分散模式時，將測試案例腳本檔輸入進JMeter的Master節點，同時給定Slave所在的IP位置。如圖 1，開始測試後，JMeter Master會將腳本檔統一分配與各個JMeter Slave節點，並在所有的節點上同時執行。各Slave會回傳測試結果至Master，Master會統一將所有測試結果整合成一份測試結果，並產生結果記錄檔。

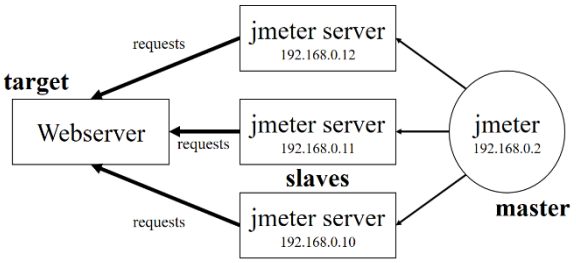

圖 1 JMeter Distributed Mode 示意圖[4]

2-2 Testing as a Service

Gao et al.[5]指出，雲端測試即服務的好處在於測試資源的共用、測試環境的擴充性、隨叫即用的自動測試服務、隨測隨付(Pay as you Testing)的計費方案、多租戶各自獨立的測試服務，及第三方測試品質認證等。

Yen et al.[6]提出並實做了一個提供網頁負載測試的雲端測試即服務平台，以網頁應用程式的方式提供測試人員方便操作的測試環境，並且將測試節點佈署在不同的地理位置，以模擬現實世界中的網路狀況，讓測試人員能夠觀察因不同地方網路狀況不盡相同而產生的差異。

Lee et al.[7]整合了三種測試工具，提供負載測試工具服務。透過讓使用者透過 FTP 上傳測試腳本至服務平台的方式，簡化使用者自行設定及操作負載測試工具的步驟。該系統在測試執行期間，會對受測系統進行系統足跡監控，藉以紀錄受測系統在接受測試時的 CPU、記憶體使用率，磁碟 I/O 及網路傳輸量等，供測試者了解系統的各項瓶頸所在。

Yu et al.[8]列舉了 TaaS 系統所會面對的幾項問題，包括如何依照各種測試種類準備相對應的測試環境，如何排成讓測試能夠在時限內完成，監控所有可用測試資源及各測試的執行狀態，即時管理包括建立、管理、搬移測試程序等。作者針對以上議題，設計並實做了一個雲端測試即服務平台，將

Xen 虛擬機使用自製的雲端管理器及叢集管理器，達到測試工具的虛擬化及自動佈署。

三、 Testing-as-a-Service Web Portal

圖 2 為目前 TaaS Web Portal 的架構圖，包含開發人員進行測試時使用的 TaaS Web Portal 伺服器，以及 TaaS Web Portal 背後實際執行自動測試工具的 JMeter 及 Selenium 的遠端測試伺服器。

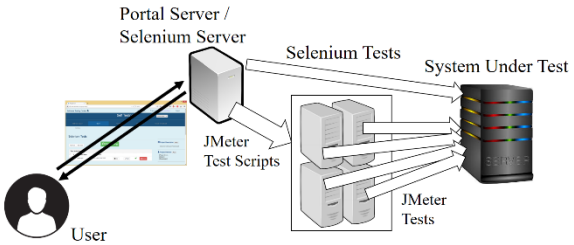

圖 2 TaaS Web Portal 架構圖

開發人員在使用此雲端測試平台時，所有的操作都是透過 TaaS Web Portal 網頁介面進行。使用者以帳號密碼登入 TaaS Web Portal 後，會進入專案管理頁面，其中表列所有此使用者有權限存取的專案，如圖 3。

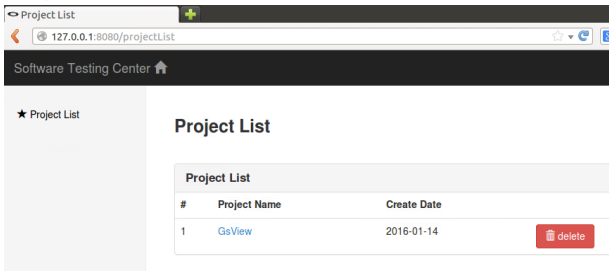

圖 3 TaaS Web Portal 列出使用者具存取權限的專案列表

每一個專案會有自己獨立的測試案例集，不同專案的測試案例不會共用，讓不同的開發團隊能夠有各自獨立的測試專案。使用者選擇欲操作的專案後，可以進到該專案的頁面中，此時可以選擇測試案例列表、測試計畫列表、測試執行結果列表等項目進行查看。在測試案例列表中，使用者可以上傳測試腳本，例如 Selenium 的 HTML 腳本檔，並儲存該測試案例於 TaaS Web Portal 中。上傳腳本後，使用者可以透過如圖 4 的 TaaS Web Portal 介面調整腳本內的相關參數，例如 Selenium 腳本中記錄的受測系統網址，JMeter 腳本中記錄的受測系統網址、欲模擬之同時上線使用者(Concurrent Users)數量、同時上線使用者數量爬升時間(Ramp Up Period)、執行測試次數(Iteration)、測試執行時間(Duration)等。

在建立完測試案例後，使用者可以選擇複數個測試案例建立為一個測試計畫(Test Plan)，並以測試計畫為單位執行自動測試。當測試計畫被使用者手動選定執行，或由持續整合工具觸發自動執行後，TaaS Web Portal 會利用測試工具所包含的遠端執行功能，依序將測試計畫中的測試案例藉由在

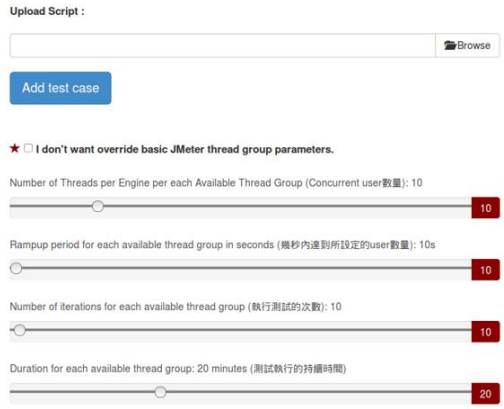

圖 4 在 TaaS Web Portal 網頁介面中調整 JMeter 腳本的相關參數

TaaS Web Portal 伺服器上執行中的測試工具 Master 端向預先設定好的遠端測試工具 Slave 端發送測試要求，待遠端的 Slave 端測試結束將結果回傳給 Master 端後，Master 端會把測試結果以檔案的方式存在 TaaS Web Portal 伺服器中。

測試計畫執行結束後，使用者可於專案頁面中選擇測試執行結果頁面查看各次的測試結果，如圖 5 所示。點選進入各次執行結果的連結後，可以看到該次測試計畫中的測試案例列表，並可以點選觀看各測試案例在該次自動測試的結果。Selenium 測試案例的測試結果會以表格方式，如圖 6 所示，在自動測試過程中有通過的執行步驟會以綠色顯示，若該步驟未通過則會以紅色顯示。JMeter 的測試結果分為兩個部分，圖表式測試結果及純文字測試記錄。圖表式測試結果如圖 7 所示，使用者可於網頁中選取如延遲時間、同時上線人數、錯誤數等項目，即可在 TaaS Web Portal 的網頁介面中看到以每分鐘為單位的測試結果趨勢折線圖及測試結果表格。若是測試過程中因測試腳本有錯誤而產生錯誤的測試結果，使用者仍能在 TaaS Web Portal 上查看 JMeter 的原始純文字測試結果及測試記錄檔，以確認錯誤所在。

四、 系統實作

為了增加雲端測試平台的擴充性，將測試工具的 Master 端與 TaaS Web Portal 伺服器分離，及達成測試工具 Slave 端相關設定的自動化，我們選擇將測試工具整合進 OpenStack 中。同時我們對 TaaS Web Portal 作了一些架構改進，使其能夠更輕易的新增可供使用者使用的測試工具。圖 8 為本篇研究實作的 TaaS Web Portal 架構圖。

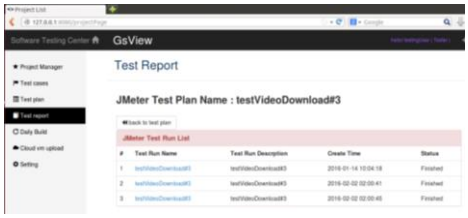

圖 5 測試計畫的自動執行結果列表

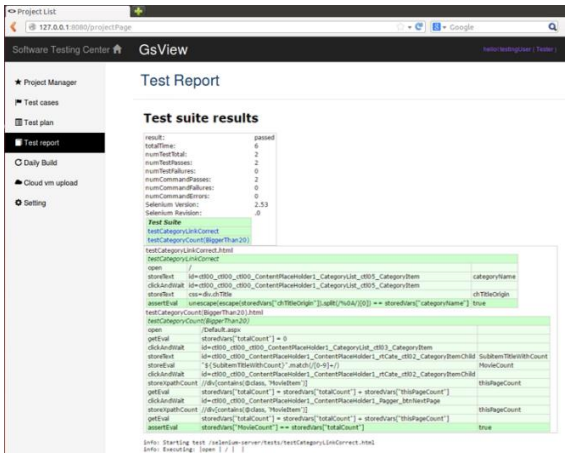

圖 6 在 TaaS Web Portal 中檢視 Selenium 的自動測試結果

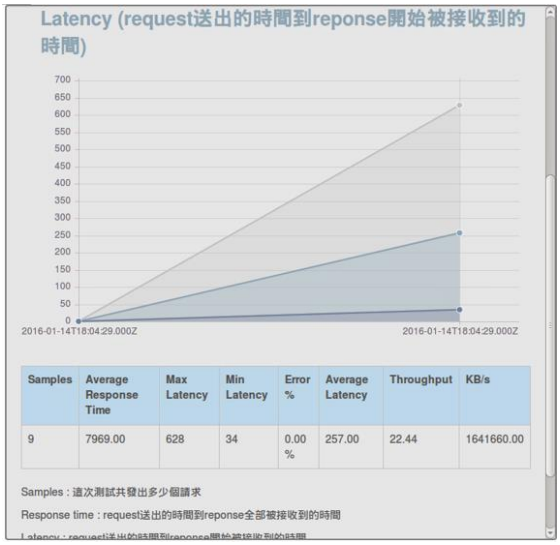

圖 7 在 TaaS Web Portal 上檢視 JMeter 實驗結果中的最大、最小、平均延遲時間(Latency)

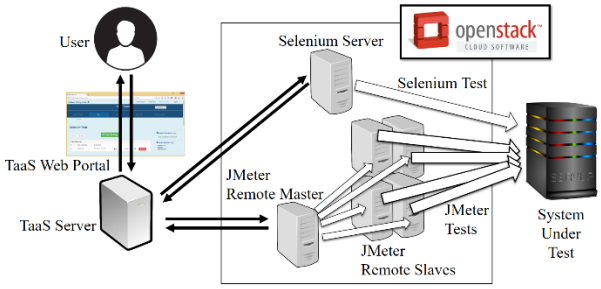

圖 8 本篇研究提出之 TaaS Web Portal 架構圖

4-1 OpenStack

為了讓測試工具能夠被自動佈署，我們選擇使用 OpenStack 作為 IaaS 平台。每一個測試工具會先由 TaaS 系統管理者包裝成一個虛擬機映像檔，這些映像檔會儲存在 OpenStack 中供 TaaS 選用。當測試人員從 TaaS 啟動測試時，TaaS 會連上 OpenStack 並選擇對應工具的映像檔啟動為雲實例(Cloud Instance)。啟動時 TaaS 會一併將雲實例啟動後要執行的指令送入 OpenStack，讓雲實例在開機完成後能夠自動開始執行已事先安裝好於映像檔中的測試工具。

如圖 9 所示，Selenium 等只需單機即可執行的測試工具，會由 TaaS 要求 OpenStack 執行對應雲實例並開啟測試。測試結束後，雲實例會將測試結果傳回 TaaS 的 API 上，待 TaaS 收到測試結果檔案後，TaaS 會要求 OpenStack 將該測試所使用的雲實例刪除掉，以清理測試所使用的 OpenStack 運算資源。

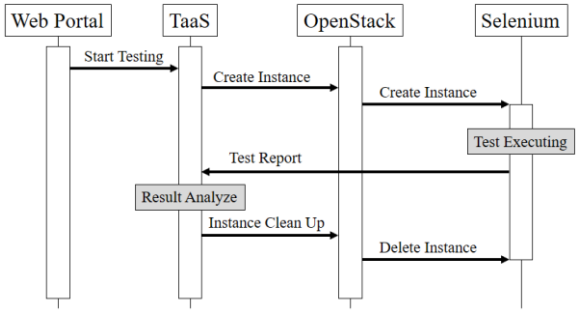

圖 9 Selenium 測試執行流程

針對 JMeter 等有分散模式的測試工具，TaaS 會先計算該次測試需要多少個 Slave 節點，要求 OpenStack 開啟足夠數量的 Slave 雲實例，並記錄 OpenStack 回傳的雲實例 IP 位置。接著，TaaS 會要求 OpenStack 開啟一個 Master 節點，並傳入測試腳本及前一步驟所儲存的 Slave 節點 IP 清單。待 Master 節點開啟後，會執行 TaaS 要求其開機後執行的指令，開始與 Slave 節點連線並開始測試。JMeter Master 會回收所有 JMeter Slave 的測試資料後統整成測試結果，之後會將測試結果透過 TaaS 的 API 上傳回 TaaS。在 TaaS 收到測試結果後，會要求 OpenStack 將測試所用的雲實例刪除掉。上述流程如圖 10。

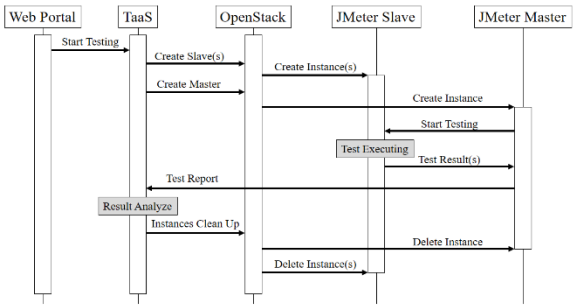

圖 10 JMeter 測試執行流程

4-2 TaaS Web Portal

為了使 TaaS 便於擴充新的測試工具，我們將其做插件化的設計。如圖 11，JMeter 與 Selenium 各為一個模組，其中包含各自的工具執行模組(Tool Executor)及結果分析模組(Result Analyzer)。當 TaaS 開始執行測試案例時，TaaS 會依據該測試案例使用哪一個測試工具，而選擇對應工具的工具執行模組進行測試。

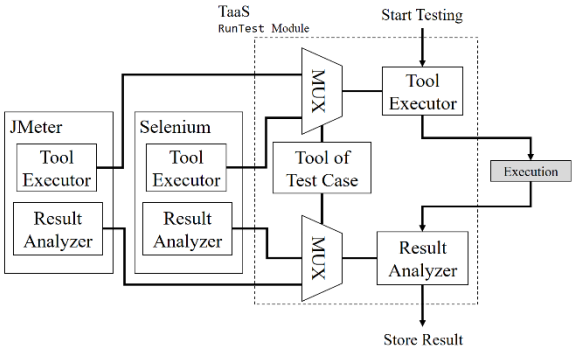

圖 11 TaaS 中，針對測試工具的插件化設計

測試結束後，TaaS 會將測試結果送入對應工具的 Result Analyzer 進行結果分析。測試人員可以在撰寫 Result Analyzer 時，額外加入自行定義的結果分析運算，例如透過分析 JMeter 的 JTL 結果檔案，算出每一分鐘區間內的最大延遲(Latency)、總吞吐量(Throughput)等，再回傳運算後的結果。TaaS 允許測試人員自行撰寫各工具的 Result Analyzer，只要 Result Analyzer 回傳的格式符合 TaaS 所定義的格式。

圖 12 為 Result Analyzer 回傳格式的範例，回傳值為一個 Array 物件，此物件最後將被轉為字串存入資料庫中。Array 物件內包含許多 JSON Object，每一個 JSON Object 內有 type、name、data 三種資料，分別表示該筆資料的資料類型、資料標題、資料內容。資料類型目前支援三種，HTML 類型會將資料內容中給定路徑的 HTML 網頁檔直接以內嵌框架的方式顯示於 Web Portal 上，Selenium 就是採用這樣的方式讓測試人員能夠直接觀看其產生出來的 HTML 報表。CHART 類型會將資料內容中的 Timestamp 陣列當作 X 軸，Value 當作 Y 軸，在網頁上繪製折線圖，讓觀看測試報告的人能夠以較直覺的方式觀察測試過程中各項數據的走勢。PLAINTEXT 類型則是將資料內容中給定路徑的檔案，例如 JMeter 測試過程中所產生出的記錄檔，以純文字的形式顯示於網頁中。JMeter 測試結果中，各分鐘的最大、最小、平均延遲等數據

```
[
  {
    "type": "chart",
    "name": "Throughput",
    "data": {
      "timestamp": [ 17, 18, 19 ],
      "value": [ 200, 300, 400 ]
    }
  },
  {
    "type": "chart",
    "name": "Latency",
    "data": {
      "timestamp": [ 17, 18, 19 ],
      "value": [ 0, 0, 30 ]
    }
  },
  {
    "type": "plaintext",
    "name": "Test Result",
    "data": {
      "filepath": "uploads/9_48.jtl"
    }
  },
  {
    "type": "plaintext",
    "name": "Test Log",
    "data": {
      "filepath": "uploads/9_48.log"
    }
  }
]
```

圖 12 Result Analyzer 回傳格式範例

五、 案例研究

TaaS Web Portal 是本實驗室與工業技術研究院合作的研究計畫，目的在為工研院內部建立一套測試即服務私有雲平台，使未來工研院內部各開發團隊皆能使用此系統對其內部開發中的軟體進行測試。在合作初期，工研院挑選了集視[9]，一套由工研院開發的線上影音串流網站作為受測系統，希望能對其進行功能性測試、負載測試及壓力測試。在此案例研究中，我們將該系統作為目標的受測系統，測試與 OpenStack 整合後的 TaaS Web Portal 是否能夠對其進行自動測試。

功能性測試的部分，我們首先在個人電腦上透過 Selenium IDE 製作了四個 Selenium 測試案例腳本，包括檢查由首頁點入影片類別後載入的影片列表是否正確、影片類別中的影片數量是否計算正確、是否有產生正確數量的影片頁數、點入影片後是否載入正確的影片等。在個人電腦上建立完並確認測試腳本正常無誤後，將其上傳至 TaaS Web Portal 建立測試計畫並執行自動測試。測試結果與在個人電腦上的執行結果相同，測試計畫中的所有測試案例均能在 TaaS Web Portal 上通過自動測試，顯示 TaaS Web Portal 能夠正常的執行 Selenium 對網頁進行功能性自動測試。

非功能性測試的部分，我們以 JMeter 製作了兩個腳本，模擬大量使用者載入網站首頁，以及模擬大量使用者同時觀看影片。我們首先在個人電腦上製作模擬大量使用者載入網站首頁的 JMeter 腳本。由於個人電腦運算能力受限，我們將同時上線使用者數量調整為 10，並以此參數測試 JMeter 腳本運行無誤。而後將腳本上傳至 TaaS Web Portal，並在上傳後透過網頁介面將腳本的同時上線使用者數量調整至 200。

對於模擬大量使用者同時觀看影片的測試案例，我們需先定義如何量化品質的好壞。以一個使用者觀看長度為五分鐘的影片為例，若使用者在沒有手動將影片暫停的情況下，花費剛好五分鐘觀看該影片，我們認為影音串流服務確實有即時的將使用者欲觀看的串流影片撥放完畢，我們便認定這樣的服務是正常的。反之，若使用者觀看五分鐘影片所花費的時間為六分鐘，表示除了影片長度的五分鐘外，使用者額外花了一分鐘在等待緩衝器從遠端伺服器讀取影片串流資料到本機，這樣的結果對於使用者來說是破壞其使用者體驗的，我們認定這樣的服務是有錯誤的。我們依照此定義製作 JMeter 腳本，當模擬大量使用者觀看影片時，會記錄有多少使用者體驗到的服務是錯誤的，藉此觀察系統在多少使用者同時觀看時會使集視系統滿載。

然而在製作測試腳本時，我們遇到了一個問題。JMeter 自動測試工具為了減少模擬使用者所需的運算資源，以發送及接收 HTTP Request 的方式模擬使用者透過瀏覽器瀏覽網站的行為。這樣的模擬方式對靜態網站而言是沒有問題的，但對於許多使用較先進網頁技術的動態網站來說，特別是如串流影音網站使用 Ajax 非同步載入技術的網站，這樣模擬 HTTP Request 單純載入網站沒有辦法正確的對其影音功能做負載測試。同時，以 HTTP Request 直接存取影音在伺服器上的檔案位置也無法正確重現網頁上的影音撥放器為了減緩影片載入對伺服器造成的負擔，不選擇直接載入影片而以分段串流載入的特性。

對於這類的動態網站，我們需要讓 JMeter 實際開啟瀏覽器進行測試。由於 JMeter WebDriver Plugin Set[10]可以讓 JMeter 實際開啟瀏覽器進行測試，對於模擬使用者觀看影片的測試案例，我們選擇 JMeter 配合 Web Driver Plugin 進行測試腳本製作。我們的待測系統所使用的影片撥放器是 JWPlayer[11]，透過 JWPlayer 提供的 Javascript API，我們可以獲取影片撥放器中，緩衝器已從伺服器載入多少百分比的影片。在撰寫 JMeter 腳本時，我們以緩衝器完全載入影片所花費的時間與影片長度做比較，若載入影片的時間比影片本身還長，表示在影片撥放的時候一定會因為尚未載入影片段落而暫停撥放，此時我們便會記錄該模擬使用者為失敗的測試。

在 JMeter 的文件中[12]，WebDriver Plugin 開發團隊建議測試人員在使用此 Plugin 時，以一個網頁使用一個計算核心的方式，計算測試案例所需使用的計算資源數量。由此可以看出，雖然 WebDriver Plugin 可以解決前段所述動態網頁及 Ajax 技術的問題，執行測試所需使用的運算資源也較一般的 JMeter 測試腳本多上許多倍。我們先在個人電腦上先以 5 個同時上線使用者進行測試，確認測試腳本能夠正常運行後，上傳測試腳本至 TaaS Web Portal，透過網頁介面調整同時上線使用者為 20 人，並進行自動測試。自動測試的結果顯示使用 TaaS Web Portal 能夠正常執行 JMeter 自動負載測試腳本，並在 TaaS Web Portal 上顯示測試結果。

上述案例研究說明，TaaS Web Portal 與 OpenStack 結合後，確實能夠使用現有的自動測試工具向待測系統進行自動測試，包括使用 Selenium 進行功能測試，及使用 JMeter 對網頁載入進行壓力測試。更進一步，我們可以透過 JMeter 及其 WebDriver Plugin 對影音網站進行影音串流服務的壓力及負載測試，並提供測試結果讓開發團隊了解其系統目前能承受的負載大小，進一步找尋系統瓶頸並改進。

六、 結論與未來展望

本論文為降低開發團隊使用自動測試工具對其程式碼進行軟體測試所需成本，將實驗室過去開發的系統進一步與 OpenStack 整合，實作一個能夠自動佈署測試工具且易於擴充其測試運算資源的 TaaS Web Portal 雲端測試即服務系統。使用此系統進行網頁測試時，開發團隊只需將撰寫完的測試腳本上傳至本服務平台，並於 TaaS Web Portal 所提供的網頁介面中調整同時上線使用者數量等測試相關參數，無需顧慮測試工具的佈署及網路設定等問題，即可進行自動測試。對於 TaaS Web Portal 平台管理者，只需擴增 OpenStack 的運算節點數量，即可快速擴增測試即服務平台的規模，讓所有測試工具均能利用新加入的運算資源執行測試案例。

在案例研究的過程中，我們發現現有的測試工具對於動態網頁的支援並不是非常完善。儘管可以透過執行瀏覽器的方式對動態網頁進行測試，但測試所需的運算資源過於龐大，即使是以易於擴充的雲端測試環境進行測試，欲模擬大量的同時上線使用者，仍然是個問題。對於此類非功能性測試，研究並開發運算資源需求較小的自動測試工具，是值得深入研究的方向。

參考文獻

[1] O. Inc. (1 May). *OpenStack*. Available: https://www.openstack.org/

[2] T. S. Project. (2016, 25 Apr). *Selenium*. Available: http://www.seleniumhq.org/

[3] T. A. S. Foundation. (2016, 25 Apr). *Apache JMeter*. Available: http://jmeter.apache.org/

[4] T. A. S. Foundation. (27 Apr). *JMeter Distributed Testing Step-by-step*. Available: https://jmeter.apache.org/usermanual/jmeter_distributed_testing_step_by_step.pdf

[5] J. Gao, X. Bai, W. T. Tsai, and T. Uehara, "Testing as a Service (TaaS) on Clouds," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, 2013, pp. 212-223.

[6] M. Yan, H. Sun, X. Wang, and X. Liu, "WS-TaaS: A Testing as a Service Platform for Web Service Load Testing," in *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, 2012, pp. 456-463.

[7] S.-J. Lee, Y.-C. Lin, K.-H. Lin, and J. L. You, "Design and Implementation of a Composite Web Load Testing Service," presented at the 2015 Taiwan Conference on Software Engineering, Yunlin, Taiwan, 2015.

[8] L. Yu, W. T. Tsai, X. Chen, L. Liu, Y. Zhao, L. Tang, *et al.*, "Testing as a Service over Cloud," in *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, 2010, pp. 181-188.

[9] I. T. R. Institute. (24. May). *G's VIEW*. Available: http://www.gsview.tv

[10] T. A. S. Foundation. (24. May). *JMeter WebDriver Plugin Set*. Available:

http://jmeter-plugins.org/wiki/WebDriverSet/
[11] L. A. S. Incorporated. (24. May). *JW Player*. Available: https://www.jwplayer.com
[12] T. A. S. Foundation. (24. May). *JMeter Webdriver Tutorial since 1.1.0*. Available: http://jmeter-plugins.org/wiki/WebDriverTutorial/