МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

‘Операционные системы’

Выполнил:

Студент группы P33312

Соболев Иван
Александрович

Преподаватель:

Пашнин Александр
Денисович

Санкт-Петербург, 2023

Оглавление

Задание:

Основная цель лабораторной работы — знакомство с системными инструментами анализа производительности и поведения программ. В данной лабораторной работе Вам будет предложено произвести нагрузочное тестирование Вашей операционной системы при помощи инструмента stress-ng.

В качестве тестируемых подсистем использовать: cpu, cache, io, memory, network, pipe, scheduler.

Для работы со счетчиками ядра использовать все утилиты, которые были рассмотренны на лекции (раздел 1.9, кроме kdb)

Ниже приведены списки параметров для различных подсистем (Вам будет выдано 2 значения для каждой подсистемы согласно варианту в журнале). Подбирая числовые значения для выданных параметров, и используя средства мониторинга, добиться **максимальной** производительности системы (BOGOPS, FLOPS, Read/Write Speed, Network Speed).

Построить графики (подходящие по заданию.):

- Потребления программой CPU;
- Нагрузки, генерируемой программой на подсистему ввода-вывода;
- Нагрузки, генерируемой программой на сетевую подсистему;
- Другие графики, необходимые для демонстрации работы.

Исходный код всех скриптов: https://github.com/Ivanio1/itmo-os/tree/main/lab1

Выполнение:

Вариант:

cpu: [int128decimal128, decimal64];

cache: [cache-ways, l1cache];

memory: [lockbus, fork-vm];

network: [sockdiag, netlink-proc];

io: [iomix, ioport];

pipe: [pipe-size, pipeherd-yield];

sched: [sched-runtime, sched-prio]

CPU:

1) Запустим stress-ng с первым параметром.

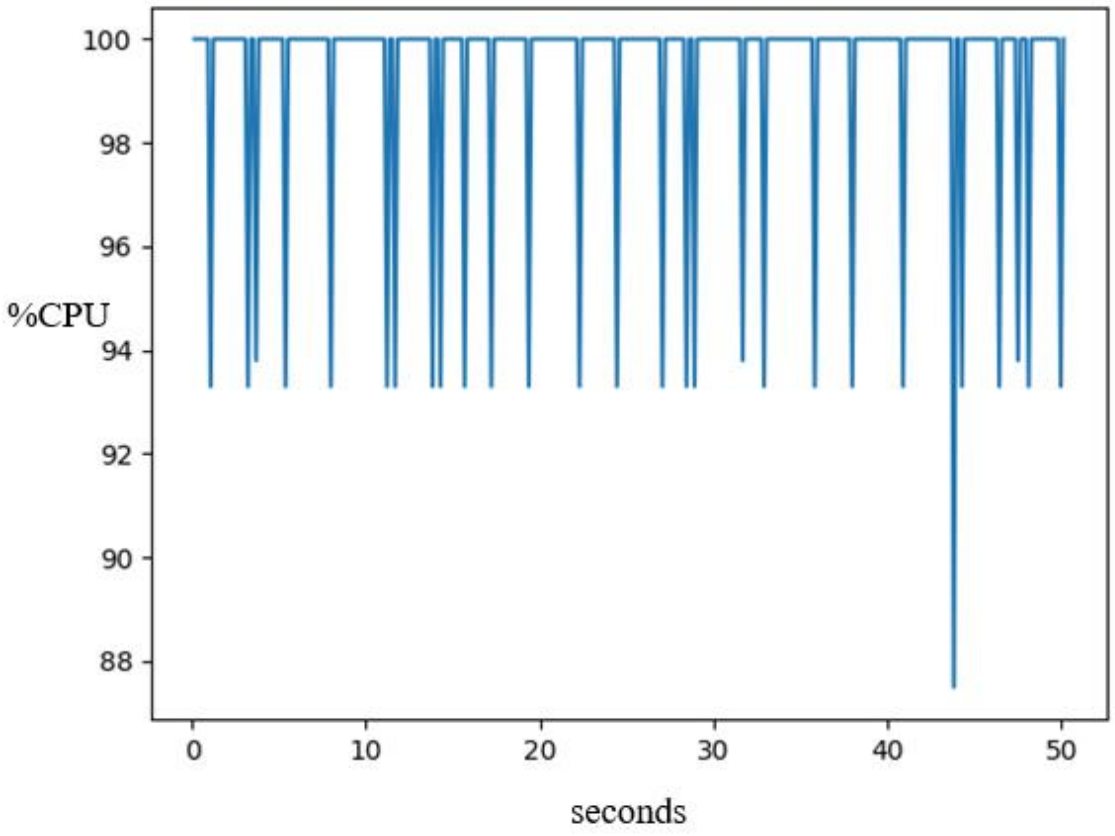
--int128decimal128 1000 итераций сочетания 128-разрядных целочисленных и 128-разрядных десятичных операций с плавающей запятой.

Команда запуска: `stress-ng --cpu 1 --cpu-method int128decimal128`

Запуск:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 1 --cpu-method int128decimal128
stress-ng: info:  [4100] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [4100] dispatching hogs: 1 cpu
```

Построим график с помощью питоновского скрипта (top):



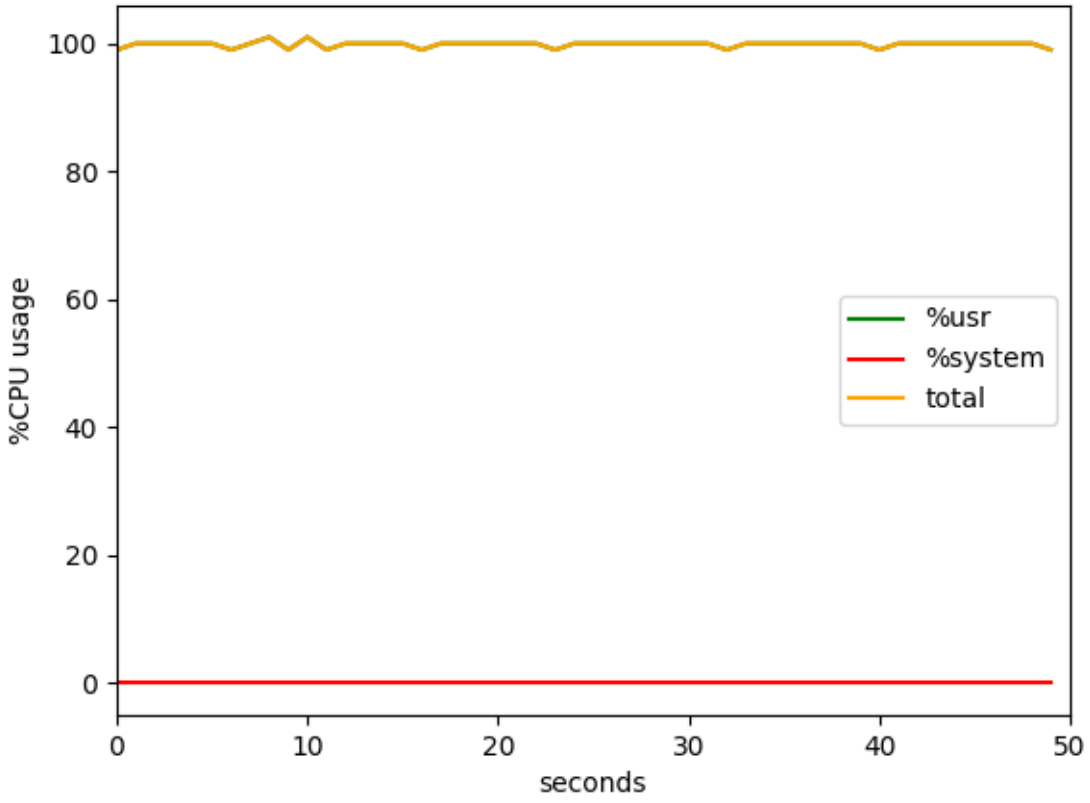
Также посмотрим на потребление CPU с помощью команды pidstat:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 4101 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:14:04      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:14:05     1000      4101  100,00    0,00    0,00    0,00  100,00     7  stress-ng
22:14:06     1000      4101  100,00    0,00    0,00    0,00  100,00     7  stress-ng
22:14:07     1000      4101  100,00    0,00    0,00    0,00  100,00     7  stress-ng
22:14:08     1000      4101  100,00    0,00    0,00    0,00  100,00     7  stress-ng
22:14:09     1000      4101  100,00    0,00    0,00    0,00  100,00     7  stress-ng
```

Данная команда выводит статистику по потоку -p PID раз в секунду.

Как мы можем видеть 1 ядро загружено на 100%.

График (pidstat):



Попробуем запустить программу на 8 процессорных ядрах. (В системе 8 ядер)

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 8 --cpu-method int128decimal128
stress-ng: info:  [5287] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5287] dispatching hogs: 8 cpu
```

Имеем похожую картину:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 5288 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:16:59      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:17:00     1000      5288  100,00    0,00    0,00    0,00  100,00     5  stress-ng
22:17:01     1000      5288  100,00    0,00    0,00    0,00  100,00     5  stress-ng
22:17:02     1000      5288  100,00    0,00    0,00    0,00  100,00     5  stress-ng
22:17:03     1000      5288  100,00    0,00    0,00    0,00  100,00     5  stress-ng
22:17:04     1000      5288  100,00    0,00    0,00    0,00  100,00     5  stress-ng
22:17:05     1000      5288   99,00    0,00    0,00    0,00   99,00     5  stress-ng
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 5289 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:17:16      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:17:17     1000      5289   99,01    0,00    0,00    0,00   99,01     6  stress-ng
22:17:18     1000      5289  100,00    0,00    0,00    0,00  100,00     6  stress-ng
22:17:19     1000      5289  100,00    0,00    0,00    1,00  100,00     6  stress-ng
22:17:20     1000      5289  100,00    0,00    0,00    0,00  100,00     6  stress-ng
22:17:21     1000      5289  100,00    0,00    0,00    0,00  100,00     6  stress-ng
22:17:22     1000      5289  100,00    0,00    0,00    0,00  100,00     6  stress-ng
```

Каждое из 8 ядер нагружается практически на 100% своим тестом. Можно увидеть на каком ядре работает процесс, обратив внимание на параметр CPU в выводе pidstat.

Посмотрим, что будет при запуске 9 тестов:

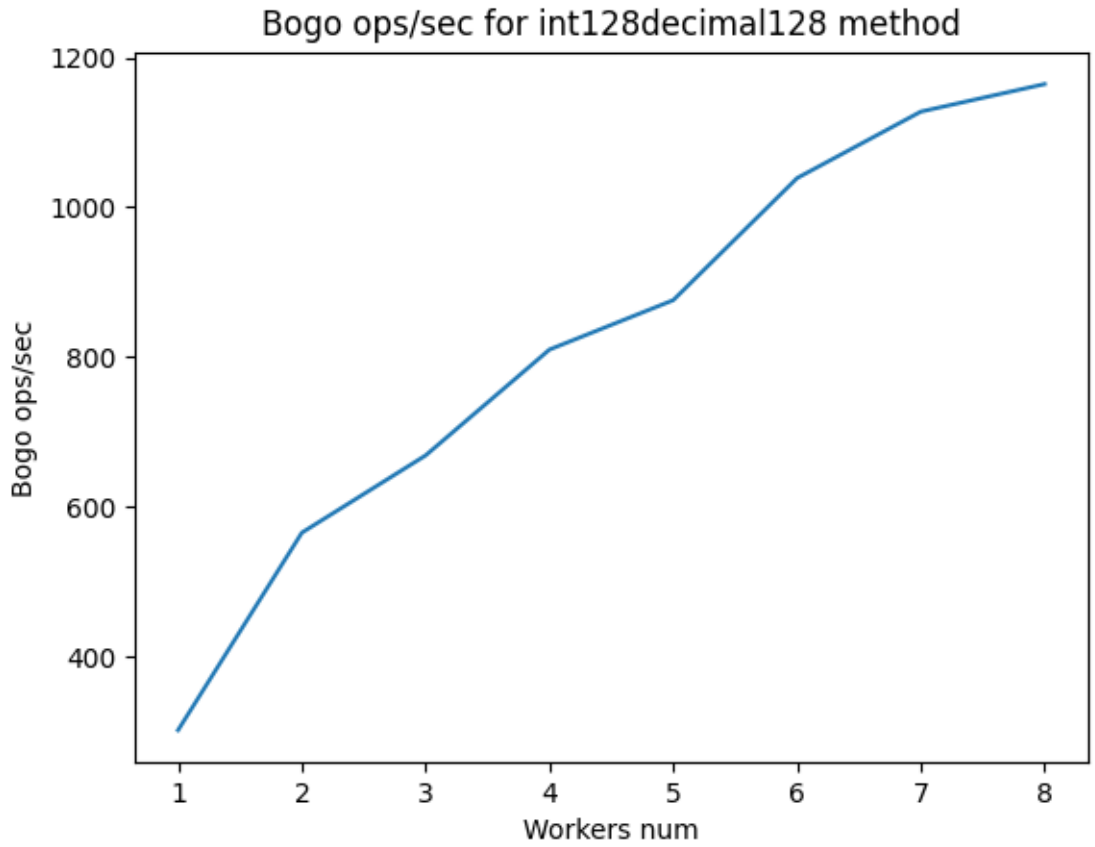
```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 9 --cpu-method int128decimal128
stress-ng: info:  [5366] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5366] dispatching hogs: 9 cpu
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 5370 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:19:35      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:19:36     1000      5370   95,00    0,00    0,00    5,00   95,00     3  stress-ng
22:19:37     1000      5370   80,00    0,00    0,00   20,00   80,00     0  stress-ng
22:19:38     1000      5370   88,00    0,00    0,00   13,00   88,00     0  stress-ng
22:19:39     1000      5370   86,00    0,00    0,00   13,00   86,00     3  stress-ng
22:19:40     1000      5370   86,00    0,00    0,00   14,00   86,00     6  stress-ng
22:19:41     1000      5370   89,00    0,00    0,00   11,00   89,00     6  stress-ng
22:19:42     1000      5370   79,00    0,00    0,00   21,00   79,00     6  stress-ng
```

Теперь же ситуация другая. Так как процессов больше, чем ядер, то возникает конкуренция за ресурсы. Она на многоядерных процессорах возникает, когда более одного процесса или потока пытаются использовать общие аппаратные ресурсы, такие как ядра процессора, кэш-память, оперативная память (RAM) и другие системные ресурсы. Эта конкуренция может влиять на производительность и загрузку процессора.

Чтобы добиться максимальной загрузки системы, нужно запускать n тестов на n-ядерном процессоре, тогда каждое ядро будет использоваться на 100%.

Посмотрим также на график bogoops:



Количество bogo операций увеличивается с увеличением количества тестов.

2) Посмотрим второй параметр

-- decimal64 1000 итераций сочетания 64-разрядных десятичных операций с плавающей запятой

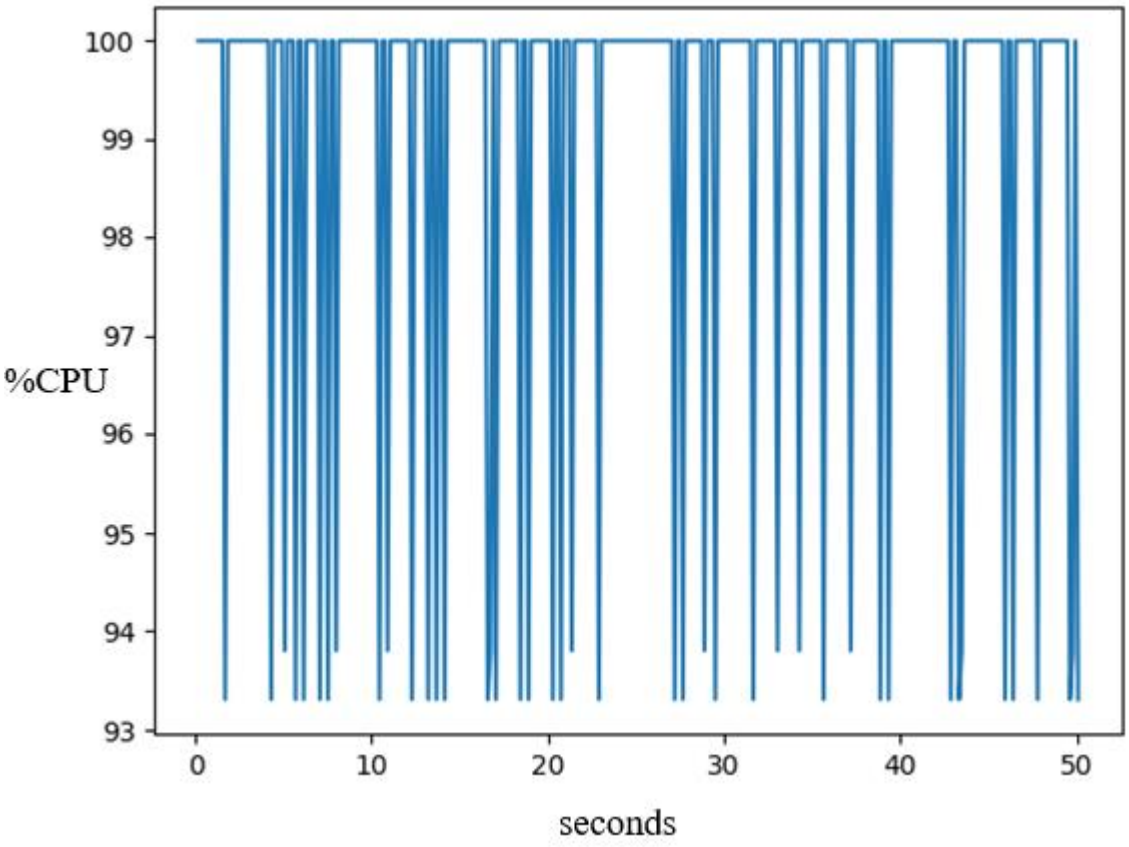
Команда запуска: `stress-ng --cpu 8 --cpu-method decimal64`

Запуск:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 8 --cpu-method decimal64
stress-ng: info:  [5424] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5424] dispatching hogs: 8 cpu
```

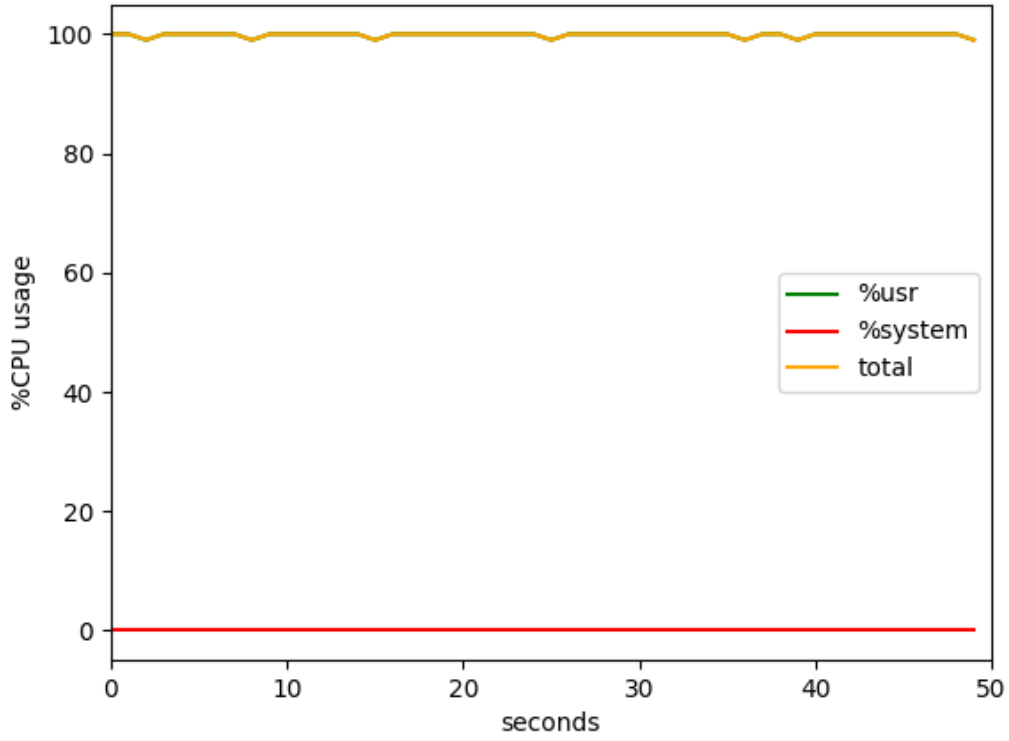
Запустим сразу на 8 ядрах.

График загрузки одного из ядер (top).



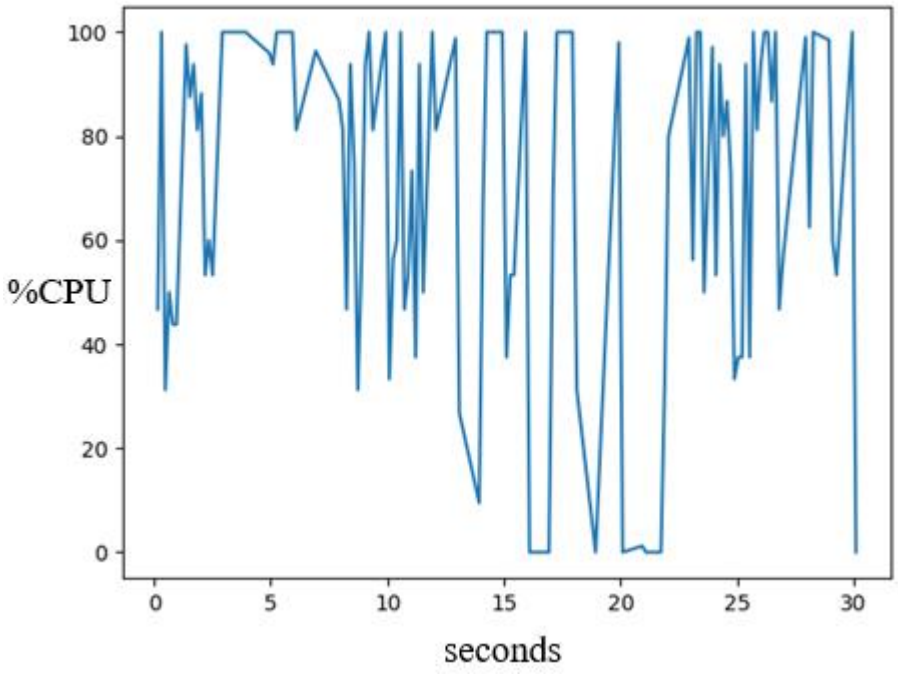

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 5430 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:22:19      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:22:20     1000      5430  100,00    0,00    0,00    0,00  100,00     5  stress-ng
22:22:21     1000      5430   99,00    0,00    0,00    0,00   99,00     5  stress-ng
22:22:22     1000      5430  100,00    0,00    0,00    1,00  100,00     5  stress-ng
22:22:23     1000      5430   99,00    0,00    0,00    1,00   99,00     5  stress-ng
22:22:24     1000      5430   99,00    0,00    0,00    0,00   99,00     5  stress-ng
```

График (pidstat):



Запустим 9 тестов:

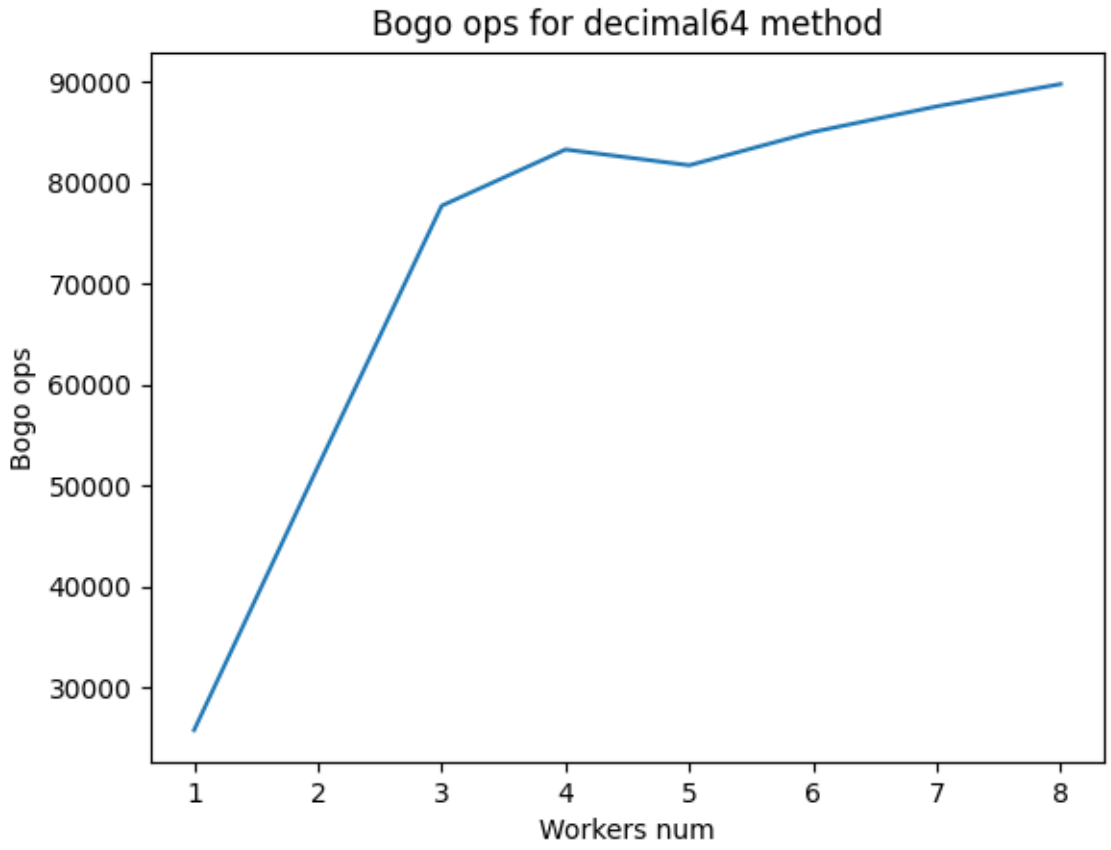
```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 9 --cpu-method decimal64
stress-ng: info:  [5489] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5489] dispatching hogs: 9 cpu
```



```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 5492 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:23:58      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:23:59     1000      5492   85,00    0,00    0,00   15,00   85,00     5  stress-ng
22:24:00     1000      5492   95,00    0,00    0,00    4,00   95,00     2  stress-ng
22:24:01     1000      5492   88,00    0,00    0,00   12,00   88,00     2  stress-ng
22:24:02     1000      5492   95,00    0,00    0,00    5,00   95,00     2  stress-ng
22:24:03     1000      5492   88,00    0,00    0,00   13,00   88,00     6  stress-ng
22:24:04     1000      5492   92,00    0,00    0,00    8,00   92,00     6  stress-ng
```

Имеем аналогичную ситуацию. Это ожидалось, так как мы просто изменили метод нагрузки.

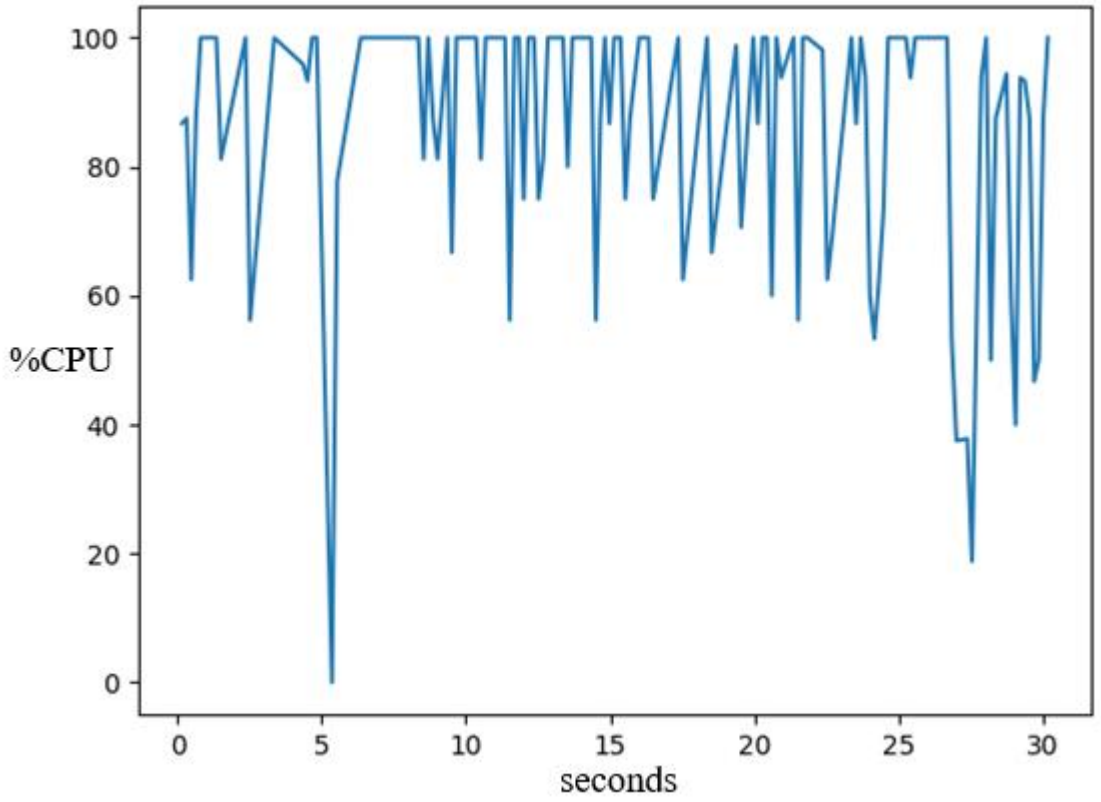
Посмотрим также на график bogoops:



Количество bogo операций увеличивается с увеличением количества тестов.

3) Запустим тесты сразу с двумя параметрами.

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 8 --cpu-method int128decimal128 --cpu-method decimal64
stress-ng: info:  [5546] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5546] dispatching hogs: 8 cpu
```




```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -p 5547 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:26:13      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
22:26:14     1000      5547  100,00    0,00    0,00    0,00  100,00     1  stress-ng
22:26:15     1000      5547   97,00    0,00    0,00    2,00   97,00     1  stress-ng
22:26:16     1000      5547   99,00    0,00    0,00    2,00   99,00     1  stress-ng
22:26:17     1000      5547   98,00    0,00    0,00    1,00   98,00     1  stress-ng
22:26:18     1000      5547   98,00    0,00    0,00    3,00   98,00     1  stress-ng
22:26:19     1000      5547   98,00    0,00    0,00    2,00   98,00     1  stress-ng
22:26:20     1000      5547   99,00    0,00    0,00    1,00   99,00     1  stress-ng
22:26:21     1000      5547   97,00    0,00    0,00    3,00   97,00     1  stress-ng
22:26:22     1000      5547   98,00    0,00    0,00    2,00   98,00     1  stress-ng
```

Можем заметить, что при запуске двух методов нагрузки мы видим некие просадки в производительности. Процессор вынужден работать более интенсивно, что может вызвать просадки из-за следующих факторов:

- Увеличение общей нагрузки: Запуск двух методов увеличивает общую нагрузку на процессор, что может создавать более сильные просадки.
- Конфликт ресурсов: Разные методы могут конфликтовать за доступ к вычислительным ресурсам (Например, оперативная память).
- Увеличение конкуренции: Запуск двух методов может вызвать конкуренцию между ними, что может привести к борьбе за ресурсы процессора и, следовательно, к просадкам в процентной загрузке.

Выводы по мониторингу CPU: для максимальной производительности n-ядерного процессора, необходимо запускать нагрузочные тесты ровно на n ядер. При увеличении данной цифры производительность падает. Также при увеличении числа методов нагрузочного тестирования производительность также снижается.

Утилита top и pidstat показали примерно одинаковые цифры.

Cache:

1) Первый параметр

--cache-ways N указывает количество способов кэширования для тестирования.

Команда запуска: sudo perf stat -e cache-misses,cache-references stress-ng --cache 1 --cache-ways 1 --timeout 30s

Кэш-память в современных процессорах обычно имеет несколько уровней, такие как L1, L2 и L3. Каждый уровень кэша может иметь определенное количество способов ассоциации, что влияет на то, сколько блоков данных может храниться в кэше.

Параметр "cache-ways" в stress-ng позволяет определить, сколько способов ассоциации должно использоваться в кэше.

С помощью данной команды мы запустим один тестер кэша с одним способом ассоциации, далее с помощью утилиты perf соберем информацию о кэш-промахах и обращениях к кэш памяти за 30 секунд работы тестера.

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e cache-misses,cache-references stress-ng --cache 1 --cache-ways 1 --timeout 30s
[sudo] password for ivan:
stress-ng: info:  [30527] setting to a 30 second run per stressor
stress-ng: info:  [30527] dispatching hogs: 1 cache
stress-ng: info:  [30527] successful run completed in 30.00s

 Performance counter stats for 'stress-ng --cache 1 --cache-ways 1 --timeout 30s':

         7 674 440      cache-misses                     #    0,116 % of all cache refs
     6 603 810 205      cache-references

      30,009759223 seconds time elapsed

      28,927993000 seconds user
       0,399187000 seconds sys


ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$
```

Увеличим количества способов ассоциации:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e cache-misses,cache-references stress-ng --cache 1 --cache-ways 8 --timeout 30s
stress-ng: info:  [30576] setting to a 30 second run per stressor
stress-ng: info:  [30576] dispatching hogs: 1 cache
stress-ng: info:  [30576] successful run completed in 30.00s

 Performance counter stats for 'stress-ng --cache 1 --cache-ways 8 --timeout 30s':

       875 187 966      cache-misses                     #    7,680 % of all cache refs
    11 395 141 869      cache-references

      30,011467609 seconds time elapsed

      29,861720000 seconds user
       0,085344000 seconds sys


ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$
```

Статистика по обращениям и промахам увеличилась. Ожидаемо, ведь мы заставляем систему обрабатывать различные конфигурации кэша.

Также проведем мониторинг утилитой free:

До запуска стресс-теста:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7793        2934         910        1642        3948        2934
Swap:           2047          34        2013
```

После запуска стресс-теста:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7793        2846         812        1850        4134        2814
Swap:           2047          34        2013
```

Значение памяти, которая ииспользуется буферами ядра и кэшем увеличилось.

2) Второй параметр

-- l1cache N запускает N рабочих процессов, которые используют кэш ЦП уровня 1 при чтении и записи.

Команда запуска: sudo perf stat -e cache-misses,cache-references,L1-dcache-loads stress-ng --l1cache 1 --timeout 30s

L1-dcache-loads – статистика по загрузке кэша первого уровня.

Статистика утилитой perf:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e cache-misses,cache-references,L1-dcache-loads stress-ng  --l1cache 1 --timeout 30s
stress-ng: info:  [31142] setting to a 30 second run per stressor
stress-ng: info:  [31142] dispatching hogs: 1 l1cache
stress-ng: info:  [31143] stress-ng-l1cache: l1cache: size: 32.0K, sets: 64, ways: 8, line size: 64
stress-ng: info:  [31142] successful run completed in 30.04s

 Performance counter stats for 'stress-ng --l1cache 1 --timeout 30s':

         1 051 358      cache-misses                     #   15,683 % of all cache refs
         6 703 903      cache-references
    13 431 971 886      L1-dcache-loads

      30,049937748 seconds time elapsed

      30,039038000 seconds user
       0,008641000 seconds sys
```

Увеличим количество рабочих процессов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e cache-misses,cache-references,L1-dcache-loads stress-ng  --l1cache 8 --timeout 30s
stress-ng: info:  [31150] setting to a 30 second run per stressor
stress-ng: info:  [31150] dispatching hogs: 8 l1cache
stress-ng: info:  [31151] stress-ng-l1cache: l1cache: size: 32.0K, sets: 64, ways: 8, line size: 64
stress-ng: info:  [31150] successful run completed in 30.30s

 Performance counter stats for 'stress-ng --l1cache 8 --timeout 30s':

        57 339 037      cache-misses                     #    0,209 % of all cache refs
    27 406 512 597      cache-references
    51 065 453 025      L1-dcache-loads

      30,312523734 seconds time elapsed

     237,311646000 seconds user
       0,092090000 seconds sys
```

Загрузка кэша увеличивается.

Также проведем мониторинг утилитой free:

До запуска стресс-теста:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7793        2937        1025        1502        3830        3071
Swap:           2047          34        2013
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng  --l1cache 1
```

После запуска стресс-теста:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7793        2953        1004        1507        3834        3050
Swap:           2047          34        2013
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$
```

3) Запуск с двумя параметрами

Команда запуска: sudo perf stat -e cache-misses,cache-references,L1-dcache-loads stress-ng --l1cache 1 --cache 1 --cache-ways 1 --timeout 30s

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$  sudo perf stat -e cache-misses,cache-references,L1-dcache-loads stress-ng  --l1cache 1 --cache 1 --cache-ways 1 --timeout 30s
stress-ng: info:  [31322] setting to a 30 second run per stressor
stress-ng: info:  [31322] dispatching hogs: 1 l1cache, 1 cache
stress-ng: info:  [31323] stress-ng-l1cache: l1cache: size: 32.0K, sets: 64, ways: 8, line size: 64
stress-ng: info:  [31322] successful run completed in 30.28s

 Performance counter stats for 'stress-ng --l1cache 1 --cache 1 --cache-ways 1 --timeout 30s':

        13 778 638      cache-misses                     #    0,199 % of all cache refs
     6 928 146 497      cache-references
    50 595 695 760      L1-dcache-loads

      30,297631266 seconds time elapsed

      43,410734000 seconds user
       0,147305000 seconds sys
```

Увеличим нагрузку:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$  sudo perf stat -e cache-misses,cache-references,L1-dcache-loads stress-ng  --l1cache 8 --cache 1 --cache-ways 8 --timeout 30s
[sudo] password for ivan:
Sorry, try again.
[sudo] password for ivan:
stress-ng: info:  [31342] setting to a 30 second run per stressor
stress-ng: info:  [31342] dispatching hogs: 8 l1cache, 1 cache
stress-ng: info:  [31343] stress-ng-l1cache: l1cache: size: 32.0K, sets: 64, ways: 8, line size: 64
stress-ng: info:  [31342] successful run completed in 30.24s

 Performance counter stats for 'stress-ng --l1cache 8 --cache 1 --cache-ways 8 --timeout 30s':

       495 194 842      cache-misses                     #    3,721 % of all cache refs
    13 309 627 933      cache-references
    74 951 631 937      L1-dcache-loads

      30,248631681 seconds time elapsed

     235,397681000 seconds user
       1,368225000 seconds sys
```

Выводы по мониторингу кэша: Был проведен мониторинг использования и нагрузки кэша с помощью утилит perf и free. По окончанию мониторинга можно сделать вывод, что чем больше процессов работает с кэшом, тем больше увеличивается на него нагрузка, увеличивается число кэш-промахов.

Memory

1) Первый параметр

--lockbus N запускает N рабочих процессов, которые быстро блокируют и увеличивают 64 байта случайно выбранной памяти из области mmap размером 16 МБ (только для

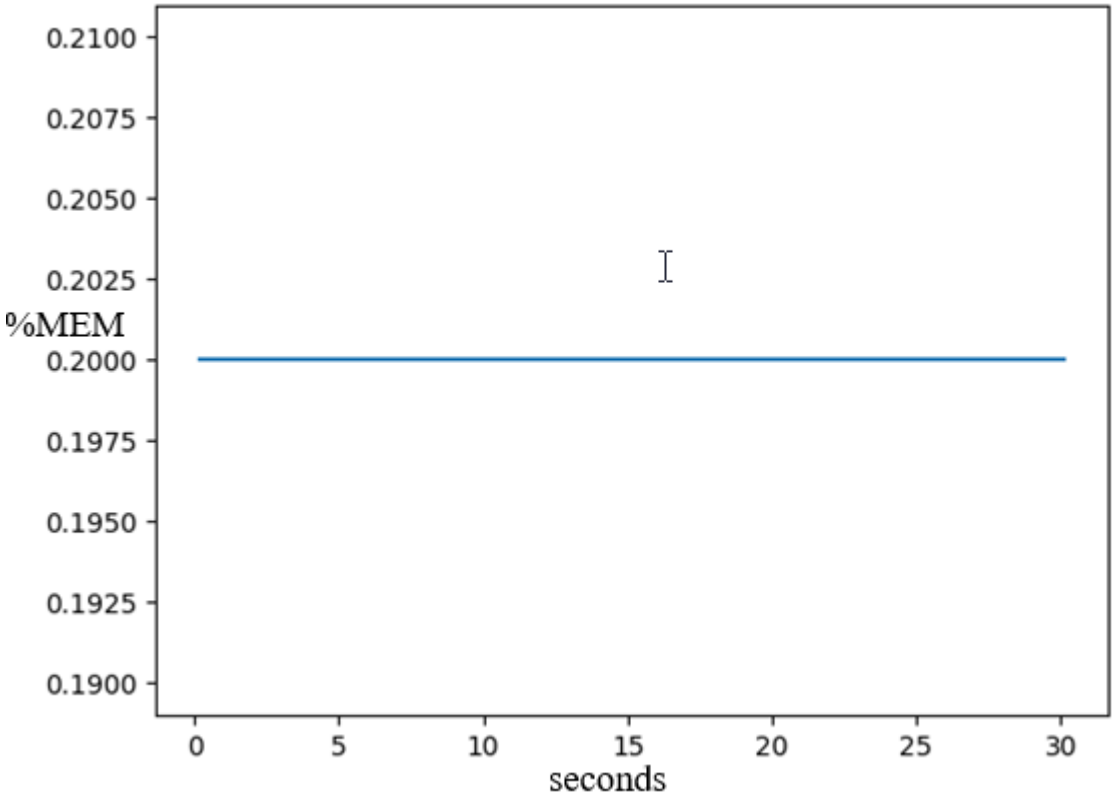
процессоров Intel x86 и ARM). Это приведет к пропускам строк кэша и остановке работы процессоров.

Команда запуска: stress-ng --lockbus 1

Запустим питон скрипт, который с помощью команды **top** считывает нагрузку на память и строит график.

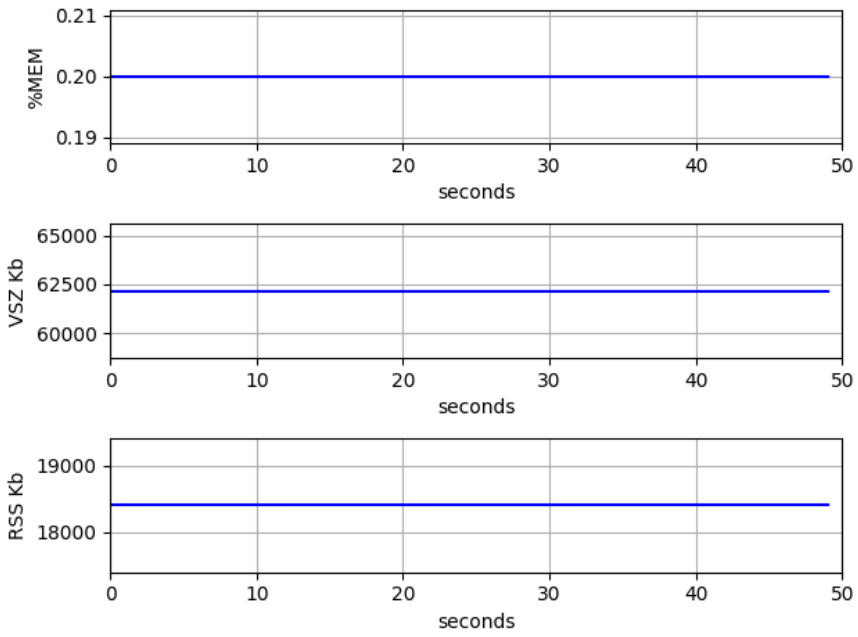
```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --lockbus 1
stress-ng: info:  [6307] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [6307] dispatching hogs: 1 lockbus
```

График:

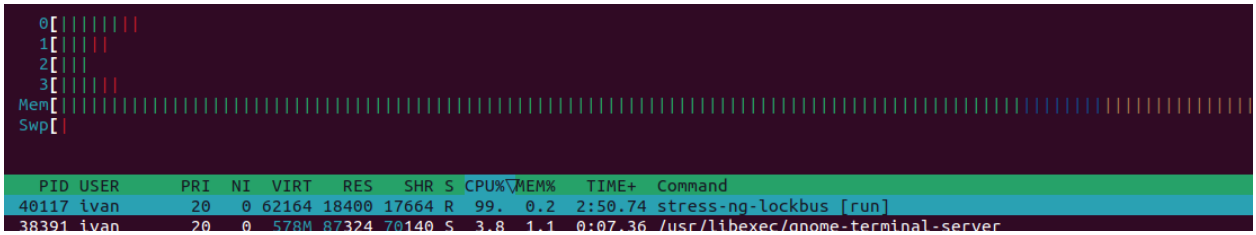


Можем видеть, что потребление памяти константное. Это обуславливается тем, что запущенный стресс-тест блокирует фиксированный размер памяти.

Посмотрим также на график утилиты pidstat:

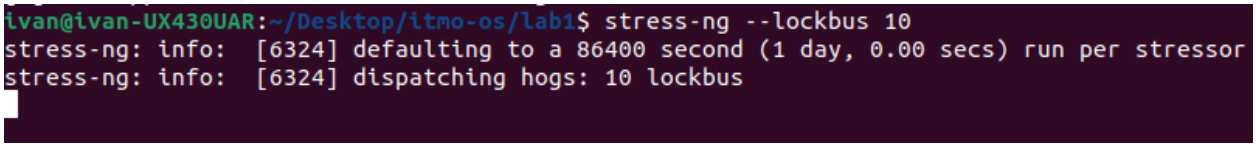


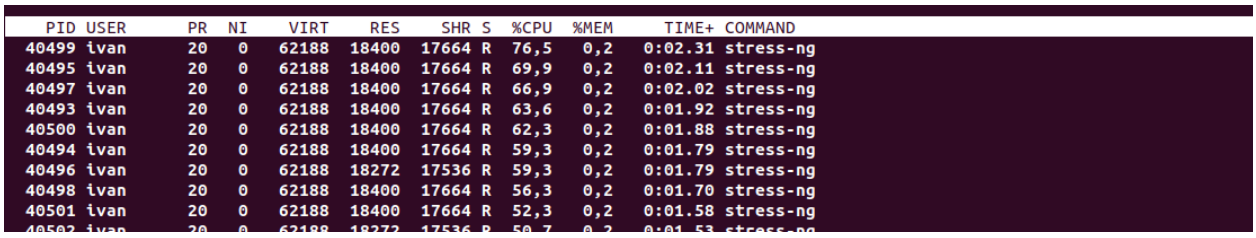
Посмотрим на данный процесс с помощью утилиты htop:



Имеем такие же значения — 0.2.

Попробуем запустить больше процессов:

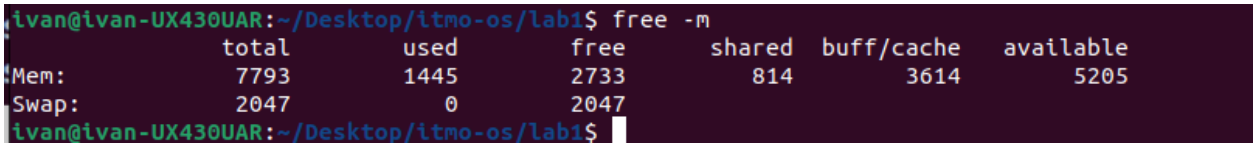




Видно, что появились новые процессы, но каждый из них использует одинаковый процент памяти — 0.2

Проведем мониторинг утилитой free:

Посмотрим сколько свободно памяти, когда стресс тесты не работают:



free — 2733

А теперь запустим тесты и посмотрим изменения.

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --lockbus 10
stress-ng: info:  [6324] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [6324] dispatching hogs: 10 lockbus
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7793        1434        2581         977        3777        5055
Swap:           2047           0        2047
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$
```

Ожидаемо значение параметра free уменьшилось.

2) Второй параметр

--fork-vm включает рекомендации по использованию виртуальной памяти, снижающие производительность, с помощью madvise на всех страницах разветвленного процесса.

Системный вызов madvise выдает предложения ядру об использовании постраничного ввода/вывода.

Команда запуска: stress-ng --vm 1 --fork-vm

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --vm 1 --fork-vm
stress-ng: info:  [6405] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [6405] dispatching hogs: 1 vm
```

Запускаем один тест виртуальной памяти с функцией fork-vm

График (top):

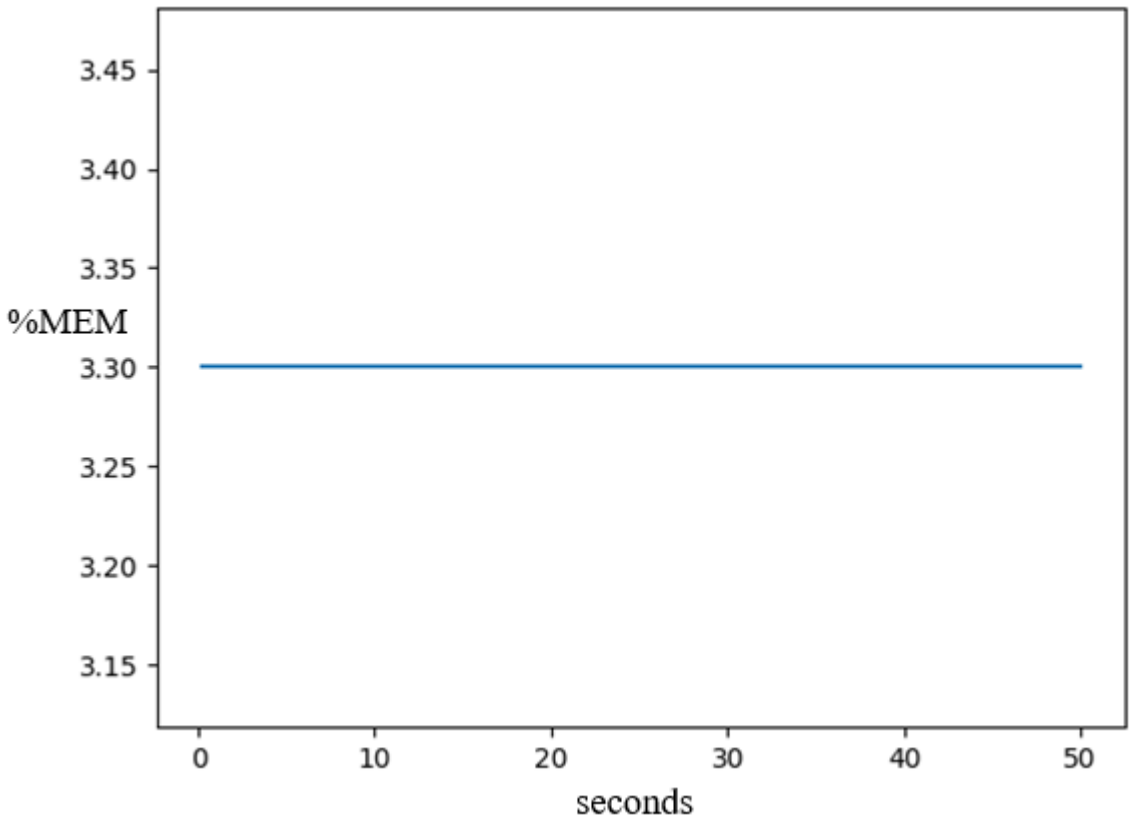
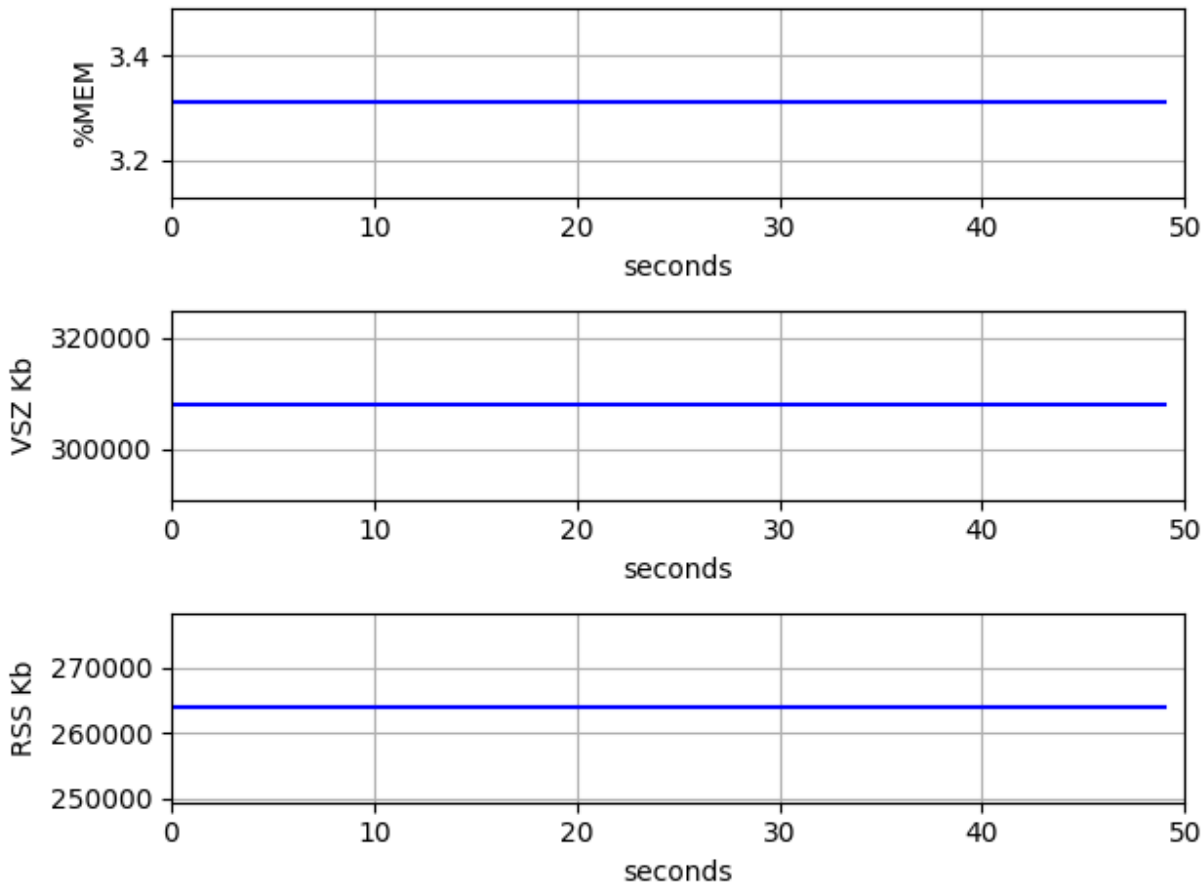
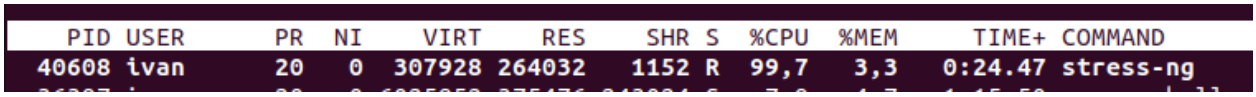
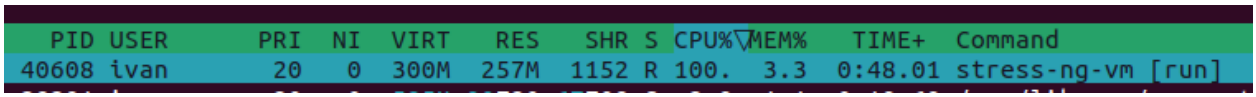

3.45
3.40
3.35
%MEM
3.30
3.25
3.20
3.15
0
10
20
30
40
50
seconds

График (pidstat):



Запустим тесты еще раз и сравним выводы значения загрузки памяти утилит top и htop:





Значения одинаковы.

3) Запуск с двумя параметрами

Команда запуска: stress-ng --vm 1 --fork-vm --lockbus 1

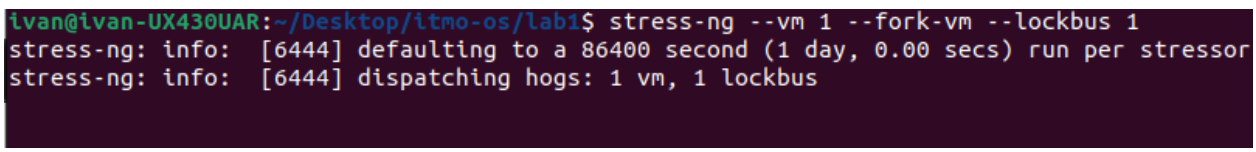
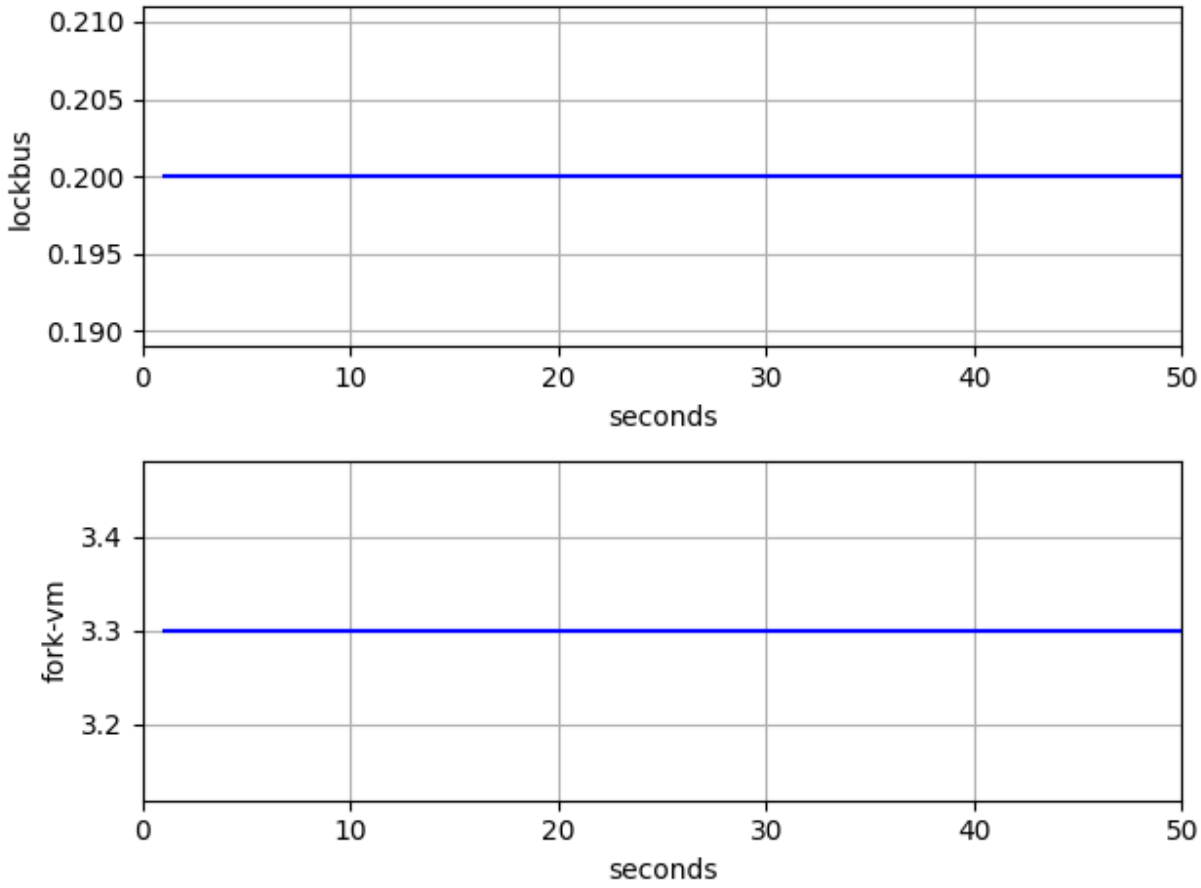


График:



Выводы по мониторингу памяти:

Был произведен мониторинг с помощью двух видов стресс-тестов. Первый параметр всегда выделял одинаковое количество памяти, второй же менял значения. При это процент загрузки памяти зависит от количества запущенных процессов работы с ней.

Утилиты top, pidstat и htop показали примерно равные результаты мониторинга.

Network

1) Первый параметр

-- sockdiag N запускает N рабочих процессов, которые выполняют диагностику сетевых сокетов Linux sock_diag (только для Linux). В настоящее время запрашивается диагностика с использованием UDIAG_SHOW_NAME, UDIAG_SHOW_VFS, UDIAG_SHOW_PEER, UDIAG_SHOW_ICONS, UDIAG_SHOW_RQLEN и UDIAG_SHOW_MEMINFO для семейства сокетных подключений AF_UNIX.

Команда запуска: stress-ng --sockdiag 1

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --sockdiag 1
stress-ng: info:  [6481] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [6481] dispatching hogs: 1 sockdiag
```

Запустим 3 питон скрипта, которые с помощью утилит bmon, ifstat и ip считают сумму полученных бит (RX) и переданных бит (TX) для сравнения утилит.

График bmon:

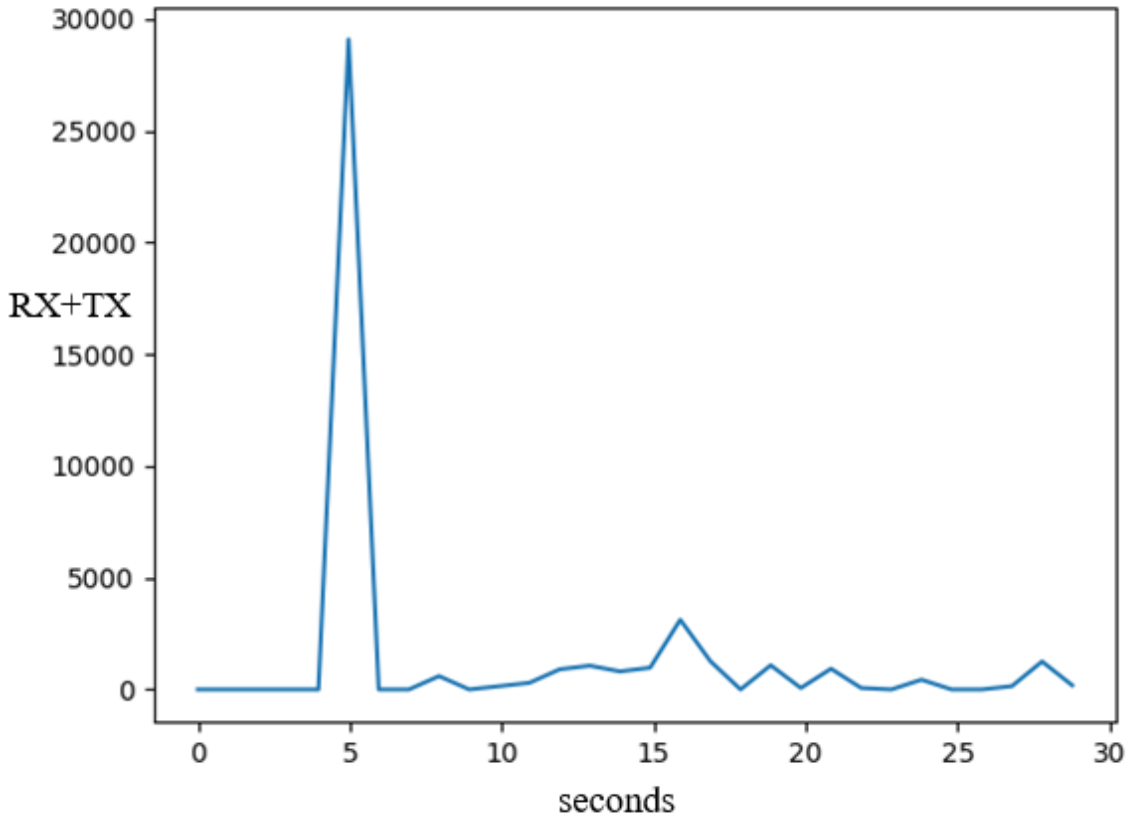


График ip:

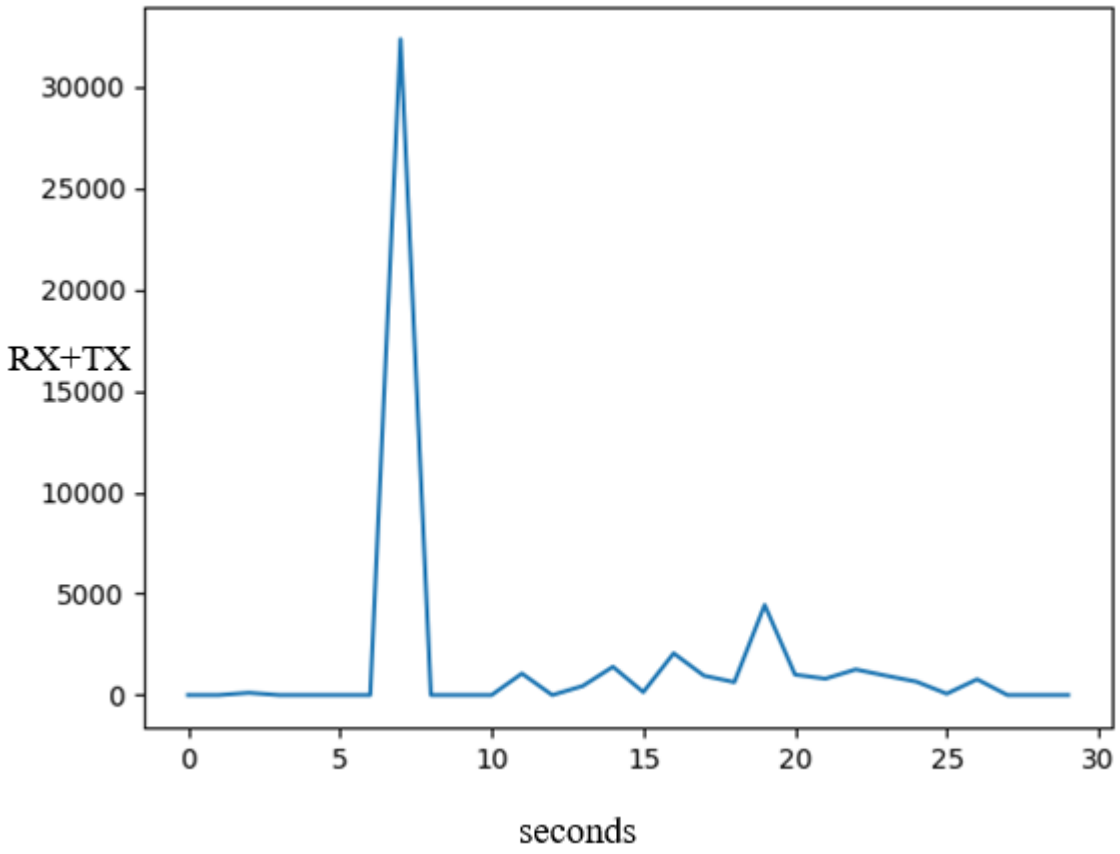
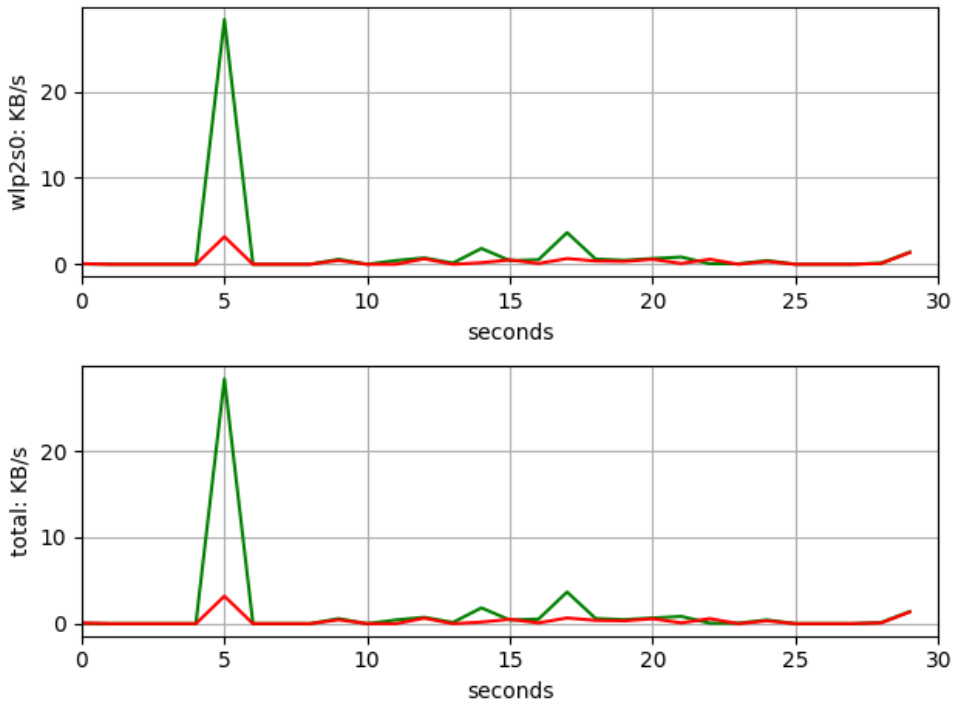


График ifstat:

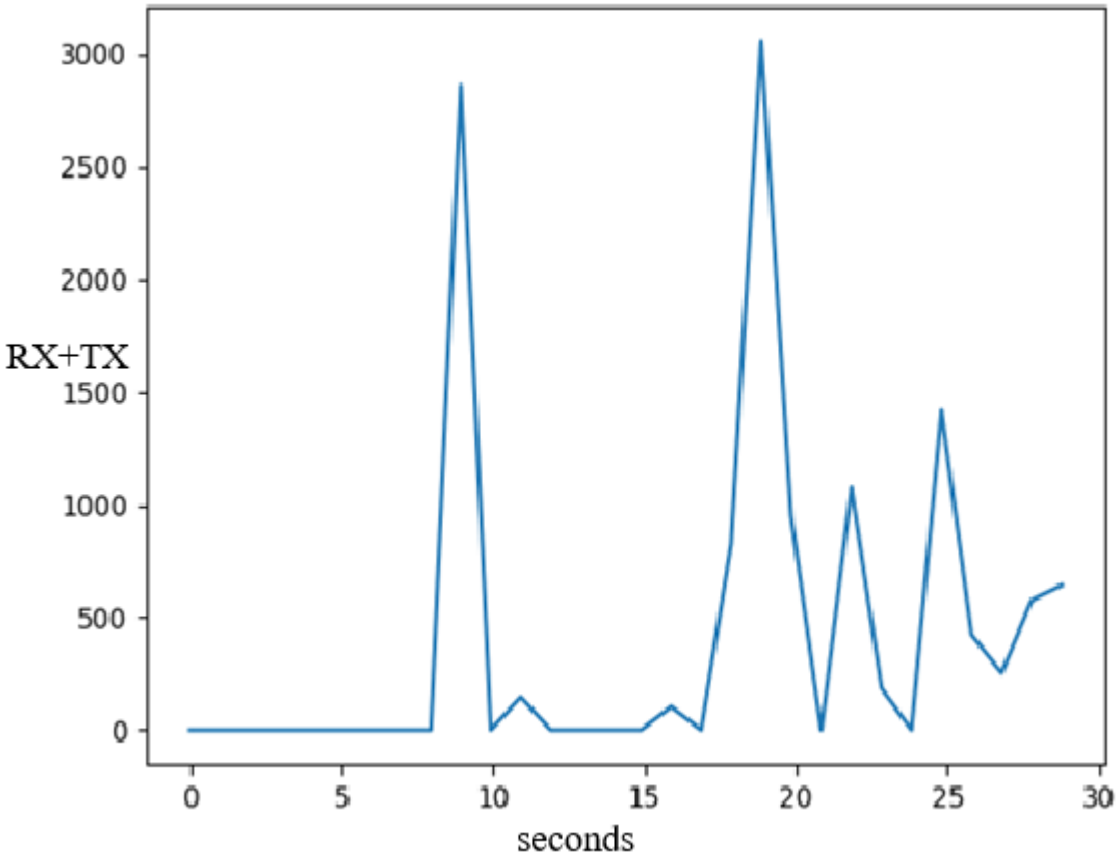


Видим,
что значения одинаковые. (Небольшое смещение по x обуславливается тем, что скрипты запускались вручную друг за другом — то есть была небольшая задержка)

Увеличим количество рабочих процессов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --sockdiag 25
stress-ng: info:  [16873] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [16873] dispatching hogs: 25 sockdiag
```

График:



Можно заметить, что с увеличением количества процессов, уменьшается нагрузка на сетевую подсистему.
Увеличение числа рабочих процессов может привести к уменьшению нагрузки на сетевую подсистему по нескольким причинам:

1. Распределение нагрузки: При увеличении числа рабочих процессов утилиты stress-ng, нагрузка на сетевую подсистему распределяется между этими процессами. Это означает, что каждый процесс будет обрабатывать меньше сетевых запросов и отправлять меньше данных, что в конечном итоге уменьшает общую нагрузку на сетевую подсистему.

2. Конкуренция за ресурсы: Увеличение числа рабочих процессов может привести к конкуренции за ресурсы, например, за доступ к сетевым портам или сетевому интерфейсу. Если есть только один процесс, он может монополизировать доступ к ресурсам и создавать высокую нагрузку на сетевую подсистему. Однако, с увеличением числа рабочих процессов, нагрузка становится более равномерной, что может привести к снижению нагрузки на сеть.

2) Второй параметр

-- **netlink-proc N** запускает N рабочих процессов, которые порождают дочерние процессы и отслеживают события процесса fork/exec/exit через коннектор proc netlink.

Команда запуска: sudo stress-ng --netlink-proc 1

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng --netlink-proc 1
stress-ng: info:  [7296] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [7296] dispatching hogs: 1 netlink-proc
```

График bmon:

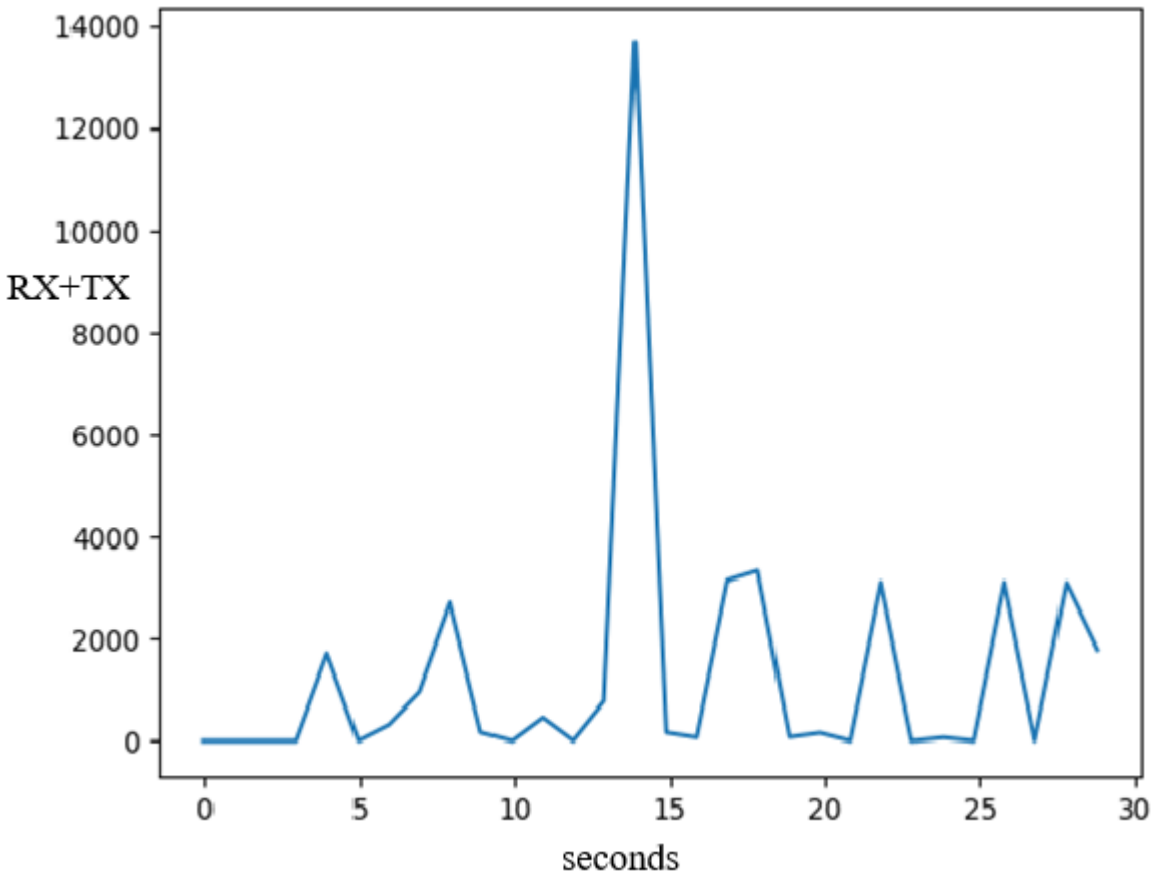


График ifstat:

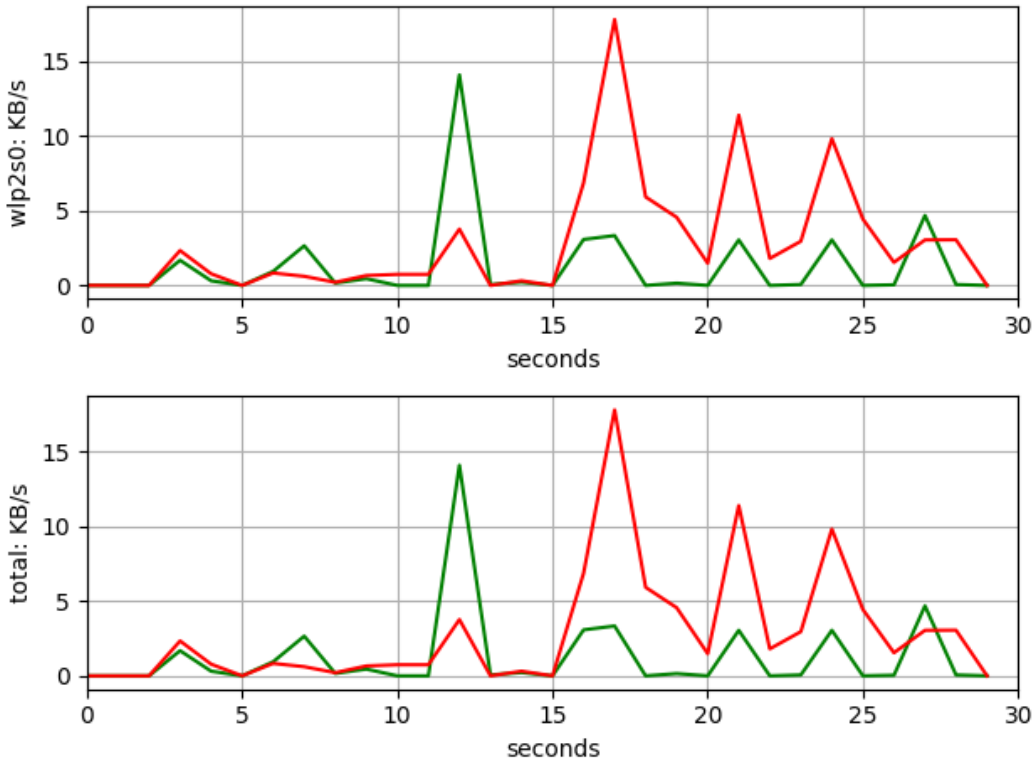
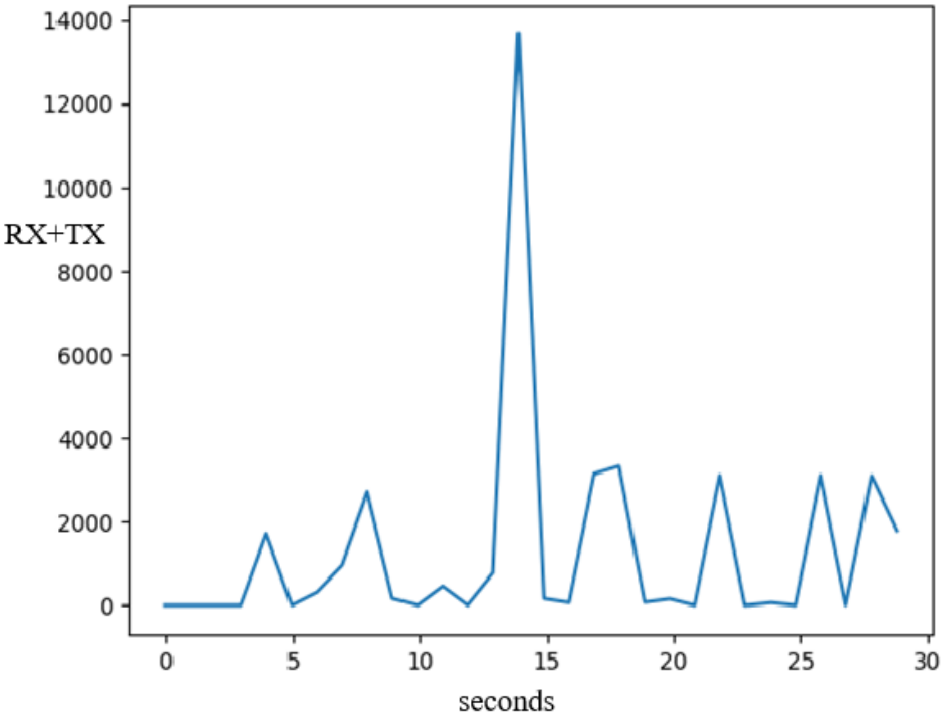


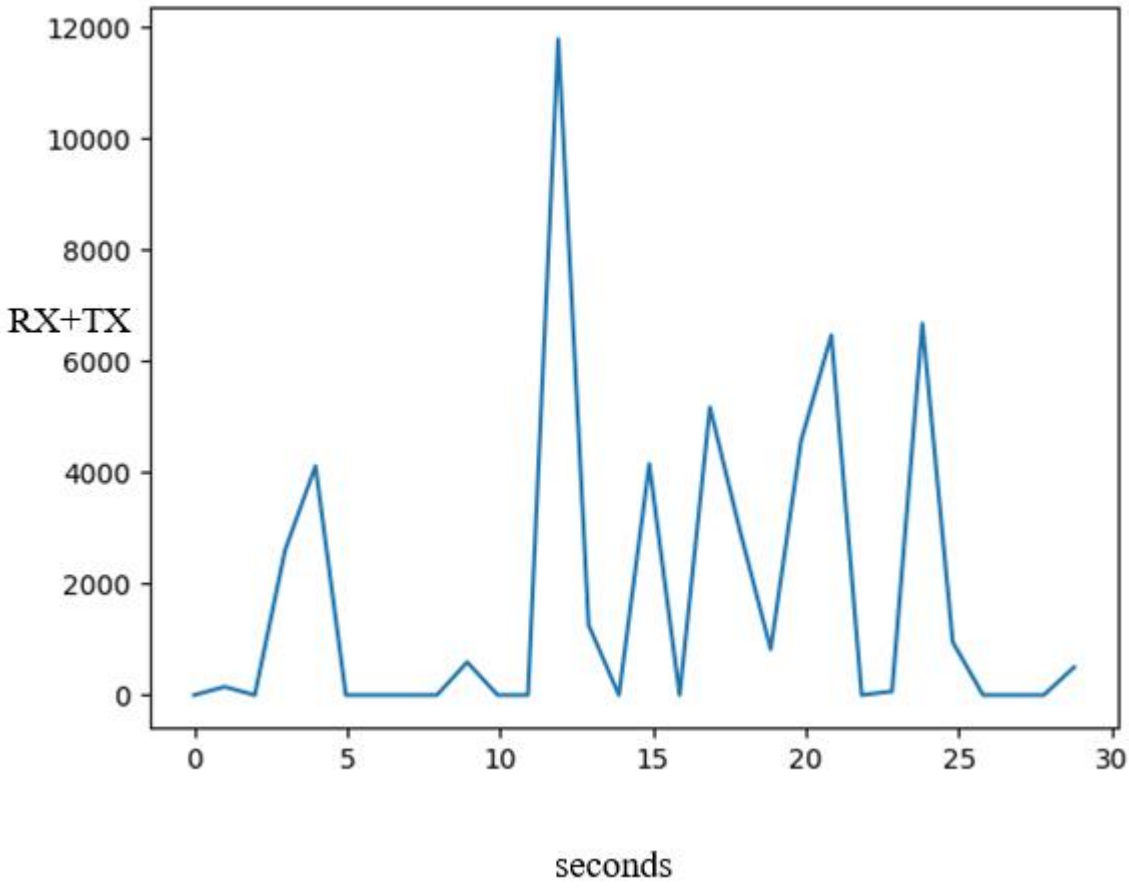
График ip:



Увеличим количество рабочих процессов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng --netlink-proc 100
stress-ng: info:  [76277] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [76277] dispatching hogs: 100 netlink-proc
```

График:



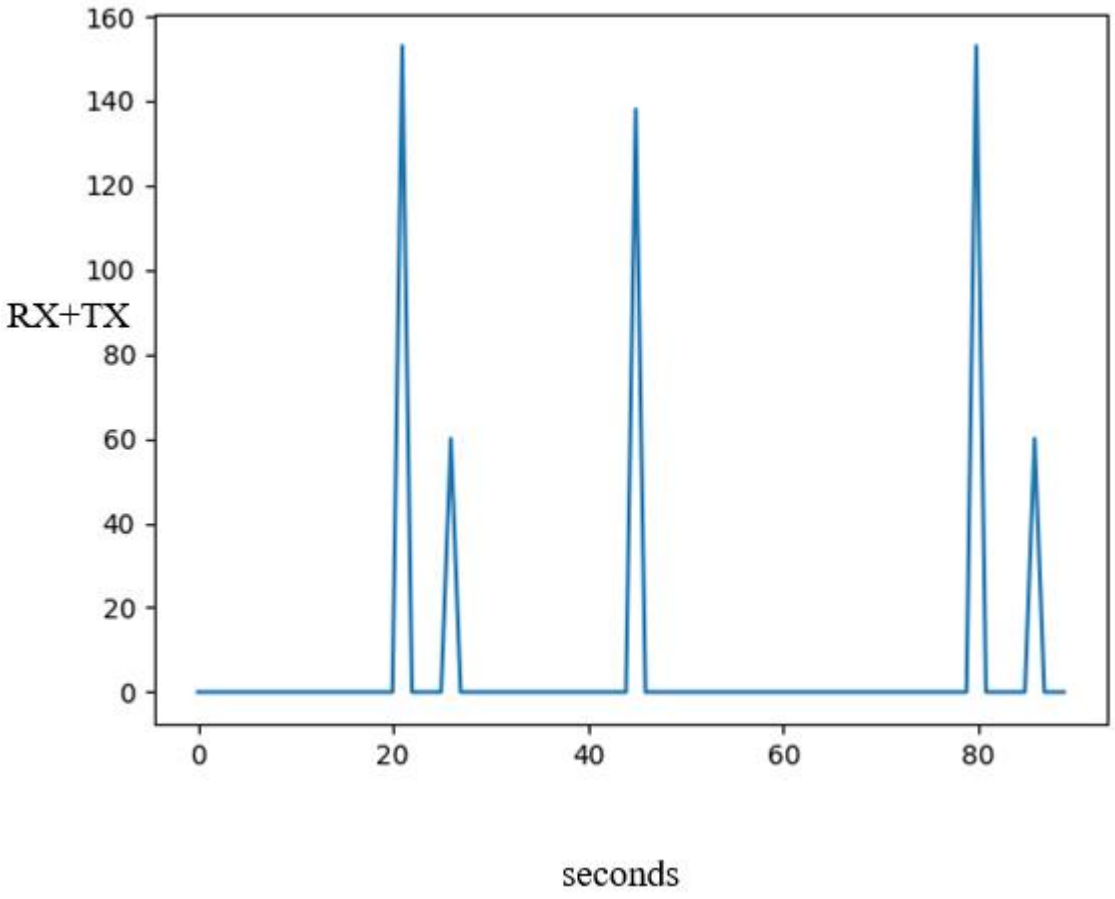
Аналогично, увеличение процессов несет за собой уменьшение нагрузки сетевой подсистемы.

3) Запуск с двумя параметрами

Команда запуска: sudo stress-ng --netlink-proc 10 --sockdiag 10

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng --netlink-proc 10 --sockdiag 10
stress-ng: info:  [423097] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [423097] dispatching hogs: 10 netlink-proc, 10 sockdiag
```

График:



Выводы по мониторингу сетевой подсистемы:

С увеличением количества процессов стресс-теста нагрузка на сетевую подсистему может уменьшаться по причинам, рассмотренным в 1 пункте.

Утилиты bmon, ifstat и ip показали практически одинаковые значения.

IO:

1) Первый параметр

-- iomix N запускает N рабочих процессов, которые выполняют сочетание последовательных, случайных и отображаемых в память операций чтения/записи. Создается несколько дочерних процессов, которые совместно используют один файл и выполняют различные операции ввода-вывода с одним и тем же файлом.

Команда запуска: stress-ng --iomix 1

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --iomix 1
stress-ng: info:  [5599] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5599] dispatching hogs: 1 iomix
```

Запустим iomix с аргументом 1, так как будут создаваться еще дочерние процессы.

Посмотрим на питоновский график процента нагрузки подсистемы ввода-вывода.

Внутри питоновского скрипта используется команда iotop с флагами "-P" "-b" "-n" "-d" и считывается процент загрузки io.

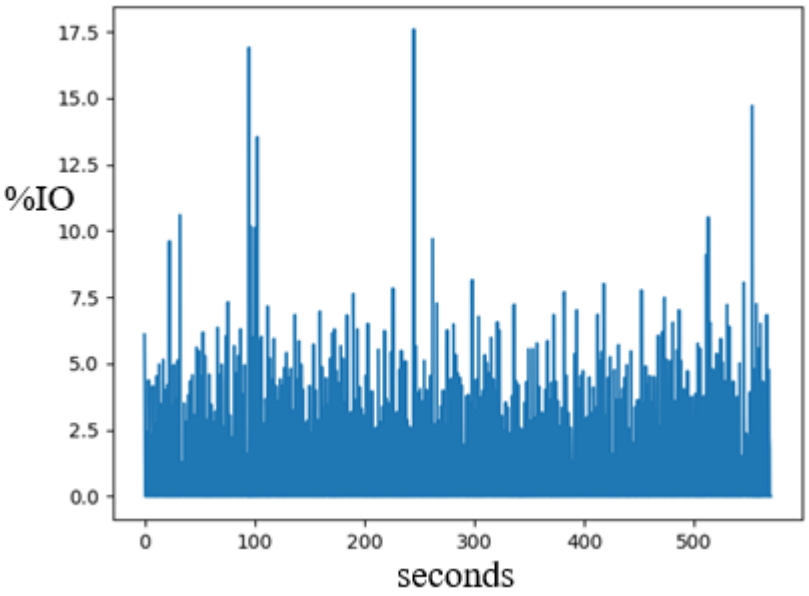
Флаги:

-P: Этот флаг используется для определения фильтрации вывода iotop по процессам.

-b: Этот флаг указывает iotop на использование режима "батч-режима". В этом режиме iotop не выводит интерактивную таблицу, а вместо этого она выводит обновления активности ввода-вывода.

-n: Этот флаг определяет количество итераций.

-d: Этот флаг устанавливает интервал между обновлениями активности ввода-вывода в секундах.



Посмотрим на процессы (top):

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5612	ivan	20	0	46296	1504	768	S	10,9	0,0	0:01.00	stress-ng
5616	ivan	20	0	45784	1120	384	D	7,9	0,0	0:00.89	stress-ng
211	root	20	0	0	0	0	D	6,9	0,0	0:00.89	jbd2/sda5-8
153	root	0	-20	0	0	0	I	2,6	0,0	0:00.37	kworker/6:1H-kblockd
5613	ivan	20	0	45784	1248	512	S	1,7	0,0	0:00.18	stress-ng
5614	ivan	20	0	45784	1120	384	D	1,7	0,0	0:00.18	stress-ng
5619	ivan	20	0	45784	1120	384	S	1,7	0,0	0:00.13	stress-ng
2652	ivan	20	0	3521852	339224	179496	S	1,3	4,3	0:29.50	firefox
2911	ivan	20	0	2463412	108640	78864	S	1,3	1,4	0:06.48	Privileged Cont
5603	ivan	20	0	45784	1120	384	D	1,3	0,0	0:00.13	stress-ng

Можем увидеть, что и в правду плодятся дополнительные дочерние процессы.

Также посмотрим на один процесс, с помощью команды pidstat:

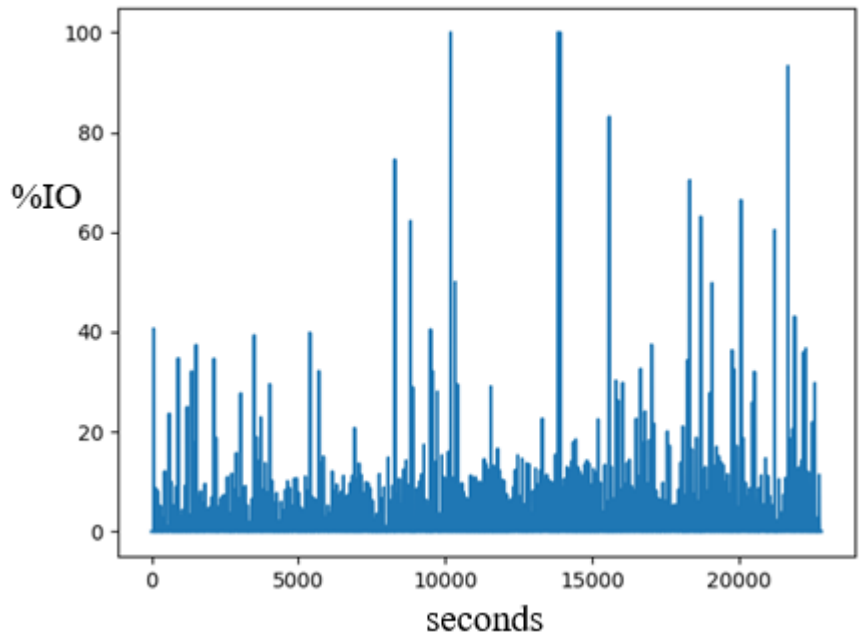
```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -d -p 5613 1
Linux 6.2.0-34-generic (ivan-UX430UAR)  13.10.2023      _x86_64_        (8 CPU)

22:32:21      UID       PID   kB_rd/s   kB_wr/s kB_ccwr/s iodelay  Command
22:32:22     1000      5613      0,00    388,00      0,00       0  stress-ng
22:32:23     1000      5613      0,00    396,00      0,00       0  stress-ng
22:32:24     1000      5613      0,00    372,00      0,00       0  stress-ng
22:32:25     1000      5613      0,00    620,00      0,00       0  stress-ng
22:32:26     1000      5613      0,00    872,00      0,00       0  stress-ng
22:32:27     1000      5613      4,00    388,00      0,00       0  stress-ng
22:32:28     1000      5613      0,00    452,00      0,00       0  stress-ng
22:32:29     1000      5613      0,00    488,00      0,00       0  stress-ng
```

Можем видеть, что чтение/запись идут, но подсистема полностью не загружена. Попробуем увеличить количество рабочих процессов и построим график:

Запущенная команда stress-ng --iomix 20

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --iomix 20
stress-ng: info:  [5695] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [5695] dispatching hogs: 20 iomix
```

График:



Ожидаемо, что с увеличением процессов работы с подсистемой увеличивается и ее загрузка.

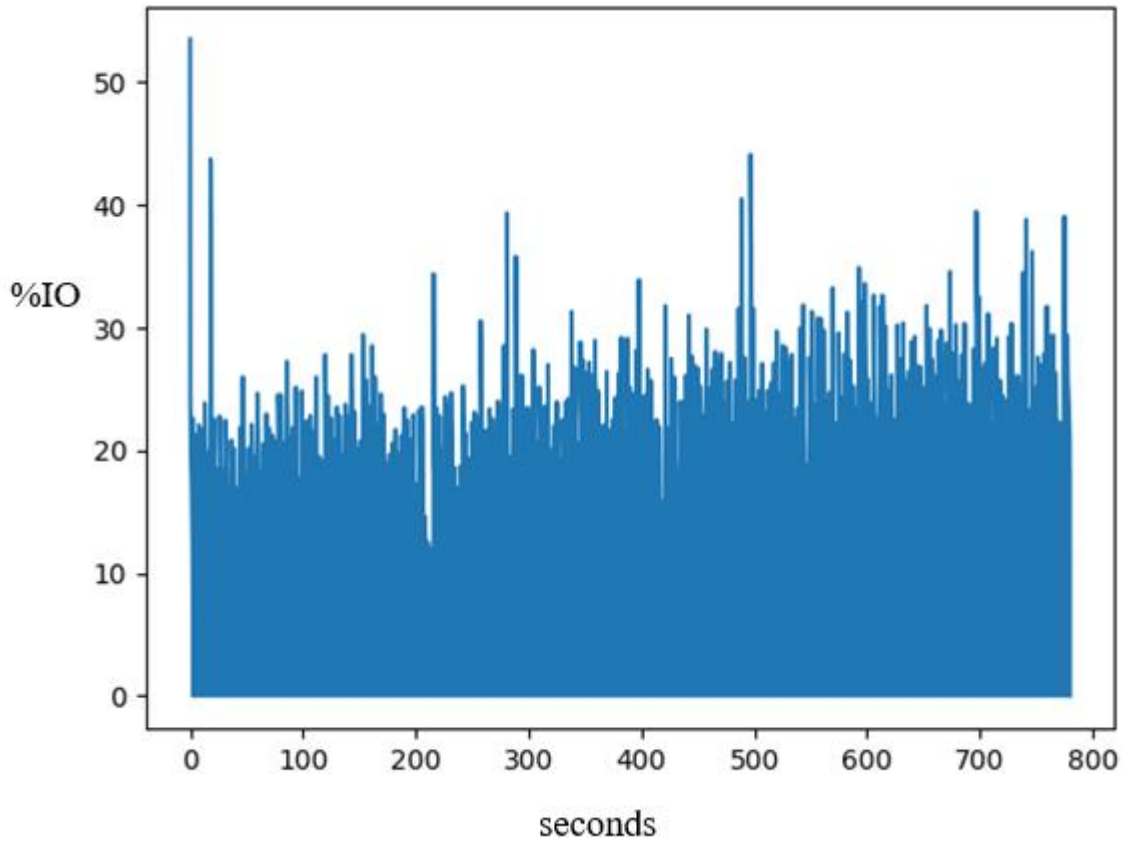
2) Второй параметр

-- ioport N запускает N рабочих процессов, которые выполняют пакеты из 16 операций чтения и 16 операций записи ioport 0x80

Команда запуска: sudo stress-ng --ioport 1

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng --ioport 1
[sudo] password for ivan:
stress-ng: info:  [3125] defaulting to a 86400 second (1 day, 0.00 secs) run per
 stressor
stress-ng: info:  [3125] dispatching hogs: 1 ioport
```

График:



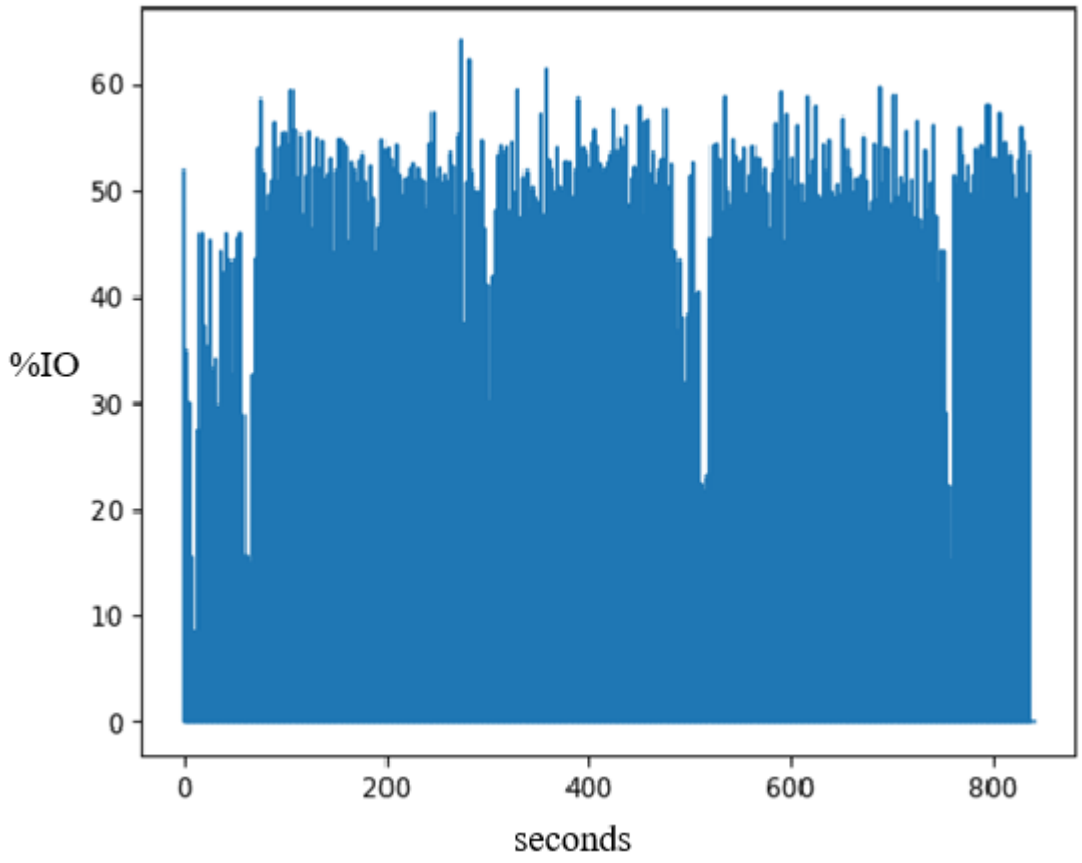
Посмотрим на процесс с помощью pidstat:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo pidstat -d -p 7195 1
[sudo] password for ivan:
Linux 6.2.0-34-generic (ivan-UX430UAR)  17.10.2023      _x86_64_        (8 CPU)

16:05:54      UID       PID   kB_rd/s   kB_wr/s kB_ccwr/s iodelay  Command
16:05:55     1000      7195      0,00   3528,00      0,00      46  stress-ng
16:05:56     1000      7195      0,00   3608,00      0,00      45  stress-ng
16:05:57     1000      7195      0,00   2516,00      0,00      37  stress-ng
16:05:58     1000      7195      0,00   3668,00      0,00      46  stress-ng
16:05:59     1000      7195      0,00   3372,00      0,00      43  stress-ng
16:06:00     1000      7195      0,00   3996,04      0,00      46  stress-ng
16:06:01     1000      7195      0,00   3548,00      0,00      46  stress-ng
```

Увеличим количество рабочих процессов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng --ioport 25
stress-ng: info:  [3152] defaulting to a 86400 second (1 day, 0.00 secs) run per
 stressor
stress-ng: info:  [3152] dispatching hogs: 25 ioport
```

График:



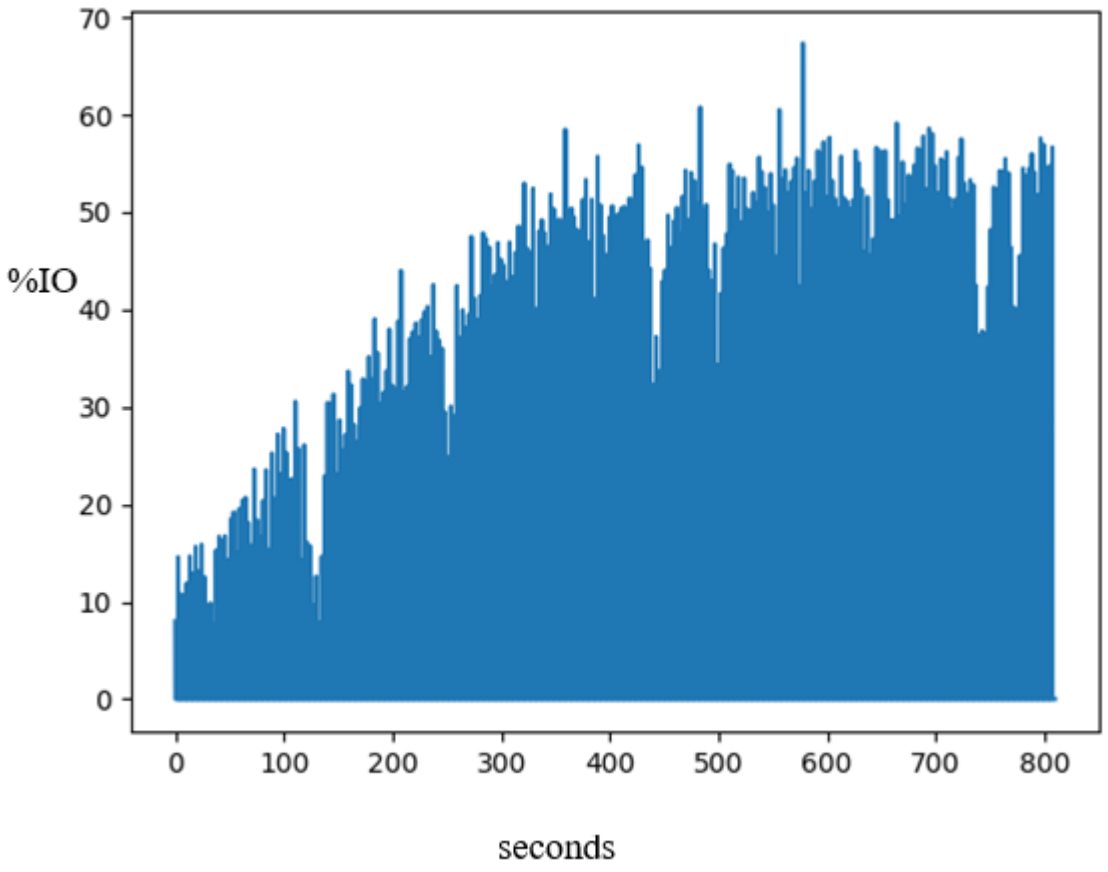
Ожидаемо, что с увеличением процессов работы с подсистемой увеличивается и ее загрузка.

3) Запуск с двумя параметрами

Команда запуска: sudo stress-ng --iomix 1 --ioport 1

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo stress-ng --iomix 1 --ioport 1
stress-ng: info:  [3224] defaulting to a 86400 second (1 day, 0.00 secs) run per
 stressor
stress-ng: info:  [3224] dispatching hogs: 1 iomix, 1 ioport
```

График:



Выводы по мониторингу подсистемы ввода-вывода: Был произведен мониторинг системы ввода/вывода утилитами iotop и pidstat. С увеличением процессов работы с подсистемой увеличивается и ее загрузка.

Pipe

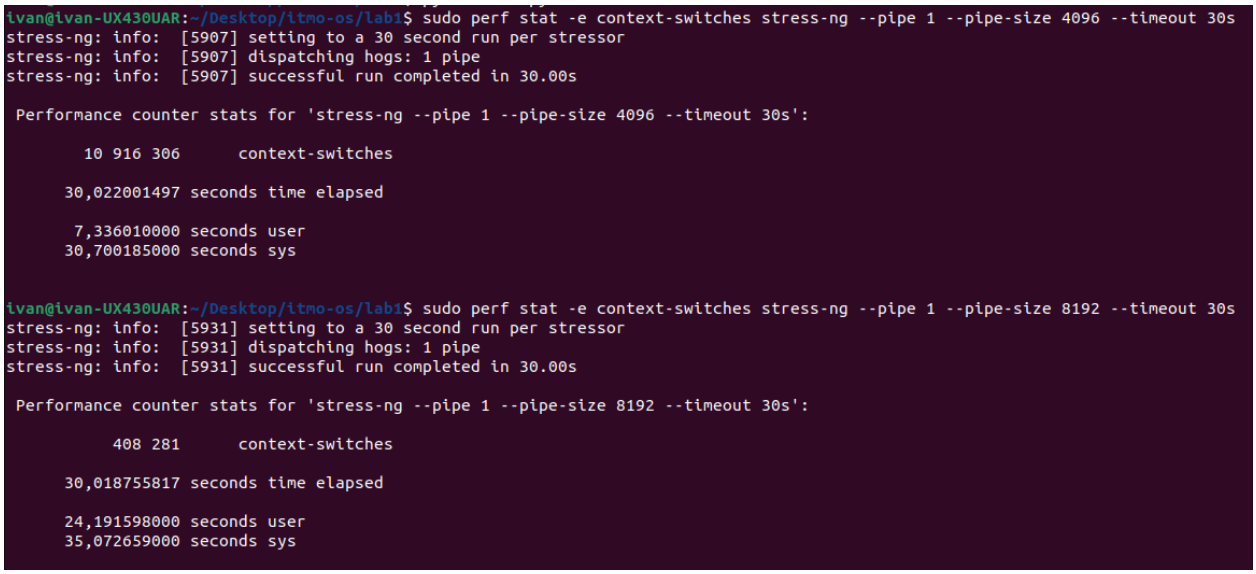
1) Первый параметр

--pipe-size N указывает размер канала в байтах
--pipe N запускает N рабочих процессов, выполняющих операции записи и чтения больших объемов данных по каналу. Осуществляется запись и чтение памяти, а также переключение контекста.

Команда запуска: stress-ng --pipe 1 --pipe-size 4096

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --pipe 1 --pipe-size 4096
stress-ng: info:  [31702] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [31702] dispatching hogs: 1 pipe
```

Будем смотреть статистику по переключению контекста.
Для мониторинг context-switching воспользуемся утилитой perf:
sudo perf stat -e context-switches stress-ng --pipe 1 --pipe-size 4096 --timeout 30s



Можно заметить, что при увеличении размера канала количество переключений контекста уменьшается. При увеличении размера pipe (канала) в операционной системе, число context-switches может уменьшиться из-за оптимизации процесса ввода-вывода и управления процессами. Это может произойти:

1. **Увеличение пропускной способности**: Увеличение размера pipe может увеличить пропускную способность для передачи данных между процессами. Это означает, что больше данных может быть передано за один раз, что уменьшает необходимость в частых контекстных переключениях между процессами.
2. **Снижение ожидания**: Меньший размер pipe может привести к тому, что процессы чаще ожидают доступа к pipe, так как они должны чаще уступать CPU другим процессам. Увеличение размера pipe снижает вероятность ожидания и, следовательно, число контекстных переключений.
3. **Увеличение эффективности**: Увеличение размера pipe может сделать взаимодействие между процессами более эффективным, так как меньше времени будет затрачено на управление самим pipe и контекстными переключениями.

Однако, важно отметить, что слишком большой размер pipe также может иметь негативные эффекты, такие как использование большего объема памяти или потенциальные проблемы с производительностью.

Попробуем изменить количество тестов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e context-switches stress-ng --pipe 8 --pipe-size 4096 --timeout 30s
stress-ng: info:  [6127] setting to a 30 second run per stressor
stress-ng: info:  [6127] dispatching hogs: 8 pipe
stress-ng: info:  [6127] successful run completed in 30.01s

 Performance counter stats for 'stress-ng --pipe 8 --pipe-size 4096 --timeout 30s':

        43 970 126      context-switches

      30,030495394 seconds time elapsed

      39,340187000 seconds user
     183,864308000 seconds sys
```

Количество стресс-тестов увеличивает количество переключений контекста.

Посмотрим на как параметры влияют на количество переключений контекстов в секунду с помощью утилиты pidstat:
Запустим один тест:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7816
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:46:21      UID       PID   cswch/s nvcswch/s  Command
16:46:21     1000      7816    164,17    201,08  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7816
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:46:24      UID       PID   cswch/s nvcswch/s  Command
16:46:24     1000      7816    200,73    249,30  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7816
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:46:27      UID       PID   cswch/s nvcswch/s  Command
16:46:27     1000      7816    230,48    287,25  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7816
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:46:27      UID       PID   cswch/s nvcswch/s  Command
16:46:27     1000      7816    240,14    297,45  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7816
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:46:28      UID       PID   cswch/s nvcswch/s  Command
16:46:28     1000      7816    248,17    305,66  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7816
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:46:29      UID       PID   cswch/s nvcswch/s  Command
16:46:29     1000      7816    254,95    315,13  stress-ng
```

Увеличим размер канала:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7860
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:47:03      UID       PID   cswch/s nvcswch/s  Command
16:47:03     1000      7860      5,21      0,03  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7860
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:47:05      UID       PID   cswch/s nvcswch/s  Command
16:47:05     1000      7860      6,97      0,03  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7860
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:47:06      UID       PID   cswch/s nvcswch/s  Command
16:47:06     1000      7860      7,43      0,03  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7860
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:47:07      UID       PID   cswch/s nvcswch/s  Command
16:47:07     1000      7860      7,88      0,03  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7860
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:47:08      UID       PID   cswch/s nvcswch/s  Command
16:47:08     1000      7860      8,29      0,03  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7860
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:47:08      UID       PID   cswch/s nvcswch/s  Command
16:47:08     1000      7860      8,96      0,03  stress-ng
```

Запустим 8 тестов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7910
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:49:24      UID       PID   cswch/s nvcswch/s  Command
16:49:24     1000      7910    531,27    584,22  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7910
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:49:25      UID       PID   cswch/s nvcswch/s  Command
16:49:25     1000      7910    541,50    594,04  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7910
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:49:26      UID       PID   cswch/s nvcswch/s  Command
16:49:26     1000      7910    547,81    602,22  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7910
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:49:27      UID       PID   cswch/s nvcswch/s  Command
16:49:27     1000      7910    556,52    613,45  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7910
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:49:27      UID       PID   cswch/s nvcswch/s  Command
16:49:27     1000      7910    561,93    618,95  stress-ng
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7910
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:49:28      UID       PID   cswch/s nvcswch/s  Command
16:49:28     1000      7910    574,09    630,05  stress-ng
```

Заметим, что увеличение размера канала также уменьшают количество переключений контекстов в секунду, но увеличение числа стресс-тестов увеличивает количество переключений контекстов в секунду.

2) Второй параметр

-- pipeherd-yield принудительно выполняет планирование после каждой записи, это увеличивает скорость переключения контекста.

Команда запуска: stress-ng --pipe 1 --pipeherd-yield

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --pipe 1 --pipeherd-yield
stress-ng: info:  [42665] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [42665] dispatching hogs: 1 pipe
```

Будем смотреть статистику по переключению контекста.

Для мониторинг context-switching воспользуемся утилитой perf:
sudo perf stat -e context-switches stress-ng --pipe 1 --pipeherd-yield --timeout 30s

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e context-switches stress-ng --pipe 1 --pipeherd-yield --timeout 30s
[sudo] password for ivan:
stress-ng: info:  [6997] setting to a 30 second run per stressor
stress-ng: info:  [6997] dispatching hogs: 1 pipe
stress-ng: info:  [6997] successful run completed in 30.00s

 Performance counter stats for 'stress-ng --pipe 1 --pipeherd-yield --timeout 30s':

          978 653      context-switches

      30,008951044 seconds time elapsed

      23,568746000 seconds user
      34,562634000 seconds sys


ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e context-switches stress-ng --pipe 8 --pipeherd-yield --timeout 30s
stress-ng: info:  [7028] setting to a 30 second run per stressor
stress-ng: info:  [7028] dispatching hogs: 8 pipe
stress-ng: info:  [7028] successful run completed in 30.01s

 Performance counter stats for 'stress-ng --pipe 8 --pipeherd-yield --timeout 30s':

        41 813 740      context-switches

      30,035249402 seconds time elapsed

      53,969814000 seconds user
     168,026549000 seconds sys
```

Количество стресс-тестов увеличивает количество переключений контекста.

Посмотрим на как параметры влияют на количество переключений контекстов в секунду с помощью утилиты pidstat:
Запустим один тест:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7234
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:32:02      UID       PID   cswch/s nvcswch/s  Command
16:32:02     1000      7234     14,60    451,12  stress-ng
```

Запустим 8 тестов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ pidstat -w -p 7253
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

16:32:46      UID       PID   cswch/s nvcswch/s  Command
16:32:46     1000      7253    137,02      0,03  stress-ng
```

Можно увидеть, что количество переключений контекстов в секунду увеличивается при увеличении количества стресс-тестов.

3) Запуск с двумя параметрами

Команда запуска: sudo perf stat -e context-switches stress-ng --pipe 1 --pipe-size 4096 --pipeherd-yield --timeout 30s

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ sudo perf stat -e context-switches stress-ng --pipe 1 --pipe-size 4096 --pipeherd-yield --timeout 30s
[sudo] password for ivan:
Sorry, try again.
[sudo] password for ivan:
stress-ng: info:  [8527] setting to a 30 second run per stressor
stress-ng: info:  [8527] dispatching hogs: 1 pipe
stress-ng: info:  [8527] successful run completed in 30.00s

 Performance counter stats for 'stress-ng --pipe 1 --pipe-size 4096 --pipeherd-yield --timeout 30s':

        11 424 314      context-switches

      30,014105814 seconds time elapsed

       7,254212000 seconds user
      30,860172000 seconds sys
```

Выводы по мониторингу канальной подсистемы: По результатам мониторинга можно сделать вывод, что увеличение размера канала уменьшают количество переключений

контекстов в секунду, но увеличение числа стресс-тестов увеличивает количество переключений контекстов в секунду.
Утилиты pidstat и perf обе показали похожие данные и помогли прийти к одинаковым выводам.

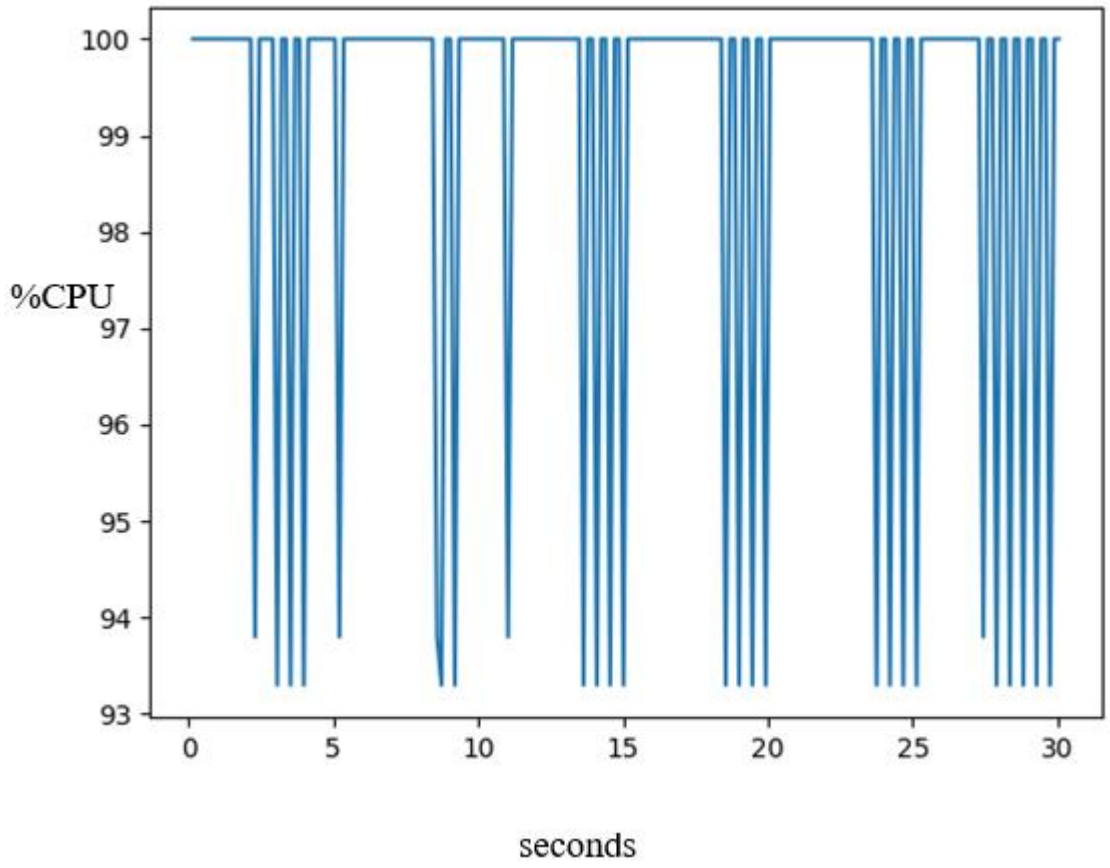
Sched

1) Первый параметр

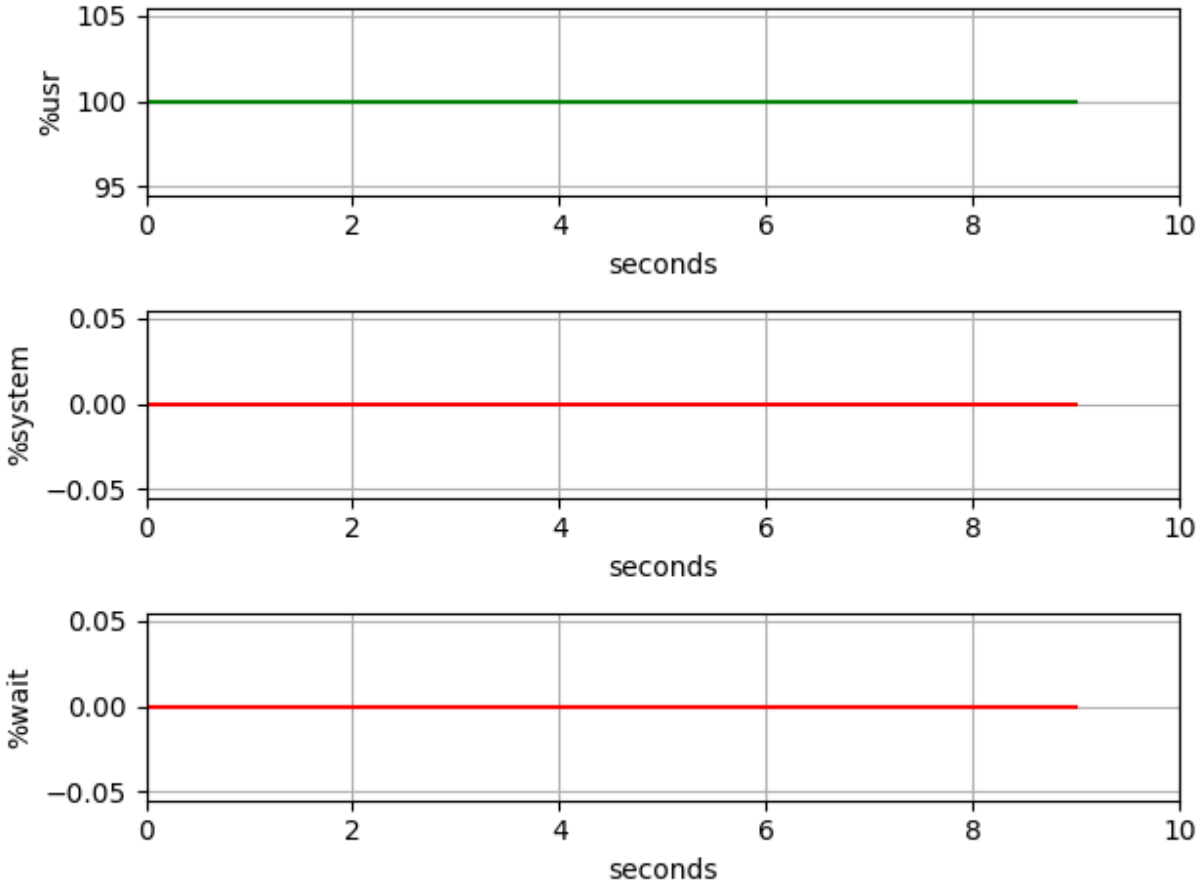
-- sched-runtime - выбирает параметр времени выполнения для планировщика сроков .
Значение по умолчанию — 99999 (в наносекундах).
Команда запуска: stress-ng --cpu 1 --sched-runtime 5000

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 1 --sched-runtime 5000
stress-ng: info:  [11315] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [11315] dispatching hogs: 1 cpu
```

Построим график загрузки процессора утилитой top:

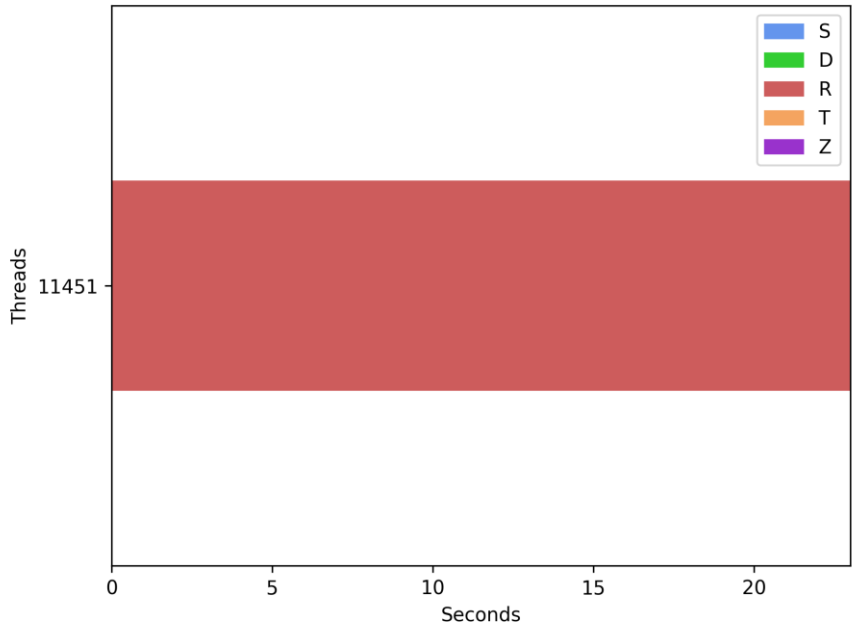


Посмотрим график процента ожидания утилитой pidstat:



Процессор не простаивает в ожидании. Процесс загружает ядро на 100%, так же как и показала утилита top.

Посмотрим в каком состоянии находится процесс:



Всегда в Running.

Посмотрим также загрузку всего процессора утилитой mpstat:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ mpstat 1 5
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

18:18:25     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
18:18:26     all   13,36    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,64
18:18:27     all   13,14    0,00    0,00    0,88    0,00    0,00    0,00    0,00    0,00   85,98
18:18:28     all   13,14    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00   86,61
18:18:29     all   13,03    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,97
18:18:30     all   13,25    0,00    0,12    0,00    0,00    0,00    0,00    0,00    0,00   86,62
Average:     all   13,18    0,00    0,08    0,18    0,00    0,00    0,00    0,00    0,00   86,56
```

Заметим, что процессор занят всего на 13% пользовательским процессом.

Также посмотрим статистику планировщика:

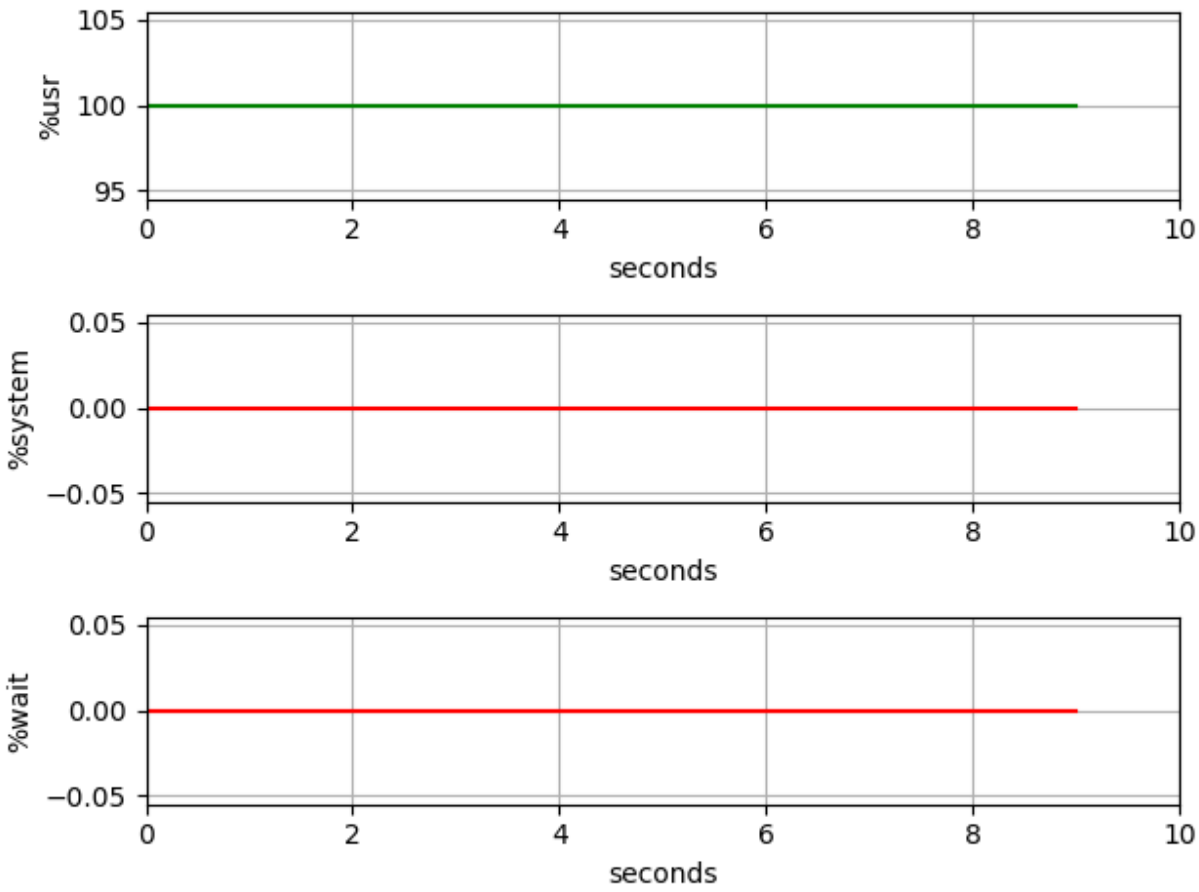
```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/11451/schedstat
147565129399 12895154 1170
```

Первый параметр показывает время работы на процессоре, второй — время ожидания.

Попробуем увеличить время выполнения для планировщика:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 1 --sched-runtime 99999
stress-ng: info:  [11544] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [11544] dispatching hogs: 1 cpu
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ mpstat 1 5
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

18:24:34     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
18:24:35     all   13,48    0,00    0,12    0,87    0,00    0,00    0,00    0,00    0,00   85,52
18:24:36     all   13,27    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,73
18:24:37     all   13,23    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00   86,52
18:24:38     all   12,66    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   87,34
18:24:39     all   13,14    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,86
Average:     all   13,16    0,00    0,08    0,18    0,00    0,00    0,00    0,00    0,00   86,59
```

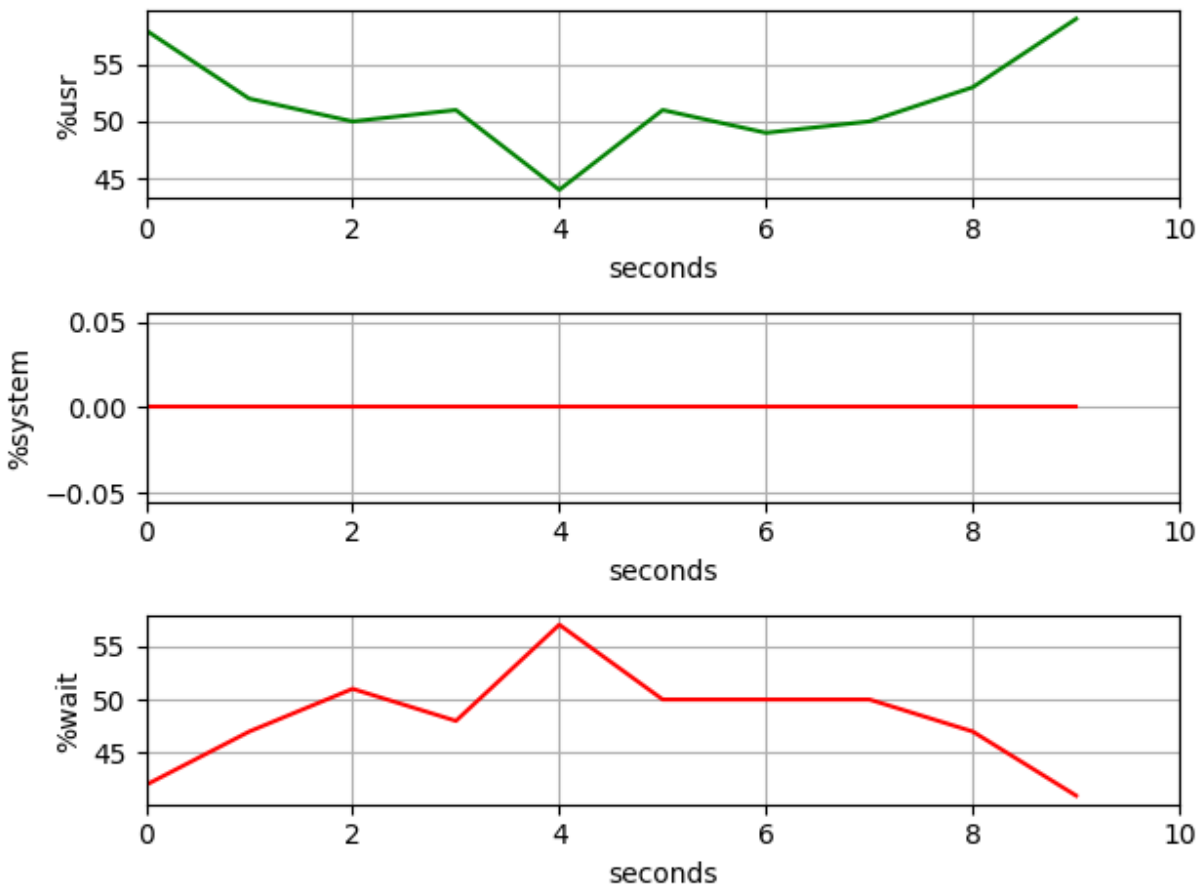



```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/11545/schedstat
74709713933 2997537 446
```

Статистика по нагрузке процессора не меняется, а вот время ожидания сокращается, так как увеличение этого параметра фактически увеличивает максимальное время выполнения процесса, что может привести к уменьшению времени ожидания процесса.

Попробуем запустить несколько процессов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 15 --sched-runtime 5000
stress-ng: info:  [11661] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [11661] dispatching hogs: 15 cpu
```




```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/11672/schedstat
81526743911 78607329840 9578
```

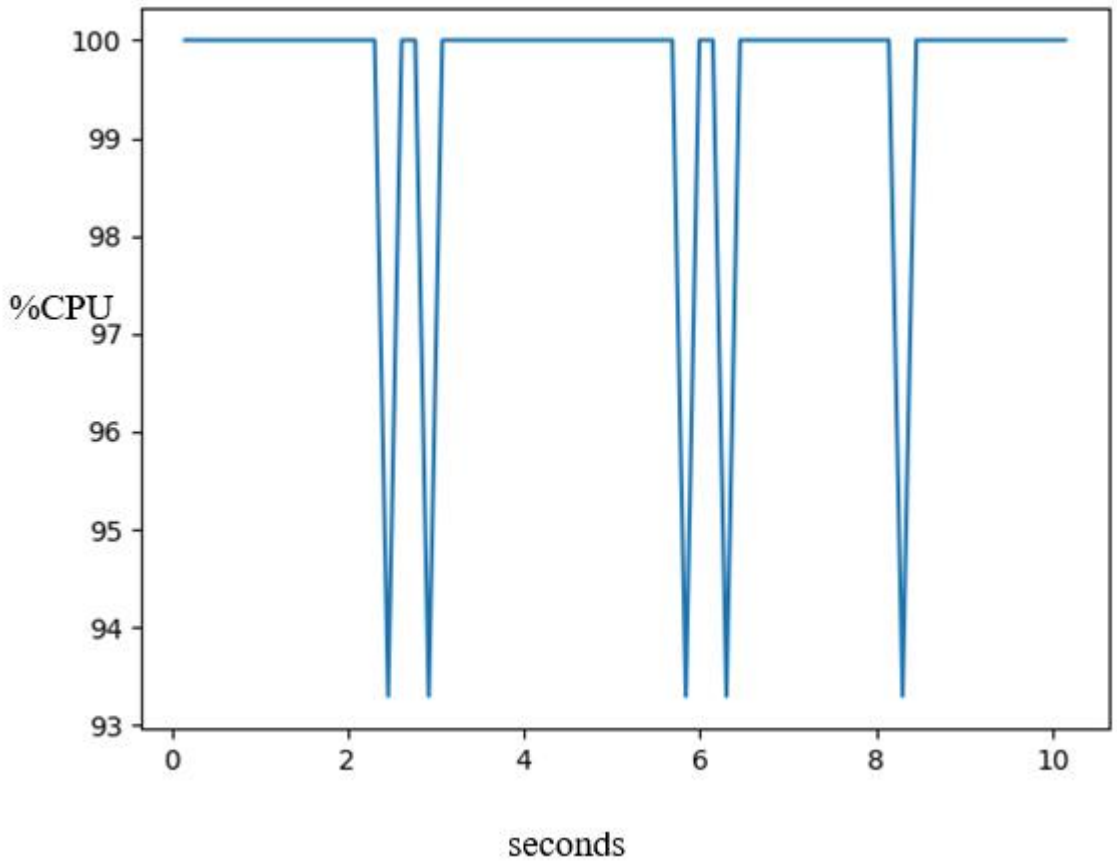
Ожидаемо увеличивается нагрузка на процессор и время ожидания процесса, также увеличивается и процент ожидания.

2) Второй параметр

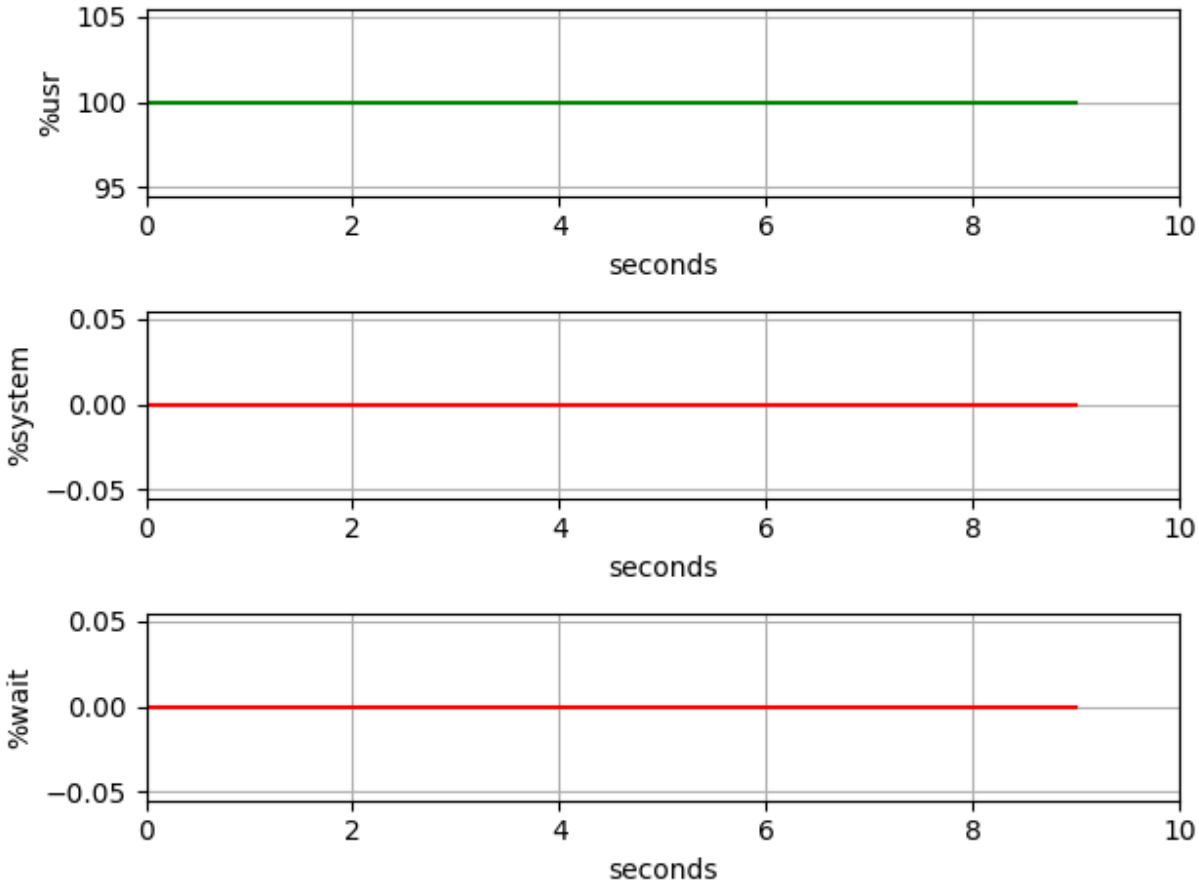
--sched-prio выбирает уровень приоритета планировщика (только в Linux). Если планировщик не поддерживает это, то выбирается уровень приоритета по умолчанию 0.
Команда запуска: stress-ng --cpu 1 --sched-prio 1

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 1 --sched-prio 1
stress-ng: info:  [12473] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [12473] dispatching hogs: 1 cpu
```

Построим график загрузки процессора утилитой top:

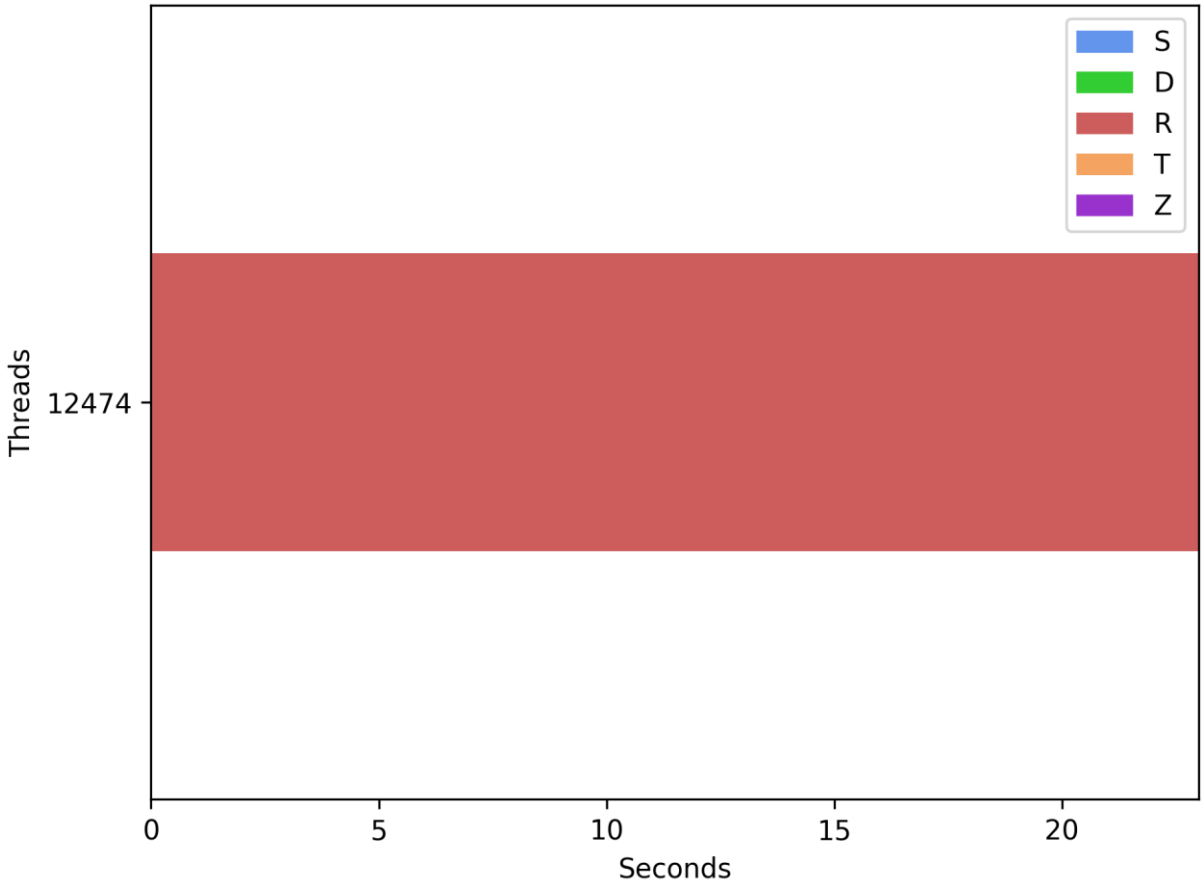


Посмотрим график процента ожидания утилитой pidstat:



Процессор не простаивает в ожидании. Процесс загружает ядро на 100%, так же как и показала утилита top.

Посмотрим в каком состоянии находится процесс:



Всегда в Running.

Посмотрим также загрузку процессора утилитой mpstat:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ mpstat 1 5
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

18:46:58     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
18:46:59     all   13,28    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00   86,47
18:47:00     all   13,59    0,00    0,50    0,00    0,00    0,00    0,00    0,00    0,00   85,91
18:47:01     all   13,88    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00   85,88
18:47:02     all   13,00    0,00    0,50    0,00    0,00    0,00    0,00    0,00    0,00   86,50
18:47:03     all   13,30    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,70
Average:     all   13,41    0,00    0,30    0,00    0,00    0,00    0,00    0,00    0,00   86,29
```

Аналогично заметим, что процессор занят всего на 13% пользовательским процессом.

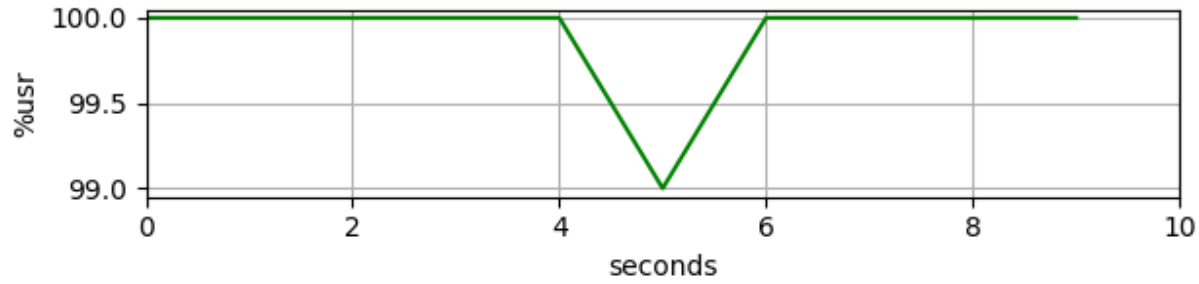
Также посмотрим статистику планировщика:

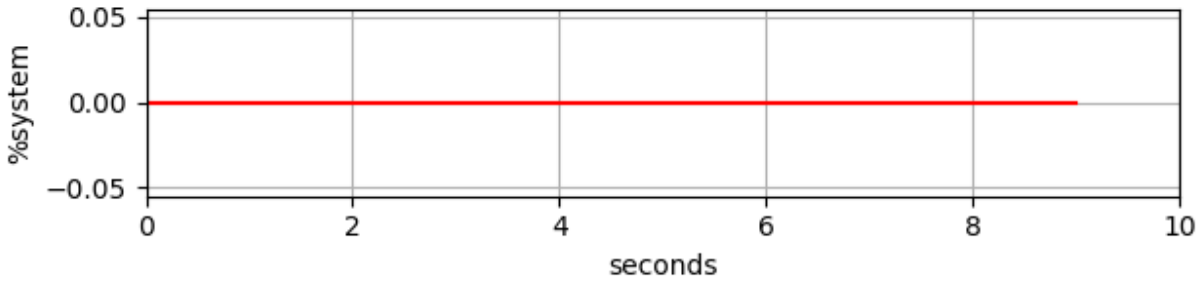
```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/12474/schedstat
76821649025 2524143 330
```

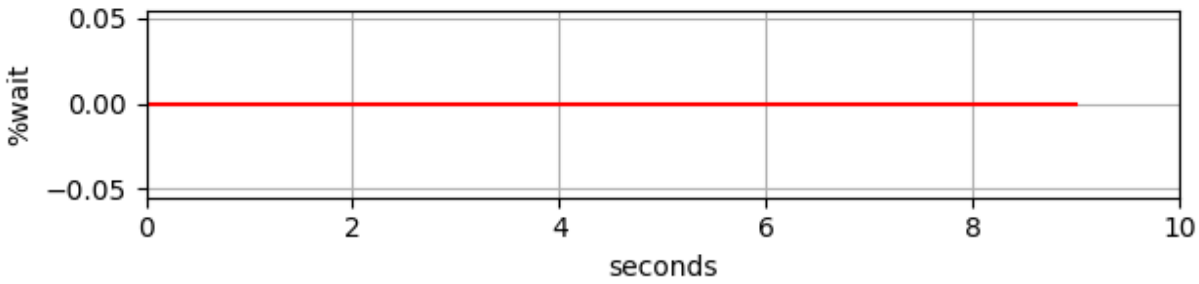
Попробуем увеличить приоритет:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 1 --sched-prio 20
stress-ng: info:  [12952] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [12952] dispatching hogs: 1 cpu
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ mpstat 1 5
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

19:11:45     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
19:11:46     all   13,30    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,70
19:11:47     all   13,03    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00   86,72
19:11:48     all   13,34    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00   86,41
19:11:49     all   13,03    0,00    0,13    0,00    0,00    0,00    0,00    0,00    0,00   86,84
19:11:50     all   13,52    0,00    0,13    0,00    0,00    0,00    0,00    0,00    0,00   86,36
Average:     all   13,24    0,00    0,15    0,00    0,00    0,00    0,00    0,00    0,00   86,60
```





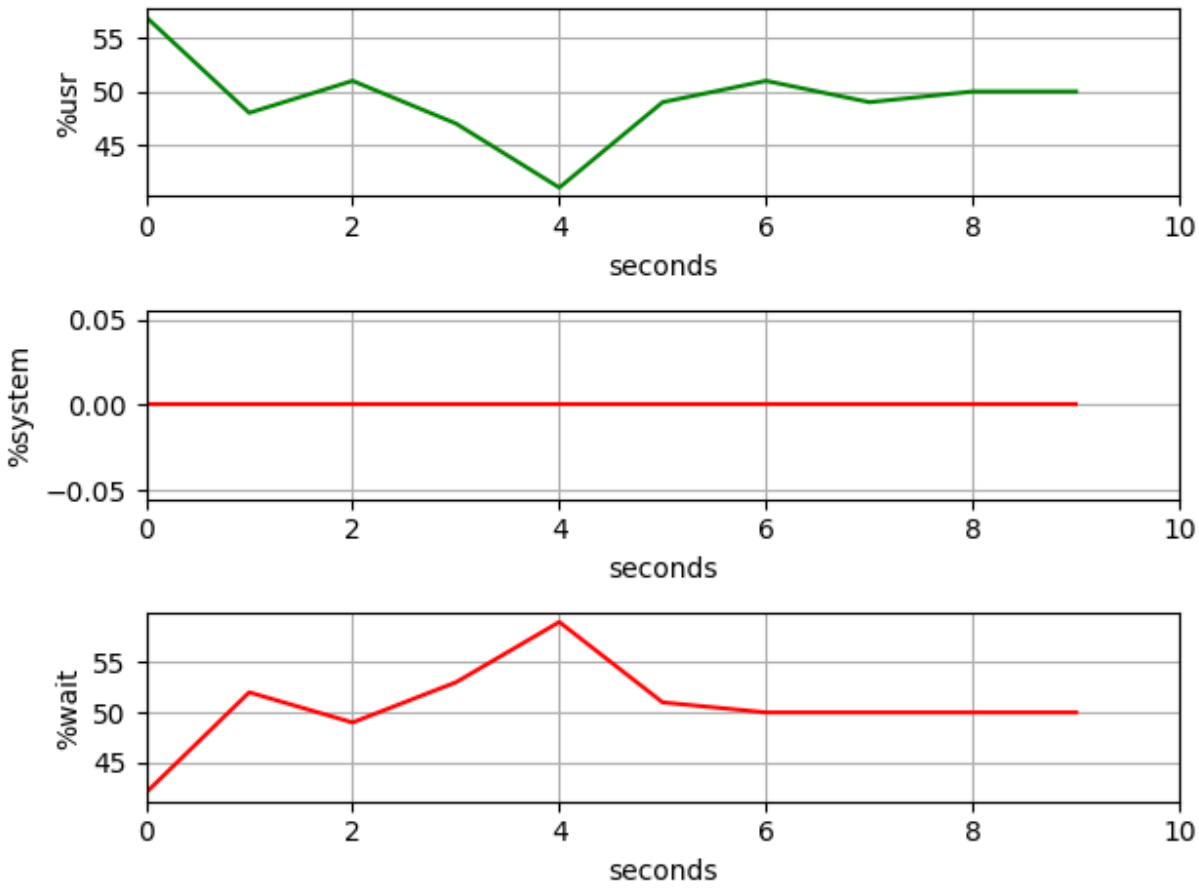



```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/12953/schedstat
34936958722 1093569 150
```

Аналогично нагрузка на процессор не меняется, но, так как повышен приоритет, уменьшается время ожидания.

Попробуем увеличить количество процессов:

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 15 --sched-prio 1
stress-ng: info:  [13016] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [13016] dispatching hogs: 15 cpu
```




```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ mpstat 1 5
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

19:16:05     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
19:16:06     all  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
19:16:07     all  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
19:16:08     all   99,88    0,00    0,12    0,00    0,00    0,00    0,00    0,00    0,00    0,00
19:16:09     all  100,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
19:16:10     all   99,75    0,00    0,25    0,00    0,00    0,00    0,00    0,00    0,00    0,00
Average:     all   99,92    0,00    0,08    0,00    0,00    0,00    0,00    0,00    0,00    0,00
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/13031/schedstat
15111430906 12878155919 1227
```

Ожидаемо, нагрузка увеличивается, время простоя процессора и время ожидания тоже, так как запускается много процессов.

3) Запуск с двумя параметрами

Команда запуска: stress-ng --cpu 1 --sched-prio 1 --sched-runtime 5000

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ stress-ng --cpu 1 --sched-prio 1 --sched-runtime 5000
stress-ng: info:  [13141] defaulting to a 86400 second (1 day, 0.00 secs) run per stressor
stress-ng: info:  [13141] dispatching hogs: 1 cpu
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ mpstat 1 5
Linux 6.2.0-34-generic (ivan-UX430UAR)  14.10.2023      _x86_64_        (8 CPU)

19:19:38     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
19:19:39     all   13,28    0,00    0,13    0,00    0,00    0,00    0,00    0,00    0,00   86,59
19:19:40     all   13,05    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   86,95
19:19:41     all   13,66    0,00    0,13    0,00    0,00    0,00    0,00    0,00    0,00   86,22
19:19:42     all   13,38    0,00    0,38    0,00    0,00    0,00    0,00    0,00    0,00   86,25
19:19:43     all   13,14    0,00    0,13    0,00    0,00    0,00    0,00    0,00    0,00   86,73
Average:     all   13,30    0,00    0,15    0,00    0,00    0,00    0,00    0,00    0,00   86,55
```

```
ivan@ivan-UX430UAR:~/Desktop/itmo-os/lab1$ cat /proc/13142/schedstat
20096037233 666314 92
```

Выводы по мониторингу подсистемы планировщика: С увеличением числа процессов, нагрузка на процессор увеличивается и время простоя процессора также увеличивается. На время ожидания процесса влияет его приоритет и время, выделенное на его выполнение.

Выводы по лабораторной работе:

В ходе выполнения лабораторной работы я применил на практике утилиты для мониторинга процессов и системного анализа.