# 為什麼 Redis 這麼快

### 關鍵要點

- 看來，YouTube 影片「系統設計：為什麼單線程 Redis 如此快？」由 ByteByteGo 解釋了 Redis 的速度，原因是內存存儲、IO 多工和高效的數據結構。
- 研究表明，這些因素使 Redis 能夠處理每秒高達 100,000 個查詢，儘管在請求處理方面是單線程。
- 證據顯示，最近的 Redis 版本為特定任務添加了多線程，但核心仍然是單線程。

---

### 介紹

這篇博客文章基於 ByteByteGo 的 YouTube 影片「系統設計：為什麼單線程 Redis 如此快？」，這是他們系統設計系列的一部分。Redis 以其高性能著稱，能夠在單台機器上處理每秒高達 100,000 個查詢，這對於單線程系統來說是非常驚人的。讓我們探討這是如何可能的，以及 Redis 為什麼如此快。

### Redis 速度的原因

Redis 的速度可以歸因於幾個關鍵因素，這些因素可能在影片中有所涵蓋：

- **內存存儲**：Redis 將數據存儲在 RAM 中，這比磁碟存儲快得多。這減少了延遲，增加了吞吐量，因為內存訪問時間是納秒，而磁碟訪問時間是毫秒。
- **IO 多工和單線程執行**：IO 多工，使用像 Linux 上的 epoll 這樣的機制，允許單線程高效地處理多個客戶端連接。這避免了上下文切換的開銷，單線程循環通過消除同步問題簡化了操作。
- **高效的數據結構**：Redis 使用優化的數據結構，如哈希表（O(1) 查找）、鏈表和跳表，這些數據結構通過最小化內存使用和加快操作來提高性能。

### 擴展和演變

對於高並發，Redis 可以通過多個實例或集群水平擴展。一個意外的細節是，儘管核心請求處理仍然是單線程，但自 4.0 版本以來的版本引入了多線程來處理背景對象刪除等任務，進一步提高了性能，而不改變主要模型。

---

### 調查筆記：Redis 單線程性能的詳細分析

這一部分提供了對為什麼單線程 Redis 如此快的全面分析，基於 ByteByteGo 的 YouTube 影片「系統設計：為什麼單線程 Redis 如此快？」以及相關研究。該影片於 2022 年 8 月 13 日發布，是一個專注於系統設計的系列，由《系統設計面試》書籍的作者創作。考慮到頻道的專注點，影片可能提供了適合技術面試和系統設計討論的詳細見解。

**背景和上下文** Redis 是一個開源的內存鍵值存儲，廣泛用作緩存、消息代理和流引擎。它支持字符串、列表、集合、哈希、排序集和概率結構（如布隆過濾器和 HyperLogLog）。影片的標題暗示探討 Redis 為什麼在單線程請求處理的情況下保持高性能，這是其設計的核心。

根據相關文章，Redis 可以在單台機器上處理每秒高達 100,000 個查詢（QPS），這是性能基準測試中常見的數字。這個速度在單線程模型下顯得驚人，但研究表明這是由於幾個架構選擇。

### 促使 Redis 速度的關鍵因素

1. **內存存儲** Redis 將數據存儲在 RAM 中，這比隨機磁碟訪問快至少 1000 倍。這消除了磁碟 I/O 的延遲，RAM 訪問時間約為 100-120 納秒，而 SSD 為 50-150 微秒，HDD 為 1-10 毫秒。影片可能強調這一點，因為它與頻道對系統設計基礎的專注點一致。

| 方面 | 詳細信息 |
|---|---|
| 存儲介質 | RAM（內存） |
| 訪問時間 | ~100-120 納秒 |
| 與磁碟比較 | 1000 倍快於隨機磁碟訪問 |
| 對性能的影響 | 降低延遲，增加吞吐量 |

2. **IO 多工和單線程執行循環** IO 多工允許單線程使用系統調用（如 `select`、`poll`、`epoll`（Linux）、`kqueue`（Mac OS）或 `evport`（Solaris））同時監控多個 I/O 流。這對於在不阻塞的情況下處理多個客戶端連接至關重要，這一點可能在影片中詳細說明。單線程執行循環避免了上下文切換和同步開銷，簡化了開發和調試。

| 機制 | 描述 |
|---|---|
| epoll/kqueue | 高並發，非阻塞，高效 |
| select/poll | 更舊，可擴展性較差，O(n) 復雜度 |
| 影響 | 降低連接開銷，啟用管道 |

   然而，客戶端阻塞命令（如 `BLPOP` 或 `BRPOP`）可能會延遲流量，這是相關文章中提到的潛在缺點。影片可能討論這種設計選擇如何在簡單性和性能之間取得平衡。

3. **高效的低層數據結構** Redis 利用哈希表進行 O(1) 鍵查找、鏈表進行列表和跳表進行排序集。這些數據結構優化了內存操作，最小化了內存使用，最大化了速度。影片可能包括圖表或示例，例如哈希表如何實現快速鍵值操作，這是系統設計面試中的常見話題。

| 數據結構 | 用例 | 時間複雜度 |
|---|---|---|
| 哈希表 | 鍵值存儲 | O(1) 平均 |
| 鏈表 | 列表，兩端高效 | O(1) 兩端 |
| 跳表 | 排序集，有序存儲 | O(log n) |

   這種優化至關重要，因為大多數 Redis 操作都是基於內存的，瓶頸通常在內存或網絡，而不是 CPU。

**額外考量和演變**　儘管核心請求處理是單線程的，但最近的 Redis 版本為特定任務引入了多線程。自 Redis 4.0 以來，實現了非同步內存釋放（懶惰刪除），自 6.0 以來，在高並發下為協議解析添加了多線程。這些變化可能在影片中提到，進一步提高了性能，而不改變主操作的單線程模型。

對於超出單個實例的擴展，Redis 支持集群和運行多個實例，這可能會討論以應對高並發需求。這是系統設計的重要方面，與頻道對大規模系統的專注點一致。

**潛在缺點和比較**　單線程模型有沒有鎖爭用和更簡單的調試的優點，但在高負載下可能面臨阻塞操作和內存/網絡瓶頸的挑戰。相關文章建議，對於 CPU 密集型任務，多線程數據庫可能表現更好，但對於 Redis 的典型用例，單線程設計是最佳的。

**結論**　ByteByteGo 的影片「系統設計：為什麼單線程 Redis 如此快？」可能涵蓋內存存儲、IO 多工和高效數據結構作為 Redis 速度的關鍵原因。這些因素使其能夠處理高 QPS，最近的版本為特定優化添加了多線程。這種分析提供了全面的理解，適合技術學習者和系統設計專業人士。

---

**關鍵引用**

- 為什麼 Redis 如此快的博客文章
- 為什麼 Redis 如此快，儘管它是單線程的文章
- Redis 線程模型面試文章
- 為什麼單線程 Redis 如此快的文章