基於 SST 的智慧型 Apache Ozone 次世代分散式檔案系統之 RocksDB 壓縮優化

SST-Based Intelligent RocksDB Compaction Optimization for Apache Ozone, a Next-Generation Distributed File System

指導教授：莊坤達
專題成員：李緒成
開發工具：Java, VS Code, Maven, Docker, Prometheus, Grafana, Apache Ozone, RocksDB
測試環境：MacOS, Ubuntu

一、簡介：

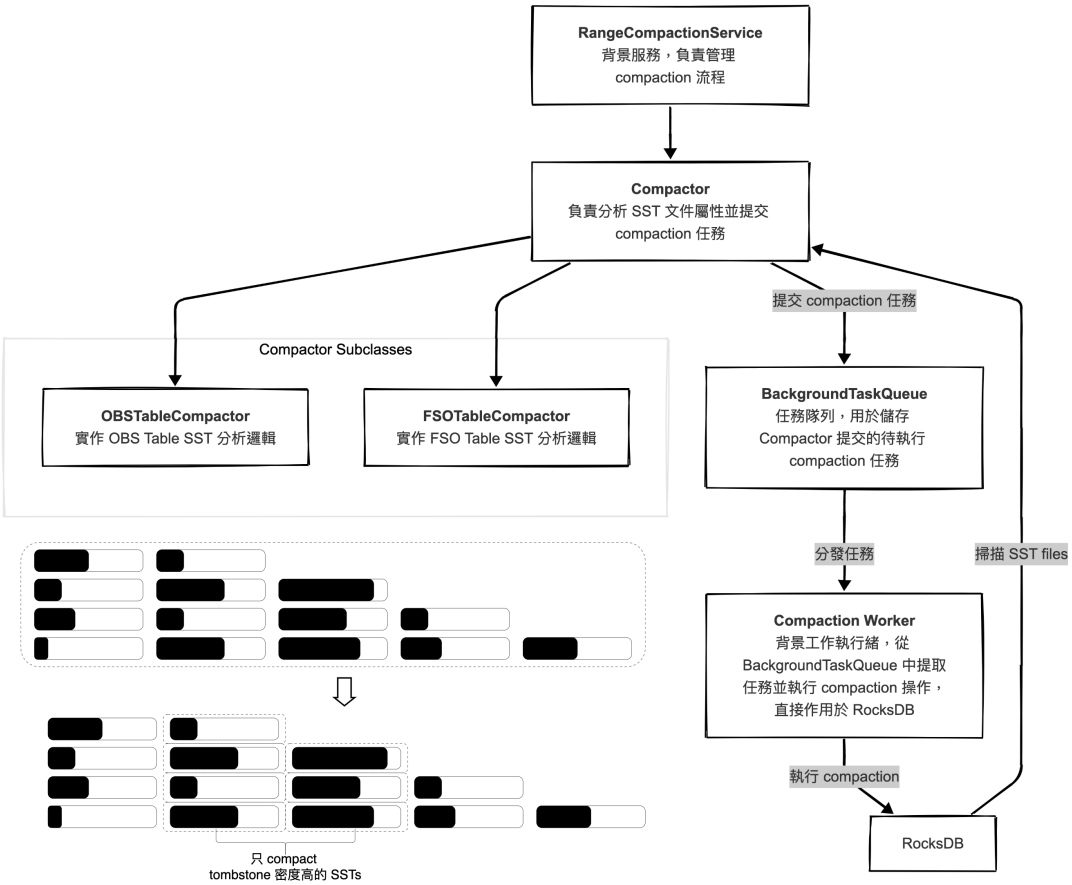
Apache Ozone 是一個高效能、可擴展且可靠的分散式物件儲存系統，旨在滿足大規模資料儲存需求。其核心組件之一是 Ozone Manager (OM)，負責管理元數據，並使用 RocksDB 作為底層鍵值儲存引擎。RocksDB 是一款高性能的嵌入式儲存庫，廣泛應用於分散式系統中。然而，隨著資料量的增加和頻繁的刪除操作，RocksDB 中會累積大量的墓碑（tombstone），這些無效資料不僅浪費儲存空間，還會降低查詢性能。為了解決此問題，RocksDB 提供了 compaction（壓縮）機制，通過清理墓碑和重組資料來優化儲存結構。

目前 Ozone Manager 的 compaction 策略是對整個 column family（列族）進行壓縮，這種粗粒度的方式會導致大量的寫入放大（write amplification），進而影響線上系統的性能。尤其在處理大規模資料時，這種方法效率低下，無法精準定位墓碑密集的區域，無法有效減少不必要的資料移動。

本專題旨在設計並實現一種基於 SST（Sorted String Table）檔案的智慧型 compaction 策略。通過分析 SST 檔案的屬性（如 num_entries 和 num_deletion），識別墓碑密集的 key range，並對這些範圍進行針對性壓縮（targeted compaction）。此方法旨在：**減少不必要的資料移動，提升 compaction 效率、降低寫入放大，最大程度減少對線上系統性能的影響、優化 Ozone Manager 的儲存空間利用率**

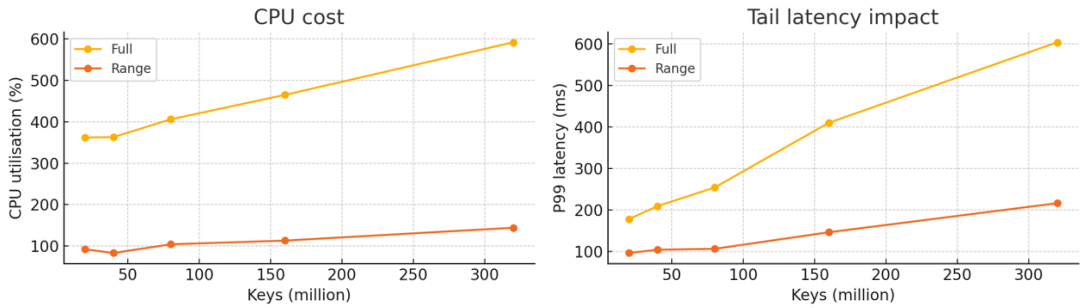
為實現上述目標，本專題採用以下技術手段：**1. SST 屬性分析**：利用 RocksDB Java API 的 getPropertiesOfTablesInRange，獲取指定 key range 內 SST 檔案的統計資訊（如條目數和刪除數）**2. Compactor 設計**：實現一個抽象的 Compactor 類別，根據 SST 屬性判斷是否需要對某個 key range 執行

compaction，並針對不同的 bucket layout（OBS 和 FSO）設計專屬的子類別，包括 **OBSTableCompactor**（適用於 OBJECT_STORE 和 legacy layout 的表格）、**FSOTableCompactor**（適用於 FILE_SYSTEM_OPTIMIZED layout 的表格） **3. 高度可調參數**：以 *ozone.om.range.compaction.** 系列參數，控制掃描週期、每次可處理的範圍數、壓縮觸發門檻與單次壓縮 timeout



二、測試結果：

(因為實驗出了點問題, 所以目前只能提供"期望"的結果)



較細粒度的壓縮能大幅減少壓縮時對服務的負擔及 RocksDB GET / Iterate 的 tail latency