Задание: провести тестирование и найти лучший планировщик ввода-вывода среди других. Усложнение: модифицировать существующий планировщик на уровне ядра.

Task: to test and find the best I/O scheduler among others. Complication: Modify an existing scheduler at the kernel level.

Ход работы

1. Для тестирования был использован следующий Bash-скрипт, который для каждого из планировщиков (mq-deadline, bfq, kyber, none) проводил по 15 тестов:

```
#!/bin/bash
DISC="sda";
cat /sys/block/$DISC/queue/scheduler;
for T in kyber bfq none mq-deadline; do
        echo $T > /sys/block/$DISC/queue/scheduler;
        cat /sys/block/$DISC/queue/scheduler;
        for i in {1..15}; do
                sync && /sbin/hdparm -tT /dev/$DISC && echo "----";
        done;
done
```

Фрагмент вывода программы:

```
root@ubuntu:/home/travis/Desktop# ./my_script.sh
mq-deadline [kyber] bfq none
mq-deadline [kyber] bfq none

/dev/sda:
 Timing cached reads:   11438 MB in  2.00 seconds = 5725.27 MB/sec
SG_IO: bad/missing sense data, sb[]:  70 00 05 00 00 00 00 0a 00 00 00 00 20 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 Timing buffered disk reads: 3280 MB in  3.00 seconds = 1092.66 MB/sec
----

/dev/sda:
 Timing cached reads:   11400 MB in  2.00 seconds = 5705.77 MB/sec
SG_IO: bad/missing sense data, sb[]:  70 00 05 00 00 00 00 0a 00 00 00 00 20 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 Timing buffered disk reads: 3116 MB in  3.00 seconds = 1037.33 MB/sec
----

/dev/sda:
 Timing cached reads:   11444 MB in  2.00 seconds = 5727.99 MB/sec
SG_IO: bad/missing sense data, sb[]:  70 00 05 00 00 00 00 0a 00 00 00 00 20 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 Timing buffered disk reads: 3206 MB in  3.00 seconds = 1068.05 MB/sec
----
```

Сравнение проводилось по скорости чтения (MB/s) из дискового буфера (Timing buffered disk reads) и по скорости чтения из кеша (Timing cached reads).

mq-deadline		bfq		kyber		none	
cache	buffer	cache	buffer	cache	buffer	cache	buffer
5026,02	1067,50	5644,51	1036,55	5725,27	1092,66	5208,75	1070,52
5009,66	1052,15	5432,29	1104,47	5705,77	1037,33	5468,84	1059,27

5065,64	1032,50	4061,56	1024,64	5727,99	1068,05	5388,82	1078,43
4722,92	1111,50	1761,18	909,88	5775,70	1047,96	5393,75	1089,64
5516,24	1007,96	2614,85	1038,55	5594,15	1067,79	5570,64	1017,62
5490,40	1096,06	3935,32	1031,76	5878,93	1047,05	4016,00	1008,34
5387,14	1098,72	3392,86	1042,84	5077,17	1078,33	5286,93	1078,21
5307,48	1056,50	4756,49	1015,68	5600,81	1102,85	4989,20	1054,22
5577,50	1108,63	5202,50	1046,55	5546,89	971,86	5363,40	1051,21
5602,43	1118,15	4914,37	1053,13	5263,07	1051,52	5077,37	1106,43
5447,9	1081,16	5406,62	1011,74	5270,38	1118,3	2749,56	1043,91
5596,94	1100,94	5407,82	1056,23	5518,89	1045,53	4839,28	1073,86
5482,35	1112,82	5475,91	1079,23	5660,67	1113,84	5247,17	1019,01
5563,98	1116,45	5491,56	1045,61	5527,15	935,07	5587,71	1104,07
5590,86	1061,91	4950,73	1020,74	4365,55	1090,77	4239,72	1023,72
5359,16	1081,53	4563,24	1034,51	5482,56	1057,93	4961,81	1058,56
274,06	33,75	1177,97	42,06	376,34	50,02	762,37	31,34

Таблица 1. Результаты тестов

(зеленые значения - математические ожидания, жёлтые значения - среднеквадратичные отклонения)

2. Для модификаций был выбран планировщик bfq в связи с тем, что у него имеется наибольшее число параметров, что позволяет больше экспериментировать.

Для начала перейдем на планировщик bfq:

echo "bfq" | sudo tee /sys/block/sda/queue/scheduler

Далее в каталоге /sys/block/sda/queue/iosched можем увидеть файлы, соответствующие параметрам планировщика bfq. В них записаны значения параметров.

```
root@ubuntu:/home/travis/Desktop# echo "bfq" | sudo tee /sys/block/sda/queue/scheduler
bfq
root@ubuntu:/home/travis/Desktop# ls /sys/block/sda/queue/iosched
back_seek_max      fifo_expire_sync  slice_idle         timeout_sync
back_seek_penalty  low_latency       slice_idle_us
fifo_expire_async  max_budget        strict_guarantees
```

Изменим следующие параметры:

- slice_idle – значение в миллисекундах, указывает, как долго простаивать, ожидая следующего запроса в пустой очереди. По умолчанию равно 8 – повысим до 20.
- fifo_expire_async – значение в миллисекундах, используется для установки тайм-аута асинхронных запросов. По умолчанию 250 – снизим до 150.
- fifo_expire_sync – тайм-аут для синхронных запросов. По умолчанию 125 – повысим до 200.
- timeout_sync – максимальное время в миллисекундах обслуживания задачи (очереди) после ее выбора. По умолчанию 124 – повысим до 160.

Получаем следующие результаты:

cache	buffer
5006,21	1048,34
5215,39	997,74
4997,47	1051,75
5196,63	1023,7

5313,83	1053,21
5185,26	1054,4
5180,52	1075,93
5275,79	1033,16
5510,94	1009,67
5102,59	1080,94
4521,58	1012,74
5017,1	1097,96
4713,74	1057,32
4803,89	1057,64
5175,07	1082,14
5081,07	1049,11
250,68	29,06

Таблица 2. Результаты для bfq после модификаций
(зеленые значения - математические ожидания, жёлтые значения - среднеквадратичные отклонения)

Вывод:

По результатам тестов с точки зрения чтения из кеша наилучшим образом себя показал планировщик ввода-вывода kyber с почти наименьшим отклонением. По чтению из дискового буфера значение несильно отстает от максимально. Следовательно, в данном тестировании лучшим планировщиком можно считать kyber. Больше всего колебались значения для неулучшенного bfq (СКО больше четверти математического ожидания). Однако, изменив его параметры, удалось повысить среднее значение и снизить СКО как для чтения из кеша, так и для чтения из буфера.

Таким образом, поставленные цели были выполнены.