Redis 簡介

Redis 是一款開源、高性能、支持可靠持久化的內存數據庫。

它可以用作分佈式緩存、消息隊列、網站訪問計數器等,廣泛應用於高流量的互聯網應用中。

Redis 提供了多種數據類型,操作簡單易上手,是許多開發者的首選數據庫之一。

Redis 的出現背景

Redis 誕生於 2009 年,當時軟件工程師 Salvatore Sanfilippo 為了解決高流量網站的性能瓶頸而開發了這款開源內存數據庫。

Redis 憑藉其出色的讀寫性能、靈活的數據結構和豐富的功能特性,迅速成為解決此類問題的首選。

Redis 想解決的問題

隨著互聯網的高速發展,網站和應用程式對資料庫的讀寫效能提出了更高的要求。傳統關係型資料庫在處理高並發、大數據量的情況下,存在效能瓶頸。**Redis**的出現,填補了這一空白,為開發者提供了一個快速、靈活的**高效能內存資料庫**。

與傳統資料庫相比,**Redis**擁有更快的讀寫效能、更豐富的資料類型,能夠更好地滿足網站和應用程式對即時性、高可用性的需求。同時,**Redis**還支援持久化機制,確保資料安全性,是當前高流量網站的首選資料庫之一。

Redis 的應用場景

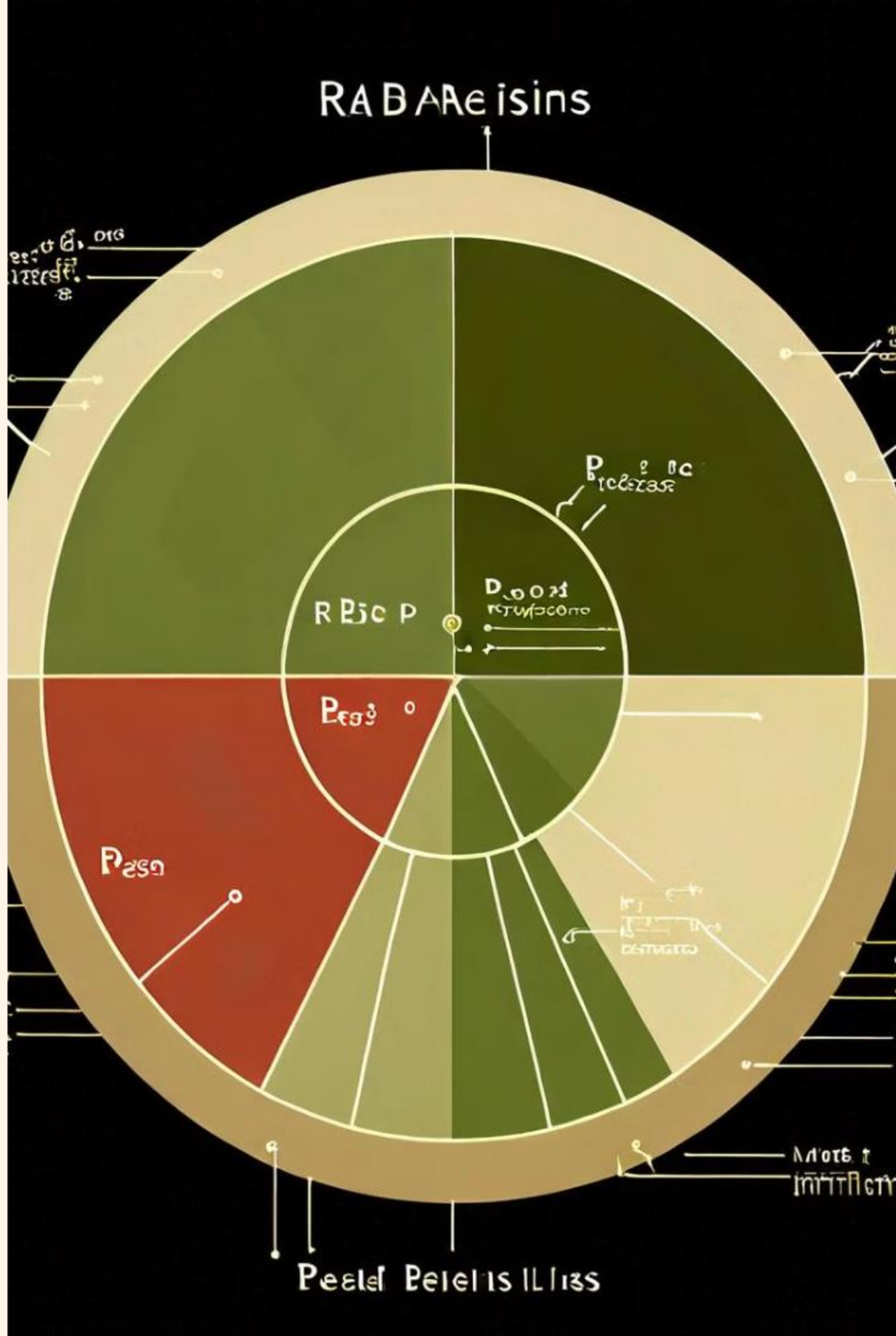
1. 作為高性能的緩存系統:Redis 優異的讀寫性能使其能夠快速緩存網站或應用的熱點數據,大幅提升系統響應速度。
2. 實現消息隊列系統:Redis 的發布/訂閱功能可以輕鬆實現高性能的消息隊列系統,用於解耦應用程序的異步處理。
3. 支持分布式計數器:Redis 內置的原子操作可以實現高並發的分布式計數器,適用於網站訪問統計、電商下單數量等場景。

Redis 的持久化(RDB & AOF)

Redis 支持兩種持久化機制:RDB(Redis Database)和 AOF(Append-Only File)。

RDB 是定期將內存中的數據快照保存到磁盤，而 AOF 則記錄每一個寫操作，在重啟時可以重放這些命令來恢復數據。

通過靈活配置這兩種方式，可以滿足不同應用對數據持久性和性能的需求。



定時刪除、惰性刪除、內存淘汰

1. 定時刪除（**Time-To-Live, TTL**）：Redis允許為每個key設置一個過期時間,到達該時間後系統會自動刪除該鍵值對。這是一種主動式的內存管理策略,可以有效控制Redis的內存佔用。
2. 惰性刪除（**Lazy Eviction**）：與定時刪除不同,惰性刪除是在訪問某個key時,才檢查該key是否已過期,如果過期則將其刪除。這種方式可以減輕系統的CPU負擔,但可能會導致過期數據暫時滯留在內存中。
3. 內存淘汰（**Memory Eviction**）：當Redis的內存使用達到事先設定的上限時,會觸發內存淘汰機制,根據設定的策略(如LRU、LFU等)自動刪除一些不常

緩存穿透、緩存擊穿、緩存雪崩

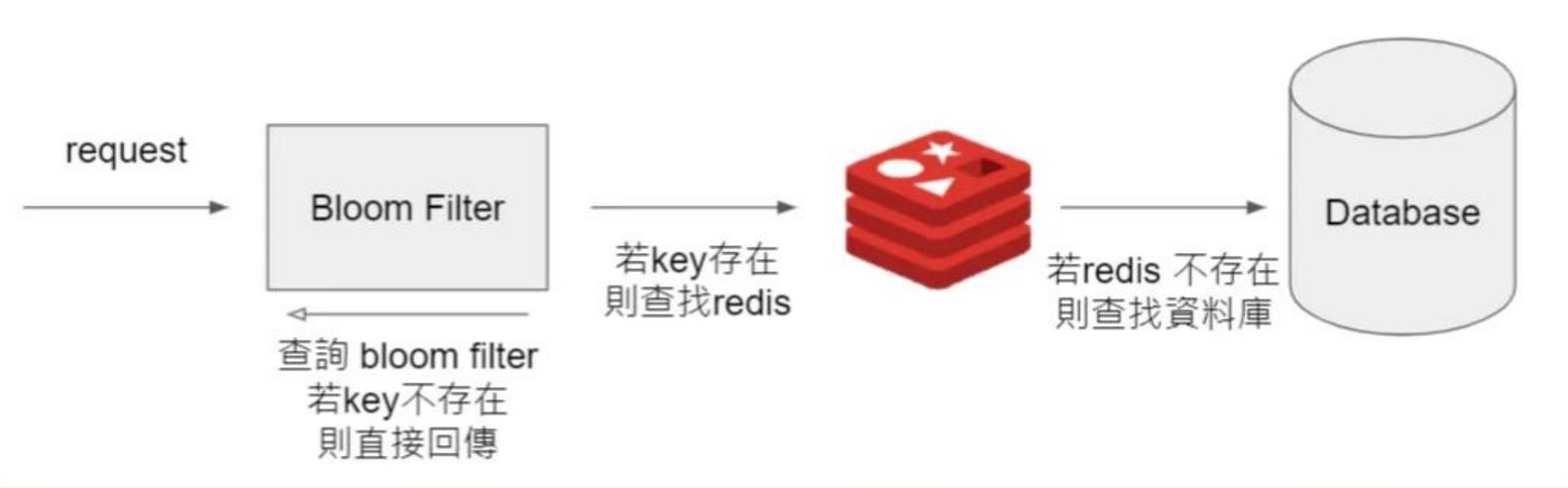
緩存穿透是指查詢一個一定不存在的數據，導致請求直接到達數據庫，無法被緩存。

而緩存擊穿是指熱點資料失效時，大量並發訪問會導致瞬間高峰流量打到數據庫。

緩存雪崩則是由於大量緩存同時失效，導致數據庫負載激增的情況。

緩存穿透

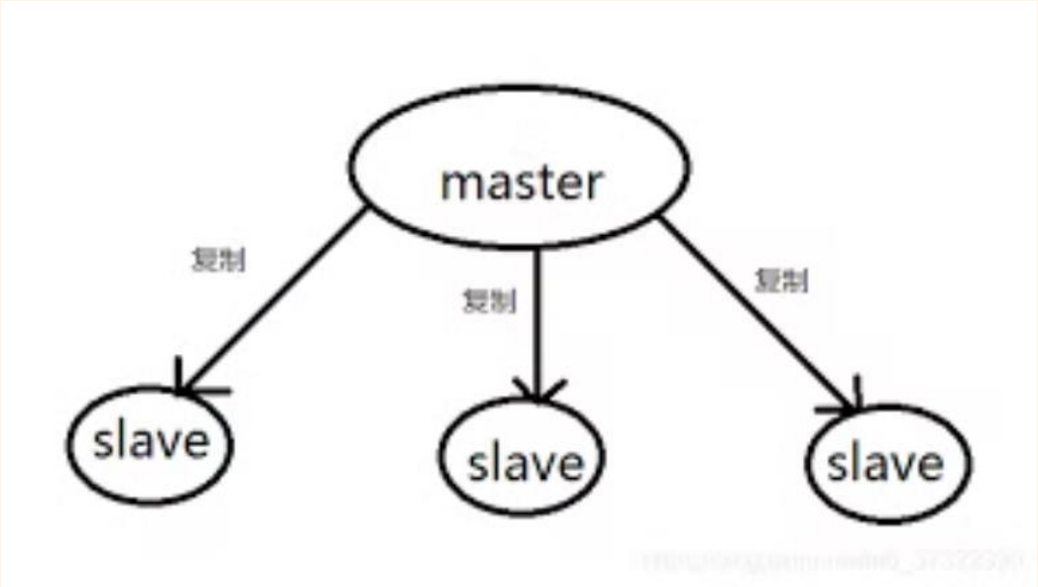
- 緩存穿透指的是客戶端請求一個一定不存在的數據，這類請求會直接到達後端數據庫，造成不必要的負擔。
- 這種情況下，即使有再強大的緩存也無法達到緩解效果。**常見的解決方案包括：**
- **使用布隆過濾器：**將所有可能的查詢鍵事先存儲到布隆過濾器中，當判斷一個key不在過濾器中時，直接返回不存在的結果，避免對數據庫的查詢。



Redis 的可靠問題(主從模式)

在主從模式中，主服務器負責寫操作並將數據同步到從服務器。但是一旦主服務器出現故障，從服務器無法自動接管讀請求，會導致整個系統無法服務。這種單點故障問題需要額外的機制來應對。

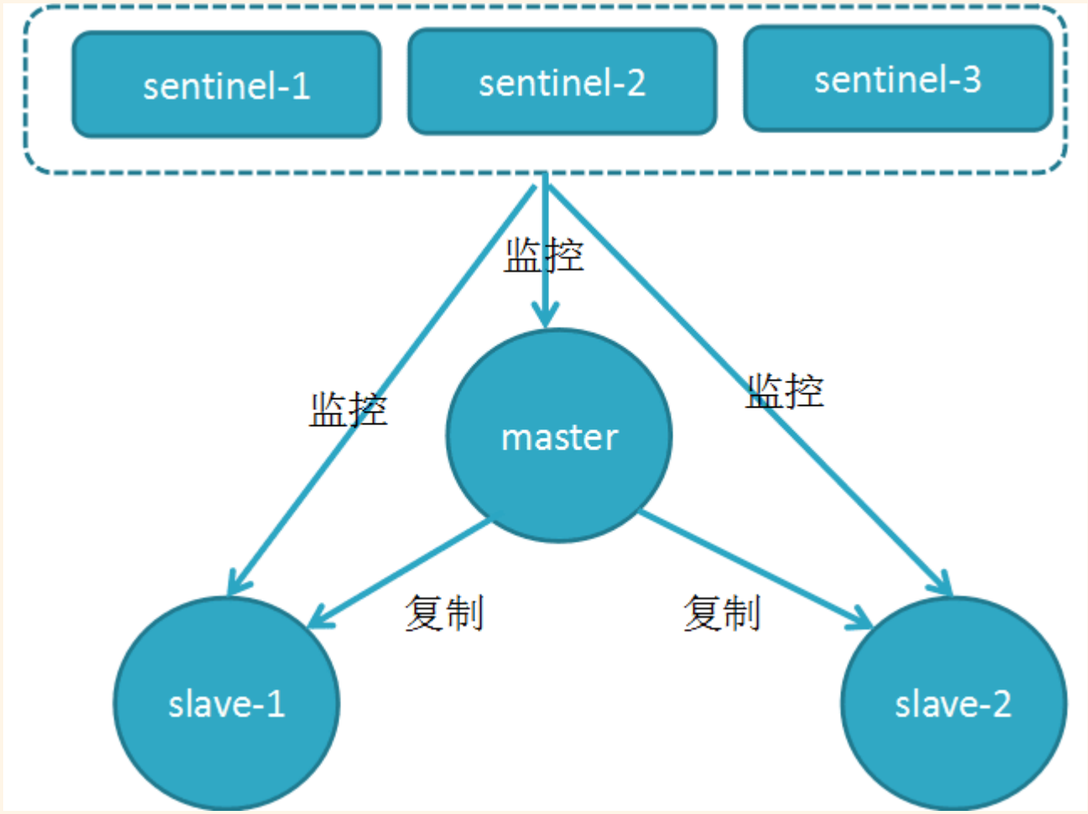
此外，主從模式也存在數據一致性問題。在主從切換過程中可能會出現短暫的數據不一致，需要謹慎處理。



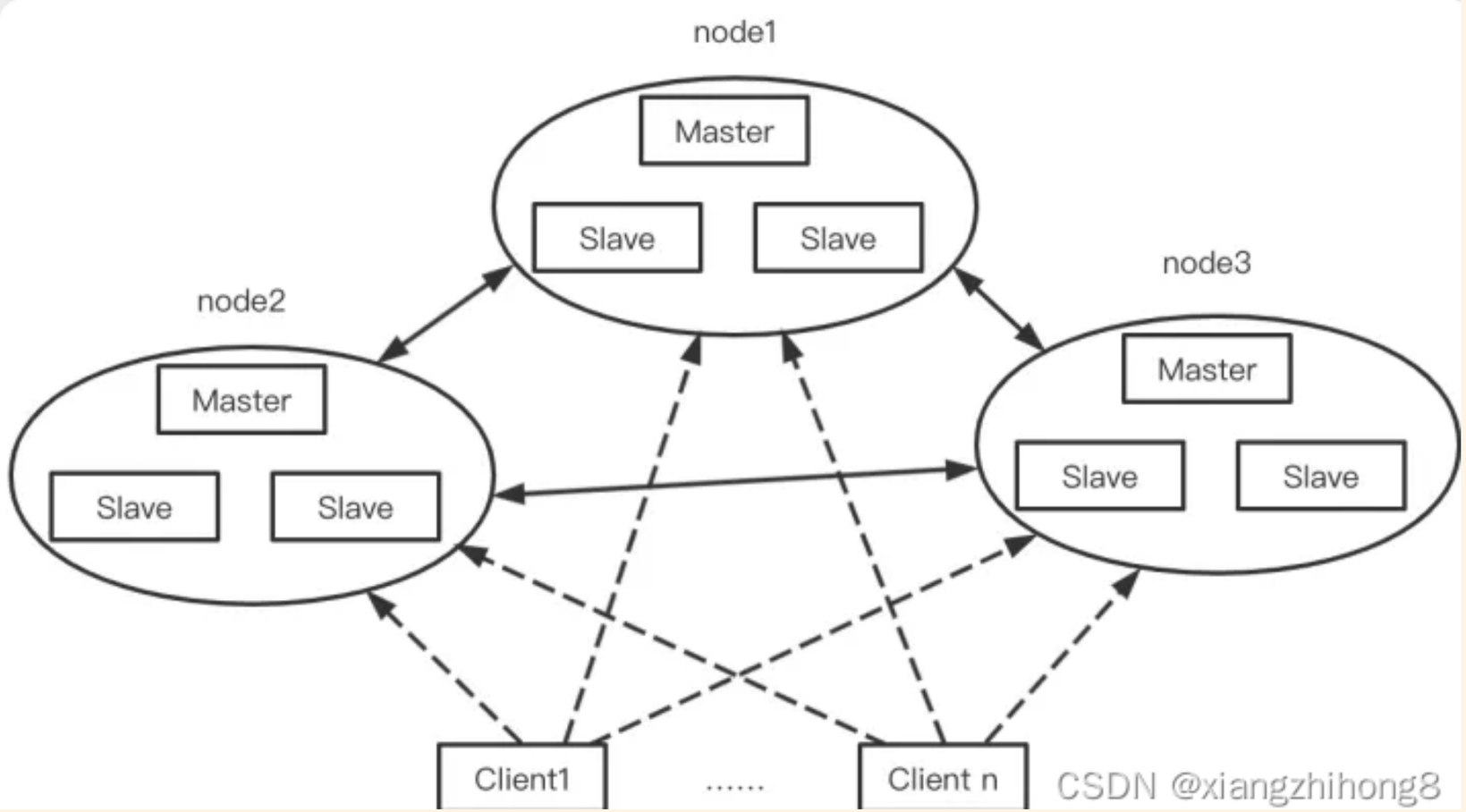
Redis 的可靠問題(哨兵模式)

Redis 哨兵模式是一種分散式的高可用性解決方案。它由一組 Redis 節點組成,負責監控主節點和從節點,並在主節點發生故障時自動進行故障轉移,確保系統的高可用性。

當發現主節點故障時,哨兵節點會自動從從節點中**選舉出新的主節點**，重新建立主從複製關係。

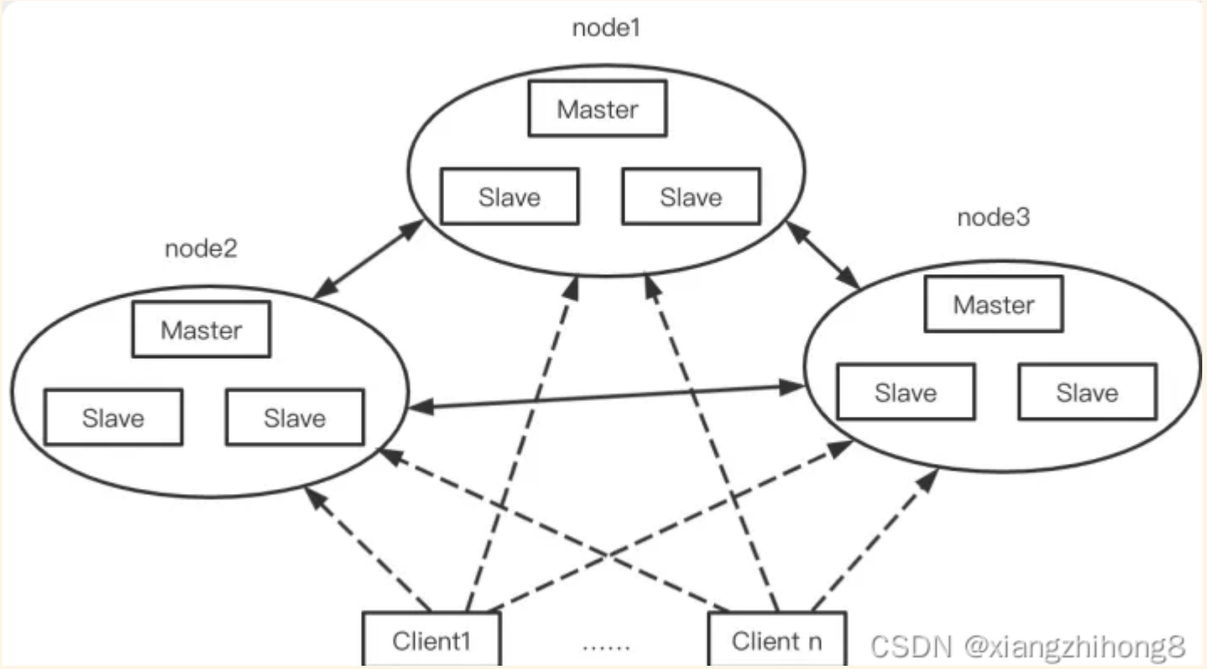


Redis 的可靠問題(集群模式)



Redis 的可靠問題(集群模式)

分片集群主要解決的是，海量資料存儲的問題，集群中有多個master，每個master保存不同資料，並且還可以給每個master設置多個slave節點，就可以繼續增大集群的高併發能力



Consistent Hashing(一致性哈希)

3個機器節點，10個資料 的雜湊值分別為1,2,3,4,...,10。使用的雜湊函數為：(m=hash(o) mod 3)

機器0 上保存的資料有：3，6，9

機器1 上保存的資料有：1，4，7，10

機器2 上保存的資料有：2，5，8

當增加一台機器後，此時n = 4，各個機器上存儲的資料分別為：

機器0 上保存的資料有：4，8

機器1 上保存的資料有：1，5，9

機器2 上保存的資料有：2，6，10

機器3 上保存的資料有：3，7