Домашнее задание 4

Домашнее задание 4

1. Скрипт, получающий информацию о CPU

Написать скрипт, получающий информацию о CPU: количество cpu, модель, рабочая/максимальная частота, количество ядер/потоков, размер кэшей; а также о физической памяти: количество, тип и размер плашек, какие слоты они заним ают.

Для получения доступа к железу была использована утилита dmidecode
Для получения информации о cpu воспользовался утилитой lscpu.
Ниже на скриншоте представлен вывод для моей VPS

```
root@cv4222321:~# nano cpu.sh
root@cv4222321:~# chmod +x cpu.sh
root@cv4222321:~# sudo ./cpu.sh
Информация о CPU
количество cpu: 2
количество ядер: 2
количество потоков: 2
рабочая нагрузка: 100%
средняя нагрузка (1, 5, 15 минут): 0.00,0.00,0.00
Размеры кэшей:
  L1d cache: 64
  L1i cache: 64
  L2 cache: 8
  L3 cache: 32
Информация о физической памяти:
Слот: Not Specified
        Type, Размер: 2 GB
        Form Factor, Тип: Multi-bit ECC
        Maximum Capacity, Скорость: U, Номер детали: QEMU
        Serial Number
nОбщее количество модулей памяти: 1
root@cv4222321:~#
```

2. hugepages

Настроить использование обычных hugepages (размер 2 мб), смонтировать их в какой-либо раздел, написать программу, которая создает memory mapped file (аллоцировать несколько страниц) и запишет в него произвольное данные (любое слово). Убедиться, что страницы и правда задействованы. Проверить наличие файла в смонтированном разделе, прочитать данные из него из консоли штатными средствами.

У меня размер hugepages составляет 2048 кБ (2 МБ) (по умаолчанию), проверил через команду grep Huge /proc/meminfo.
Вывод на скриншоте ниже:

```
root@cv4222321:~# grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
root@cv4222321:~#
```

Зарезервировал 10 hugepages: sudo sysctl -w vm.nr_hugepages=10,
в /etc/sysctl.conf добавил vm.nr_hugepages=10,
применил настройки sudo sysctl -p

На скрине ниже видно теперь что их 10 и они свободны:

```
root@cv4222321:~# sudo sysctl -w vm.nr_hugepages=10
vm.nr_hugepages = 10
root@cv4222321:~# nano /etc/sysctl.conf
root@cv4222321:~# sudo sysctl -p
vm.nr_hugepages = 10
root@cv4222321:~# grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:      10
HugePages_Free:       10
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:           20480 kB
root@cv4222321:~# nano /etc/sysctl.conf
```

Далее смонтировал hugepages

sudo mkdir /mnt/huge

sudo mount -t hugetlbfs none /mnt/huge

Решил написать программу на Java которая будет создавать файл в /mnt/huge, устанавливать его размер на несколько мб, отображать файл в память (memory mapped file). ну и записывать что-то в память

Программу написал, скомпилил и запустил, скриншот ниже:

```
root@cv4222321:~# nano HugePageTest.java
root@cv4222321:~# javac HugePageTest.java
root@cv4222321:~# java HugePageTest
Данные записаны в memory-mapped файл.
root@cv4222321:~#
```

Видим выше вывод нашей программы об успешной записи в memory-mapped файл.

Теперь введем снвоа команду grep Huge /proc/meminfo чтобы проверить уменьшилось ли количество hugepages свободных, вывод на скриншоте ниже:

```
root@cv4222321:~# nano HugePageTest.java
root@cv4222321:~# javac HugePageTest.java
root@cv4222321:~# java HugePageTest
Данные записаны в memory-mapped файл.
root@cv4222321:~# grep Huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
FileHugePages:         0 kB
HugePages_Total:      10
HugePages_Free:        9
HugePages_Rsvd:        4
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:           20480 kB
root@cv4222321:~#
```

Теперь убедимся и проверим какие файлы открыты в /mnt/huge:

ls -l /mnt/huge

Вывод команды на скриншоте ниже

```
ShmemHugePages:         0 kB
FileHugePages:          0 kB
HugePages_Total:       10
HugePages_Free:         9
HugePages_Rsvd:         4
HugePages_Surp:         0
Hugepagesize:        2048 kB
Hugetlb:            20480 kB
root@cv4222321:~#   sudo lsof | grep hugetlbfs

root@cv4222321:~#   sudo lsof | grep hugetlbfs
root@cv4222321:~#   ls -l /mnt/huge
total 2048
-rw-r--r-- 1 root root 10485760 Oct 25 20:45 testfile
root@cv4222321:~#
```

Можем прочитать данные из файла при помощи hexdump -C /mnt/huge/testfile | head

```
root@cv4222321:~# hexdump -C /mnt/huge/testfile | head
00000000  54 68 69 73 20 69 73 20  48 75 67 65 50 61 67 65  |This is HugePage|
00000010  73 65 50 61 67 65 73 21  00 00 00 00 00 00 00 00  |sePages!........|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00a00000
root@cv4222321:~#
```

Или например при помощи strings /mnt/huge/testfile

```
root@cv4222321:~# strings /mnt/huge/testfile
This is HugePagesePages!
root@cv4222321:~#
```

3. Запуск процесса с привязкой к одному ядру

Запустить процесс с привязкой к одному ядру (процесс должен потреблять 100% ядра). Сменить ему политику на realtime FIFO, разрешив потреблять не более 75% процессорного времени. Запустить на этом же ядре вторую копию процесса с любой обычной политикой, убедиться, что он может выполняться и потребляет оставшееся ему время cpu.

Сделал программу потребляющую cpu:

```java
public class CpuBurn {
    public static void main(String[] args) {
        while (true) {
        }
    }
}
```

Выполнил компиляцию при помощи javac.

Запустим наш Java-процесс, привязав его к первому ядру (ядро 0), и установим политику планирования SCHED_FIFO с приоритетом 50

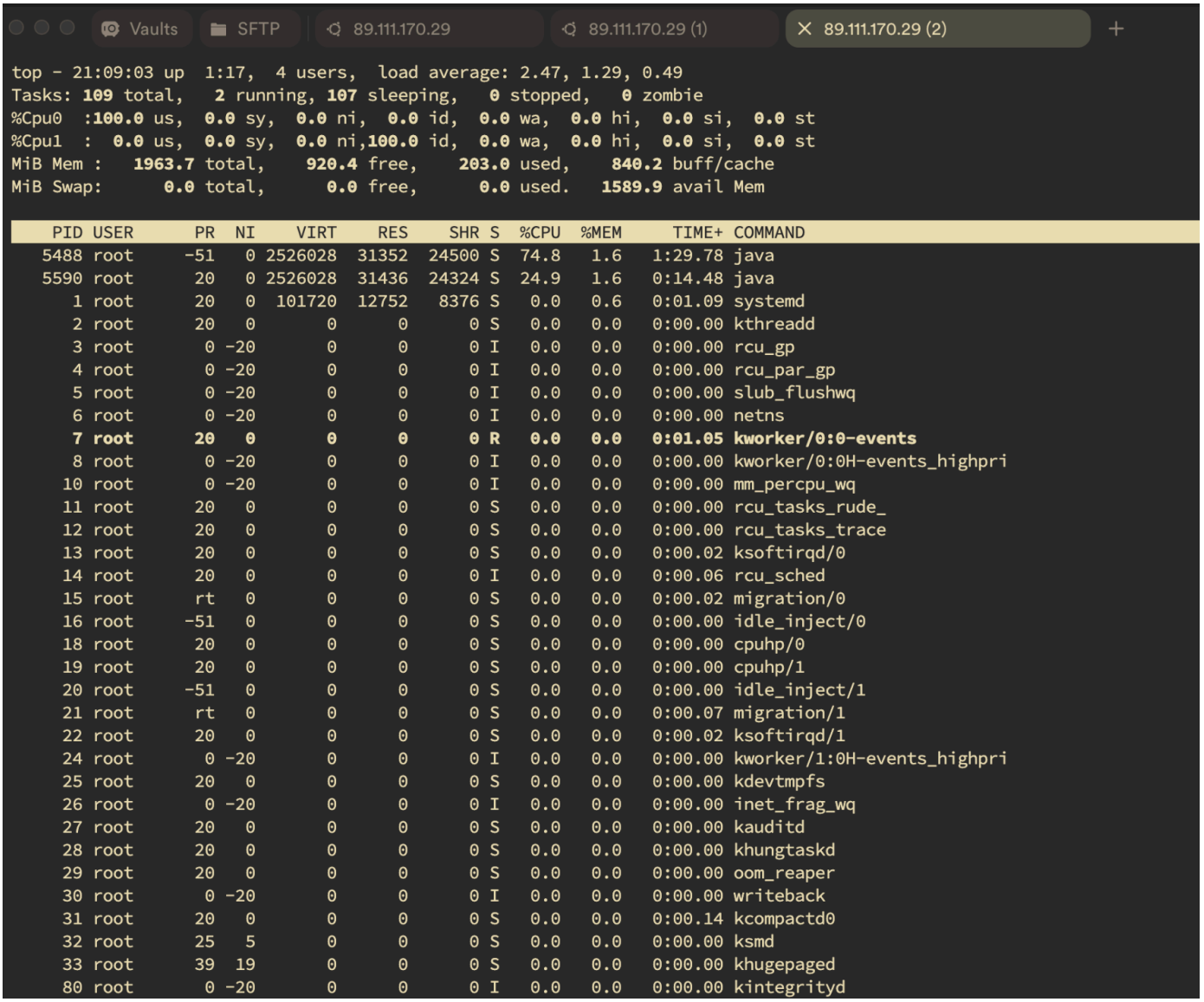
sudo taskset -c 0 chrt -f 50 java CpuBurn

И также запустим вторую копию этого жава процесса с обычной политикой планирования, привязанную к тому же ядру

taskset -c 0 java CpuBurn

Теперь выполним топ и увидим что у нас есть два джава процесса которые заняли cpu, один на 75 другой на 25 процентов

Вывод команды топ на скрине ниже



Таким образом мы запустили на этом же ядре вторую копию процесса с любой обычной политикой, убеделиись, что он может выполняться и потребляет оставшееся ему время cpu.

4. raid1 массив

Создайте программный raid1 массив и добавьте в него hot spare диск. Проверьте, что все работает. Для этого надо “сломать” один из дисков массива и убедиться, что запустился ребилд на резервный диск.

```
root@cv4222321:~/raid_test# sudo dd if=/dev/zero of=disk1.img bs=1M count=100
sudo dd if=/dev/zero of=disk2.img bs=1M count=100
sudo dd if=/dev/zero of=spare.img bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.064841 s, 1.6 GB/s
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.0711219 s, 1.5 GB/s
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.0713835 s, 1.5 GB/s
root@cv4222321:~/raid_test# sudo losetup -fP disk1.img
sudo losetup -fP disk2.img
sudo losetup -fP spare.img
root@cv4222321:~/raid_test# losetup -a
/dev/loop1: [2049]:129598 (/root/raid_test/disk2.img)
/dev/loop2: [2049]:129616 (/root/raid_test/spare.img)
/dev/loop0: [2049]:129597 (/root/raid_test/disk1.img)
root@cv4222321:~/raid_test#
```

• /dev/loop0 – диск1

• /dev/loop1 – диск2

• /dev/loop2 – резервный диск

Создаем raid1 массив из двух основных дисков и одного горячего резервного.

`sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/loop0 /dev/loop1 --spare-devices=1 /dev/loop2`

```
root@cv4222321:~/raid_test# sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/loop0 /dev/loop1 --spare-devices=1 /dev/loop2
mdadm: Note: this array has metadata at the start and
    may not be suitable as a boot device.  If you plan to
    store '/boot' on this device please ensure that
    your boot-loader understands md/v1.x metadata, or use
    --metadata=0.90
mdadm: size set to 101376K
Continue creating array? y
Continue creating array? (y/n) y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
root@cv4222321:~/raid_test#
```

Создаем файловую систему на новом raid-массиве и монтируем его

sudo mkfs.ext4 /dev/md0

sudo mkdir /mnt/raid

sudo mount /dev/md0 /mnt/raid

Теперь проверим что все работает введем команду sudo touch /mnt/raid/testfile

а потом команду ls /mnt/raid

Вывод на скрине ниже

```
root@cv4222321:~/raid_test# sudo touch /mnt/raid/testfile
ls /mnt/raid
lost+found  testfile
root@cv4222321:~/raid_test#
```

Симуляция отказа

Откажем /dev/loop1 след образом:

sudo mdadm /dev/md0 --fail /dev/loop1

sudo mdadm /dev/md0 --remove /dev/loop1

```
root@cv4222321:~/raid_test# sudo mdadm /dev/md0 --fail /dev/loop1
sudo mdadm /dev/md0 --remove /dev/loop1
mdadm: set /dev/loop1 faulty in /dev/md0
mdadm: hot removed /dev/loop1 from /dev/md0
root@cv4222321:~/raid_test# cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid1 loop2[2] loop0[0]
      101376 blocks super 1.2 [2/2] [UU]

unused devices: <none>
root@cv4222321:~/raid_test#
```

Что важно увидеть на скрине выше: горячий резервный диск /dev/loop2 автоматически вошел в состав массива
(данный вывод команды отличается от того что я делал до отказа, просто забыл заскринить).