**HTTP 伺服器效能改進 - 專案報告**

**111110543 凃玫妮**

**簡介**

此專案旨在使用 Python 建立一個 HTTP 伺服器，並專注於效能的提升。主要目標是通過實現多線程和非同步 I/O 操作，增強基本伺服器的能力，以有效處理多個客戶端請求。本報告概述了開發過程、所做的改進以及通過這些改進實現的效能提升。

**專案描述**

此專案涉及建立一個能夠有效處理多個客戶端請求的 HTTP 伺服器。專案分為三個階段：

1. **基本 HTTP 伺服器**：使用 Python 的內置 **http.server** 模塊實現一個簡單的 HTTP 伺服器。
2. **多線程**：通過引入多線程來提高伺服器的能力，以處理每個請求的獨立線程，從而實現同時處理多個請求的能力。
3. **非同步 I/O**：進一步提高效能，使用 **aiohttp** 库實現非同步 I/O 操作，從而使伺服器能夠有效處理 I/O 操作，提高整體效能。

**階段 1：基本 HTTP 伺服器**

使用 Python 的內置 http.server 模塊實現了一個基本的 HTTP 伺服器。此伺服器能夠處理簡單的 GET 請求並返回靜態響應。

**階段 2：多線程**

為了提高伺服器處理多個請求的能力，引入了多線程。每個請求在單獨的線程中進行處理，這使得伺服器能夠同時處理多個請求，而不需要等待每個請求按順序完成。

**階段 3：非同步 I/O**

為進一步提高伺服器的效能，使用 **aiohttp** 库實現了非同步 I/O。這使得伺服器能夠非同步處理 I/O 操作，從而減少等待 I/O 操作完成的時間，提高整體效能。

**效能評估**

根據伺服器處理並發請求的能力進行了每個階段的效能評估：

1. **基本 HTTP 伺服器**：按順序處理請求。在高負載下，性能明顯下降。
2. **多線程**：通過使用多線程處理每個請求，提高了處理並發請求的能力。每個請求在單獨的線程中進行處理，從而在負載下性能得到改善。
3. **非同步 I/O**：通過實現非同步 I/O 操作，進一步提高了效能。伺服器可以非同步處理 I/O 操作，從而有效地處理大量的並發請求。

**結論**

本專案成功展示了通過實現多線程和非同步 I/O 操作在 HTTP 伺服器中實現的效能改進。初始的基本伺服器得到了顯著改進，能夠更有效地處理並發請求，提供了處理高負載場景的堅實解決方案。

**參考文獻**

* Python **http.server** 文件：[https://docs.python.org/3/library/http.server.html](https://docs.python.org/3/library/http.server.html" \t "_new)
* Python **threading** 文件：[https://docs.python.org/3/library/threading.html](https://docs.python.org/3/library/threading.html" \t "_new)
* aiohttp 文件：https://docs.aiohttp.org/en/stable/

**專案開發備註**

* **原創工作**：本專案的程式碼和概念是原創的，根據任務要求和目標開發。
* **參考資料**：雖然本專案是原創的，但參考了標準庫文檔以了解和實現多線程和非同步 I/O 操作。
* **無複製**：沒有直接從外部來源複製程式碼。

本報告和相關程式碼提供了對此專案中 HTTP 伺服器效能改進的全面概述和演示。

