

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И
ОПТИКИ»**

Факультет безопасности информационных технологий
Кафедра проектирования и безопасности компьютерных систем

Дисциплина:
«Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

Выполнили:
Студент группы.N3248
Назаров Максим Вячеславович
Проверил:
Савков Сергей Витальевич

Санкт-Петербург 2022г.

Лаб 4. Планировщик

Провести тестирование и найти лучший планировщик ввода-вывода среди других.

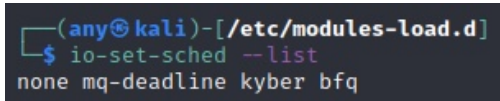
Для того, чтобы удобно менять планировщики установим утилиту ioschedset:

`git clone https://github.com/kata198/ioschedset ./install.sh`

Используя команду `io-set-sched <name>`, можно устанавливать планировщики задач.

Посмотрим какие планировщики задач у нас уже имеются:

`io-set-sched --list`



```
(any@kali)-[/etc/modules-load.d]
$ io-set-sched --list
none mq-deadline kyber bfq
```

Посмотреть активный планировщик можно через

`cat /sys/class/block/sda/queue/scheduler`

Выбранный планировщик будет выделен квадратными скобками.

Будем тестировать с помощью двух параметров:

- **Скорость чтения из буфера и чтения из кэша с помощью утилиты `hdparm`.** Провести тестирование с помощью данной утилиты достаточно нетрудно. Используем команду `hdparm -tT /dev/sda`, выбрав нужный планировщик.

После этого произведем 5 замеров, приведём пример одного замера:

```
File Actions Edit View Help
(any@kali)-[~]
$ io-set-sched --list
none mq-deadline kyber bfq

(any@kali)-[~]
$ io-set-sched mq-deadline
Must be root to set IO Scheduler. Rerunning under sudo ...

[sudo] password for any:
Sorry, try again.
[sudo] password for any:
+ Successfully set nvme0n1 to 'mq-deadline'!
+ Successfully set sda to 'mq-deadline'!

(any@kali)-[~]
$ io-set-sched none
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'none'!
+ Successfully set sda to 'none'!

(any@kali)-[~]
$ io-set-sched mq-deadline
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'mq-deadline'!
+ Successfully set sda to 'mq-deadline'!

(any@kali)-[~]
$ io-set-sched kyber
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'kyber'!
+ Successfully set sda to 'kyber'!

(any@kali)-[~]
$ io-set-sched bfq
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'bfq'!
+ Successfully set sda to 'bfq'!

(any@kali)-[~]
$

File Actions Edit View Help
(any@kali)-[/etc/modules-load.d]
$ sudo hdparm -tT /dev/sda

/dev/sda:
Timing cached reads: 25964 MB in 2.00 seconds = 12997.06 MB/sec
Timing buffered disk reads: 606 MB in 3.01 seconds = 201.50 MB/sec

(any@kali)-[/etc/modules-load.d]
$ sudo hdparm -tT /dev/sda

/dev/sda:
Timing cached reads: 25526 MB in 2.00 seconds = 12777.75 MB/sec
Timing buffered disk reads: 606 MB in 3.01 seconds = 201.51 MB/sec

(any@kali)-[/etc/modules-load.d]
$ sudo hdparm -tT /dev/sda

/dev/sda:
Timing cached reads: 26206 MB in 2.00 seconds = 13118.60 MB/sec
Timing buffered disk reads: 606 MB in 3.01 seconds = 201.55 MB/sec

(any@kali)-[/etc/modules-load.d]
$ sudo hdparm -tT /dev/sda

/dev/sda:
Timing cached reads: 26030 MB in 2.00 seconds = 13030.35 MB/sec
Timing buffered disk reads: 604 MB in 3.00 seconds = 201.22 MB/sec

(any@kali)-[/etc/modules-load.d]
$
```

Все результаты `hdparm` сохраним и найдём среднее значение по каждому параметру.

- **Время разархивирование тяжелого архива** Скачаем архив с ядром

линукса версии 5.x:

```
(any@kali)-[~]
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-5.17.5.tar.xz
--2022-05-03 03:53:40-- https://mirrors.edge.kernel.org/pub/linux/kernel/v5.x/linux-5.17.5.tar.xz
Resolving mirrors.edge.kernel.org (mirrors.edge.kernel.org)... 147.75.101.1, 2604:1380:2001:3900::1
Connecting to mirrors.edge.kernel.org (mirrors.edge.kernel.org)|147.75.101.1|:443 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 128430464 (122M) [application/x-xz]
Saving to: 'linux-5.17.5.tar.xz'

linux-5.17.5.tar.xz      100%[=====>] 122.48M  6.18MB/s   in 28s

2022-05-03 03:54:08 (4.39 MB/s) - 'linux-5.17.5.tar.xz' saved [128430464/128430464]
```

После этого замерим время разархивирования этого архива, используя различные планировщики ввода и вывода по 5 раз на каждый.

Время `real` будет расчётным, по которым позже посчитаем средние значения, пример вывода:

```

(any@kali)-[~]
$ io-set-sched --list
none mq-deadline kyber bfq

(any@kali)-[~]
$ io-set-sched none
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'none'!
+ Successfully set sda to 'none'!

(any@kali)-[~]
$ io-set-sched mq-deadline
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'mq-deadline'!
+ Successfully set sda to 'mq-deadline'!

(any@kali)-[~]
$ io-set-sched kyber
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'kyber'!
+ Successfully set sda to 'kyber'!

(any@kali)-[~]
$ io-set-sched bfq
Must be root to set IO Scheduler. Rerunning under sudo ...

+ Successfully set nvme0n1 to 'bfq'!
+ Successfully set sda to 'bfq'!

real    6.95s
user    6.85s
sys     2.27s
cpu     131%

(any@kali)-[~]
$ time tar xJf linux-5.17.5.tar.xz

real    8.84s
user    7.00s
sys     3.31s
cpu     116%

(any@kali)-[~]
$ time tar xJf linux-5.17.5.tar.xz

real    9.50s
user    6.98s
sys     3.01s
cpu     105%

(any@kali)-[~]
$ time tar xJf linux-5.17.5.tar.xz

real    7.29s
user    6.79s
sys     3.05s
cpu     135%

(any@kali)-[~]
$

```

После проделанных тестов соберем все данные вместе, посчитаем средние значения и попробуем выявить лучший планировщик для Kali linux 5.16.3.

timing cached reads	none	timing cached read mq-deadline	timing cached read kyber	timing cached read bfq	
12 997,05		12 777,75	13 118,60	13 030,35	
13,337,21		13 462,02	13 773,74	13 629,90	
13 708,92		13 423,36	13 804,47	13 353,12	
13 617,17		13 530,11	13 103,57	13 143,63	
13 521,56		13 652,83	13 388,62	13 253,10	
timing buffered disk reads		timing buffered disk reads	timing buffered disk reads	timing buffered disk reads	
201,50		201,51	201,55	201,22	
201,10		201,18	201,16	200,99	
201,63		201,77	201,71	201,59	
201,62		201,43	200,93	190,18	
200,97		201,88	201,82	194,95	
13 461,18	M	13 369,21	M	13 282,02	M
201,36	M	201,55	M	197,79	M
time disarhive					
7,75		11,62		9,64	
9,39		6,89	7,06	7,30	
6,83		6,83	7,04	6,92	
6,93		6,89	6,89	6,91	
6,84		7,12	6,91	7,20	
M		M	M	M	
7,548		7,87	6,975	7,594	
		None	mq-deadline	kyber	bfq
Чтение из кэша(Mb/s)		13461,175	13369,214	13437,8	13282,02
Чтение из дискового буфера(Mb/s)		201,364	201,554	201,434	197,786
Распаковка архива(s)		7,548	7,87	6,975	7,594

Как мы видим, лидер по скорости чтения из кэша и из дискового буфера - BFQ(причём для дискового буфера у BFQ хороший отрыв). По скорости распаковки тяжеловесного

архива лидер с большим отрывом - kyber. **Вывод:** из всех представленных планировщиков ввода и вывода для kali linux 5.16.3 наилучшим оказался BFQ, но посоревноваться с ним может kyber, т.к отрыв при распаковке тяжеловесного архива у него достаточно большой, а остальные показатели не так уж и далеки от BFQ. Так же мы видим как сильно отстает по времени none, основанный на простых очередях и mq-deadline, основанный на вычислении времени запроса в очереди, то есть он гарантирует обслуживание каждого запроса планировщиком.

Усложнённый вариант:

Выберем планировщик BFQ и посмотрим какие у него есть параметры с помощью:

```
(any@kali)-[/sys/block/sda/queue/iosched]
$ ls
back_seek_max back_seek_penalty fifo_expire_async fifo_expire_sync low_latency max_budget slice_idle slice_idle_us strict_guarantees timeout_sync
```

Изменим два значения, если быть точными то back_seek_max с 16 Кб на 32 Кб и slice_idle с 8 мс на 2 мс.

back_seek_max

Обратный поиск характеризуется более длительными задержками и негативно влияет на производительность. Этот параметр ограничивает расстояние для обратного поиска. По умолчанию составляет 16 КБ.

slice_idle — время ожидания поступления запросов.

По умолчанию 8 мс;

Посмотрим как изменились наши значения результата hddparm для 5-ти итераций:

```
(any@kali)-[/sys/block/sda/queue/iosched]
$ repeat 5 sudo hddparm -tT /dev/sda

/dev/sda:
Timing cached reads: 26394 MB in 2.00 seconds = 13215.52 MB/sec
Timing buffered disk reads: 610 MB in 3.00 seconds = 203.07 MB/sec

/dev/sda:
Timing cached reads: 26630 MB in 2.00 seconds = 13334.37 MB/sec
Timing buffered disk reads: 582 MB in 3.00 seconds = 193.94 MB/sec

/dev/sda:
Timing cached reads: 25812 MB in 2.00 seconds = 12923.72 MB/sec
Timing buffered disk reads: 568 MB in 3.00 seconds = 189.09 MB/sec

/dev/sda:
Timing cached reads: 26352 MB in 2.00 seconds = 13194.16 MB/sec
Timing buffered disk reads: 594 MB in 3.01 seconds = 197.42 MB/sec

/dev/sda:
Timing cached reads: 26232 MB in 2.00 seconds = 13134.21 MB/sec
Timing buffered disk reads: 606 MB in 3.00 seconds = 201.94 MB/sec
```


Матожидание времени чтения из кэша стало 13160.396, а было 13282.02(среднее уменьшилось не сильно, но отдельные замеры отличаются достаточно сильно)
Матожидания времени чтения из дискового буфера стало 194.092, а было 197.79(среднее время уменьшилось, что видно и по результатам отдельных замеров).

```
(anykali)-[~]  
$ repeat 5 time tar xJf linux-5.17.5.tar.xz  
  
real    7.88s  
user    6.88s  
sys     2.92s  
cpu     124%  
  
real    7.02s  
user    6.68s  
sys     3.46s  
cpu     111%  
  
real    6.86s  
user    6.78s  
sys     3.48s  
cpu     149%  
  
real    6.87s  
user    6.79s  
sys     3.59s  
cpu     151%  
  
real    6.92s  
user    6.71s  
sys     3.40s  
cpu     146%
```

На скриншоте представлено время разархивирования тяжеловесного архива, посчитаем же матожидание реального времени, оно равно 7.11 с, для сравнения предыдущий результат был 7.594, как мы видим результаты действительно улучшились. Таким образом мы модифицировали существующий планировщик ввода-вывода на уровне ядра, о чём мы можем судить исходя из результатов измерений.