56 | 套路篇:優化性能問題的一般方法

JohnChen
2020/12/09

前言 1/3

從系統資源瓶頸的角度來說，USE 法是最為有效的方法，即從使用率、飽和度以及錯誤數這三個方面，來分析 CPU、內存、磁碟和文件系統 I/O、網路以及內核資源限制等各類軟硬件資源。

前言 2/3

從應用程序瓶頸的角度來說，可以把性能問題的來源，分為資源瓶頸、依賴服務瓶頸以及應用自身的瓶頸這三類。

資源瓶頸的分析思路，跟系統資源瓶頸是一樣的。

依賴服務的瓶頸，可以使用全鏈路跟蹤系統，進行快速定位。

而應用自身的問題，則可以通過系統調用、熱點函數，或者應用自身的指標和日志等，進行分析定位。

前言 3/3

當然，雖然系統和應用是兩個不同的角度，但在實際運行時，它們往往相輔相成、相互影響。

系統是應用的運行環境，系統瓶頸會導致應用的性能下降。

而應用程序不合理的設計，也會引發系統資源的瓶頸。

我們做性能分析，就是要結合應用程序和操作系統的原理，揪出引發問題的“真兇“。

系統優化

1. CPU 優化
2. 內存優化
3. 磁碟和文件系統 I/O 優化
4. 網路優化

CPU 優化

首先來看 CPU 性能的優化方法。在CPU 性能優化的幾個思路中，我曾經介紹過，CPU 性能優化的核心，在於排除所有不必要的工作、充分利用 CPU 緩存並減少進程調度對性能的影響。

第一種，把進程綁定到一個或者多個 CPU 上，充分利用 CPU 緩存的本地性，並減少進程間的相互影響。

第二種，為中斷處理程序開啟多 CPU 負載均衡，以便在發生大量中斷時，可以充分利用多 CPU 的優勢分攤負載。

第三種，使用 Cgroups 等方法，為進程設置資源限制，避免個別進程消耗過多的 CPU。同時，為核心應用程序設置更高的優先級，減少低優先級任務的影響。

內存優化

第一種，除非有必要，Swap 應該禁止掉。這樣就可以避免 Swap 的額外 I/O ，帶來內存訪問變慢的問題。

第二種，使用 Cgroups 等方法，為進程設置內存限制。這樣就可以避免個別進程消耗過多內存，而影響了其他進程。對於核心應用，還應該降低 oom_score，避免被 OOM 殺死。

第三種，使用大頁、內存池等方法，減少內存的動態分配，從而減少缺頁異常。

磁碟和文件系統 I/O 優化

第一種，也是最簡單的方法，通過 SSD 替代 HDD、或者使用 RAID 等方法，提升 I/O 性能。

第二種，針對磁碟和應用程序 I/O 模式的特征，選擇最適合的 I/O 調度算法。比如，SSD 和虛擬機中的磁碟，通常用的是 noop 調度算法；而數據庫應用，更推薦使用 deadline 算法。

第三，優化文件系統和磁碟的緩存、緩沖區，比如優化髒頁的刷新頻率、髒頁限額，以及內核回收目錄項緩存和索引節點緩存的傾向等等。

除此之外，使用不同磁碟隔離不同應用的數據、優化文件系統的配置選項、優化磁碟預讀、增大磁碟隊列長度等，也都是常用的優化思路。

網路優化

首先，從內核資源和網路協議的角度來說，我們可以對內核選項進行優化，比如：

你可以增大套接字緩沖區、連接跟蹤表、最大半連接數、最大文件描述符數、本地端口範圍等內核資源配額；

也可以減少 TIMEOUT 超時時間、SYN+ACK 重傳數、Keepalive 探測時間等異常處理參數；

還可以開啟端口覆用、反向地址校驗，並調整 MTU 大小等降低內核的負擔。

網路優化 1/2

其次，從網路接口的角度來說，我們可以考慮對網路接口的功能進行優化，比如

你可以將原來 CPU 上執行的工作，卸載到網卡中執行，即開啟網卡的 GRO、GSO、RSS、VXLAN 等卸載功能；

也可以開啟網路接口的多隊列功能，這樣，每個隊列就可以用不同的中斷號，調度到不同 CPU 上執行；

還可以增大網路接口的緩沖區大小以及隊列長度等，提升網路傳輸的吞吐量。

網路優化 2/2

最後，在極限性能情況（比如 C10M）下，內核的網路協議棧可能是最主要的性能瓶頸，所以，一般會考慮繞過內核協議棧。

你可以使用 DPDK 技術，跳過內核協議棧，直接由用戶態進程用輪詢的方式，來處理網路請求。同時，再結合大頁、CPU 綁定、內存對齊、流水線並發等多種機制，優化網路包的處理效率。

你還可以使用內核自帶的 XDP 技術，在網路包進入內核協議棧前，就對其進行處理。這樣，也可以達到目的，獲得很好的性能。

應用程序優化 1/4

雖然系統的軟硬件資源，是保證應用程序正常運行的基礎，但你要知道，性能優化的最佳位置，還是應用程序內部。為什麼這麼說呢？我簡單舉兩個例子你就明白了。

第一個例子，是系統 CPU 使用率（sys%）過高的問題。有時候出現問題，雖然表面現象是系統 CPU 使用率過高，但待你分析過後，很可能會發現，應用程序的不合理系統調用才是罪魁禍首。這種情況下，優化應用程序內部系統調用的邏輯，顯然要比優化內核要簡單也有用得多。

應用程序優化 2/4

再比如說，數據庫的 CPU 使用率高、I/O 響應慢，也是最常見的一種性能問題。這種問題，一般來說，並不是因為數據庫本身性能不好，而是應用程序不合理的表結構或者 SQL 查詢語句導致的。這時候，優化應用程序中數據庫表結構的邏輯或者 SQL 語句，顯然要比優化數據庫本身，能帶來更大的收益。

所以，在觀察性能指標時，你應該先查看應用程序的響應時間、吞吐量以及錯誤率等指標，因為它們才是性能優化要解決的終極問題。以終為始，從這些角度出發，你一定能想到很多優化方法，而我比較推薦下面幾種方法。

應用程序優化 3/4

第一，從 CPU 使用的角度來說，簡化代碼、優化算法、異步處理以及編譯器優化等，都是常用的降低 CPU 使用率的方法，這樣可以利用有限的 CPU 處理更多的請求。

第二，從數據訪問的角度來說，使用緩存、寫時覆制、增加 I/O 尺寸等，都是常用的減少磁碟 I/O 的方法，這樣可以獲得更快的數據處理速度。

第三，從內存管理的角度來說，使用大頁、內存池等方法，可以預先分配內存，減少內存的動態分配，從而更好地內存訪問性能。

應用程序優化 4/4

第四，從網路的角度來說，使用 I/O 多路覆用、長連接代替短連接、DNS 緩存等方法，可以優化網路 I/O 並減少網路請求數，從而減少網路延時帶來的性能問題。

第五，從進程的工作模型來說，異步處理、多線程或多進程等，可以充分利用每一個 CPU 的處理能力，從而提高應用程序的吞吐能力。

除此之外，你還可以使用消息隊列、CDN、負載均衡等各種方法，來優化應用程序的架構，將原來單機要承擔的任務，調度到多台服務器中並行處理。這樣也往往能獲得更好的整體性能。

小結 1/2

從系統的角度來說，CPU、內存、磁碟和文件系統 I/O、網路以及內核數據結構等各類軟硬件資源，為應用程序提供了運行的環境，也是我們性能優化的重點對象。你可以參考咱們專欄前面四個模塊的優化篇，優化這些資源。

從應用程序的角度來說，降低 CPU 使用，減少數據訪問和網路 I/O，使用緩存、異步處理以及多進程多線程等，都是常用的性能優化方法。除了這些單機優化方法，調整應用程序的架構，或是利用水平擴展，將任務調度到多台服務器中並行處理，也是常用的優化思路。

小結 2/2

雖然性能優化的方法很多，不過，我還是那句話，一定要避免過早優化。性能優化往往會提高複雜性，這一方面降低了可維護性，另一方面也為適應複雜多變的新需求帶來障礙。

所以，性能優化最好是逐步完善，動態進行；不追求一步到位，而要首先保證，能滿足當前的性能要求。發現性能不滿足要求或者出現性能瓶頸後，再根據性能分析的結果，選擇最重要的性能問題進行優化。

思考

最後，我想邀請你一起來聊聊，當碰到性能問題後，你是怎麼進行優化的？有沒有哪個印象深刻的經歷可以跟我分享呢？你可以結合我的講述，總結自己的思路。