УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Информационный системы и базы данных»

**Лабораторная работа №1**

*Вариант* cpu: [crc16,decimal32]; cache: [l1cache-sets,l1cache-line-size]; io: [ioport,io-uring]; memory: [memrate,memthras]; network: [sockdiag,netlink-proc]; pipe: [pipe-data-size,pipeherd-yield]; sched: [sched-prio,sched-runtime]

Студент

*Кочнев Р. Д.*

*P33081*

Преподаватель

*Клименков С.В.*

Санкт-Петербург, 2023 г.

Описание задания

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).

Параметры для cpu: crc16,decimal32

Параметры для cache: l1cache-sets,l1cache-line-size

Параметры для io: ioport,io-uring

Параметры для memory: memrate,memthras

Параметры для network: sockdiag,netlink-proc

Параметры для pipe: pipe-data-size,pipeherd-yield

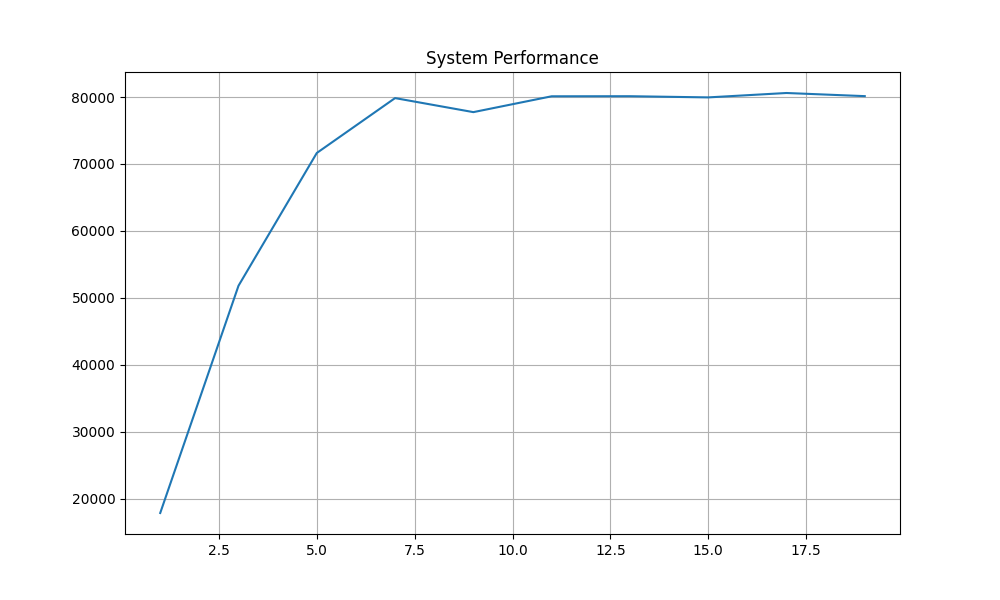
Параметры для sched: sched-prio,sched-runtime

CPU

*stress-ng --cpu {i} --cpu-method crc16 --timeout 10 --metrics-brief*

*top -b -n 1 | grep -m 123 stress-ng*

compute 1024 rounds of CCITT CRC16 on random data

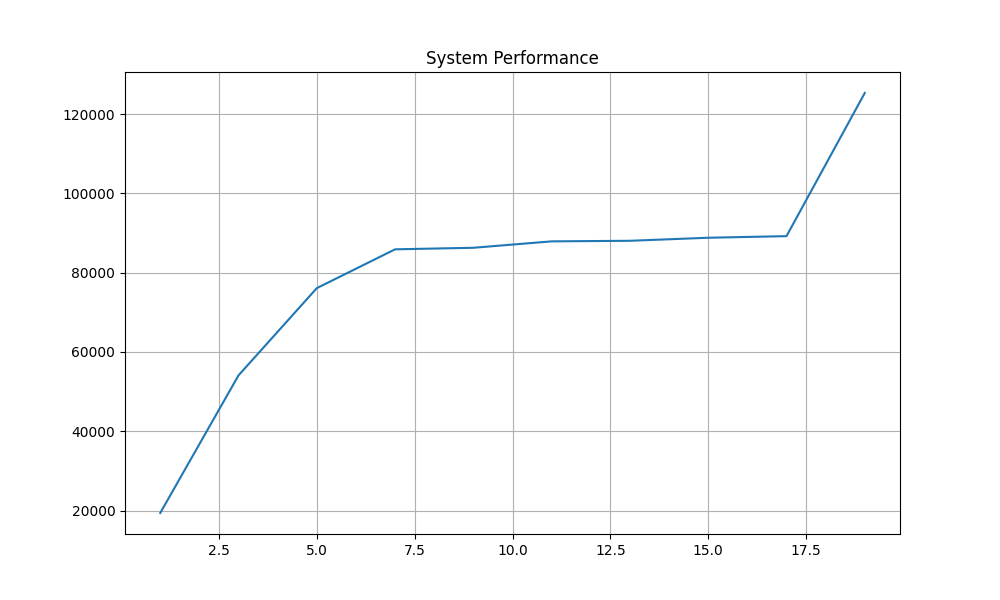
**

*stress-ng --cpu {i} --cpu-method decimal32 --timeout 60 --metrics-brief*

*top -b -n 1 | grep -m 123 stress-ng*

1000 iterations of a mix of 32 bit decimal floating point

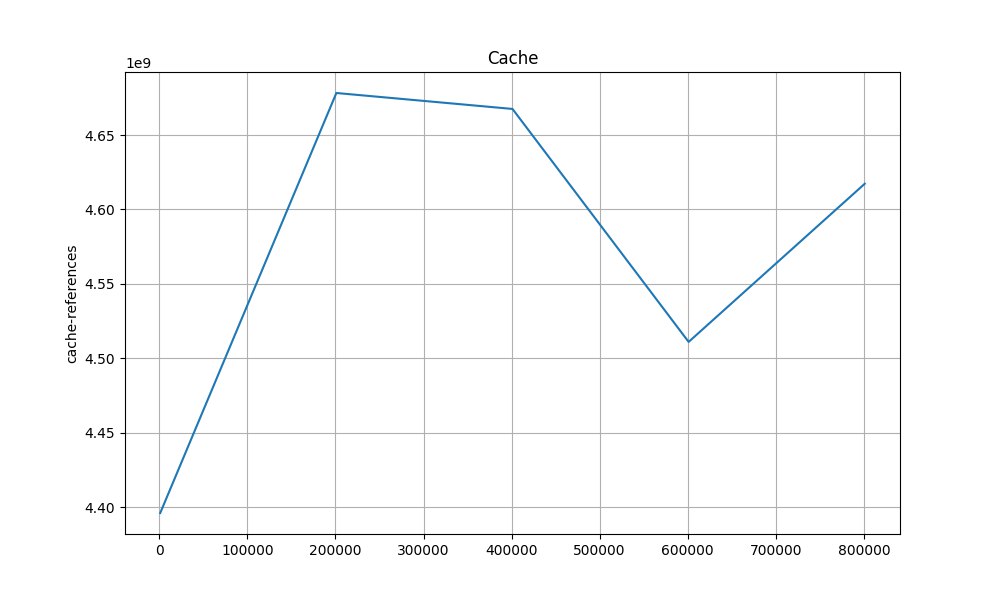
operations (GCC only)

**

Cache

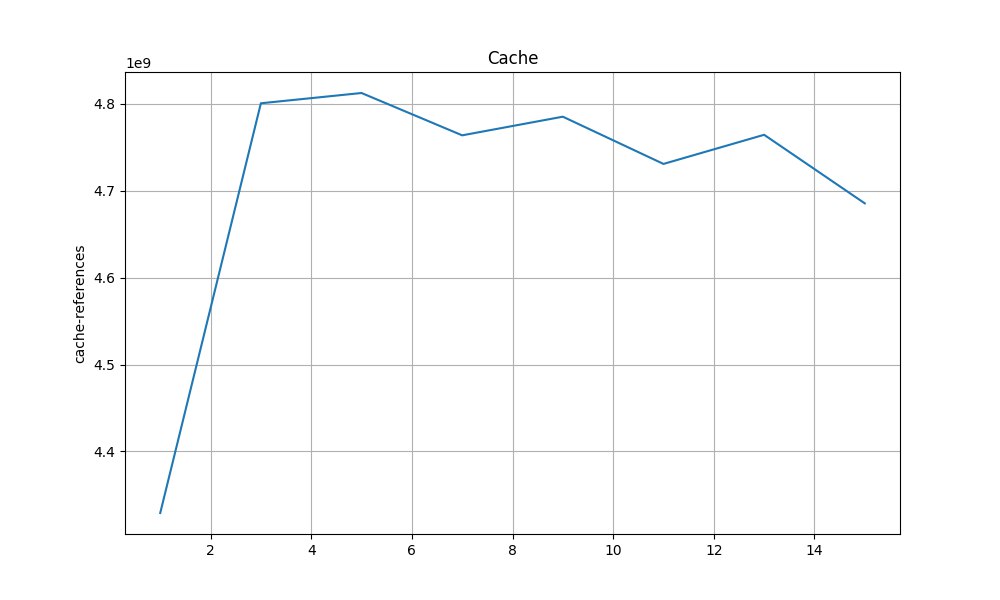
*sudo perf stat -e cache-references,cache-misses stress-ng --cache 1 --l1cache-sets {i} --timeout 20*

specify the number of level 1 cache sets

**

*sudo perf stat -e cache-references,cache-misses stress-ng --cache 1 --l1cache-line-size {i} --timeout 20*

specify the level 1 cache line size (in bytes)

**

IO

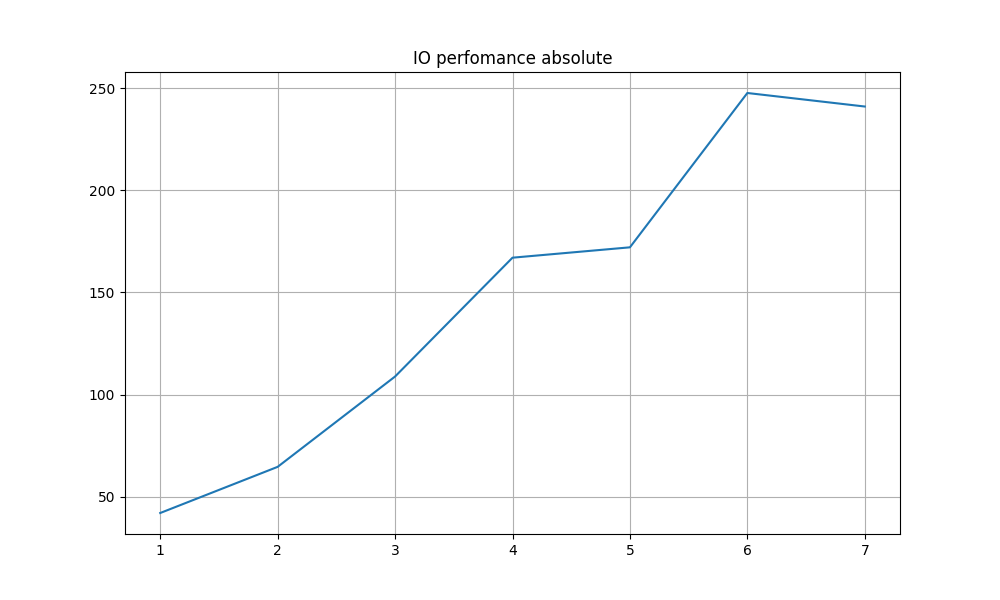
*stress-ng --io 1 --ioport {i} --timeout 10*

*iotop -b -P -n 1 | grep -m 123 stress-ng*

start N workers than perform bursts of 16 reads and 16 writes of ioport 0x80 (x86

Linux systems only). I/O performed on x86 platforms on port 0x80 will cause delays

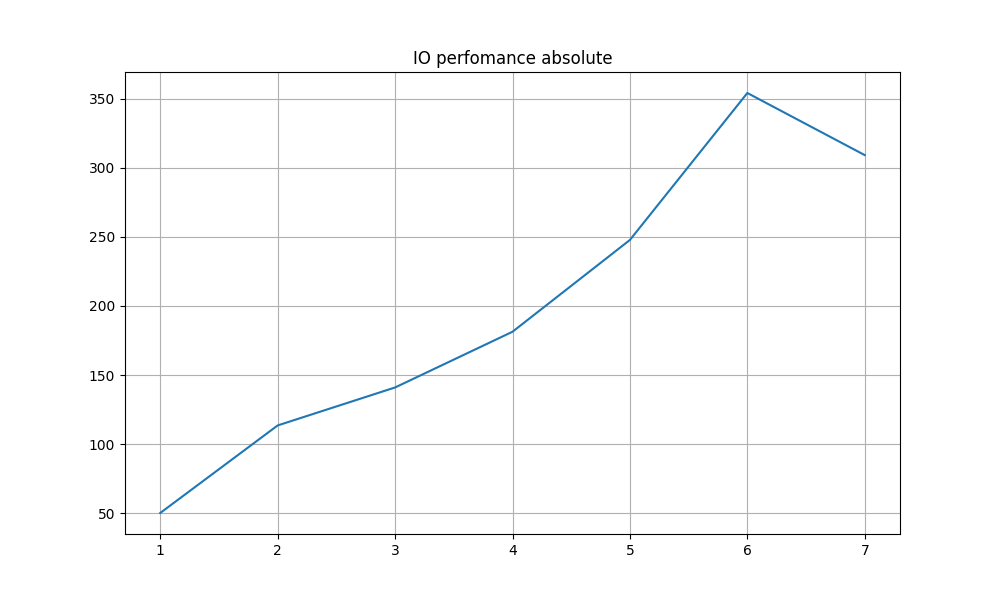
on the CPU performing the I/O.



*stress-ng --io 1 --io-uring {i} --timeout 10*

*iotop -b -P -n 1 | grep -m 123 stress-ng*

start N workers that perform iovec write and read I/O operations using the Linux io-uring interface. On each bogo-loop 1024 × 512 byte writes and 1024 × reads are performed on a temporary file.

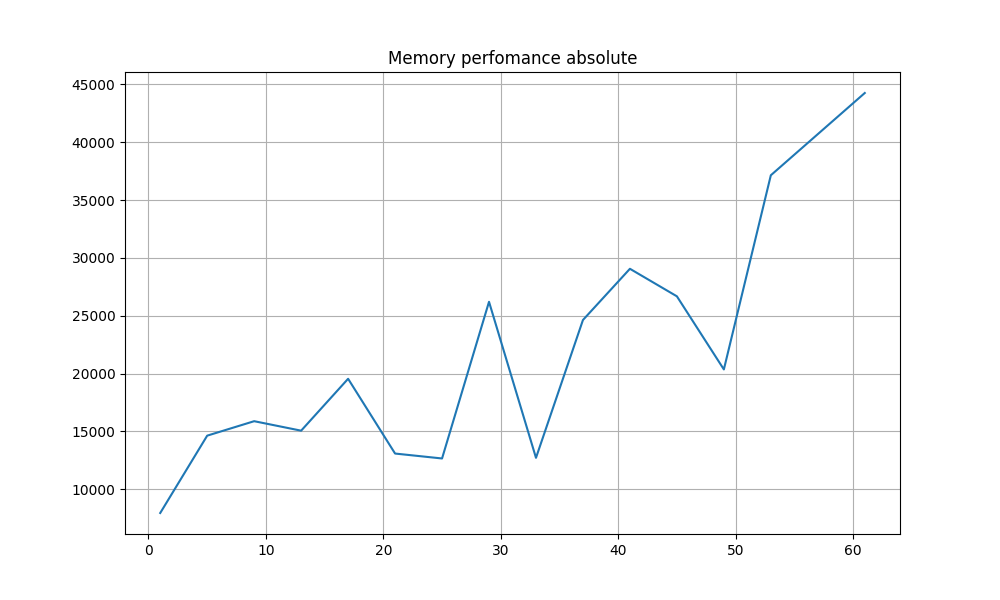
**

Memory

*stress-ng --memrate {i} --timeout 10 --metrics-brief*

*top -b -n 1 | grep -m 123 stress-ng*

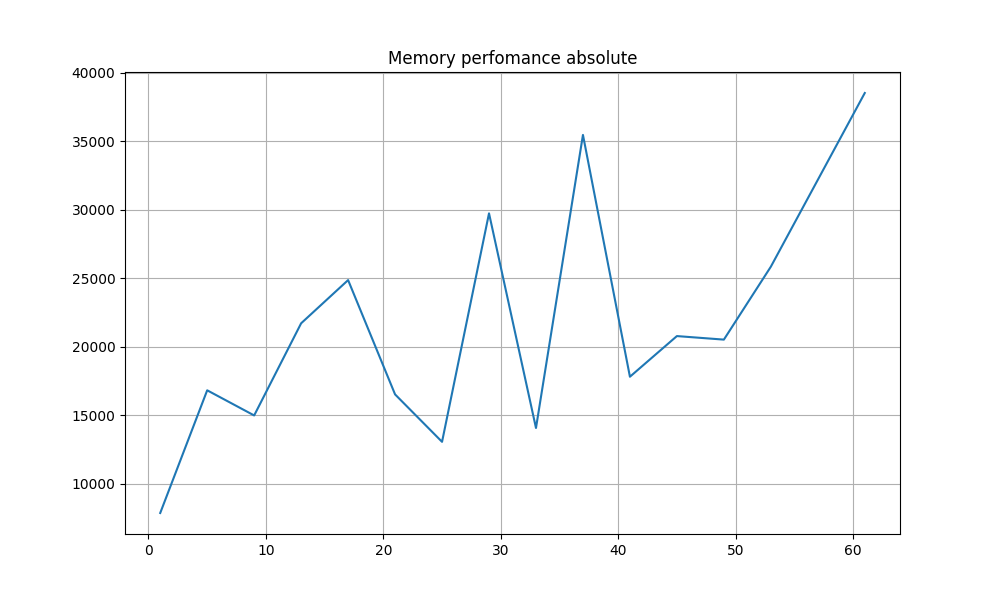
start N workers that exercise a buffer with 1024, 512, 256, 128, 64, 32, 16 and 8 bit reads and writes

**

*stress-ng --memthrash {i} --timeout 60 --metrics-brief*

*top -b -n 1 | grep -m 123 stress-ng*

start N workers that thrash and exercise a 16MB buffer in various ways to try and trip thermal overrun

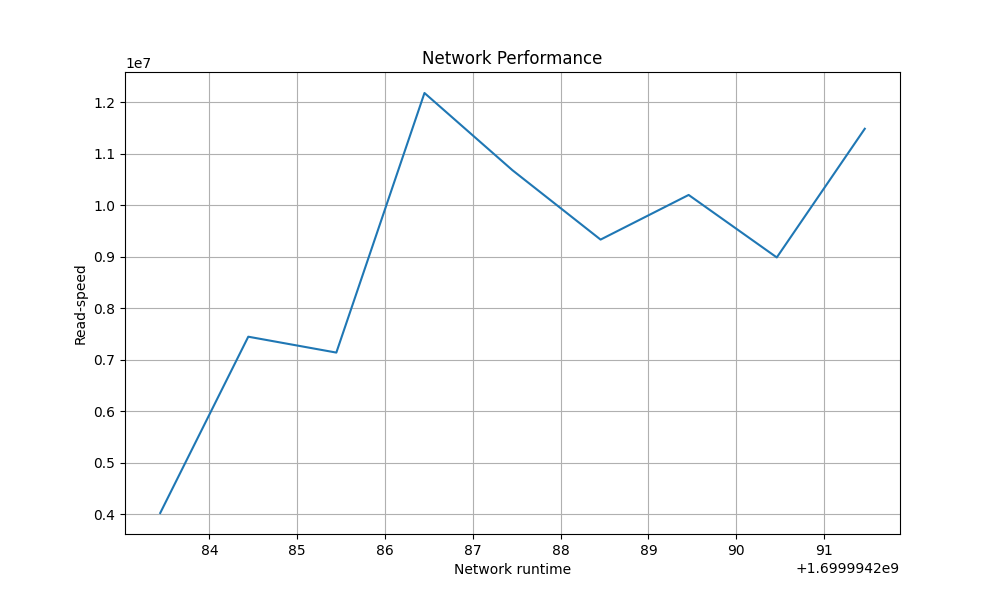
**

Network

*stress-ng --netdev {i} --timeout 10*

*ip -s link show eth0*

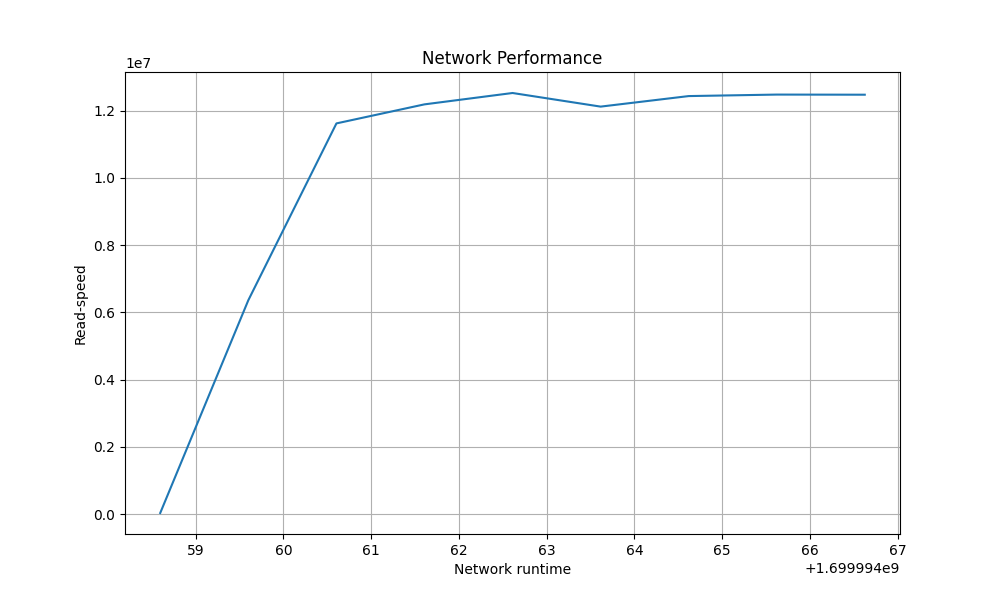
start N workers that exercise various netdevice ioctl commands across all the available network devices

**

*stress-ng --netlink-task {i} --timeout 10*

*ip -s link show eth0*

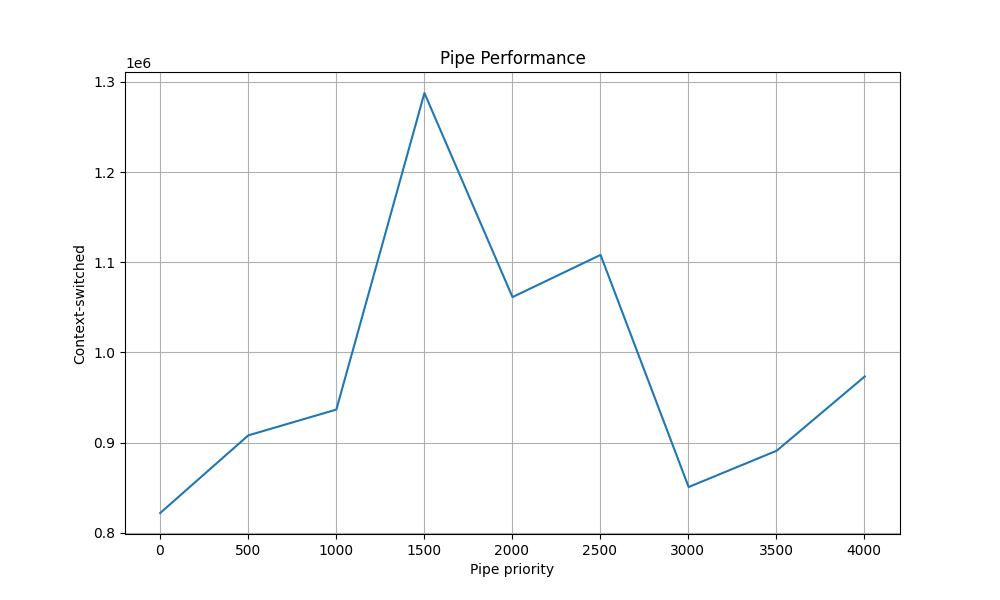
start N workers that collect task statistics via the netlink taskstats interface. This stressor can only be run on Linux and requires CAP\_NET\_ADMIN capability.

**

Pipe

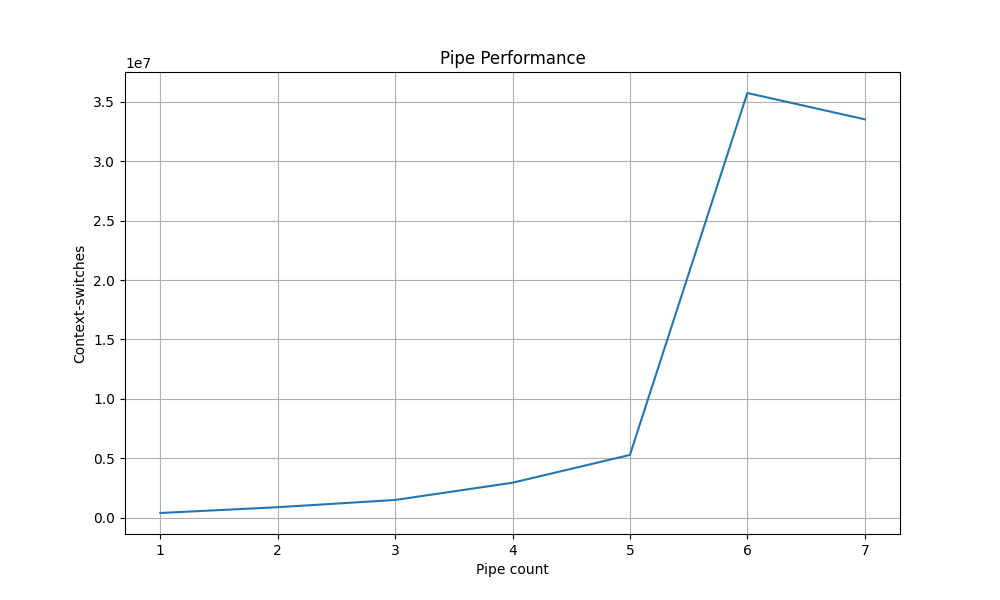
*sudo perf stat -e context-switches stress-ng --pipe 2 --pipe-data-size {i} --timeout 10*

specifies the size in bytes of each write to the pipe (range from 4 bytes to 4096 bytes). Setting a small data size will cause more writes to be buffered in the pipe, hence reducing the context switch rate between the pipe writer and pipe reader processes. Default size is the page size.

**

*sudo perf stat -e context-switches stress-ng --pipe {i} --pipeherd-yield --timeout 10*

force a scheduling yield after each write, this increases the context switch rate.

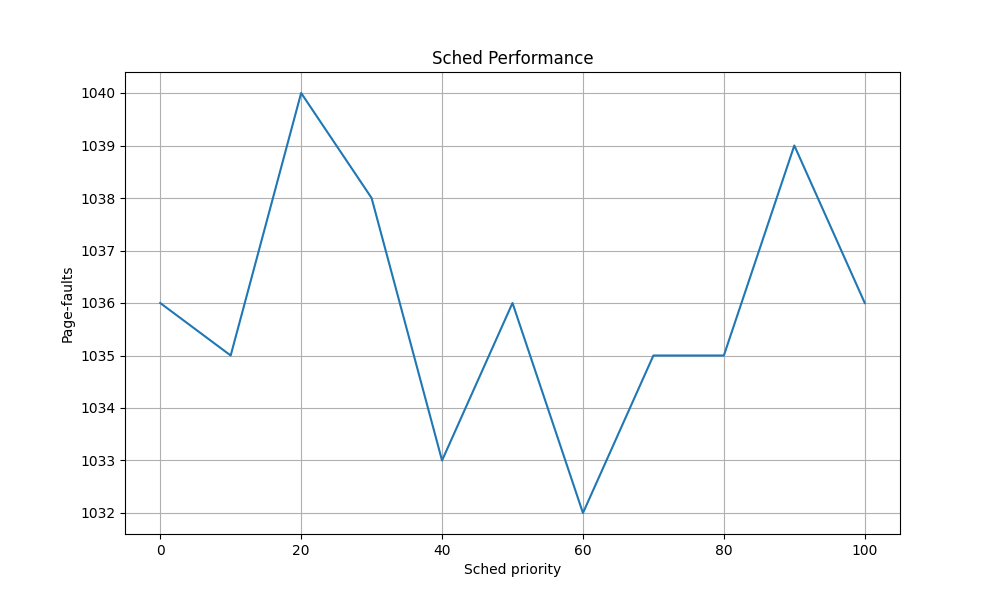
**

Sched

*sudo perf stat -e page-faults stress-ng --schedpolicy 2 --sched-prio {i} --timeout 10*

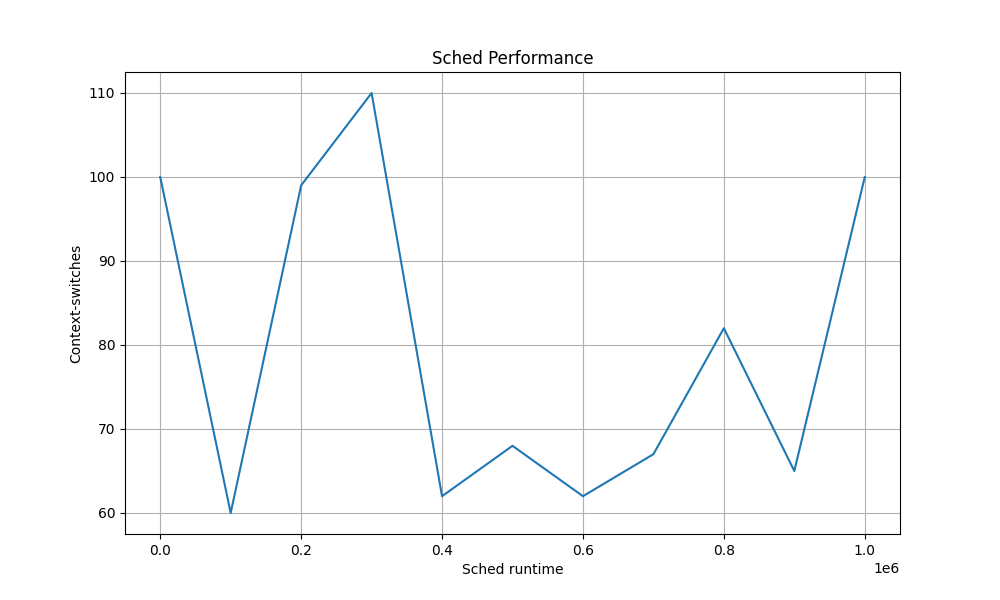
select the scheduler priority level (only on Linux). If the scheduler does not

support this then the default priority level of 0 is chosen.

**

*sudo perf stat -e context-switches stress-ng --schedpolicy 2 --sched-runtime {i} --timeout 10*

select the runtime parameter for deadline scheduler (only on Linux). Default value is 99999 (in nanoseconds).

**

Вывод

Познакомился с нагрузочным тестированием различных частей операционной системы. Понял, что Windows не дает полноценной информации о система, это видно из некоторых графиков, в которых ничего не меняется или результаты случайны.