Лабораторная работа №5
Операционные системы, Lite-поток
Подготовил Сергей Гусев, M3237

Эксперимент 1

Этап 1

```
getconf PAGE_SIZE = 4096
Текущий размер массива: 42000000
Системная память и подкачка:
               total        used        free      shared  buff/cache
available
Mem:           3,8Gi       3,8Gi       106Mi        21Mi        80Mi
20Mi
Swap:          1,0Gi       1,0Gi       140Ki
Верхние пять процессов:
    PID COMMAND         %MEM %CPU
  17521 bash            66.9 92.4
    621 gnome-shell      8.0 19.5
  15555 nautilus         6.6  8.1
    484 Xorg             1.3  2.1
  17786 gedit            0.5 12.1
Информация о текущем скрипте:
    PID COMMAND         %MEM %CPU
  17521 bash            66.9 92.2

[14050.884946] [  17521]  1000 17521   834931   684384  6737920   148736
200 bash

[14050.884950]
oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=
0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000.service/app.
slice/app-org.gnome.Terminal.slice/vte-spawn-232ac80f-51f3-4eba-b9b2-c0914
b27a587.scope,task=bash,pid=17521,uid=1000
[14050.884965] Out of memory: Killed process 17521 (bash)
total-vm:3339724kB, anon-rss:2737152kB, file-rss:384kB, shmem-rss:0kB,
UID:1000 pgtables:6580kB oom_score_adj:200
```

Страничная Организация Памяти и Раздел Подкачки

1. Страничная Организация Памяти: Современные операционные системы используют страничную организацию памяти для эффективного распределения и управления памятью. Это позволяет ОС загружать и выгружать страницы памяти между физической памятью (RAM) и разделом подкачки на диске.

2. Раздел Подкачки (Swap): Swap используется как расширение физической памяти, позволяя системе продолжать работать, когда физическая память полностью занята. Данные, которые редко используются, могут быть перемещены в swap, освобождая RAM для более важных задач.

В скрипте происходит непрерывное увеличение размера массива `arr`, что приводит к постоянному увеличению потребления памяти. По мере увеличения размера массива:

- Системная Память (RAM): Наблюдается увеличение использования RAM. Это происходит из-за того, что массив `arr` непрерывно растет, занимая всё больше и больше памяти.
- Swap-память: Изначально swap-память остается относительно стабильной, что указывает на то, что физическая память справляется с нагрузкой. Однако, когда физическая память приближается к своему пределу, операционная система начинает использовать swap, чтобы предотвратить сбои и обеспечить дополнительное пространство для памяти.

Пороговые Величины и Аварийная Остановка

Аварийная остановка произошла из-за исчерпания доступной физической и swap-памяти. Массив достиг размера, при котором операционная система не смогла выделить ему больше памяти. Это может произойти, когда:

1. Исчерпание RAM: Все доступные страницы памяти в RAM заняты.
2. Исчерпание Swap: Swap-память также заполнена, и ОС не может перенести данные из RAM в swap для освобождения места.

Эти пороговые значения зависят от общего объема физической памяти и swap-памяти, а также от других процессов, использующих память в системе.

Выводы

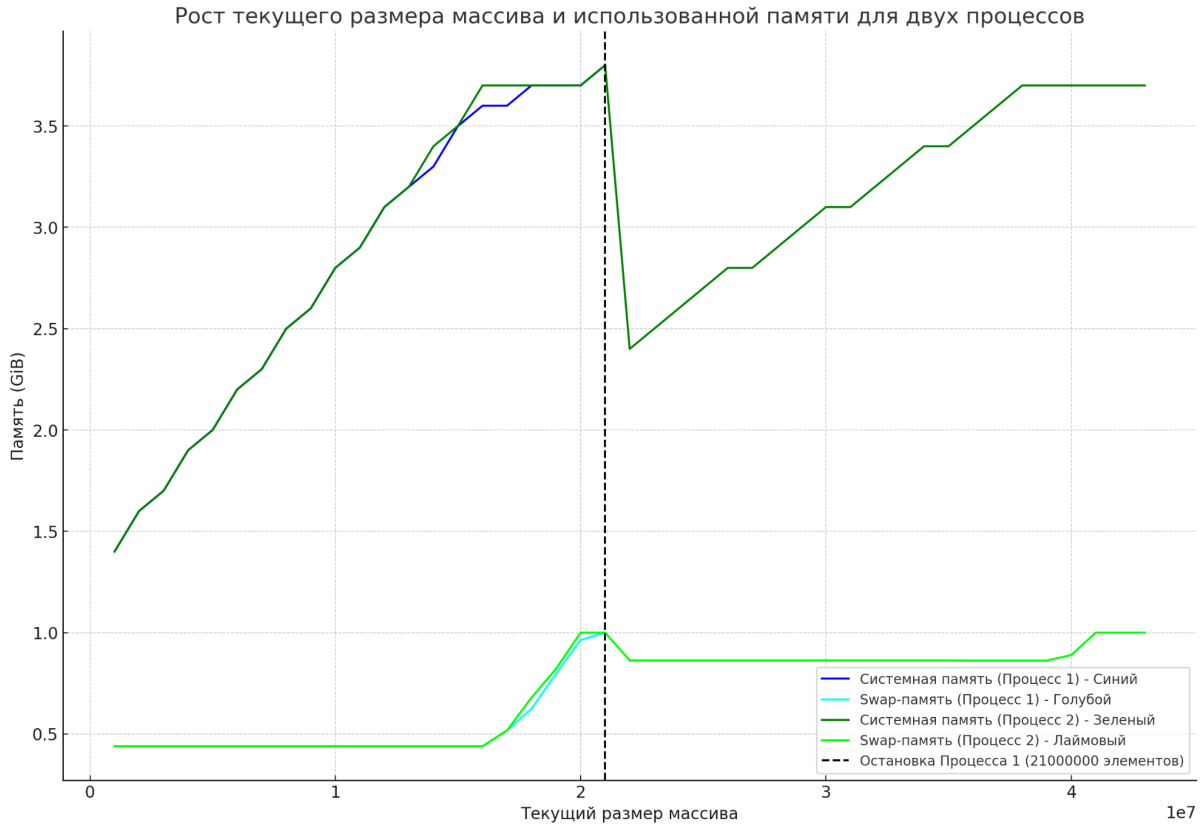
- Наблюдаемый рост использования памяти напрямую связан с увеличением размера массива в скрипте.
- Система справляется с нагрузкой до определенного предела, после чего начинает использовать swap-память, что является признаком исчерпания физической памяти.
- Аварийное завершение скрипта, вероятно, вызвано исчерпанием всех доступных ресурсов памяти, включая swap.
- Это демонстрирует важность управления ресурсами памяти в приложениях и необходимость предусмотрения механизмов защиты от переполнения памяти, особенно в сценариях с потенциально неограниченным ростом используемых ресурсов.

Этап 2

Для mem.bash

```
Текущий размер массива: 21000000
Системная память и подкачка:
               total        used        free      shared  buff/cache
available
Mem:           3,8Gi       3,8Gi       110Mi        14Mi       111Mi
43Mi
Swap:          1,0Gi       1,0Gi          0B
Верхние пять процессов:
    PID COMMAND         %MEM %CPU
```

```
  19634 bash            35.6 94.9
  19635 bash            35.5 94.9
  15555 nautilus         6.3  6.1
    621 gnome-shell      5.8 20.0
    484 Xorg             1.2  2.1

Информация о текущем скрипте:
    PID COMMAND         %MEM %CPU
  19634 bash            35.6 94.8

[15789.504599] [  19634]  1000 19634   424576   369952  3440640    52736
200 bash
[15789.504607]
oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=
0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000.service/app.
slice/app-org.gnome.Terminal.slice/vte-spawn-f2f6b88f-660a-4a9f-af3b-a2c42
fcdb63d.scope,task=bash,pid=19634,uid=1000
[15789.504630] Out of memory: Killed process 19634 (bash)
total-vm:1698304kB, anon-rss:1479808kB, file-rss:0kB, shmem-rss:0kB,
UID:1000 pgtables:3360kB oom_score_adj:200

Для mem2.bash

Текущий размер массива: 43000000
Системная память и подкачка:
               total        used        free      shared  buff/cache
available
Mem:           3,8Gi       3,7Gi        82Mi        12Mi        61Mi
82Mi
Swap:          1,0Gi       1,0Gi        48Ki
Верхние пять процессов:
Информация о текущем скрипте:
    PID COMMAND         %MEM %CPU
  19635 bash            76.6 81.1

[15845.038321] [  19635]  1000 19635   844217   770976  6803456    71296
200 bash
[15845.038324]
oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=
0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000.service/app.
slice/app-org.gnome.Terminal.slice/vte-spawn-f2f6b88f-660a-4a9f-af3b-a2c42
fcdb63d.scope,task=bash,pid=19635,uid=1000
[15845.038337] Out of memory: Killed process 19635 (bash)
total-vm:3376868kB, anon-rss:3083776kB, file-rss:128kB, shmem-rss:0kB,
UID:1000 pgtables:6644kB oom_score_adj:200
```



Анализ Двух Процессов

Оба процесса в эксперименте выполняют одну и ту же задачу — постоянно увеличивают размер массива. Из-за этого происходит:

- Постепенное Накопление Памяти: Каждый процесс постепенно увеличивает использование системной памяти. Поскольку оба процесса работают одновременно, они конкурируют за ресурсы памяти.
- Активизация Swap: Когда суммарное потребление памяти обоих процессов приближается к пределам доступной RAM, операционная система начинает активнее использовать swap.

Пороговые Величины и Аварийная Остановка

Аварийная остановка одного из процессов происходит при достижении определенного размера массива, что указывает на исчерпание доступной памяти. Пороговая величина — 21000000 элементов для одного из процессов — означает, что в этот момент системе стало недостаточно ресурсов для продолжения работы процесса, что привело к его аварийной остановке.

Выводы

- Запуск двух интенсивно использующих память процессов одновременно приводит к более быстрому исчерпанию ресурсов системной и swap-памяти.
- По мере приближения к пределу доступной памяти, ОС активно использует swap, но это может быть недостаточно для предотвращения аварийной остановки процесса.
- Аварийная остановка процесса служит важным сигналом о необходимости оптимизации использования памяти в приложениях и о возможности конфликта за ресурсы в системе с несколькими интенсивными потребителями памяти.
- Эти наблюдения подчеркивают важность мониторинга и управления памятью, особенно в системах, где множество процессов работают параллельно.

Эксперимент 2

Предыдущий скрипт аварийно остановился на размере массива 42000000. Если взять в качестве N величину в 10 раз меньшую - 4200000, то при K=10 обеспечивается безаварийная работа. Предел использования памяти взят из newmem.log

Системная память и подкачка:

```
               total        used        free      shared  buff/cache
available
Mem:           3,8Gi       3,0Gi       810Mi        16Mi       307Mi
882Mi
Swap:          1,0Gi       634Mi       389Mi
```

Верхние пять процессов:

```
    PID COMMAND         %MEM %CPU
  22547 newmem.bash      7.8 55.8
  15555 nautilus         7.2  6.6
  22554 newmem.bash      6.8 51.4
    621 gnome-shell      6.6 20.6
  22561 newmem.bash      5.9 48.3
```

При K=30 и сохранении значения N=4200000 произошло аварийное завершение:

```
[18290.081828] Out of memory: Killed process 15555 (nautilus)
total-vm:2497828kB, anon-rss:215072kB, file-rss:180kB, shmem-rss:4476kB,
UID:1000 pgtables:1196kB oom_score_adj:200

start.bash: строка 11: 26654 Убито              ./newmem.bash $N
start.bash: строка 11: 26660 Убито              ./newmem.bash $N
start.bash: строка 11: 26671 Убито              ./newmem.bash $N
```

```
start.bash: строка 11: 26677 Убито              ./newmem.bash $N
start.bash: строка 11: 26689 Убито              ./newmem.bash $N
start.bash: строка 11: 26700 Убито              ./newmem.bash $N
start.bash: строка 11: 26707 Убито              ./newmem.bash $N
start.bash: строка 11: 26714 Убито              ./newmem.bash $N
start.bash: строка 11: 26721 Убито              ./newmem.bash $N
start.bash: строка 11: 26737 Убито              ./newmem.bash $N
start.bash: строка 11: 26745 Убито              ./newmem.bash $N
start.bash: строка 11: 26760 Убито              ./newmem.bash $N
start.bash: строка 11: 26767 Убито              ./newmem.bash $N
start.bash: строка 11: 26775 Убито              ./newmem.bash $N
```

Сценарий запускает множество экземпляров `newmem.bash` с заданным параметром `N`, который определяет размер массива в каждом процессе. При увеличении количества запущенных процессов (`K`) до 30, система сталкивается с ограничениями по памяти, что приводит к аварийным завершениям процессов.

Причины Аварийных Завершений

1. Исчерпание RAM: Каждый экземпляр скрипта выделяет память для своего массива. При `K=30`, общий объем используемой памяти увеличивается в соответствии с размерами массивов всех процессов. Если суммарный объем памяти превышает доступный объем RAM, начинаются проблемы с нехваткой памяти.

2. Использование Swap-памяти: Когда RAM исчерпана, система начинает использовать swap-память. Однако swap-память медленнее и не способна компенсировать нехватку RAM полностью, особенно при высокой нагрузке.

3. Системные Ограничения: ОС имеет ограничения на количество процессов и объем используемой памяти на пользователя или систему в целом. Превышение этих ограничений также может привести к аварийным завершениям.

Подбор Максимального N при K=30

Чтобы подобрать максимальное значение `N`, при котором не происходит аварийных завершений при `K=30`, нужно учесть ограничения системной памяти. Предположим, что каждый элемент массива занимает примерно одинаковое количество памяти. В этом случае общее потребление памяти всеми процессами можно приблизительно рассчитать как K * N * размер элемента массива.

При N=3000000 отработал без аварийных завершений
При N=3600000 были аварийные завершения:

```
start.bash: строка 11: 28736 Убито              ./newmem.bash $N
start.bash: строка 11: 28747 Убито              ./newmem.bash $N
```

```
start.bash: строка 11: 28754 Убито              ./newmem.bash $N
start.bash: строка 11: 28761 Убито              ./newmem.bash $N
start.bash: строка 11: 28768 Убито              ./newmem.bash $N
start.bash: строка 11: 28775 Убито              ./newmem.bash $N
start.bash: строка 11: 28782 Убито              ./newmem.bash $N
start.bash: строка 11: 28793 Убито              ./newmem.bash $N
start.bash: строка 11: 28800 Убито              ./newmem.bash $N
start.bash: строка 11: 28807 Убито              ./newmem.bash $N
start.bash: строка 11: 28814 Убито              ./newmem.bash $N
start.bash: строка 11: 28829 Убито              ./newmem.bash $N
```
При N=3200000 отработал без аварийных завершений
При N=3300000 были аварийные завершения:
```
start.bash: строка 11: 30759 Убито              ./newmem.bash $N
start.bash: строка 11: 30773 Убито              ./newmem.bash $N
start.bash: строка 11: 30787 Убито              ./newmem.bash $N
start.bash: строка 11: 30794 Убито              ./newmem.bash $N
start.bash: строка 11: 30801 Убито              ./newmem.bash $N
```
При N=3250000 отработал без аварийных завершений
.....
При дальнейших подборах оказалось, что повторный запуск N=3250000 может быть и с аварийными завершениями.

То, что процессы сначала успешно выполнялись без аварийных завершений при `N=3250000`, а затем начали аварийно завершаться даже при том же значении `N`, может быть объяснено несколькими факторами, связанными с динамикой управления системными ресурсами:

1. Динамическое Изменение Условий Системы: Возможно, что во время первого запуска система имела больше свободных ресурсов (RAM и CPU). В последующих запусках условия могли измениться из-за дополнительной нагрузки от других процессов или системных служб, что привело к нехватке ресурсов.

2. Фрагментация Памяти: Со временем RAM может стать фрагментированной, что означает, что большие блоки непрерывной памяти могут быть недоступны, даже если общий объем доступной памяти кажется достаточным. Это может повлиять на способность системы выделять память для новых процессов.

3. Кэширование и Swap: Операционная система может кэшировать данные и использовать swap-память для управления недавно использованными данными и приложениями. Эти механизмы могут изменять доступный объем памяти для новых процессов со временем.

4. Влияние Операционной Системы: ОС могла автоматически настроить параметры управления памятью и процессами в ответ на предыдущие запуски скриптов, что повлияло на результаты последующих запусков.

5. Тепловой Режим и Производительность: В некоторых случаях, особенно в системах с ограниченным охлаждением, длительная высокая нагрузка может привести к снижению производительности из-за перегрева компонентов.

Итого, можно сказать, что максимальное значение N при K=30, с которым не происходит аварийного завершения процессов - 3250000. Это значение действительно является пороговым, т.к. при фоновых изменениях системы во времени, без влияния со стороны пользователя, в определенный момент времени при заданных значениях аварийные завершения все-таки начинают происходить. В качестве решения можно перезагрузить систему и запустить скрипт повторно, после чего аварийных завершений снова не будет