Android App 伺服器效能測試平台
A Performance Testing Platform for Android App Servers

陳偉凱、劉建宏、陳炳宏、朱宏文
國立臺北科技大學資訊工程系
Woie-Kae Chen, Chien-Hung Liu, Ping-Hung Chen, Hung-Wen Chu,
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Email: {wkchen, cliu, t100599001, t102598041}@ntut.edu.tw

摘　要

隨著雲端服務興起，支援雲端應用的 Android App 也不斷地增加。這類應用通常必須同時服務大量的使用者，營運人員必需知道 Android App 伺服器的負載能力，以作為硬體設備擴充或調校的依據，因此效能測試對於持續開發並擴張規模的 Android App 顯得非常重要。本論文透過雲端良好的延展性、低成本及高運算能力的特性，提出一個 Android App 伺服器的效能測試平台，可以平行執行多個功能性測試專案，並開啟大量的虛擬機器執行 Android 模擬器，來模擬大量的使用者操作 Android App，對 Android App 伺服器產生負載。實驗結果顯示，使用本平台確實可以對 Android App 伺服器產生真實的負載，而且負載量與使用 JMeter 執行測試時相當，然而卻不需額外撰寫 JMeter 專屬的測試腳本。因此當測試人員已經有現成的功能性測試腳本時，即可重複利用該腳本，來減輕測試人員進行壓力測試的負擔。

關鍵字：Android、OpenStack、CTP、效能測試

一、 前言

近年來 Android [1]智慧型手機普及，使得越來越多的 Android App 出現，加上 Wi-Fi 和行動網路的速度越來越快，以及雲端服務的興起，使得支援雲端服務的 Android App 也不斷地增加。因此，提供 Android App 雲端服務的業者，必須對伺服器進行效能測試，作為硬體設備擴充或調校的依據，所以 Android App 伺服器的效能測試越趨重要。

一般而言，執行功能性測試及效能測試時，會使用不同的測試工具。例如對 Android App 進行功能性測試時，通常會使用 Monkey [2]、MonkeyTalk [3] 、Robotium [4] 、UiAutomator [5] 及 Robot Framework [6]等測試工具。然而上述的功能性測試工具，並無法直接對伺服器產生大量的負載，以至於要進行負載測試時，測試人員無法重覆利用現有的功能性測試工具以及測試腳本，而必須另外使用 Apache JMeter [7]、BlazeMeter [8]、NeoLoad [9]或是 LoadRunner [10]等效能測試工具來進行效能測試。有時測試人員對於功能性測試工具已經很熟悉，但是對於效能測試工具卻不夠熟悉，以致測試人員必須重新學習這些效能測試工具，因而增加測試人員的負擔。

為了減少測試人員的負擔，本論文希望重覆利用功能性測試腳本，對 Android App 伺服器產生負載，進而使得測試人員，不須要重新學習效能測試工具及撰寫新的測試腳本，即可進行效能測試。本論文提出一個以 CTP [24]為基礎的 Android App 伺服器效能測試平台，並使用 OpenStack [11]開啟大量的虛擬機器以及 Android 模擬器，模擬大量的使用者操作程序，對 Android App 伺服器產生大量的負載，以達到效能測試的目的。

本論文後續章節組織如下：第二節介紹本論文的相關研究；第三節說明 Android App 伺服器效能測試的方法；第四節敘述本論文所提出 Android App 伺服器效能測試平台的系統架構與設計；第五節為實驗的說明與結果比較；第六節為結論以及研究展望。

二、 相關研究

在軟體測試中，效能測試(Performance testing)一般透過量測或驗證系統執行時的資源使用情況，來了解系統在特定的工作量下執行的反應時間以及穩定性。而負載測試(Load testing)為效能測試的一部分，通常用來了解系統在預期的負載情況下，系統依舊能正常的運行。壓力測試(Stress testing)則是用來了解在超出負荷的情況下，系統是否穩定及功能是否正常，以應付預期之外的大量負載。當測試人員進行效能測試時，通常會使用 Apache JMeter [7]，它是由 Apache 組織所開發的開放原始碼工具，使用 Java 撰寫，用於 client/server software 的負載測試。

整合式雲端測試平台(Cloud Testing Platform，簡稱 CTP)是由臺北科技大學資訊工程系開發的系統[21][24][25][26][27][28][29][30][31][32][33][34]，支援 Android 與 Web 的相關測試，其中 Android 的部分則支援以下測試工具：(1) Monkey 是一個 command-line 工具，它可以在 Android 模擬器或是實體機器上使用，藉由產生隨機事件對 Android 應用程式進行壓力測試；(2) MonkeyTalk 是一套行動裝置的測試工具，支援 Android 以及 IOS 應用程式的測試，它可進行簡單的冒煙測試(smoke test)到複雜的資料驅動測試(data-driven test suites)；(3)

Robot Framework 是使用 Python 撰寫的自動化測試框架，主要用於驗收測試(acceptance testing)和驗收測試驅動開發(acceptance test-driven development, ATDD)，支援 KDT (keyword driven testing)方法；(4) Robotium 是個開放原始碼的 Android 自動化測試框架，支援灰箱測試，對於 Native [12]與 Web 類型的 Android 應用程式有全面性的支援；(5) Espresso 是 Google 開發的 Android 自動化測試框架，提供了一系列的 API 可以讓測試人員建構 Android 應用程式的介面測試流程；(6) UiAutomator 是由 Android 提供的使用者介面測試的測試框架，支援黑箱測試，可以對裝置當前畫面上的元件進行操作，支援 Android 4.1 以上版本。而 Web 則支援 Functional Testing 及 Performance Testing。

本論文以 CTP 為基礎並加以擴充，使得 CTP 具有 Android App 伺服器效能測試的功能。

三、 Android App Server 效能測試方法

3.1 測試方法簡介

在軟體開發的過程中，測試人員通常已經寫好一些功能性的測試腳本，並使用這些腳本進行功能性測試。但是當測試人員想進行 Andoird App 伺服器的負載測試時，通常會使用 JMeter 進行效能測試，使得測試人員必須再學習 JMeter，並另外撰寫效能測試腳本，對測試人員會形成額外的負擔。如果能夠利用功能性的測試腳本直接對 Android 伺服器產生負載，就可以重複的利用這些腳本進行效能測試，測試人員不需要花額外的時間學習和撰寫新的腳本，進而減少開發人員或測試人員的負擔。

圖 1 為 CTP 功能性測試流程圖，在使用 CTP 進行功能測試時，有以下四個主要的工作：1.撰寫功能性測試腳本：測試人員必須先對待測試的應用軟體，撰寫功能性測試腳本，並且要驗證功能性腳本是正確無誤的。2.上傳測試所需要的測試腳本以及待測 APK 至 CTP Server：測試人員先登入到 CTP Server，根據測試類型選擇測試專案，例如：要測試 Android 應用程式時，必須要選擇 Android 的測試專案。並且上傳對應的測試腳本及待測的 APK。3.執行測試：測試人員使用 CTP Server 執行所選擇的專案開始進行功能性測試。4.觀看測試報表：當測試結束時，CTP Server 會產生功能性測試的報表給測試人員觀看。

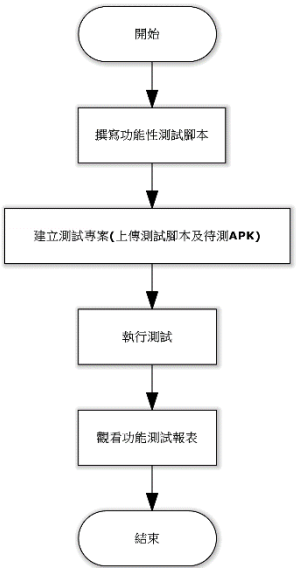

圖 1 CTP 功能性測試流程圖

圖 2 為 Android 應用程式伺服器效能測試的流程圖，當開始執行壓力測試時，分成五個主要的工作，前兩個工作與 CTP 功能性測試的前兩個工作相同，其餘的三個工作則為：1.設定 Composite Project：測試人員依據第二步設定好測試專案後，可以透過一個特殊的專案 Composite Project，將測試專案加入到 Composite Project，並且可以設定測試專案想要產生的建置次數。2.執行壓力測試：本平台會將加入到 Composite Project 的專案進行平行化的測試。3.觀看效能測試報表：當測試完成時，會產生一份效能測試報表給測試人員觀看。

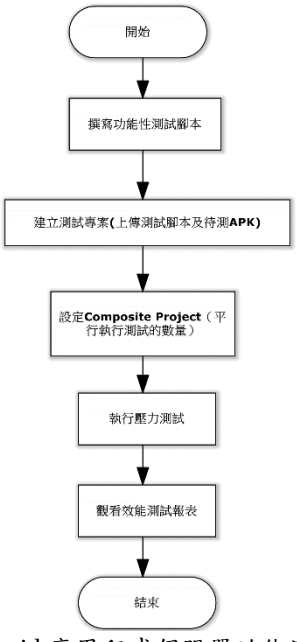

圖 2 Android 應用程式伺服器效能測試流程

本研究是建立在 CTP 上，此平台提供 Android 應用程式的測試，並支援以下 5 種測試工具：Monkey、MonkeyTalk、Robotium、RobotFramework

及 UiAutomator。以上這 5 種 Android 測試工具除了 Monkey 以外其它都屬於功能性的測試工具，所以本系統可以透過功能性的測試工具來產生 Android 伺服器的負載。為了產生大量的負載，如果使用實體手機會因成本考量而有數量的限制，所以本論文使用 Android 模擬器來替代實體手機。同時考量到單一個人電腦能開啟的 Android 模擬器數量有限，因此，本論文採用雲端運算的概念，將 Android 模擬器部署在雲端的虛擬機上。

CTP 支援以下三種測試專案種類：Web Testing、CTS 及 Android Testing，為了能夠重複使用這些設定好測試的專案，並且將同一個專案產生多個實例，就可以產生大量的相同測試一起執行。因此，提出一個 Composite Project 概念，此種類型 project 稱為 Composite Project，它可以將以上提到的三種測試專案複合起來，並可以將專案產生成數個實例進行建置。當使用本方法時，如果壓力測試的情境與功能性測試的情境相同，則測試腳本不需要變動，可以直接重複的使用；但若是兩者的情境不同的話，則可能需要調整原有的功能性測試腳本。

3.2 Composite Project

Composite Project 概念類似 Gang of Four 所提出的 Composite Pattern 如圖 3 所示。在 Composite Pattern 中，可以將 Leaf 加入到 Composite 形成一個群組，再透過統一的介面去操作這一個群組。假設有很多不同的測試專案，希望能透過統一的介面，使得這些測試專案能夠同時執行測試，因此，我們利用 Composite Pattern 的概念，可以將不同的測試專案類型視為 Leaf，然後再增加一個特殊的測試專案 Composite Project，這種專案可以視為 Composite，因為這種專案可以將其他複合起來，則進而透過統一的 Component 介面來操作被加入到 Composite Project 執行測試。為了模擬大量的使用者，還必須將專案產生大量的 instance，假設一個專案只能模擬一個使用者，所以可以將某一個專案產生大量 instance 後，就可產生大量的使用者，再把這些專案加入到 Composite Project 中， 就可以同時模擬大量的使用者進行測試。

圖 4 為根據以上提到的方法，所產生的 Composite Project 類別圖，在 CTP 有 Web Testing、Compatibility Test Suite (簡稱 CTS) [23]及 Android Testing 這三種不同的測試專案，所以可以把這三種測試專案視為 Composite Pattern 的 Leaf，然後 Composite Project 視為 Composite Pattern 的 Composite，根據圖 4 的架構顯示，可以將三種測試類型加入到 Composite Project，再透過統一的介面 Project 來操作這三種測試專案。

我們就使用者以及開發者的角度，進行探討使用 Composite Project 的優缺點：以使用者角度就系統的操作上進行分析，其優點是假如使用者希望將不同類型的測試專案做平行測試，則可以利用 Composite Project 將不同的測試專案複合起來一起執行測試，而其缺點是使用者介面多了 Composite Project，使得使用者必須多學習 Composite Project 的用法。從開發者角度對系統設計上進行分析，優點是很容易的可以建立及操作一個樹狀結構的物件，缺點則是需要花大量的時間，將原本的架構進行重構。

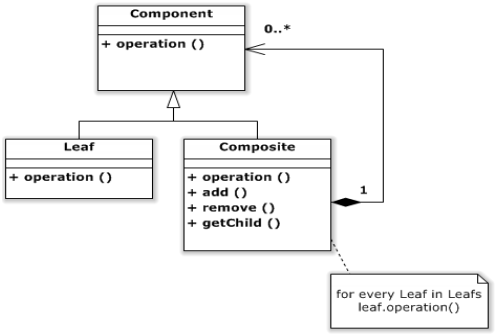

圖 3 Composite Pattern

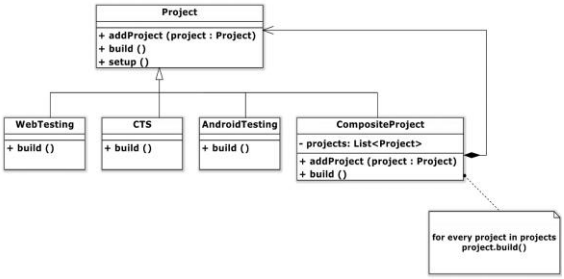

圖 4 Composite Project

四、 系統設計與實作

本論文使用 CTP 做為基礎系統，擴充其系統架構與設計，以支援 Android App 伺服器效能測試。

4.1 系統架構

Android App 伺服器效能測試系統運作流程，如圖 5 所示，分為三個主要的模組：CTP Server、Integration Station 及 Mobile Device。本系統的 CTP Server 模組為整個系統對測試人員的溝通媒介，除了提供友善的使用者介面外，還負責管理整個測試工作流程，包含工作分派以及整合報表的彙整等操作。Integration Station 為 OpenStack 虛擬機器的一支程式，此模組負責執行接收到的測試工作，產生測試報表，並將測試報表回傳給 CTP Server 彙整。Android Emulator 則負責執行 Android 應用程式的測試。

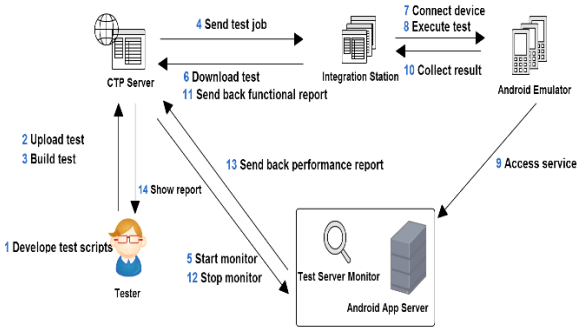

圖 5 Android App 伺服器效能測試系統運作流程

以下說明由這些模組所構成的測試系統，其執行測試的流程：

1. 由 Tester 或其他開發人員撰寫測試腳本。
2. Tester 驗證腳本正確後，將測試所需要的資料或檔案上傳至 CTP Server，例如：測試腳本及待測試的 APK。
3. Tester 透過 CTP Server 執行測試。
4. CTP Server 將測試工作傳送給 Integration Station。
5. CTP Server 啟動 Test Server Monitor
6. Integration Station 收到測試工作後，向 CTP Server 下載測試檔案並執行測試前的準備工作。
7. Integration Station 根據 CTP Server 所給予的測試資訊連接 Android Emulator，安裝測試用的 APK。
8. Integration Station 開始執行 Android App 的測試。
9. Android Emulator 開始對 Android App 伺服器執行測試。
10. Android Emulator 測試完成後，Integration Station 向 Android Emulator 取得測試結果。
11. Integration Station 將測試結果回傳給 CTP Server。
12. CTP Server 關閉 Test Server Monitor。
13. Test Server Monitor 回傳效能測試報表。
14. CTP Server 顯示測試結果給使用者觀看。

4.2 Composite Project 實作

圖 6 為 Composite Project 類別圖，本論文實作的部分使用粗框標示。原本的架構沒有 WebTesting、CTS 及 AndroidTesting 這三種類別，這三種測試的建置流程都寫在同一個 BuildAction 類別，但是這個物件本身是作為 Controller 使用並不適合作為一個專案進行建置的類別。於是本論文對原架構進行重構，把測試類型分成 WebTesting、CTS 及 AndroidTesting 這三種類別，並且新增一個 Composite Project 類別，它可以包含多個 Project。以上提到的類別都繼承 Project，將共同的 setup 及 build 動作寫在 Project 中，當不同的測試有不同 setup 時，利用 template method 來解決此問題。當未來要加入新的測試專案時，只要繼承 Project 並實作該介面即可。

ProjectInfoCenter 負責管理執行測試需要的測試資訊，以及請求 Project 執行測試的類別。當壓力測試被要求時，ProjectInfoCenter 會先產生多個 ProjectInfo 的 instance，根據 ProjectInfo 的 projectType 產生對應的 concrete class，例如圖 6 中的 WebTesting、CTS 及 AndroidTesting，再把這些 concrete class 加入到 CompositeProject，透過統一的介面 Project 呼叫 build 方法，然後被加入到 CompositeProject 的 Project 就會透過 BuildFacade 產生 Integrator 開始平行的執行測試。在同一時間 CompositeProject 使用 CompositeIntegrator 通知 Test Server Monitor 開始監控 Android 應用程式伺服器，之後 CompositeIntegrator 等待所有加入到 CompositeProject 的測試結束。當測試結束時 CompositeIntegrator 會停止 Test Server Monitor 對 Android App 伺服器的監控，並下載效能測試報表，最後將報表呈現給使用者觀看。

4.3 Test Server Monitor 實作

本論文的 Test Server Monitor [19]是使用張耀庭所實作出來的待測伺服器監控程式。透過此監控程式記錄 Android 伺服器在提供服務時的系統效能，並且將記錄結果回傳至 CTP Server。圖 7 及圖 8 為監控程式記錄系統效能的模組類別，圖 7 為 CTP Server 負責通知 Test Server Monitor 開始與結束記錄的 CompositeIntegrator；圖 8 則為負責記錄及彙整效能報表的模組。在此模組中，主要實作一個 SystemMonitor 的類別，用來達成監控伺服器系統效能的流程。在監控流程中主要有三個方法，分別為 startCollect、stopCollect 與 converReport。開始執行測試時，TestServerConnection 類別收到由 MonitorController 所傳送的訊息後開始記錄系統效能；當測試執行結束時，TestServerConnection 收到執行結束之訊息，即停止記錄並將蒐集的系統效能數據轉換為圖檔，經壓縮完畢後，通知 MonitorDownloader，並由 MonitorDownloader 從測試伺服器將效能資訊回傳至 CTP Server。

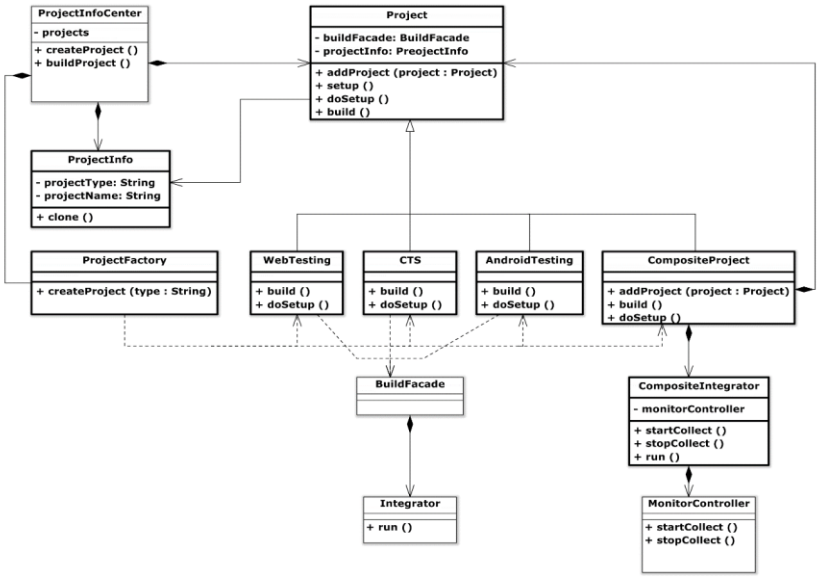

圖 6 Composite Project 類別圖

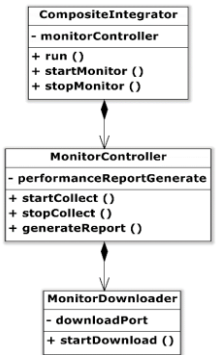

圖 7 Composite Integrator 模組類別

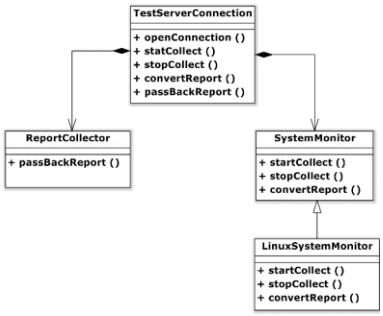

圖 8 Test Server Monitor 模組類別圖

五、 實驗

本節說明使用 Android App 伺服器效能測試系統與 JMeter，分別對待測伺服器產生負載，並將實驗結果做比較。首先簡述實驗目的，接著介紹實驗環境，包括待測伺服器及 Client 規格、JMeter 的版本與 Client 規格，然後是測試案例的說明，最後為實驗之結果比較。

5.1 實驗目的

本論文之實驗目的在於驗證實作出來的 Android App 伺服器效能測試系統，能夠確實對待測 Android App 伺服器產生負載，達到效能測試的目的。因此利用 Android App 伺服器效能測試平台及 JMeter，分別對 FTP Server 產生負載，再根據量測的結果數據進行比較其結果是否相似。本論文選擇 FTP Server 作為測試對象的原因，是因為 FTP 是一個很普及的網路服務，網路頻寬通常是網路服務的瓶頸，因此對 FTP Server 產生足夠的負載後，就很容易可以量測到網路頻寬滿載的情況。另外，容易取得現成的 FTP Client 和 Server，則有利於實驗的進行。

5.2 實驗環境

本實驗所使用的 CTP Server 硬體規格，如表 1 所示，CTP Server 版本為 2.2.18。虛擬機器使用 OpenStack 管理，表 2 為 OpenStack 主機的硬體規格，使用兩台相同規格的主機組成一個 OpenStack。

表 1 CTP Server 硬體規格

CPU	Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz
RAM	8GB
OS	Windows 7 Enterprise 64bit
DISK	SSD 256GB

表 2 OpenStack 硬體規格

CPU	Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz x2
RAM	66GB
OS	Ubuntu 14.04 LTS
Disk	200GB

本實驗所使用的 Android App 為 FtpCafe FTP Client [22]，提供簡單的使用者介面及 FTP、FTPS 及 SFTP 三種傳送檔案的方法，在 Google Play 有免費版及付費版本可以下載。模擬器使用 Google

官方的 Android Emulator，硬體規格如表 3，Android 版本為 4.2.2 Jelly Bean，並部署在 OpenStack 虛擬機器上。

表 3 Android Emulator 硬體規格

CPU	ARMv7
RAM	2GB
OS	Android 4.2.2 Jelly Bean
Internal Storage	200MB
SD Card	300MB

本實驗所使用的待測伺服器硬體規格，如表 4 所示， FTP Server 為 vsftpd，是一個 GNU General Public License [20]的應用程式，可在 Unix-like 系統上使用，包含 Linux 系統。當實驗進行時，所在的網路環境，如表 5 所示。

表 4 FTP Server 硬體規格

CPU	Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
RAM	4GB
OS	Ubuntu 14.04 LTS
DISK	SSD 240GB

表 5 網路環境

Subnet	Bandwidth
140.124.X.X	100Mbps

本實驗的比較對像為 JMeter2.12 版，JMeter Client 的硬體規格，如表 6 所示。

表 6 JMeter Client 硬體規格

CPU	Intel(R) Core(TM) i5-3470 CPU @ 3.20 GHz
RAM	8GB
OS	Windows 7 Enterprise 64bit
DISK	SSD 256GB

5.3 測試案例

表 7 為使用 Android 模擬器進行壓力測試的測試案例，表 8 為使用 JMeter 進行壓力測試的測試案例。這兩個測試案例基本上是相同的，但是由於測試工具不同所以略做調整，使得測試情境更為接近，以下說明測試案例的設計方式。為了要對 Android App 伺服器產生負載，我們設計了至 FTP Server 下載檔案的測試案例。在此測試案例中，當二十位使用者同時下載檔案時，因為實驗環境的網路速度為 100Mbps，下載檔案大小為 128MB，對 Android App 伺服器量測負載時，可以量測到網路頻寬滿載的情況。兩個測試案例不同之處在於使用 Android 模擬器進行 FTP 下載檔案測試，必須透過一系列的使用者操作程序，才能開始進行下載檔案，最後再進行刪除檔案的動作。使用 JMeter 則只需直接發送 requet 的方法，即可開始下載檔案，不需要經過一系列的使用者操作程序，所以在設計 JMeter 的測試腳本時　加入延遲時間，在下載檔案前增加五分鐘的等待時間當作下載檔案的前置動作，最後再加入五分鐘的等待時間，以觀察伺服器在下載後一段時間的效能情形。

表 7 Android 應用程式的測試案例

Test Case:登入 FTP 下載檔案	
Description:以 20 位使用者，登入 FTP Server 下載 128MB 大小的檔案	
Data Requirement: ${帳號} : cloud ${密碼} : cloudtesting ${檔案}: test_file	
Step number	Step Description
1	使用${帳號}及${密碼}登入 FTP Server
2	選擇 download 資料夾
3	選擇${檔案}
4	下載檔案
5	刪除檔案

表 8 JMeter 的測試案例

Test Case:登入 FTP 下載檔案	
Description:以 20 位使用者，登入 FTP Server 下載 128MB 大小的檔案	
Data Requirement: ${帳號} : cloud ${密碼} : cloudtesting ${檔案}: test_file	
Step number	Step Description
1	等待 5 分鐘
2	使用${帳號}及${密碼}登入 FTP Server
3	下載${檔案}
4	等待 5 分鐘

5.4 實驗結果

本實驗依上述的規格及測試案例，使用二十台 Android 模擬器模擬二十位使用者對 FTP 伺服器下載檔案。使用本論文實作的平台對待測伺服器執行壓力測試時，在執行測試的期間，由 Test Server Monitor 監控 FTP 伺服器的負載情形。此外亦利用 Jmeter 的 performance monitor 量測相關的數據，根據實驗結果顯示，兩者所量測的結果幾乎相同。圖 9 至圖 14 為分別使用本平台(黑色的線所表示)以及 JMeter (灰色的線表示)兩種不同的方法，對 FTP 伺服器執行壓力測試時，所量測到待測伺服器的 CPU/Memory 使用率、Disk Read/Write 以及 Network Receive/Transmit 的效能。其結果顯示，當二十位使用者開始下載檔案時，兩者所產生的負載基本上是相似的。

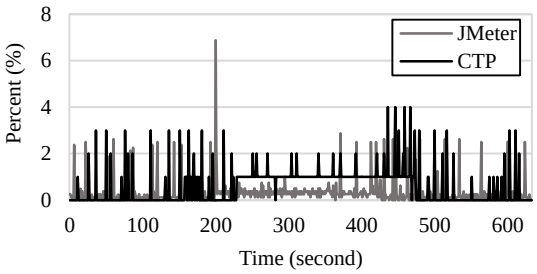

圖 9 CPU 使用率

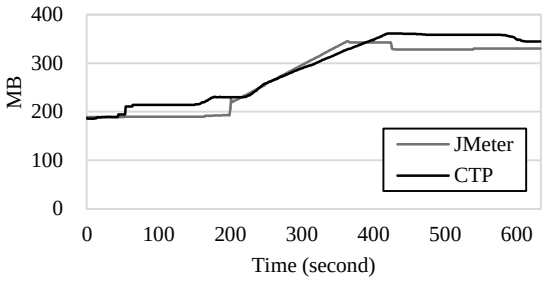

圖 10 Memory 使用率

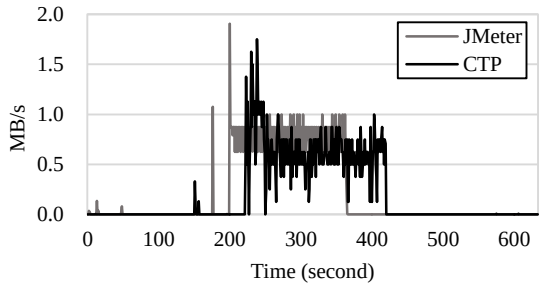

圖 11 Disk Read

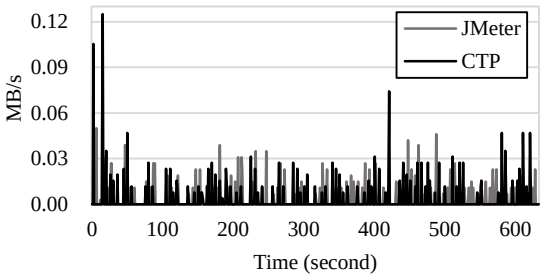

圖 12 Disk Write

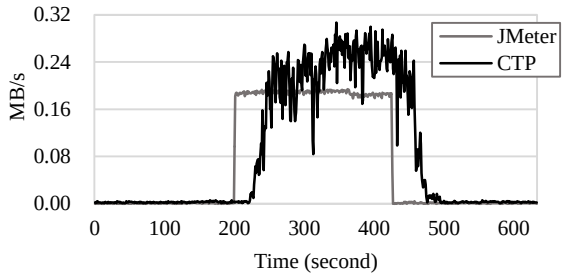

圖 13 Network Receive

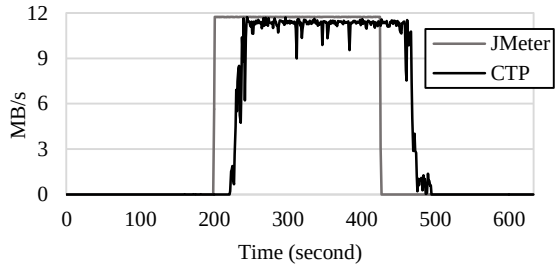

圖 14 Network Transmit

六、 結論與未來研究方向

本論文實作一個 Android App 伺服器效能測試平台，利用 OpenStack 開啟大量的虛擬機器執行 Android 模擬器，來模擬大量的使用者對 Android App 伺服器產生負載。並且提出一個 Composite Project 概念，可以將功能性測試專案平行的執行測試，進而對伺服器產生大量的負載。最後測試人員可以直接使用功能性的測試工具，以及重複利用現有的功能性測試腳本，來進行效能測試，進而減少測試人員的負擔。未來研究有以下方向：(1) 改善 Android 模擬器的穩定度：在執行效能測試時，Android 模擬器會自動重開機，導致該次的測試失敗，希望之後能夠提高 Android 模擬器的穩定度，讓效能測試能順利進行。(2)同步機制：本平台使用 RabbitMQ 來分配工作，它採用 First-in-First-out 的機制，因此，Integration Station 收到工作的時間不同，再加上 Android 模擬器在模擬使用者操作時會有延遲時間，所以每個 Client 開始測試的時間會略有差異，未來希望增加同步機制，使得不同測試情境亦能同時產生測試負載。(3)自動開啟 Android 模擬器機制：本平台的 Android 模擬器是預先開啟在 OpenStack 的虛擬機器，當需要使用大量的虛擬機執行效能測試時，會花費大量的時間開啟 Android 模擬器。希望未來能支援自動開啟 Android 模擬器機制，進而減少測試人員設置測試環境的時間。(4)支援不同版本 Android OS：本平台目前只支援 Android 4.2.2 版本，因為 Android OS 有許多不同的版本，不同版本 Android OS 的 App 在實作上會略有不同，可能對 Server 效能的影響會不太一樣，所以本平台必須要能夠支援不同版本的 Android OS。

致謝

本研究由科技部計畫 MOST 104-2221-E-027-008- 所補助，特此感謝。

參考文獻

[1] Android, Available, http://www.android.com/
[2] Monkey, Available, http://developer.android.com/tools/help/monkey.html
[3] MonkeyTalk, Available, https://www.cloudmonkeymobile.com/monkeytalk
[4] Robotium, Available,

https://code.google.com/p/robotium/
[5] UiAutomator, Available, http://www.android.com/
[6] Robot Framework, Available, http://robotframework.org/
[7] ApacheJMeter, Available, http://jmeter.apache.org/
[8] BlazeMeter, Available, http://community.blazemeter.com/
[9] NeoLoad, Available, http://www.neotys.com/product/overview-neoload.html
[10] LoadRunner, Available, http://www8.hp.com/us/en/home.html
[11] OpenStack, Available, https://www.openstack.org/
[12] Native app, Available, https://en.wikipedia.org/wiki/Native_(computing)
[13] Web app, Available, http://developer.android.com/guide/webapps/index.html
[14] System Testing, Available, https://en.wikipedia.org/wiki/System_testing
[15] Espresso, Available, https://developer.android.com/training/testing/ui-testing/espresso-testing.html
[16] The "Gang of Four": Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Design Patterns, USA: Addison-Wesley, 1994, pp. 163
[17] Collectl, Available, http://collectl.sourceforge.net/
[18] Gnuplot, Available, http://www.gnuplot.info/
[19] 張耀庭，一個以雲端運萬進行系統效能測試之系統，碩士論文，國立臺北科技大學資訊工程所，臺北，2012。
[20] GNU General Public License, Available, http://www.gnu.org/
[21] 李玠鎔，Web 應用程式之雲端效能測試系統，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。
[22] FtpCafe FTP Client, Available, https://play.google.com/store/apps/details?id=com.ftpcafe.trial&hl=zh_TW
[23] Compatibility Test Suite, Available, https://source.android.com/compatibility/cts/index.html
[24] 李友文，雲端測試服務平台 Android 測試過程錄影服務設計與實作，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[25] 賴勇安，一個以移除未使用的程式碼改善軟體維護性的方法：以 STF-CTP 為例，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[26] 康芷瑜，以雲端測試服務進行之 Web 應用程式效能測試實驗，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。
[27] 陳張正，一個使雲端平台支援多版本軟體服務的方法：以 STF-CTP 為例，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[28] 劉淯鈞，Android 雲端測試平台的裝置連線品質改善方法，碩士論文，國立臺北科技大學資訊工程所，臺北，2014。
[29] 張雄展，圖形使用者介面多選一與隨意順序之測試方法，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。
[30] 柯杏洳，基於雲端測試改善 Android 相容性測試效率方法之研究，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。
[31] 呂浩沅，改善雲端測試成本效益之研究，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。
[32] 林憲良，雲端測試策略效能評估之研究，碩士論文，國立臺北科技大學資訊工程所，臺北，2012。
[33] 李俊毅，Android 雲端測試平台的安全性防護方法，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。
[34] 洪陳佐，一個確保 Android 雲端測試裝置即用性的方法，碩士論文，國立臺北科技大學資訊工程所，臺北，2013。