Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

ИТМО»

**Лабораторная работа №1**

По дисциплине “Операционные системы”

Выполнил:

Кравец Роман Денисович

Проверил: Саржевский Иван Анатольевич

Санкт-Петербург

2023

**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.

**Вариант лабораторной работы:**

cpu: [int8, bitops]

cache: [l1cache-sets, l1cache-ssize]

io: [iomix, ioport]

memory: [memthras, memfd-fds]

network: [dccp, netlink-task]

pipe: [pipe-data-size, sigpipe]

sched: [sched-prio, sched-period]

Код, используемый для получения графиков:

**CPU:**

* Bitops:

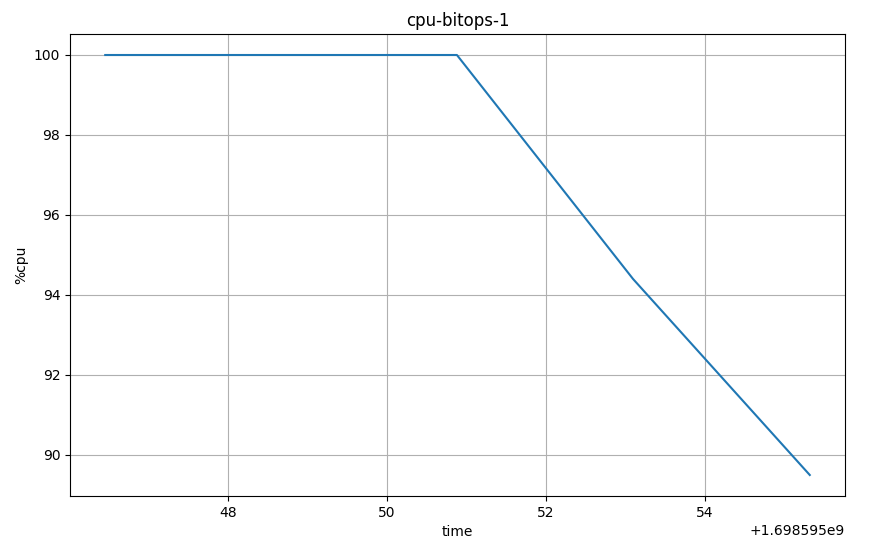
Bitops - различные битовые операции от bithack, а именно: обратные биты, проверка четности, подсчет битов, округление до ближайшей степени.

Запустим следующую команду несколько раз:

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

Затем посмотрим на графики загруженности процессора. Информацию будем получать с помощью утилиты top.

Для одного процесса:



Для 6 процессов:

Изображение выглядит как линия, График, снимок экрана, диаграмма

Автоматически созданное описание

График bogo ops:

Изображение выглядит как линия, График, диаграмма, Параллельный

Автоматически созданное описание

График bogo ops/s аналогичен:

Изображение выглядит как линия, диаграмма, График, текст

Автоматически созданное описание

* Int 8

Int 8 - 1000 итераций сочетания 8-разрядных целочисленных операций

Запустим следующую команду несколько раз:

stress-ng --cpu {i} --cpu-method int8 --timeout 10 --metrics-brief

Аналогично будем получать информацию с помощью утилиты top.

Для одного процесса:

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Для 6 процессов:

Изображение выглядит как линия, График, диаграмма, снимок экрана

Автоматически созданное описание

График bogo ops:

Изображение выглядит как линия, График, диаграмма, скат

Автоматически созданное описание

График bogo ops/s

Изображение выглядит как линия, текст, График, диаграмма

Автоматически созданное описание

**Cache:**

1. l1cache-sets

--l1cache-sets N - указывает количество наборов кэша 1-го уровня

Для мониторинга кэша будем использовать perf для получения информации об обращениях к кэшу и кэш промахах.

Команда:

sudo perf stat -e cache-references -e cache-misses stress-ng --cache 1 --l1cache-sets 1 --timeout 10

Результат работы:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Поэтому буду исследовать bogo ops:

Для этого использую команду:

stress-ng --cache {i} --l1cache-sets {j} --timeout 10 --metrics-brief

Изображение выглядит как диаграмма, линия, График, текст

Автоматически созданное описание

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

С увеличением количества наборов кэша 1-го уровня увеличивается число операций.

1. l1cache-size

[--l1cache-size N](https://manpages.debian.org/testing/stress-ng/stress-ng.1.en.html#l1cache-size) - указывает размер кэша 1-го уровня (в байтах)

Команда:

stress-ng --cache {i} --l1cache-size {j} --timeout 10 --metrics-brief

Результат работы:

Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание

Изображение выглядит как линия, График, диаграмма

Автоматически созданное описание

Размер кэша 1 уровня надо подбирать исходя из рабочих процессоров, которые нагружают cache.

**IO:**

1. iomix

iomix N - запускает N рабочих процессов, которые выполняют сочетание последовательных, случайных операций чтения/записи с отображением в память, а также операций чтения/записи файлов случайного копирования, принудительной синхронизации и (при запуске от имени root) удаления кэша. Создается несколько дочерних процессов, которые совместно используют один файл и выполняют различные операции ввода-вывода с одним и тем же файлом.

Используем команду:

stress-ng --iomix {i} --timeout 10

Мониторим с помощью следующей утилиты iotop

Для 11 процессов(запись):

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Для 11 процессов(чтение):

Изображение выглядит как текст, линия, диаграмма, График

Автоматически созданное описание

Для 26 процессов(запись):

Изображение выглядит как текст, График, линия, диаграмма

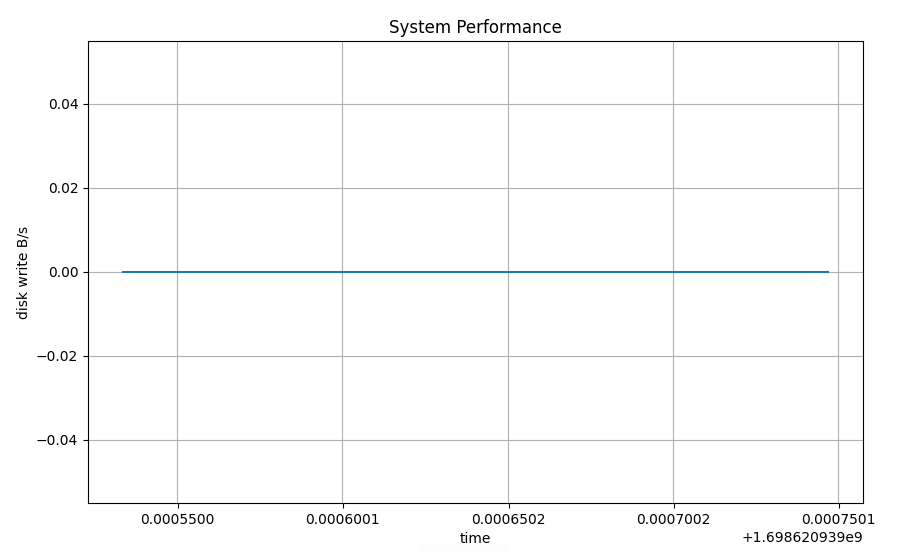
Автоматически созданное описание

Для 26 процессов(чтение):

Изображение выглядит как текст, линия, График, снимок экрана

Автоматически созданное описание

Для 31 процессов(запись):



Для 31 процессов(чтение):

Изображение выглядит как текст, линия, число, График

Автоматически созданное описание

В остальных случаях выходит, что чтения и записи не происходит.

1. Ioport

Ioport N - запускает N рабочих процессов, а затем выполняет пакеты из 16 операций чтения и 16 операций записи ioport 0x80 (только для систем Linux x86). Ввод-вывод, выполняемый на платформах x86 через порт 0x80, приведет к задержкам в работе центрального процессора, выполняющего ввод-вывод.

Команда:

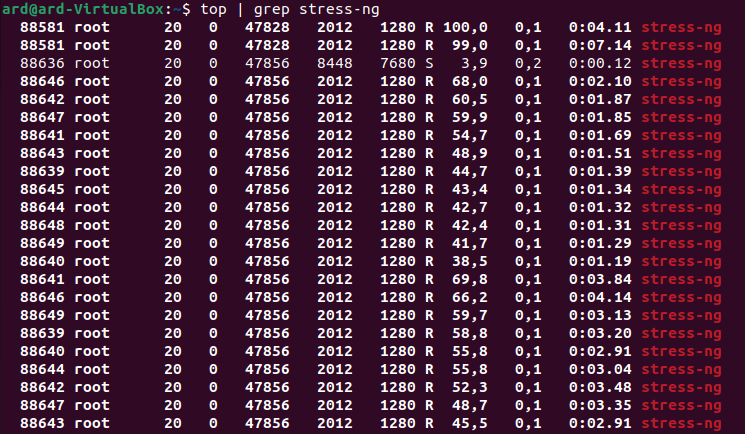
sudo stress-ng --ioport {i} --timeout 10

Мониторим с помощью следующей утилиты iotop:

Ничего не записывается и не читается:



Хотя процессы нагружают систему:



Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

**Memory:**

1. Memthrash

Memthrash N - запускает N рабочих процессов, которые различными способами обрабатывают 16-мегабайтный буфер, чтобы попытаться

предотвратить перегрев. Каждый стрессор запускает 1 или более потоков. Количество

потоков выбирается таким образом, чтобы на каждый процессор приходилось не менее 1 потока. Обратите внимание, что

оптимальным выбором для N является значение, которое делится на количество процессоров.

Воспользуемся утилитой top для нахождения %MEM

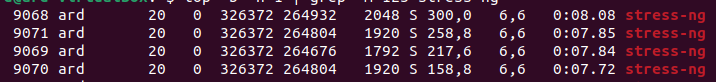
stress-ng --memthrash 2 --timeout 20

bogo ops: 3289



stress-ng --memthrash 4 --timeout 20

bogo ops: 3379



stress-ng --memthrash 8 --timeout 20

bogo ops: 3563

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Видим, что потребляемая память – константа 6,6 от общего числа.

1. memfd-fds

--memfd N - запускает N рабочих процессов, выделяющих память с помощью memfd\_create

--memfd-fds N - количество открытых fds памяти для каждого рабочего процессора Создает N файловых дескрипторов memfd, значение по умолчанию равно 256. Можно выбрать от 8 до 4096 memfd описания файлов с этой опцией.

Команда:

stress-ng --memfd {i} --memfd-fds {j} --timeout 10

1 процесс с количеством fds памяти 808:

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

4 процесса с количеством fds памяти 1608:

Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание

7 процессов с количеством памяти 4008:

Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание

**Network:**

1. dccp

dccp N -запускает N рабочих процессов, которые отправляют и получают данные, используя управление перегрузкой дейтаграмм Протокол (DCCP) (RFC4340). Это включает в себя пару клиент-серверных процессов выполнение быстрого подключения, отправки, приема и отключения на локальном хосте.

Команда:

stress-ng --dccp {i} --timeout 20

Воспользуемся утилитой ip -s link show lo. С помощью него будем измерять скорость сети, которая будет измеряться в byte/s. Добиться этого можно с помощью разности полученных и отправленных данных за каждую секунду

Для 1 процесса:

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Для 11 процессов:

Изображение выглядит как линия, График, диаграмма

Автоматически созданное описание

Для 8 процессов:

Изображение выглядит как линия, График, диаграмма

Автоматически созданное описание

Из графиков видно, что оптимальный параметр – 11 процессов.

1. netlink-task

запустите N рабочих, которые собирают статистику задач через интерфейс netlink taskstats.Этот стрессор может быть запущен только в Linux и требует возможности CAP\_NET\_ADMIN.

Команда:

sudo stress-ng --netlink-task {i} --timeout 20

Пользуемся все той же утилитой.

Для 1 процесса

Изображение выглядит как текст, линия, диаграмма, График

Автоматически созданное описаниеДля 36 процессов:

Изображение выглядит как текст, линия, диаграмма, График

Автоматически созданное описание

И во всех других случаях скорость сети – 0.

**Pipe:**

1. pipe-data-size

pipe-data-size N - определяет размер в байтах каждой записи в канал (диапазон от 4 байт до 4096 байт). Установка небольшого размера данных приведет к буферизации большего количества записей в канале, что приведет к снижению скорости переключения контекста между каналом записи и каналом передачи процессы чтения

Команда:

stress-ng --pipe 1 --pipe-data-size {i} --timeout 10 --metrics

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание Изображение выглядит как линия, График, диаграмма, скат

Автоматически созданное описание

Оптимальное значение pipe-data-size 1008.

1. sigpipe

--sigpipe N - запускает N рабочих процессов, которые многократно порождают дочерний процесс, завершающийся раньше родительского, может завершить запись по каналу, вызывая сигнал SIGPIPE.

Убийства процессов будут вызывать переключения контекста, которые мы будем мониторить утилитой perf

Команда:

sudo perf stat -e context-switches stress-ng --sigpipe {i} --timeout 10

Изображение выглядит как линия, диаграмма, График, Параллельный

Автоматически созданное описание

С ростом количества процессов растет и количество переключений контекста.

**sched:**

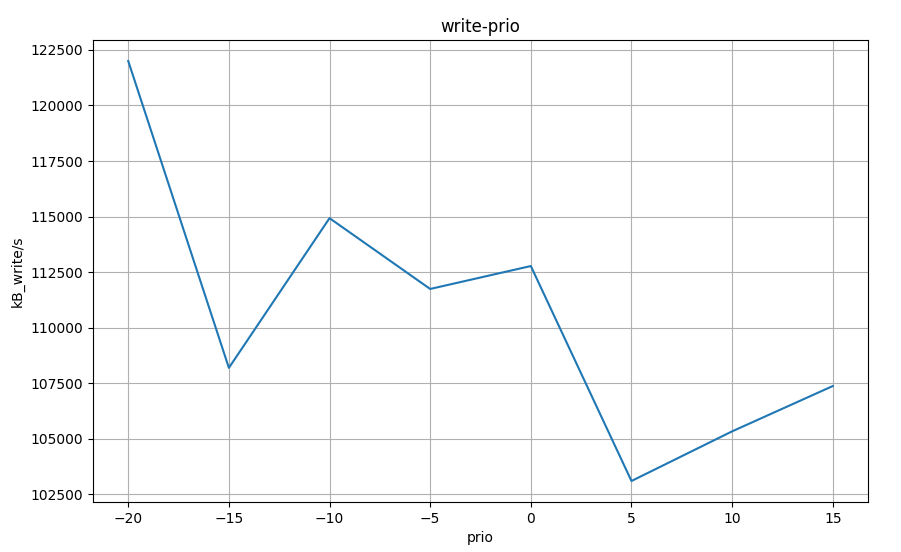
1. sched-prio

shed-prio N - установить уровень приоритета планировщика (только в Linux). Если планировщик не поддерживает это, тогда будет выбран уровень приоритета по умолчанию, равный 0.

Команда:

stress-ng --hdd 10 --sched-prio {i} --timeout 15

Отслеживать будем скорость записи на диск с помощью утилиты iostat. Максимальный приоритет имеет значение -20, а минимальный 19. Если установить максимальный приоритет, то должно максимизироваться пропускная способность, то есть количество задач, выполняемых за единицу времени.



1. sched-period

sched-period N - установить период для SCHED\_DEADLINE равным N наносекунд.

Команда:

stress-ng --hdd 10 --sched-prio {i} --timeout 15

Анализируем аналогично с помощью утилиты iostat.

Изображение выглядит как линия, текст, диаграмма, График

Автоматически созданное описание

Средняя скорость записи при возрастании периода увеличивается, т. к. временной интервал увеличивается в течение которого процесс должен быть выполнен.

**Вывод:**

В ходе выполнения данной лабораторной работы я познакомился с различными системными инструментами анализа производительности и поведения программ. Ознакомился как производить нагрузочное тестирование операционной системы при помощи утилиты stress-ng.