

專用伺服器遊戲解決方案

目錄

使用情境	2
解決方案概論	2
線上遊戲解決方案的重點	3
實作細節	5
重點導覽圖解和實作細節	6
範例應用程式	9
結論	10

隨著玩家人數不斷增加，良好的遊戲體驗背後需要有強大的電腦運算支援。這份文件將提供兼具穩固和高擴展性的遊戲解決方案，說明如何透過 Google App Engine 和 Google Compute Engine 提供玩家即時互動體驗。簡單來說，以遊戲的核心元素而言，例如遊戲配對或建立角色，都需要大量的使用 App Engine，而 Compute Engine 則會用於運行專用遊戲伺服器 and 常見的遊戲引擎。

解決方案涵蓋的主要重點如下：

- 透過自動擴展服務數百至數百萬位的玩家
- 使用 Google Cloud Platform 打造功能齊全的遊戲體驗
- 透過 App Engine 儲存使用者遊戲體驗數據並維護遊戲狀態
- 透過 App Engine 自動擴展 Compute Engine 專用遊戲伺服器的數量
- 藉由分析大量用戶資料和遊戲數據，取得具參考價值的商業洞察

線上遊戲已經從過去少少幾個人在自己家的車庫運行遊戲伺服器，發展到數以百萬計的玩家能夠同時在線上享受遊戲體驗、遊戲商城和朋友清單的服務。這些常見的遊戲模組，已經發展出足以匹配高性能電腦運算功能和大規模網頁應用的分散式系統。在遊戲界的激烈競爭下，開發者需要更妥當地管理資源，將資源著重在開發重點遊戲模組，才能創造下一個引領風潮的遊戲。若過度配置雲端資源或是關注無關緊要的細節，將會導致人力和財力資源的減少，導致無法著重於遊戲玩法和圖片資產。現在有了 Google Cloud Platform，遊戲開發人員擁有 Google 先前開發分散式系統的經驗，藉此可以更專注在研發獨特的遊戲體驗上。

這個解決方案藉由 App Engine 的擴展性和穩固性、運行遊戲 session 的 Compute Engine 遊戲伺服器(完全代管)做相互搭配。App Engine 是一個可擴展的平台，提供的功能如：玩家資料、遊戲連線、遊戲商城、社交空間和行動應用平台，App Engine 可以提供線上遊戲各方面的支援，但開發者經常需要使用虛擬機器來運作一般的遊戲引擎和 SDK。此解決方案在很多情況下當然可以單純的使用 App Engine，但最主要還是著重於需要在 Compute Engine 中架設專有遊戲伺服器的案例上。

此解決方案所使用到的產品有：

- Google App Engine
 - 將使用者界面圖像化並提供遊戲和使用者設定
 - 提供玩家配對和伺服器瀏覽的功能
 - 將負載分散到 Compute Engine
 - 藉由叢集維護管理玩家遊戲負載量
- Google Compute Engine
 - 負責運行自訂遊戲伺服器
- Google BigQuery
 - 遊戲資料分析和用戶數據分析
- Google Cloud Storage

- 儲存遊戲伺服器程式
- 讓玩家下載遊戲主程式及相關檔案
- 儲存備份紀錄, 處理並存入 BigQuery 中

使用情境




Dora 現在有時間想要玩幾場他最喜歡的線上遊戲: Giant Robot Smash 5000。他打開筆電並登入到他的遊戲介面中。在跟多人連線之前, Dora 發現遊戲商店可以買到新型導彈發射器。點擊後, 她透過 Google Wallet 購買了新設備, 並將新型導彈發射器裝備在她最愛的巨型步行機器人上。

Dora 和幾位在線上的朋友組隊, 現在她已經準備好開始新遊戲並再次拯救銀河系。這個團隊要求他們選擇喜歡的遊戲模式, 幾秒鐘後, 他們進入機甲的駕駛艙, 準備接受任何挑戰。

解決方案概論

下方架構圖(圖一)展現了如何整合 Compute Engine 和 App Engine, 並同時創造兼具穩固和高擴展性的線上遊戲解決方案。

Dedicated Server Gaming Solution on the Google Cloud Platform

-  Your Application Code running on Google App Engine, Google Compute Engine, and Client Devices
-  Google Cloud Platform Services
-  Capabilities Included

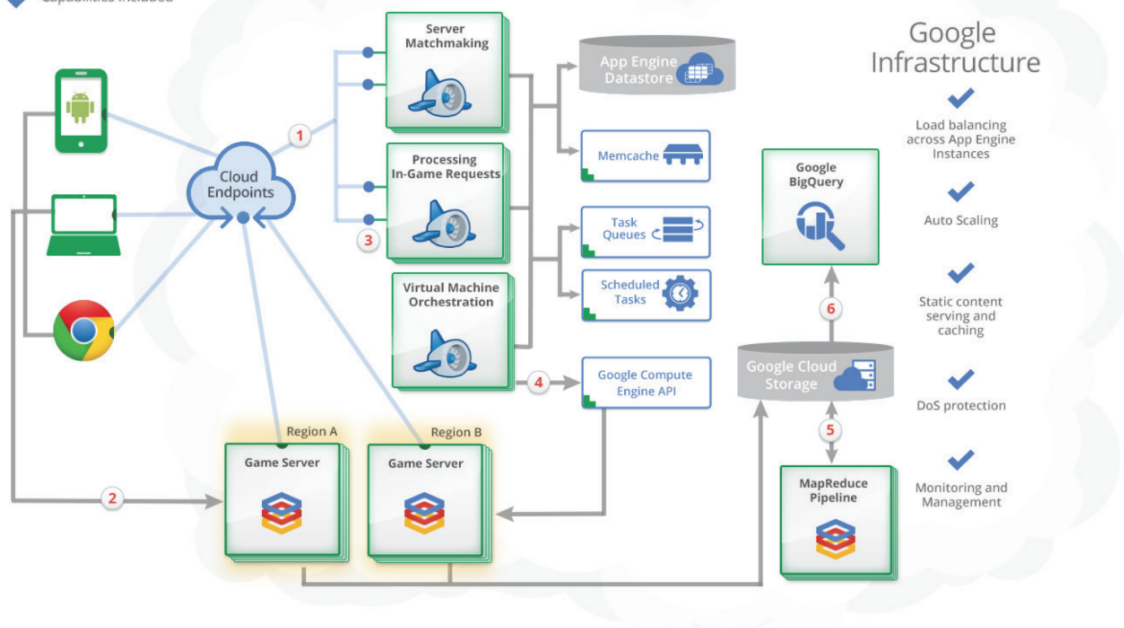


圖1:線上遊戲解決方案參考架構圖

線上遊戲解決方案的重點

1. 選擇遊戲伺服器
2. 遊戲玩家連線到專用遊戲伺服器
3. 遊戲內的請求和 Google Compute Engine Instance 健康狀況檢查
4. 自動擴展遊戲伺服器
5. 儲存記錄並利用 MapReduce 分析
6. Google BigQuery 分析大量用戶和遊戲數據

用戶可以透過下載本地應用程式或是到遊戲網站來開始遊戲。如果玩家是第一次玩，所有用戶的程式和遊戲資料將可從 Cloud Storage 下載。雖然遊戲端的行動裝置和個人電腦會有所不同，但遊戲的核心功能將適用於各裝置。核心功能包括更新使用者資訊、玩家設定管理和查看好友積分。App Engine 能夠直接提供網站服務或是提供存取所有需要資訊的 RESTful API。

以下部分將針對此遊戲解決方案的主要功能做更詳細的介紹：

1. 選擇遊戲伺服器

玩家間的互動是遊戲中最重要的一部份，而玩家配對則是遊戲解決方案中最重要的一部分，因為它可以將同一個區域和相同遊戲模式的玩家進行相互連線。透過搜尋、性能表現和可擴展性的需求，這項解決方案可以同時被延伸應用在 Google Cloud SQL、Search API 或 Datastore 的多功能性瀏覽器和搜尋功能上。

2. 玩家連線到專用遊戲伺服器

一旦玩家進入遊戲，而用戶端將會接收到專用伺服器的 IP 位置，玩家所建立的連線，會在 Compute Engine 中的專用連線伺服器上運行，並加載遊戲內部資源。Compute Engine 遊戲伺服器透過低網路傳輸處理玩家的線上互動。關於設計多人連線遊戲伺服器不在本文所探討的範圍。在設計多人連線遊戲伺服器時，建議使用現有的遊戲伺服器和軟體開發工具包。

3. 遊戲內的請求和 **Compute Engine Instance** 運行健康狀況檢查

當專用遊戲伺服器在 Compute Engine 上運行，可能需要向 App Engine 發送遊戲內部請求。若玩家在商店中購買物品或是新增遊戲設定，App Engine 將提供正確的訊息來源。此外，專用伺服器也能夠和 App Engine 相互聯繫，作為更新玩家積分、統計狀態和經驗值的用途。當遊戲比賽結束時，玩家可以選擇進行新的一輪比賽或是重新做連線配對。玩家的積分、比賽統計數據和遊戲

商品推薦會顯示在遊戲畫面中。如果專用遊戲伺服器無預期終止，玩家端必須重新將玩家連線至新的伺服器中。

4. 自動擴展遊戲伺服器

自動擴展是其中一個對遊戲內容較不會有顯著影響的後端功能，但其實這對於建構可擴展性的完整遊戲十分重要。此方法由 App Engine 開發人員設定的專用遊戲伺服器自動擴展邏輯來做執行。隨著玩家人數的增加，VM 的調控機制會自動創造新的專用伺服器來管理新增的流量。同理，如果玩家人數減少，將會回收未使用的專用伺服器節省空間。

5. 分析儲存記錄和 MapReduce

建議使用 Google Cloud Storage 儲存文件，像是伺服器記錄或是 MapReduce 的輸出紀錄。Compute Engine 中的專用遊戲伺服器會產生大量玩家行為數據和軟體除錯資料。為了長期保存這些數據，應該透過背景程序定期將資料從 Compute Engine Instances 更新至 Google Cloud Storage。如果透過 MapReduce 轉換和彙集數據，則可以從 Google Cloud Storage 下載相關文件並透過額外的 Compute Engine 加以處理。MapReduce 作業的輸出資料可以儲存至 Google Cloud Storage 中，並且作為其他應用管道的輸入資料，像是存至 Google BigQuery 或是編譯至報告中。

6. 運用 Google BigQuery 分析巨量用戶和遊戲數據

這個解決方案是將資料搜集至 Google BigQuery，一個能夠即時分析大量資料的搜尋工具。當專用遊戲伺服器同時控管上百萬的玩家時，則會生成數十億筆有價值的數據。無論是原始玩家數據或 MapReduce 輸出數據，都可以預先設定將儲存在 Google Cloud Storage 的數據提取到 Google BigQuery 中。完成資料匯入後，類似於 SQL 的查詢將會在幾秒鐘內完成，並可以獲取像是玩家的參與度、遊戲獎賞的實質影響等有價值的訊息。

實作細節

接下來的部分將提供實作細節，包含：分配玩家負載、創建一個功能完整的遊戲體驗所需的核心功能。此解決方案的主要重點在於分配遊戲伺服器以處理玩家即時互動的場景，此外也可以加以提供其他功能：更完整的商店和社交模組等等，但這超出了本文所涵蓋的範圍。

以下架構圖(Fig. 2)描述如何實施可擴展的專用遊戲伺服器解決方案。

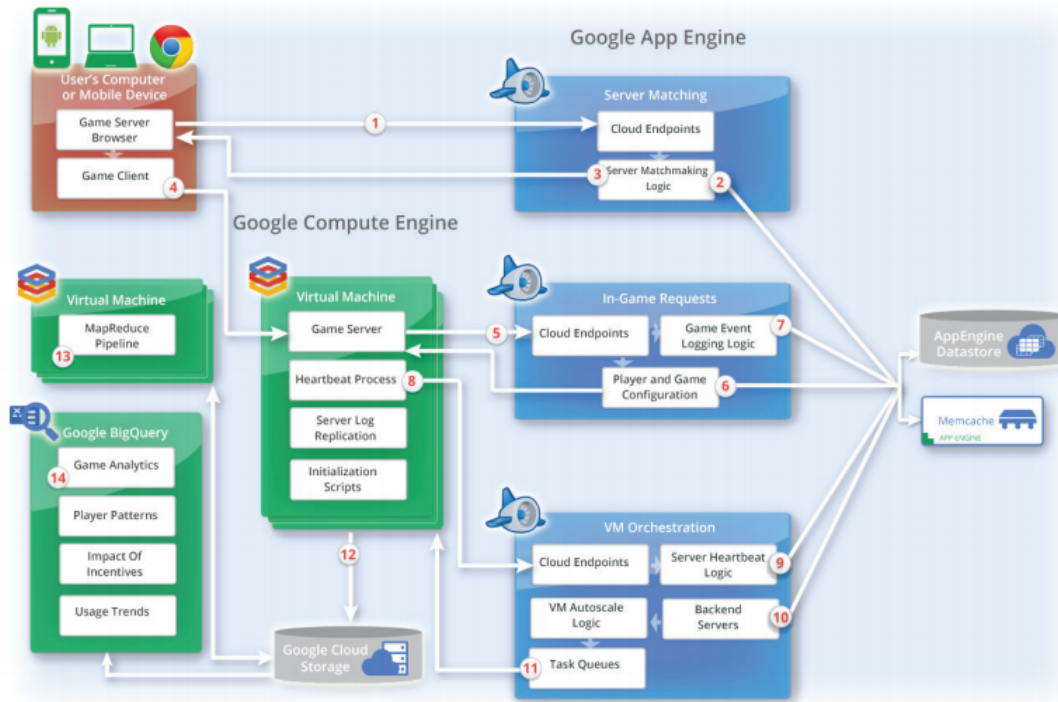


Figure 2. Implementation Details for an dedicated server gaming solution

重點導覽圖解和實作細節

1. 遊戲伺服器的選擇請求

玩家藉由遊戲伺服器瀏覽器對後端伺服器發出請求，以獲取依照遊戲配對規則計算出的推薦伺服器。這些請求經由 Cloud Endpoint 發送到提供 RESTful API 的 App Engine，而 Cloud Endpoint 提供了身份認證 (authenticate) 的功能。

2. 遊戲伺服器的配對邏輯

伺服器配對邏輯將為用戶提供受推薦的伺服器列表，根據伺服器配對請求的規模和頻率，而有不同實作此解決方案的方法。其中一種方法是使用 App Engine 的後台任務來維護每個數據中心所推薦的伺服器列表，並將列表儲存在快取中以便快速檢索。而推薦

伺服器的邏輯將取決於遊戲的類型，例如，某些遊戲透過最低的負載來盡可能的降低延遲，而有些多人遊戲則需要盡可能地將大量的玩家放入伺服器中。雖然快取提供了高性能的分佈式記憶體物件緩存系統，但還是必須儲存在 App Engine Datastore 中，以便可以搬遷快取，建議後台任務由 App Engine Cron service 每分鐘運行一次。因為玩家通常希望連接到最低延遲伺服器，所以由後台任務維護的區域推薦伺服器列表就顯得非常重要。其他伺服器選擇技術可能會不斷存取可用的伺服器，或向客戶端提供大小合理的列表，以便區別哪些是最低延遲的伺服器。更複雜的解決方案包括維護所有伺服器的玩家數量、負載、延遲和狀態，還可以動態查詢每個請求。

3. 回傳遊戲伺服器匹配的結果

遊戲伺服器配對的結果是回傳給在列表中選擇的客戶端或客戶端自動選定的伺服器。

4. 遊戲玩家連接到專用遊戲伺服器

玩家嘗試連接到所選專用遊戲伺服器的 IP 地址。如果連接失敗或是伺服器滿載，則玩家可以嘗試連接到其他伺服器或將玩家重新進行伺服器配對。

5. 遊戲伺服器的遊戲內請求

在玩家與專用遊戲伺服器連接之後，伺服器將負責處理來自玩家的所有事件並提供當前其他玩家的資訊。利用 App Engine 來處理重要的事件並提供玩家資訊，以確保玩家在專用伺服器上有一致的遊戲體驗。如果玩家有一些自訂的配置，將對 App Engine 發送相關配置信息的請求，並允許玩家存取其購買的所有項目。隨著玩家獲得經驗並觸發重要的遊戲事件，詳細信息將發送至 App Engine 以確保能接收到所有玩家的資訊。對 Cloud Endpoints 發送認證請求和提供 RESTful API 是將遊戲伺服器連接到 App Engine 的簡易方法。

6. 請求玩家設定資訊

當遊戲允許玩家購買商品或自訂玩家設定時，這些資訊就必須保存在可靠及可擴展的資料庫中。App Engine Datastore 是設計來擴展數百萬用戶使用的網頁應用程式，這項 Google Megastore technology 被用在 Google 的許多產品中。App Engine Datastore 適合用來儲存所有玩家資訊，因為它可以隨著遊戲從數百人無縫擴展至數百萬玩家。我們也可以透過快取儲存頻繁查詢 App Engine Datastore 的結果，來提高性能，但由於快取的資源有限，因此建議有效率的使用。如果需要複雜的 SQL 查詢或是有使用 MySQL 的必要，Google Cloud SQL 將提供完全託管且高度可用的關係式資料庫服務。雖然 Google Cloud SQL 與 App Engine 有高度整合，但它並沒有往無限度擴展的方向做設計，所以建議還是先進行壓力測試，以便了解在實際應用場景下資料庫的效能。

7. 儲存重要的遊戲事件

處理並儲存重要的事件(比如:玩家在遊戲後獲得的經驗),是創造受歡迎遊戲的關鍵。玩家配置的請求也是由 App Engine 處理,且重要資訊可以儲存在 App Engine Datastore。這兩種請求之間最主要的差別在於,對所有活躍玩家而言,遊戲事件發生的頻率更高。舉例來說,玩家的配置只能在比賽開始時獲得,而遊戲事件可能在每次玩家獲得經驗時觸發。雖然 App Engine Datastore 可以透過自動擴展來處理來自數百萬用戶的事件,但為了避免後續遇到與系統擴張性相關的問題,開發人員應該詳細了解 entity groups、NoSQL 資料庫、eventual consistency 的特性。有關這些主題的詳細討論可以在 App Engine 開發人員文件中找到。

8. 伺服器的 Heartbeat Process

關於如何維護專用遊戲伺服器叢集的健康,最關鍵的部分是持續追蹤每台伺服器的統計數據和運行狀況。Cloud Endpoints 再一次的被用來提供經身份驗證的 RESTful API,API 包括每個 Compute Engine instance 上運行的狀況以及提供硬體相關信息(例如:CPU 和 RAM 使用率)、遊戲特定資訊(例如:玩家平均等待的時間和伺服器上活躍的玩家數量)。

9. 儲存伺服器運行狀況和統計數據

在 Compute Engine instance 上運行的 Heartbeat process 可以提供大量有價值的資訊,且需要伺服器 heartbeat 邏輯去解析和儲存。與自動擴展相關的訊息(例如:伺服器上活躍的玩家數量和平均延遲時間)應暫存在快取中,以供後端服務能夠快速查詢。任何重要的資料也應儲存在 App Engine Datastore 中,以防止快取被清除。假如此資訊與分析和維護伺服器的歷史記錄相關,則建議將所有的數值儲存在獨立且可自動擴展的表格中。

10. 自動擴展專用遊戲伺服器並保持運行

儘管有許多方法可以根據玩家負載自動擴展 Compute Engine 的資源,但通用元件包括使用 App Engine Cron 服務運行每分鐘一次的任務排程。可以透過預設的時間表或是分析遊戲伺服器中空房的數量和玩家延遲速度,來計算理想的虛擬機數量。自動擴展的另一個關鍵輸入是透過快取或是 App Engine Datastore 取得 heartbeat process 的數據來確定當前運行狀態。理想的虛擬機器數量與當前遊戲伺服器數量的差異可以透過新增或刪除 instances 來達成。此外,任何不健康的伺服器都應該在沒有玩家時刪除。如果需要在不同的 Compute Engine zone 之間遷移遊戲伺服器,則可以使用自動擴展的方式在新的 zone 創建,同時移除原 zone 中間置的 instances。這是自動擴展遊戲伺服器的概述,還是建議您謹慎使用擴展的演算法,並盡量避免過度調整和 noisy responses 等問題。Compute Engine servers 是按秒計費(譯者按,現在 GCE 是開啟

第一分鐘之後以秒計費), 因此為了減少未使用的 Compute Engine instances 所帶來的成本, 請避免不斷創建或刪除 instances。

11. 建立與刪除專用遊戲伺服器

確認要刪除或建立遊戲伺服器後, 任務才會添加到 App Engine 任務序列中。獨立的後台任務將負責維護伺服器以及呼叫 Compute Engine API。如果 API 呼叫數量超出後端的限制也可以新增額外的後端。如果 Compute Engine API 呼叫量不多, 則可以透過任務排程來進行伺服器的維護, 這可以降低 App Engine 的資源消耗。建議在每個伺服器任務維護中包含一個 timestamp, 以便在系統中發生 backlog 時出現警報。可以把 Push queues 當作 pull queues 的替代方案, 建議進行壓力測試, 方法包含: 在 Compute Engine 上運行常見的開源技術, 或是使用第三方服務進行壓力測試。

12. 將日誌儲存在 Google Cloud Storage

遊戲中的伺服器日誌記錄將統計每個玩家的操作及移動直到遊戲結束, 每個遊戲伺服器都會生成許多日誌文件, 可以使用背景程序將這些文件定期複製到 Google Cloud Storage 中。如果關鍵數據儲存在文件中, 則應該將其儲存在硬碟上, 以防止在複製過程完成之前, instance 意外終止導致資料遺失。或是將文件儲存在較低成本的臨時磁碟上, 不過磁碟會連同 instance 一起被刪除。但無論選擇何種磁碟, 始終建議您使用自動複製流程來維護在 Google Cloud Storage 的所有日誌和統計資料。

13. 轉換和處理日誌文件

從伺服器收集大量原始日誌數據後, 我們也需要清理日誌文件, 增加額外的數據並彙總到不同的級別。MapReduce 或 Extract、Transform、Load 工具可以建立用於玩家導向的數據, 如商店項目的推薦, 或是匯入 Google BigQuery 進行分析。

14. 報告與分析

Google BigQuery 是遊戲解決方案的重要組成元素, 因為它可以針對大量玩家和遊戲資料進行特定的分析。舉例來說, 它可以用於分析遊戲獎賞機制(例如: 商品特價或雙倍經驗值) 對用戶的影響。此外, Google BigQuery 在數據擴展到兆位元組和數十億行時, 也能保持一致性。

範例應用程式

範例應用程式使用此解決方案的概念, 可以當作參考。範例應用程式核心功能如下:

- 客戶端向 App Engine 查詢專用遊戲伺服器的 IP 地址

- 客戶端透過連接到在 Compute Engine 上運行的遊戲伺服器來開始新遊戲
- 管理人員可以從 App Engine Administration UI 建立與刪除遊戲伺服器
- Compute Engine instances 定期向 App Engine 報告負載級別
- 管理人員可以查看所有遊戲伺服器 Compute Engine instances 的負載級別
- 如果叢集超過最大負載值，App Engine 會自動添加新的 instances

結論

此解決方案展現了開發人員如何在擴展支援上百萬玩家的線上遊戲的同時，提供功能完整的遊戲體驗。透過使用多個在 Google Cloud Platform 上的元件，開發人員可以在 Compute Engine 上運行符合業界標準的遊戲伺服器，同時具備 App Engine 的高擴展性及高可靠性。這使遊戲開發人員能夠快速啟動、迭代與擴展，並致力於提供更出色的遊戲。

([原文](#)出自 Google Cloud。)