建立負載均衡的 online judge 系統

組別：347-110-10
指導教授：李龍盛 教授
組員：盧柏瑜、許哲愷

目錄

簡介

- 一套基於 Coderunner & Jobe Sandbox online judge 的流量分散系統
- 基於反向代理的原理建立，透過一台代理伺服器將流量分散至數台沙盒伺服器，以達成負載均衡，並且能夠避免塞車，還能夠自動迴避故障的沙盒伺服器。

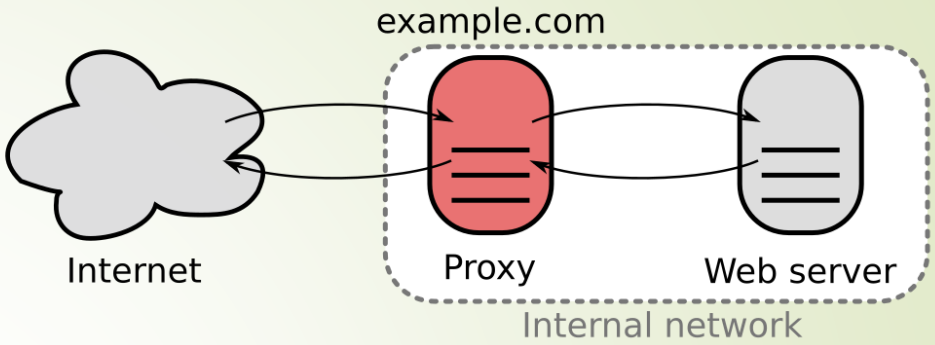
專題動機

- 現有的 online judge 都有一些缺點：
  - 老師與學生還要特別申請帳號，相當麻煩
  - 老師必須於不同系統中追蹤學生的學習狀況
  - Web server 與 judge server 綁在同一台主機，**時常造成系統塞車而無法使用**

特色及目標

- 後端主機附載可依權重分配流量
- 自動迴避崩潰的後端主機
- 能夠快速建置、擴充後端主機
- 不同程式語言支援的後端主機群
- 結合學校的輔助教學系統

系統開發相關技術: 設計核心



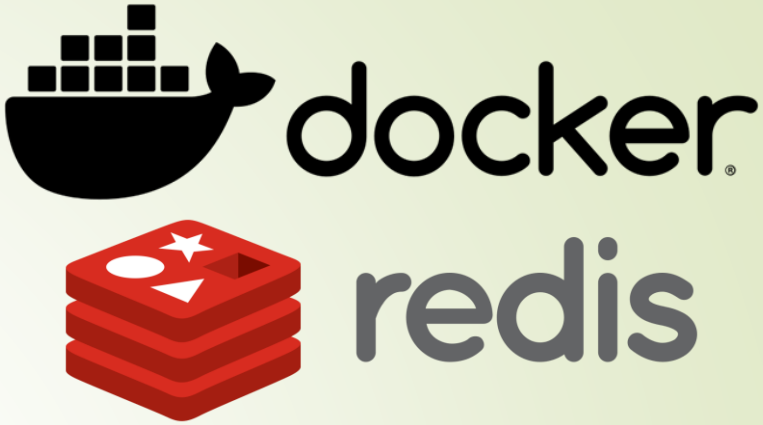
- 反向代理 (Reverse Proxy)

- 反向代理是一種代理伺服器。伺服器根據客戶端的請求，從其關聯的一組或多組後端伺服器上取得資源，然後將這些資源返回給客戶端
- 客戶端只會得知反向代理的IP位址，而不知道在代理伺服器後面的伺服器叢集的存在

系統開發相關技術: 系統基礎

- Moodle LMS
  - 開源的學習管理系統。有完善的出題系統強大且豐富。
- Coderunner & Jobe Sandbox
  - Coderunner 是一套開源的 Moodle 插件，用來線上評分程式設計問題
  - Jobe Sandbox 是為 Coderunner 所製作的遠端沙盒，一樣是開源軟體
  - 兩者使用 REST API 溝通

系統開發相關技術: 使用軟體



- Docker 是一套使用 OS-level 虛擬化和套件封包來交付軟體的平台即服務(PaaS)產品
  - 使用它來快速建立需要的背景服務
- Redis 是一套開源的記憶體資料結構儲存，用來作為資料庫、快取和訊息中間人

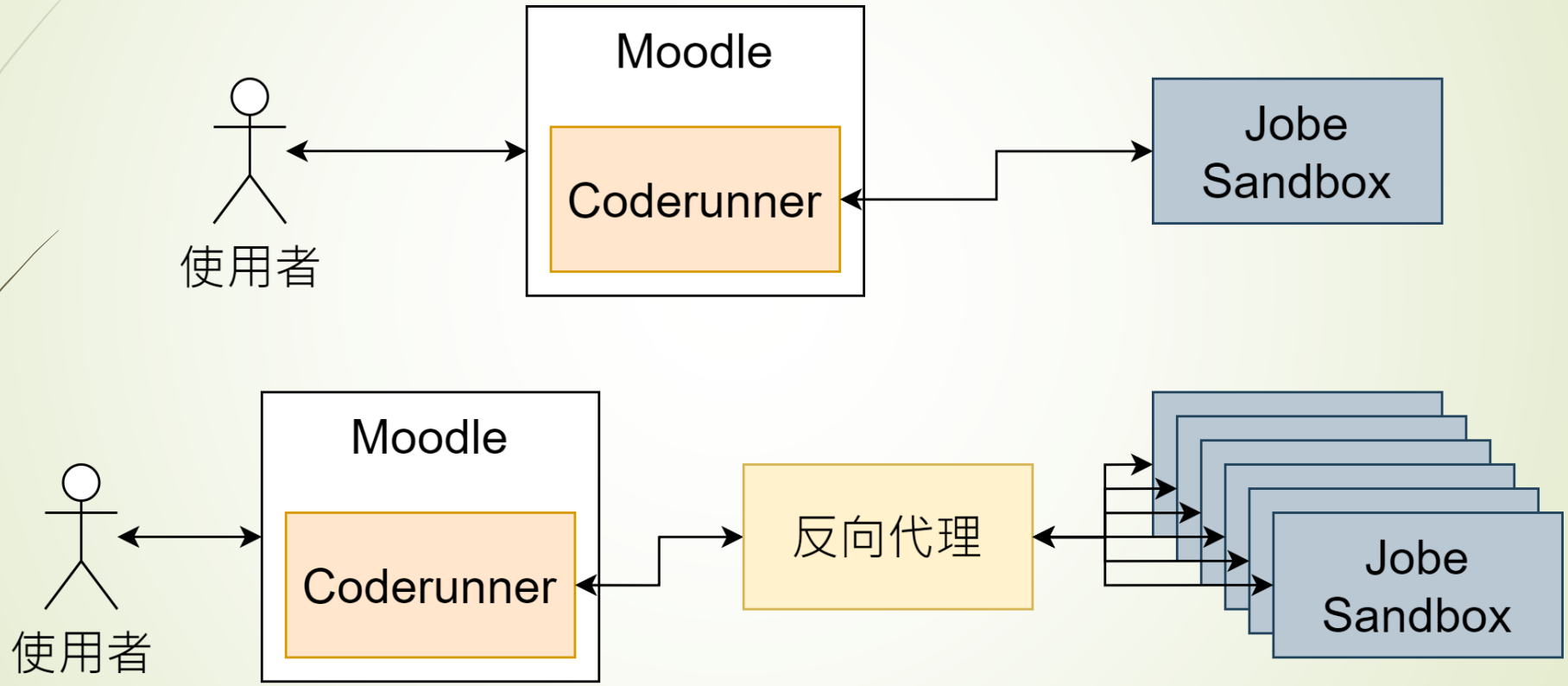
系統開發相關技術: 使用框架

- Gunicorn + Flask
  - Gunicorn 是用於UNIX的Python WSGI HTTP服務，兼容各種 Web 框架，速度快且服務器資源少
  - Flask 是一個使用Python編寫的輕量級Web應用框架
- Celery
  - 一套開源基於訊息傳遞的非同步任務佇列(Task queue)

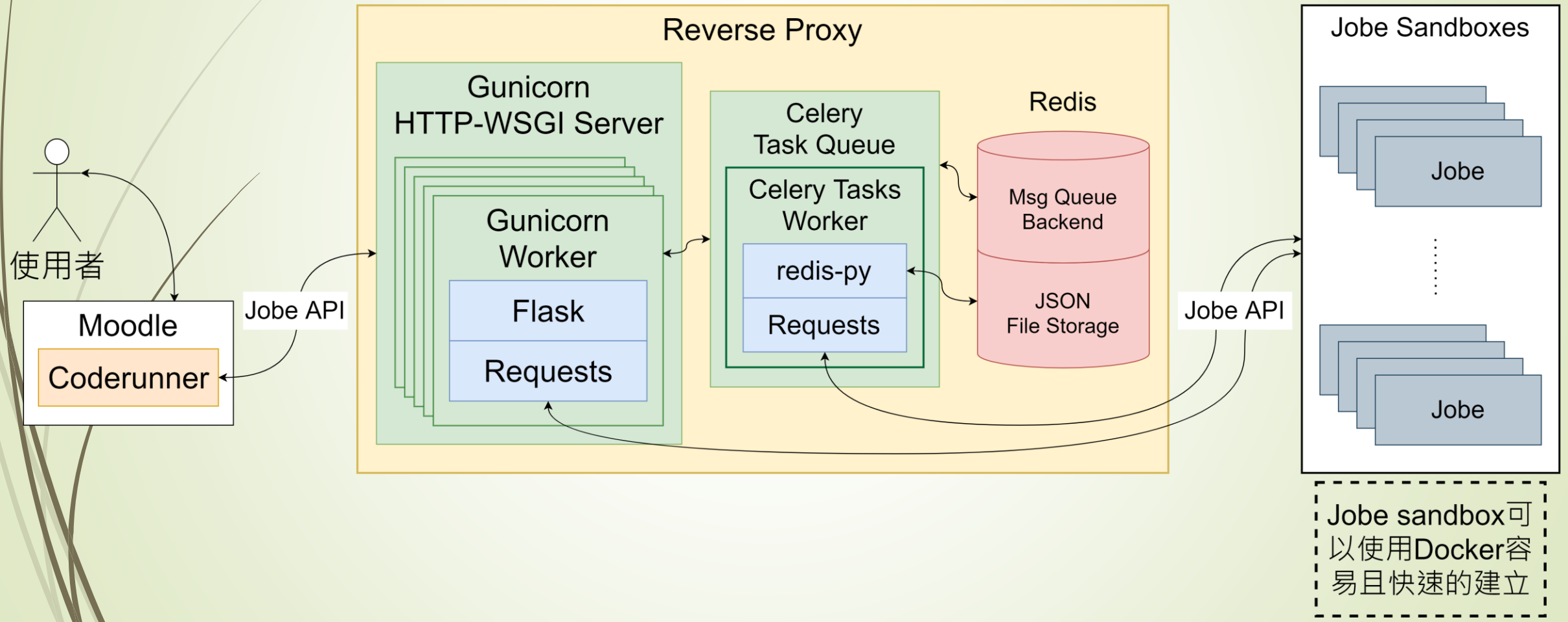
系統開發相關技術:
第三方套件

- Requests
  - 一個簡練且容易使用的 Python HTTP 函式庫
- redis-py
  - Redis 官方推薦使用的 Python Redis 用戶端

系統概觀


Moodle
Coderunner
使用者
Jobe
Sandbox
Moodle
Coderunner
使用者
反向代理
Jobe
Sandbox

系統架構


使用者
Moodle
Coderunner
Jobe API
Reverse Proxy
Gunicorn
HTTP-WSGI Server
Gunicorn
Worker
Flask
Requests
Celery
Task Queue
Celery Tasks
Worker
redis-py
Requests
Redis
Msg Queue
Backend
JSON
File Storage
Jobe API
Jobe Sandboxes
Jobe
Jobe
Jobe sandbox可以使用Docker容易且快速的建立

Jobe API

有實現

- (GET) **get_languages**

  取得支援的程式語言

- (POST) **submit_run**

  送出程式碼並取得執行結果

- (PUT) **put_file**

  上傳額外檔案至沙盒(client 決定檔名)

- (HEAD) **check_file**

  檢查沙盒中是否有某一檔案

未實現

- (GET) **get_run_status**

  取得某一次 submit_run 的執行狀況

  只回覆 404

- (POST) **post_file**

  上傳額外檔案至沙盒(server 決定檔名)

  只回覆 403

為什麼不用一般的Web軟體(如Nginx)來達成?

- 當使用者按下送出的時候，Coderunner 會做一連串的 API 請求
- 為確保正確的運行，這些 API 請求(大部分)**必需送到同一個沙盒**
- Example:
  - Bob 送出一個含有額外檔案的繳交
  1. Coderunner 向 Jobe 呼叫 get_languages，確認Jobe支援的程式語言
  2. Coderunner 確認完後，送出 submit_run 並等待結果
     - Jobe 缺少額外的檔案，回傳 404，到步驟三
     - Jobe 成功跑完繳交的程式碼，回傳 200 和結果，流程完成
  3. Coderunner 向 Jobe 送出 put_file，上傳額外檔案
  4. Coderunner 再送一次 submit_run 並等待結果

為什麼不用一般的Web軟體(如Nginx)來達成?

- 以上範例中：
  1. **get_languages** 必須能正確表示所支援的程式語言
  2. 這次繳交所有的 **put_file** 和 **submit_run** 必須送達同一台沙盒

- 若使用 Nginx 的 proxy_pass 功能，每個 API 呼叫都會被當成獨立事件，無法保障以上事項能成立

與原 API 比較

	一般 Jobe Sandbox	反向代理
get_languages	取得**該沙盒**支援的程式語言	取得**所有沙盒**支援的程式語言
submit_run	送出程式碼並取得執行結果	送出程式碼，**分配至沙盒**並取得執行結果 若中間**遇到錯誤，分配至另一個沙盒**
put_file	上傳額外檔案至**該沙盒**	上傳額外檔案至**反向代理** **需要時再上傳至沙盒**
check_file	檢查**該沙盒**中是否有某一檔案	檢查**反向代理**中是否有某一檔案

Celery 和 Redis 用來幹嘛?

- 解決開發中遇到的狀況：
  - 反向代理需要更新並暫存每個沙盒支援語言的資訊
  - 在大量接受 API 要求時，用讀寫檔案的方式會有造成有些 Gunicorn worker 讀不到東西

- 加入 Celery 和 Redis 來處理**更新、暫存**這些資訊
- 而這些 Gunicorn worker 在需要時向 Celery 要求這些資訊

結論

- 運用反向代理伺服器的設計，我們讓原先不易擴充的系統能夠容易的擴充負載能力，更不必過度擔心後端沙盒
- 加上 Jobe Sandbox 明確且詳細的 API 文件，我們能在不更動原始系統程式碼的情況下完整實現了我們想要的功能

參考文獻

- Jobe REST API - https://github.com/trampgeek/jobe/blob/master/restapi.pdf
- Jobe Github repo - https://github.com/trampgeek/jobe
- Coderunner Github repo - https://github.com/trampgeek/moodle-qtype_coderunner
- Flask 官方文件 - https://flask.palletsprojects.com/en/1.1.x/
- Gunicorn 官方文件 - https://docs.gunicorn.org/en/latest/index.html
- Requests 官方文件 - https://requests.readthedocs.io/en/latest/
- Celery 官方文件 - https://docs.celeryproject.org/en/stable/
- redis-py 官方文件 – https://redis-py.readthedocs.io/en/stable/

事後反思

- 老師主要的提問點：
  - 反向代理的效能如何？負載上限為何？
  - 為何不使用 round-robin 而採用隨機的分配方式？
  - 此系統套用到學校的系統上，實際效能又是如何？