Яковлев Илья М3239

группа В1

Отчет по лабораторной работе №5

Параметры системы:

- Общий объем оперативной памяти: 39,15 Гб
- Объем раздела подкачки: 10 Гб
- Размер страницы виртуальной памяти: 4096 б
- Объем свободной физической памяти в ненагруженной системе: 36,41 Гб
- Объем свободного пространства в разделе подкачки в ненагруженной системе: 9,46 Гб

Эксперимент №1

Этап 1:

Запускаем скрипт track_1.sh и собираем данные

Последнее значение в report.log: 638000000

Последние записи в системном журнале:

```
[31815.475566] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task

mem.bash,pid=99755,uid=1000

[31815.475577] Out of memory: Killed process 99755 (mem.bash) total-vm:49915168kB, anon-rss:39929872kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:97716kB oom_score_adj:0

[36524.507763] [ 114840]  1000 114840 12465757 10011433 99954688  2453185             0 mem.bash

[36524.507778] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task

mem.bash,pid=114840,uid=1000

[36524.507793] Out of memory: Killed process 114840 (mem.bash) total-vm:49863028kB, anon-rss:40045732kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:97612kB oom_score_adj:0
```

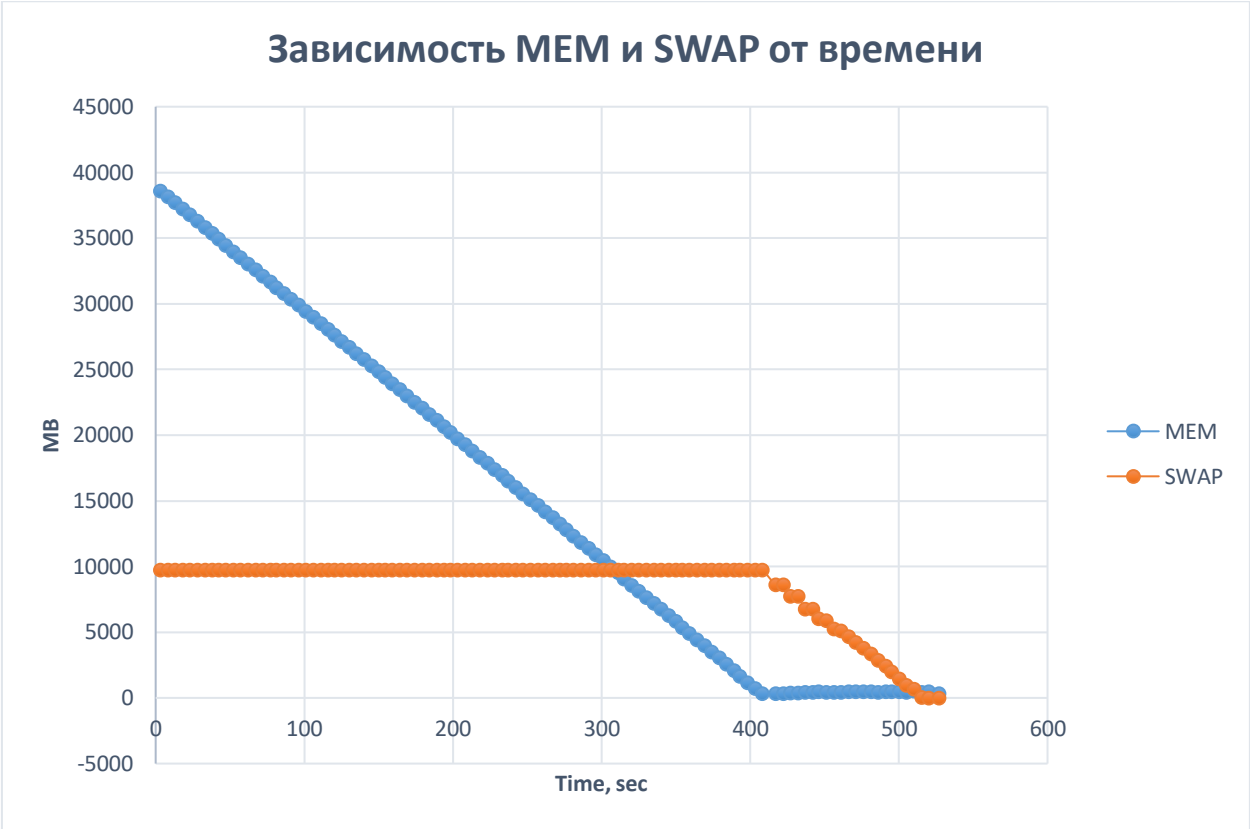

График 1

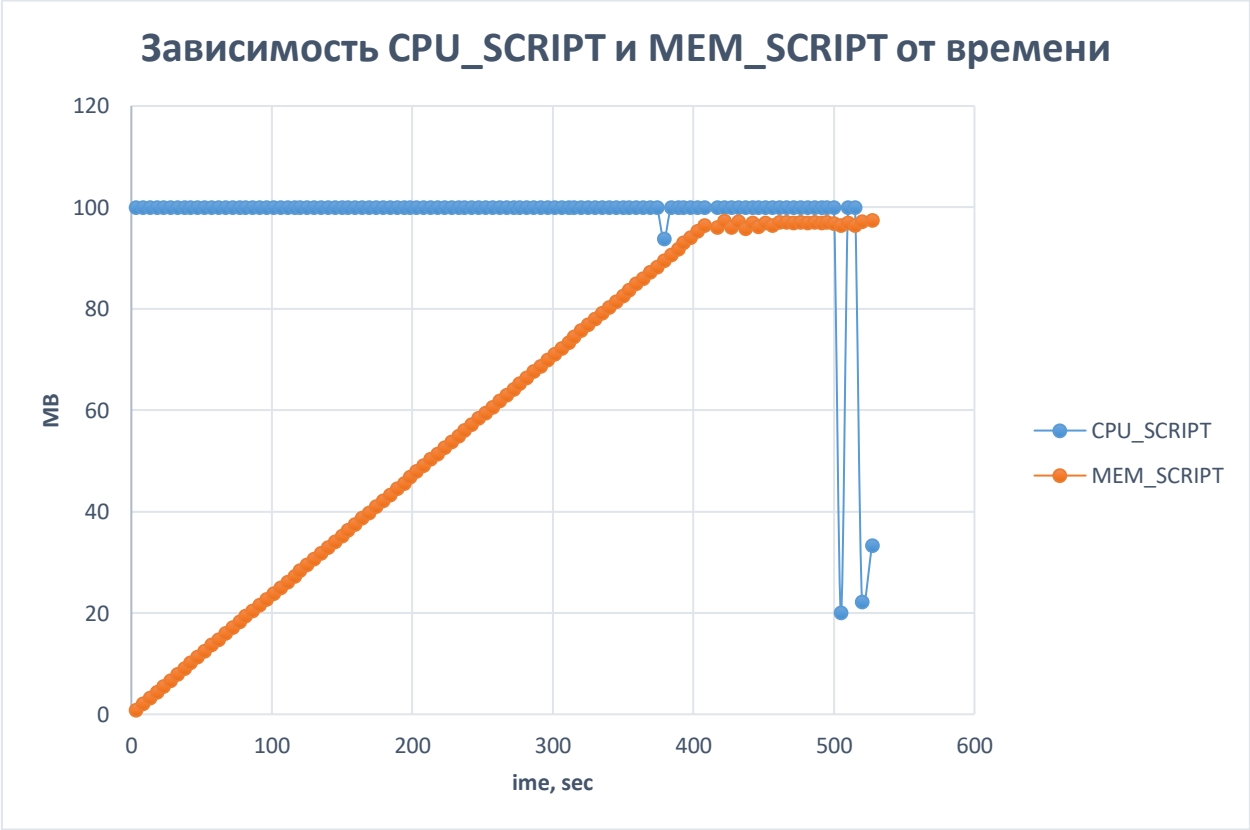

График 2

Этап 2:

Запускаем скрипт track_2.sh и собираем данные

Последнее значение в report.log: 638000000

Последнее значение в report2.log: 319000000

Последние записи в системном журнале:

[31815.475563] [99755] 1000 99755 12478792 9982468 100061184 2495183 0 mem.bash

[31815.475566] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=mem.bash,pid=99755,uid=1000

[31815.475577] Out of memory: Killed process 99755 (mem.bash) total-vm:49915168kB, anon-rss:39929872kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:97716kB oom_score_adj:0

[36524.507763] [114840] 1000 114840 12465757 10011433 99954688 2453185 0 mem.bash

[36524.507778] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=mem.bash,pid=114840,uid=1000

[36524.507793] Out of memory: Killed process 114840 (mem.bash) total-vm:49863028kB, anon-rss:40045732kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:97612kB oom_score_adj:0

[39957.340118] [132688] 1000 132688 6220507 5000886 49901568 1218474 0 mem.bash

[39957.340120] [132689] 1000 132689 6250141 5024375 50135040 1224630 0 mem2.bash

[39957.340129] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=mem2.bash,pid=132689,uid=1000

[39957.340140] Out of memory: Killed process 132689 (mem2.bash) total-vm:25000564kB, anon-rss:20097500kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:48960kB oom_score_adj:0

[40223.606299] [132688] 1000 132688 12463744 10009943 99942400 2452667 0 mem.bash

[40223.606311] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=mem.bash,pid=132688,uid=1000

[40223.606324] Out of memory: Killed process 132688 (mem.bash) total-vm:49854976kB, anon-rss:40039772kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:97600kB oom_score_adj:0

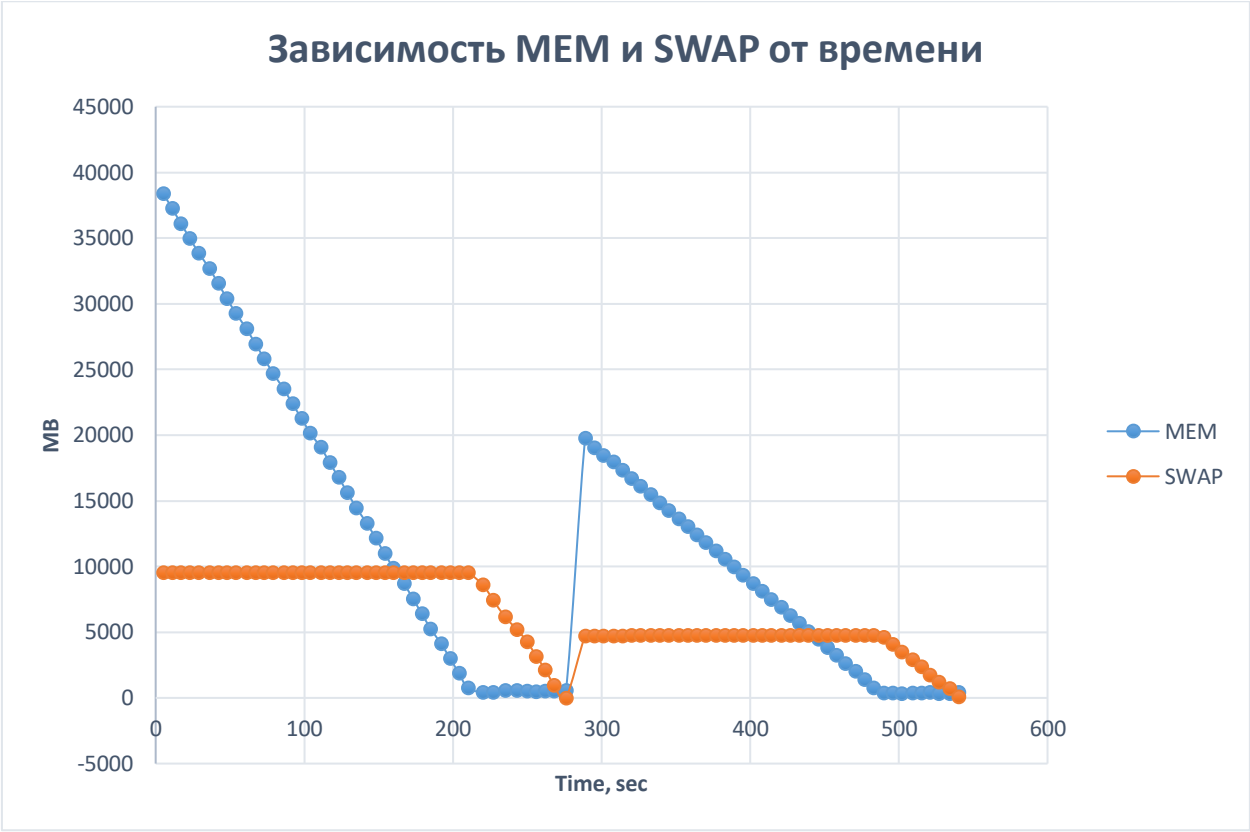

График 3

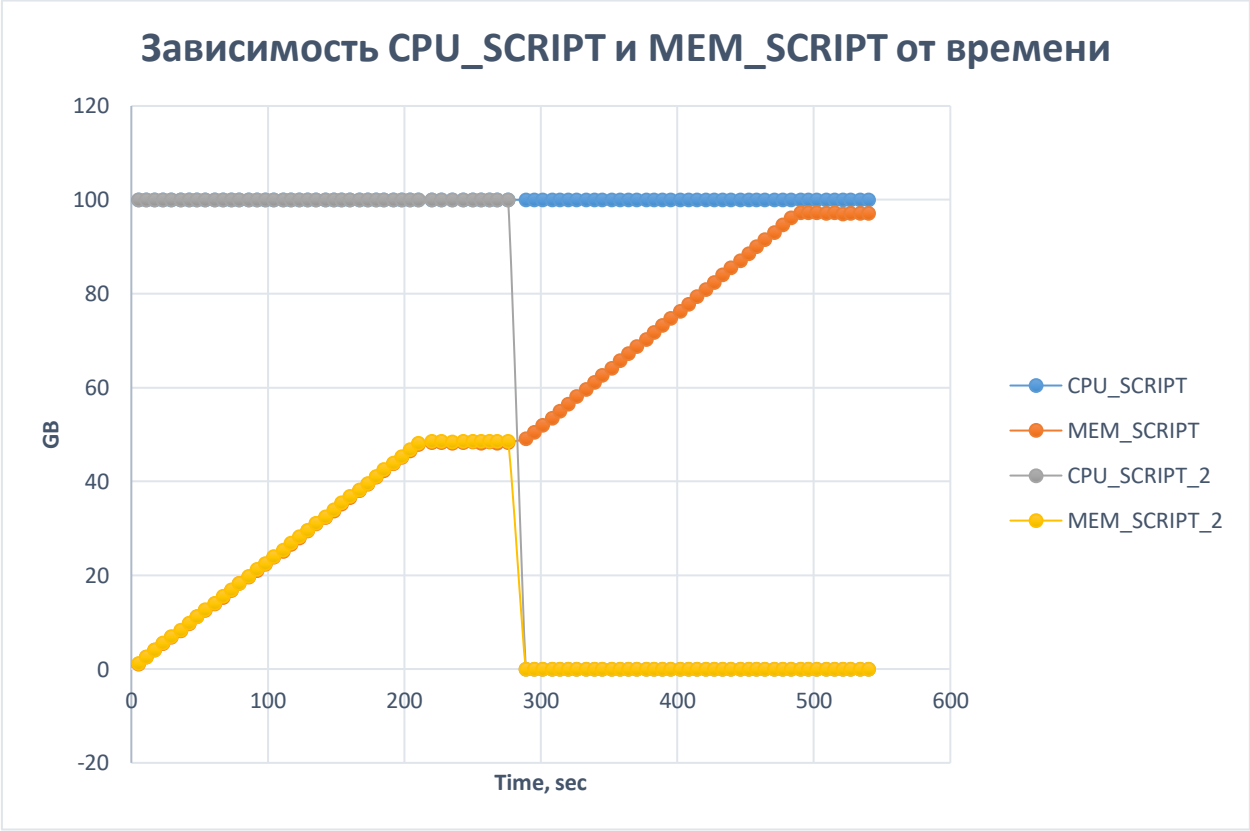

График 4

Обработка результатов:

Рассмотрим каждый график отдельно

Примечание
Далее под “линейно” и под аналогичными по смыслу конструкциями будет подразумеваться “линейно относительно времени” и тому подобное.

График 1:

Память уменьшается линейно, пока не достигает 0 на 400 секунде. До 400 секунды свободная память из подкачки не изменяется, после изменяется линейно.

Пока можно взять память из оперативной памяти, она берется. И в крайнем случае, когда нельзя, берется память из подкачки.

График 2:

До 420 секунды потребление памяти идет линейно.

С 420 секунды до 510 секунды память берется из подкачки, из-за этого потребление не меняется.

График 3:

До 280 секунды тот же график, что и “График 1”.

После 280 секунды mem2.bash убили и освободилась память, которую потратил mem.bash, что мы можем видеть с 280 секунды.

С 280 секунды и до 540 секунды тот же график, что и “График 1”.

График 4:

До 220 секунды потребление памяти у обоих процессов идет линейно с одинаковой скоростью.

С 220 секунды до 280 секунды память берется из подкачки, из-за этого потребление не меняется.

(То есть оба процесса соответствуют “График 2”)

На 280 секунде умирает mem2.bash и освобождает ресурсы.

С 280 секунды и до 540 секунды тот же график, что и “График 2”.

Итог:

Определили, как ведет себя система, когда заканчивается память, провели небольшой анализ полученных графиков.

На этапе 2 суммарный размер массивов из mem.bash и mem2.bash при первом out of memory оказался равен полученному на этапе 1 размеру. Но это не обязательно должно было случиться и об этом будет следующий эксперимент.

Эксперимент №2

Запускаем скрипт track_3.sh при разных K, N и собираем данные

Примечания
1.) Чтобы узнать был ли out of memory (далее OOM), можно посмотреть, появились ли новые записи в dmesg | grep "newmem.bash,pid" 2.) Обозначим граничный размер массива, найденный в предыдущем эксперименте за MAX_SIZE = *638000000*

- K = 10; N = MAX_SIZE / 10
  track_3.sh ожидаемо отрабатывает без OOM
  Ведь единовременно занимается не больше памяти, чем в ранее найденном граничном случае.

- K = 30; N = MAX_SIZE / 10
  track_3.sh ожидаемо не отрабатывает без OOM
  Значительно превышен максимально возможный размер массива в памяти.

- K = 30
  Бинарным поиском найдено максимально возможное N = 22500000
  22500000*30 = 675000000 > 638000000 = MAX_SIZE
  Как же такое может быть?
  При работе с множеством процессов нельзя считать, что все процессы отработали одинаковое количество времени с начала запуска (в том смысле, что отношение процессорного времени к времени жизни процесса одинаковое). Ведь планировщик процессов может исходя из оптимизаций притормозить процесс, который много потребляет памяти вперед, чтобы он, возможно, освободил, или назад, наоборот, назад, считая, видя тенденцию увеличения потребления памяти или какого-либо другого ресурса. Может появиться совершенно другой процесс, который будет более или менее приоритетный.
  Если резюмировать, у планировщика могут быть разные алгоритмы, основывающиеся на большом множестве факторов, так что справедливость совершенно не факт, что будет соблюдаться.
  Отвечая на исходный вопрос:
  Процессы не имеют справедливости в распределении ресурсов, из-за чего какой-то процесс может выполниться пораньше, освободив память и дав возможность воспользоваться этим ресурсом другим процессам.

  В заключение хочется сказать, что хоть в данном случае удалось найти N при котором нет OOM, что потребность программ в оперативной памяти больше предоставляемых ресурсов, однако в общем случае на такое

рассчитывать нельзя, ведь у нормальных программ, а не учебных, как минимум может быть совершенно не линейное потребление памяти от времени работы, не говоря о других факторах. Да и в целом, надеяться на хорошее планирование точно не стоит.