20 | 案例篇:为什么系统的Swap变高了？（下）

JohnChen

在內存資源不足時, Linux 通過直接內存回收和定期掃描的方式, 來釋放文件頁和匿名頁, 以便把內存分配給更需要的進程使用。

文件頁的回收比較容易理解, 直接清空緩存, 或者把髒數據寫回磁盤後, 再釋放緩存就可以了。

而對不常訪問的匿名頁，則需要通過 Swap 換出到磁盤中，這樣在下次訪問的時候，再次從磁盤換入到內存中就可以了。

開啟 Swap 後，你可以設置 /proc/sys/vm/min_free_kbytes，來調整系統定期回收內存的閾值，也可以設置 /proc/sys/vm/swappiness，來調整文件頁和匿名頁的回收傾向。

那麼，當 Swap 使用升高時，要如何定位和分析呢？下面，我們就來看一個磁盤 I/O 的案例，實戰分析和演練。

1. Ubuntu 18.04
2. 機器配置：2 CPU，8GB 內存
3. 需要預先安裝 sysstat 等工具，如 apt install sysstat

在終端中運行 free 命令，查看 Swap 的使用情況。比如，在我的機器中，輸出如下

```
root@ip-172-31-88-164:~# free
               total        used        free      shared  buff/cache   available
Mem:         7850276      146076     7249532         736      454668     7465136
Swap:              0           0           0
```

從這個 free 輸出你可以看到，Swap 的大小是 0，這說明我的機器沒有配置 Swap。

Linux 本身支持兩種類型的 Swap, 即 Swap 分區和 Swap 文件。以 Swap 文件為例, 在第一個終端中運行下面的命令開啟 Swap, 我這里配置 Swap 文件的大小為 8GB。

然後, 再執行 free 命令, 確認 Swap 配置成功

```
root@ip-172-31-95-99:~# dd if=/dev/zero of=/mnt/swapfile bs=1024 count=8000000
8000000+0 records in
8000000+0 records out
8192000000 bytes (8.2 GB, 7.6 GiB) copied, 23.1051 s, 355 MB/s
root@ip-172-31-95-99:~# chmod 600 /mnt/swapfile
root@ip-172-31-95-99:~# mkswap /mnt/swapfile
Setting up swapspace version 1, size = 7.6 GiB (8191995904 bytes)
no label, UUID=85ab6ad0-a937-42db-9eb9-01dc2835f670
root@ip-172-31-95-99:~# swapon /mnt/swapfile
root@ip-172-31-95-99:~# free
               total        used        free      shared  buff/cache   available
Mem:       15950164      262240     6262460         784     9425464    15352180
Swap:       7999996           0     7999996
```

$ dd if=/dev/nvme0n1p1 of=/dev/null bs=1G count=2048
$ sar -r -S 1

```
08:02:25    kbmemfree   kbavail kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
08:02:26       155556  14373184  15794608     99.02  14183268    230476   1565080      6.43   1000624  14520672     37720

08:02:25    kbswpfree kbswpused  %swpused  kbswpcad   %swpcad
08:02:26      8388092       512      0.01         8      1.56

08:02:26    kbmemfree   kbavail kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
08:02:27       176264  14373236  15773900     98.89  14162640    230496   1565080      6.43   1000308  14500332     37720

08:02:26    kbswpfree kbswpused  %swpused  kbswpcad   %swpcad
08:02:27      8388092       512      0.01         8      1.56

08:02:27    kbmemfree   kbavail kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
08:02:28       169692  14373272  15780472     98.94  14169384    230336   1565080      6.43   1000044  14507204     37720

08:02:27    kbswpfree kbswpused  %swpused  kbswpcad   %swpcad
08:02:28      8388092       512      0.01         8      1.56
```

kbcommit, 表示當前系統負載需要的內存。它實際上是為了保證系統內存不溢出, 對需要內存的估計值。%commit, 就是這個值相對總內存的百分比。

kbactive, 表示活躍內存, 也就是最近使用過的內存, 一般不會被系統回收。

kbinact, 表示非活躍內存, 也就是不常訪問的內存, 有可能會被系統回收。

清楚了界面指標的含義後，我們再結合具體數值，來分析相關的現象。你可以清楚地看到，總的內存使用率（%memused）在不斷增長，從開始的 23% 一直長到了 98%，並且主要內存都被緩沖區（kbbuffers）占用。具體來說：

剛開始, 剩余內存(kbmemfree)不斷減少, 而緩沖區(kbbuffers)則不斷增大, 由此可知, 剩余內存不斷分配給了緩沖區。

一段時間後, 剩余內存已經很小, 而緩沖區占用了大部分內存。這時候, Swap 的使用開始逐漸增大, 緩沖區和剩余內存則只在小範圍內波動。

你可能困惑了，為什麼緩沖區在不停增大？這又是哪些進程導致的呢？

顯然，我們還得看看進程緩存的情況。在前面緩存的案例中我們學過，cachetop 正好能滿足這一點。那我們就來 cachetop 一下。

$cachetop 5

```
07:57:35 Buffers MB: 4591 / Cached MB: 640 / Sort: HITS / Order: ascending
PID      UID      CMD              HITS     MISSES   DIRTIES  READ_HIT%  WRITE_HIT%
   2999 root     cachetop            2          0          0     100.0%        0.0%
   3111 root     dd             164218     164202          0      50.0%       50.0%
```

使用 cachestat 和 cachetop 前，我們首先要安裝bcc 軟件包。比如，在Ubuntu 系統中，你可以運行下面的命令來安裝：

$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 4052245BD4284CDD

$ echo "deb https://repo.iovisor.org/apt/xenial xenial main" | sudo tee /etc/apt/sources.list.d/iovisor.list

$ sudo apt-get update

$ sudo apt-get install -y bcc-tools libbcc-examples linux-headers-$(uname -r)

$ export PATH=$PATH:/usr/share/bcc/tools

通過 cachetop 的輸出，我們看到，dd 進程的讀寫請求只有 50% 的命中率，並且未命中的緩存頁數（MISSES）為 64192（單位是頁）。這說明，正是案例開始時運行的 dd，導致了緩沖區使用升高。

你可能接著會問, 為什麼 Swap 也跟著升高了呢？直觀來說, 緩沖區占了系統絕大部分內存, 還屬於可回收內存, 內存不夠用時, 不應該先回收緩沖區嗎？

這種情況, 我們還得進一步通過 /proc/zoneinfo , 觀察剩余內存、內存閾值以及匿名頁和文件頁的活躍情況。

```
Every 2.0s: grep -A 15 Normal /proc/zoneinfo

Node 0, zone   Normal
  pages free     19419
        min      13649
        low      17061
        high     20473
        spanned  3287552
        present  3287552
        managed  3217627
        protection: (0, 0, 0, 0, 0)
      nr_free_pages 19419
      nr_zone_inactive_anon 18611
      nr_zone_active_anon 258211
      nr_zone_inactive_file 2838295
      nr_zone_active_file 20986
      nr_zone_unevictable 0
      nr_zone_write_pending 0
```

你可以发現, 剩余內存(pages_free)在一個小範圍內不停地波動。當它小於頁低閾值(pages_low) 時, 又會突然增大到一個大於頁高閾值(pages_high)的值。

再結合剛剛用 sar 看到的剩余內存和緩沖區的變化情況, 我們可以推導出, 剩余內存和緩沖區的波動變化, 正是由於內存回收和緩存再次分配的循環往覆。

當剩余內存小於頁低閾值時, 系統會回收一些緩存和匿名內存, 使剩余內存增大。其中, 緩存的回收導致 sar 中的緩沖區減小, 而匿名內存的回收導致了 Swap 的使用增大。

緊接著, 由於 dd 還在繼續, 剩余內存又會重新分配給緩存, 導致剩余內存減少, 緩沖區增大。

其實還有一個有趣的現象, 如果多次運行 dd 和 sar, 你可能會发現, 在多次的循環重覆中, 有時候是 Swap 用得比較多, 有時候 Swap 很少, 反而緩沖區的波動更大。

換句話說, 系統回收內存時, 有時候會回收更多的文件頁, 有時候又回收了更多的匿名頁。

顯然, 系統回收不同類型內存的傾向, 似乎不那麼明顯。你應該想到了上節課提到的 swappiness, 正是調整不同類型內存回收的配置選項。

```
root@ip-172-31-95-99:~# cat /proc/sys/vm/swappiness
60
```

swappiness 顯示的是默認值 60, 這是一個相對中和的配置, 所以系統會根據實際運行情況, 選擇合適的回收類型, 比如回收不活躍的匿名頁, 或者不活躍的文件頁。

到這里, 我們已經找出了 Swap 发生的根源。另一個問題就是, 剛才的 Swap 到底影響了哪些應用程序呢？換句話說, Swap 換出的是哪些進程的內存？

這里我還是推薦 proc 文件系統, 用來查看進程 Swap 換出的虛擬內存大小, 它保存在 /proc/pid/status 中的 VmSwap 中(推薦你執行 man proc 來查詢其他字段的含義)。

按 VmSwap 使用量對進程排序, 輸出進程名稱、進程 ID 以及 SWAP 用量

```
root@ip-172-31-95-99:~# for file in /proc/*/status ; do awk '/VmSwap|Name|^Pid/{printf $2 " " $3}END{ print ""}' $file; done | sort
 -k 3 -n -r | head -10
snapd 1397 1056 kB
unattended-upgr 968 12 kB
networkd-dispat 859 8 kB
(sd-pam) 1859 4 kB
writeback 64
watchdogd 75
watchdog/7 50
watchdog/6 44
watchdog/5 38
watchdog/4 32
```

關閉 swap

```
root@ip-172-31-95-99:~# swapoff -a
root@ip-172-31-95-99:~# free
              total        used        free      shared  buff/cache   available
Mem:       15950164      213756     1191008         760    14545400    15427876
Swap:             0           0           0
```

在內存資源緊張時, Linux 會通過 Swap , 把不常訪問的匿名頁換出到磁盤中, 下次訪問的時候再從磁盤換入到內存中來。你可以設置 /proc/sys/vm/min_free_kbytes, 來調整系統定期回收內存的閾值;也可以設置 /proc/sys/vm/swappiness, 來調整文件頁和匿名頁的回收傾向。

當 Swap 變高時，你可以用 sar、/proc/zoneinfo、/proc/pid/status 等方法，查看系統和進程的內存使用情況，進而找出 Swap 升高的根源和受影響的進程。

反過來說，通常，降低 Swap 的使用，可以提高系統的整體性能。要怎麼做呢？這里，我也總結了幾種常見的降低方法。

禁止 Swap，現在服務器的內存足夠大，所以除非有必要，禁用 Swap 就可以了。隨著雲計算的普及，大部分雲平台中的虛擬機都默認禁止 Swap。

如果實在需要用到 Swap，可以嘗試降低 swappiness 的值，減少內存回收時 Swap 的使用傾向。

響應延遲敏感的應用，如果它們可能在開啟 Swap 的服務器中運行，你還可以用庫函數 mlock() 或者 mlockall() 鎖定內存，阻止它們的內存換出。

今天的案例中, swappiness 使用的是默認配置的 60。如果把它配置成 0 的話, 還會发生 Swap 嗎？這又是為什麼呢？

swappiness 的範圍是 0-100, 數值越大, 越積極使用 Swap, 也就是更傾向於回收匿名頁;數值越小, 越消極使用 Swap, 也就是更傾向於回收文件頁。

雖然 swappiness 的範圍是 0-100, 不過要注意, 這並不是內存的百分比, 而是調整 Swap 積極程度的權重, 即使你把它設置成 0, 當剩余內存 + 文件頁小於頁高閾值時, 還是會发生 Swap。

希望你可以實際操作一下，重點觀察 sar 的輸出，並結合今天的內容來記錄、總結。

END