Управление памятью в ОС Linux

Ерёмин Владимир

3 декабря 2023 г.

1 Конфигурация

- Общий объем оперативной памяти: **7810200** KiB
- Объем раздела подкачки: **2097152** KiB
- Размер страницы вирт. памяти: **4096**
- Объем свободной физ. памяти в ненагруж. системе: **6728628** KiB
- Объем свободного пр-ва в разделе подкачки в ненагруж. системе: **2097152** KiB

```
$ free --kibi
               total        used        free      shared  buff/cache   available
Mem:         7810200      554736     6728628        3312      526836     7019820
Swap:        2097152           0     2097152

$ getconf PAGE_SIZE
4096
```

2 Эксперимент №1

2.1 Первый этап

Макс. размер массива: **118000000**

2.1.1 Сообщение

```
$ dmesg

[ 5333.821710]
↪  oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,
↪  mems_allowed=0,
↪  global_oom,task_memcg=/,task=mem.bash,pid=5201,uid=1000
```

```
[ 5333.821738] Out of memory: Killed process 5201 (mem.bash)
↪  total-vm:9240292kB, anon-rss:7333220kB, file-rss:4kB, shmem-rss:0kB,
↪  UID:1000 pgtables:18116kB oom_score_adj:0
```

2.1.2 График

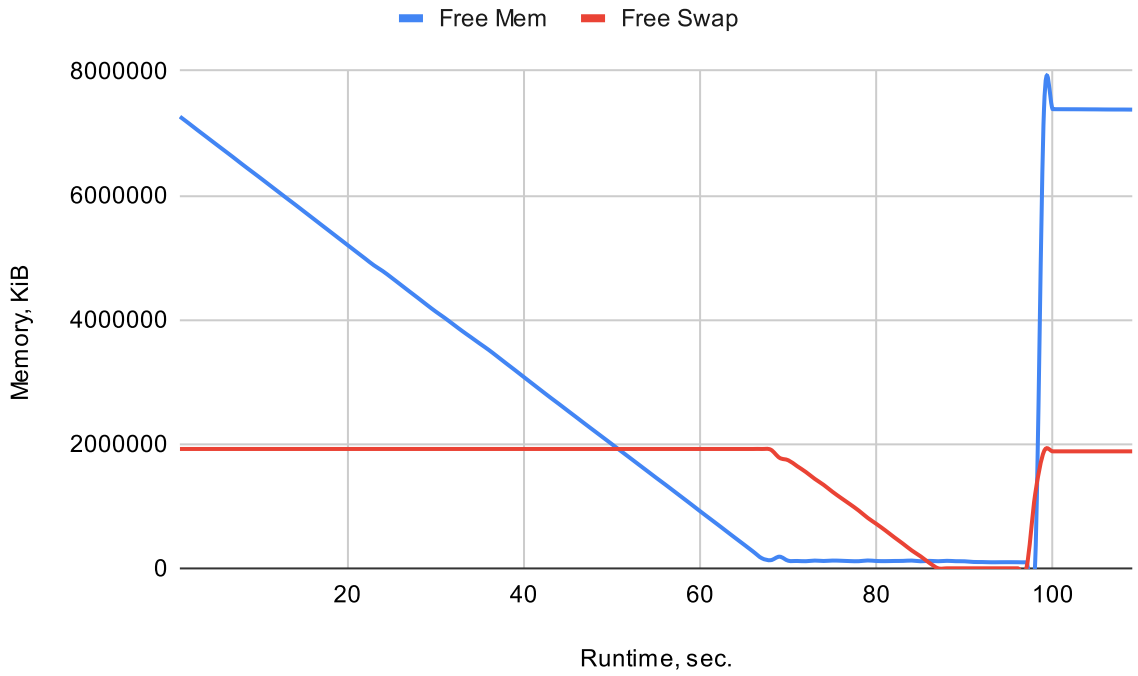

Рис. 1: Free Mem vs. Free Swap (single)

2.1.3 Вывод

График демонстрирует логичную последовательность событий: по истечении физической памяти активизировался раздел подкачки, что обеспечило продолжение выполнения процесса при использовании дополнительного объема памяти. По истечении доступного ресурса подкачки произошло аварийное завершение процесса из-за исчерпания оперативной памяти (OOM). Резкое освобождение заметного объема памяти отчетливо фиксируется в конечной части графика. Стоит отметить, что раздел подкачки не мгновенно возвращает освобожденную память, что отражено в небольшом скачке на графике физической памяти.

2.2 Второй этап

Макс. размер массива (база): **118000000** Макс. размер массива (копия): **60000000**

2.2.1 Сообщение

```
$ dmesg

[ 8353.913293]
↪  oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,
↪  mems_allowed=0, global_oom,task_memcg=/,
↪  task=mem-copy.bash,pid=6080,uid=1000
[ 8353.913421] Out of memory: Killed process 6080 (mem-copy.bash)
↪  total-vm:4703584kB, anon-rss:3676908kB, file-rss:0kB, shmem-rss:0kB,
↪  UID:1000 pgtables:9252kB oom_score_adj:0

[ 8448.533098]
↪  oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,
↪  mems_allowed=0,
↪  global_oom,task_memcg=/,task=mem.bash,pid=6087,uid=1000
[ 8448.533126] Out of memory: Killed process 6087 (mem.bash)
↪  total-vm:9226564kB, anon-rss:7349100kB, file-rss:0kB, shmem-rss:0kB,
↪  UID:1000 pgtables:18096kB oom_score_adj:0
```

2.2.2 График

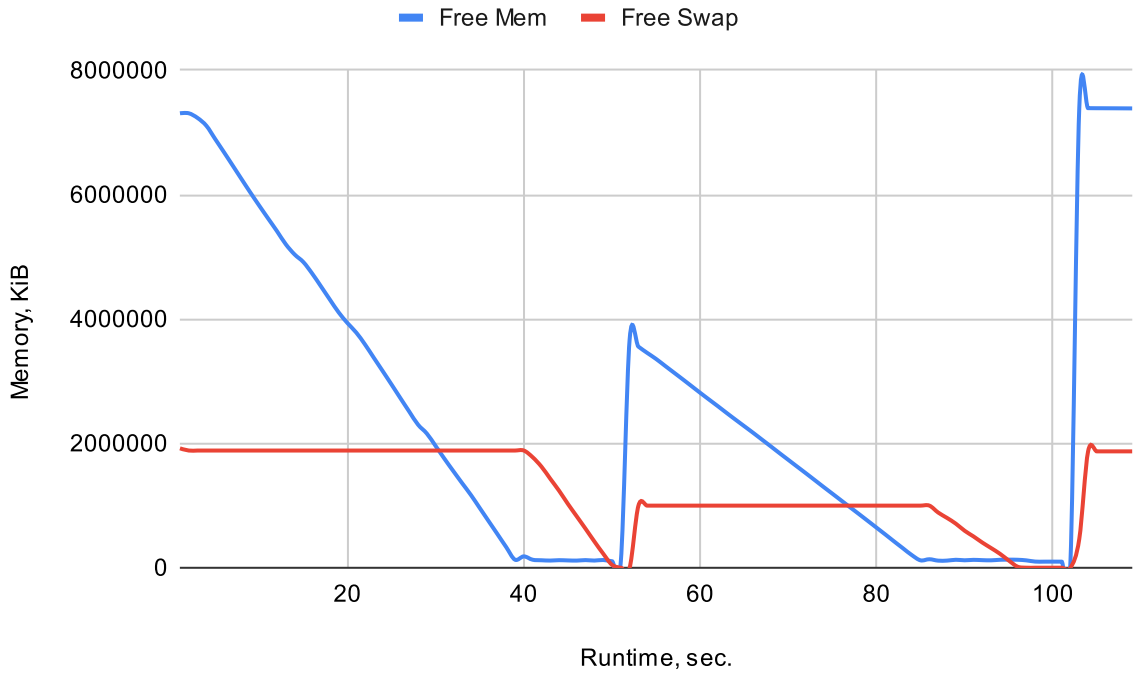

Рис. 2: Free Mem vs. Free Swap (parallel)

2.2.3 Вывод

В начале данного этапа наблюдается аналогия с первым этапом, однако это происходит быстрее. Это обусловлено наличием двух параллельно выполняющихся процессов, оба из которых требуют выделения памяти. Копия завершила свое выполнение досрочно, и этот факт привел к освобождению ресурсов памяти. Этот момент можно наблюдать на графике, где примерно половина общего объема памяти становится доступной после завершения копии. Далее процесс развивается согласно логике первого этапа.

3 Эксперимент №2

При $K = 10$ процессы не прервутся так, как запущеные $newmem.sh$ успевают завершится, высвобождая память для новых запусков - ромежуток в секунду на самом деле достаточен для этого. Однако и при $K = 30$ на моем устройстве процессы выполняются успешно. (даже физическая память не успевает заполнится). Думаю это связанно с железом на котором происходит тестирование. Поэтому эксперемент буду проводить при $K = 50$ - OOM убил 10 процессов.

С помощью метода бинарного поиска можно найти N при котором процессы завершатся успешно: **11662500**