**資料結構 – HW2 報告***2024 NOV*

姓名：徐崇恆  
學號：112503023  
系級：通訊二

# 1 摘要

本作業的目的是在 Linux 環境中，以 C 語言開發一個性能測試程式 (benchmark) 。此作業需要將作業一的資料庫程式封裝為函式庫，並比較其與開源 Redis 的效能，測試項目包含操作耗時及記憶體使用情況。

# 2 需求分析

本作業的目標是通過性能測試，評估作業一的 NoSQL 資料庫與 Redis 之間在時間和空間複雜度上的差異。為了達到此目標，測試程式生成大量假資料，並依次執行新增與讀取操作，以測量延遲。此外，程式也會監測程式的記憶體佔用情況。

## Redis

Redis (Remote Dictionary Server) 是一個以記憶體為基礎的開源 NoSQL 資料庫，具有快速的讀寫速度。然而，其記憶體佔用相對較高，適合需要快速存取的應用場景。

# 3 設計

## 資料模型

本次測試所使用的資料模型沿用作業一中的通訊錄人物記錄格式。

以下是一份 JSON 格式的資料樣本：

{  
    "name": "Alice",  
    "jobTitle": "Software Engineer",  
    "age": 30,  
    "address": "123 Main St",  
    "phoneNumbers": ["123-456-7890", "098-765-4321"],  
    "emailAddresses": ["alice@example.com", "alice.work@example.com"],  
    "isMarried": true,  
    "isEmployed": true  
}

## 假資料生成

程式的假資料生成規則以流水號參數 i 為基準，為每個 PersonSample 實例生成各個屬性。首先，名稱 name 使用 "test\_person\_<i>" 格式，以確保每筆資料的唯一性；職稱 jobTitle 和年齡 age 則依序根據 i 的不同取值循環生成，使職稱在 "job\_0" 至 "job\_99" 間變化，年齡在 0 到 68 間分佈。電話號碼和電子郵件地址的數量根據 i 的值決定，並按特定格式生成內容。這樣的設計使生成的假資料具有規律性和唯一性，適合用於測試資料庫的儲存與查詢操作。

為了確保對兩個資料庫的測試公平性，我們定義了一個結構體 PersonSample 作為假資料的格式標準。在資料寫入和讀取的過程中，我們也考慮了資料庫原始資料類型轉換為 PersonSample 結構體所需的時間成本，以便更加準確地評估性能。

## 基準測試 (benchmark)

程式引入了 time.h 庫，以利用時間相關函數實現計時功能。本作業中所比較的是資料庫內資料所佔用的記憶體空間，而非整個程式的記憶體使用量。

在作業一中，資料庫使用雜湊表 (hash table) 作為資料結構，其中每個槽位都指向一個 DBItem 結構體。DBItem 是一個資料節點，包含資料的 key、 next 指位器以及儲存資料本體的 cJSON 結構體。為了精準計算資料所佔用的記憶體空間，需要考慮計算雜湊表、所有 DBItem 節點、 cJSON 結構體以及所有結構體中的指位器所指向的記憶體空間。

而 Redis 資料庫中資料所佔用的記憶體可以使用 Redis 指令 “INFO memory” 查詢。

# 4 實作

## 開發環境

硬體：Raspberry Pi 5  
作業系統：Raspberry Pi OS Lite (64-bit) (Linux-based OS)  
IDE：Visual Studio Code (with SSH Remote)  
編譯器：GCC  
版本控制：Git, GitHub

## Tester

本作業中定義了一個 Tester 結構體，結構體中有指向與測試資料庫相關的函式的指位器，用於在測試時執行。資料庫的寫入、讀取、刪除以及記憶體佔用大小的查詢函式會被包含在 Tester 結構體中。

|  |  |
| --- | --- |
|  |  |

Fig 1. Tester 相關程式碼片段

## 計算操作用

程式引入了 time.h 庫中的 clock() 函式，用以獲取自程式開始運行以來所經過的處理器時間 (CPU time)，單位為時鐘週期數 (clock ticks)。在測試程式時，於開始和結束處分別調用 clock() 函數以記錄時間，取得的時間差再除以 CLOCKS\_PER\_SEC 進行換算，即可得到程式的運行時間。

## Hiredis

本作業中使用了 Hiredis 函式庫對 Redis 資料庫的操作提供支持。Hiredis 是一個輕量級的 C 語言 Redis 客戶端庫，用於連接、操作 Redis 資料庫。它提供簡單、高效的 API，支援同步與非同步操作，適合嵌入式系統和性能要求高的應用場景。Hiredis 以其小巧的設計和快速的處理速度而受到廣泛使用，特別適合需要直接用 C 語言與 Redis 進行高效互動的開發者。

## 計算記憶體佔用

在計算作業一資料庫的記憶體使用量過程中，考慮到作業系統在分配記憶體時會進行對齊 (alignment) 。本作業使用 malloc.h 函式庫中的 malloc\_usable\_size 函式來測量指位器所指向空間的實際大小。該函數只能被用於測量動態分配的記憶體空間。

|  |  |
| --- | --- |
|  |  |

Fig 2. 計算作業一資料庫資料記憶體空間佔用量的程式碼片段

此外，為了取得 Redis 資料庫中資料的記憶體佔用量，本作業透過 Redis 指令 INFO memory 獲取記憶體相關資訊。接著，利用 string.h 函式庫中的 strstr 函數來提取這些資訊，並使用 stdio.h 函式庫的 sscanf 函數匹配 "used\_memory\_dataset:"，以獲得資料在 Redis 資料庫中佔用的記憶體空間的確切大小。

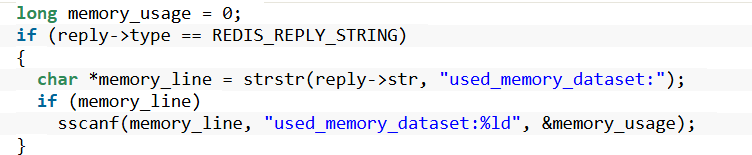


Fig 3. 計算 Redis 資料庫資料記憶體空間佔用量的程式碼片段

# 5 測試與結果

## 基準測試結果

本作業中使用自定義的通訊錄人物格式作為測試資料，測試樣本數為 100,000 筆。

|  |  |  |  |
| --- | --- | --- | --- |
| **資料庫** | **寫入用時 (ms)** | **讀取用時 (ms)** | **記憶體空間用量 (Byte)** |
| 作業一資料庫 | 456 | 552 | 132,418,832 |
| Redis | 1724.2 | 1229.6 | 43,369,440 |

Table 1. 作業一資料庫 v.s. Redis 基準測試

# 6 討論

根據測試數據顯示，作業一的資料庫在寫入和讀取延遲方面表現較佳，但記憶體佔用顯著高於 Redis，這表明作業一的資料庫適合應用於不依賴大量記憶體的高效能場景。而 Redis 雖然在延遲方面稍遜一籌，但因記憶體佔用較少，具備更高的效能。

## Redis 的 Rehash 機制

在撰寫作業過程中，我了解到 Redis 也使用雜湊表來儲存資料，但它包含一種更為複雜的 rehash 機制，能夠根據資料集大小自動調整雜湊表的大小。每個 Redis 資料庫內包含兩個哈希表，一個用於當前數據，另一個用於 rehash 過程中。當載荷因子（load factor）達到某個閾值時，Redis 會動態調整哈希表的大小，從而在內存使用和查找效率之間取得平衡。

Redis 的哈希表大小並非固定，而是隨著數據量的變化自適應地增長或縮小。當表中元素數量超過容量的某個閾值（通常為 1）時，Redis 會啟動 rehash 將哈希表擴容或縮減。這樣做有助於保持查找的高效性，使時間複雜度接近 O(1)。

## 作業一可以改進的地方

作為當前主流的開源 NoSQL 記憶體資料庫，Redis 技術穩定且安全，值得借鑒。根據測試結果，作業一的資料庫在操作速度上優於 Redis，但記憶體佔用顯著高於 Redis。為了改善這一點，以下是幾個潛在的優化方向：

1. **記憶體優化**：可以考慮引入資料壓縮技術，以減少記憶體佔用。壓縮技術有助於降低單位資料的儲存空間，這對記憶體有限的環境特別有用。
2. **改進資料結構**：作業一中使用的 cJSON 雖然在開發上帶來了便利，但增加了額外的記憶體開銷。可以考慮使用更高效的資料結構（如自定義二進位編碼），以減少對內存的需求，進一步提升效能。
3. **借鑒 Redis 的 Rehash 機制**：作業一的資料庫可以參考 Redis 的 rehash 機制，實現類似的動態調整功能，以在內存和效能之間取得更好的平衡。

## 時間複雜度與空間複雜度

在完成 100,000 筆資料的測試後，我們進行了對作業一資料庫與 Redis 資料庫性能的更全面比較，旨在分析它們在時間複雜度、空間複雜度以及資料樣本數之間的關係。

我們將樣本數從 10,000 逐步提升至 200,000，觀察到作業一資料庫的時間複雜度為 𝑂(𝑛) ，空間複雜度也為 𝑂(𝑛) ，兩者隨著樣本數增加呈線性增長趨勢。

此外，我們對每筆資料在資料庫中的相關測試總結如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **資料庫** | **寫入用時 (ms)** | **讀取用時 (ms)** | **記憶體空間用量 (Byte)** |
| 作業一資料庫 | 0.004826 | 0.005522 | 1313.834168 |
| Redis | 0.016791 | 0.012489 | 439.216724 |

Table 2. 作業一資料庫 v.s. Redis 每筆樣本資料操作用時與記憶體空間用量

測試結果顯示，作業一資料庫在寫入和讀取速度上均優於 Redis，但記憶體使用量顯著高於 Redis。具體而言，作業一資料庫的寫入時間約為 Redis 的 0.29 倍，讀取時間約為 Redis 的 0.44 倍，顯示出在操作速度上有較高的效能。然而，在記憶體空間用量方面，作業一資料庫約為 Redis 的 2.99 倍，明顯佔用了更多的記憶體資源。

# 7 結論

本作業在 Linux 環境下對作業一自建資料庫與 Redis 進行效能測試，涵蓋寫入、讀取速度及記憶體使用量的比較。結果顯示，作業一資料庫在操作速度上優於 Redis，但記憶體佔用約為 Redis 的三倍，適合高效能且記憶體不受限的應用場景；而 Redis 以較低的記憶體使用在資源受限環境中更具優勢。

為進一步提升作業一資料庫的記憶體效率，可以考慮引入壓縮技術和更高效的資料結構設計，並參考 Redis 的 rehash 機制進行優化。本次測試為 NoSQL 資料庫的選擇和改進提供了有價值的數據依據與方向。

# 8 參考文獻與資料

1. Redis - Docs  
   <https://redis.io/docs/latest/>
2. Hiredis - Redis  
   <https://redis.io/lpage/hiredis/>

# 9 附錄

1. 程式碼GitHub 倉庫  
   <https://github.com/113NCUCE/hw2-cch137>
2. 函式庫 cJSON GitHub 倉庫 - DaveGamble/cJSON: Ultralightweight JSON parser in ANSI C  
   <https://github.com/DaveGamble/cJSON>
3. 函式庫 hiredis GitHub 倉庫 - redis/hiredis: Minimalistic C client for Redis >= 1.2  
   <https://github.com/redis/hiredis>