**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Дисциплина:**

«Операционные системы»

**Лабораторная работа No4**

“Планировщик”

Выполнил:

Хоанг Вьет Зунг – Студент группы N32491

Нгуенг Хонг Ньят - Студент группы N32451

Преподаватель:

Савков С.В.

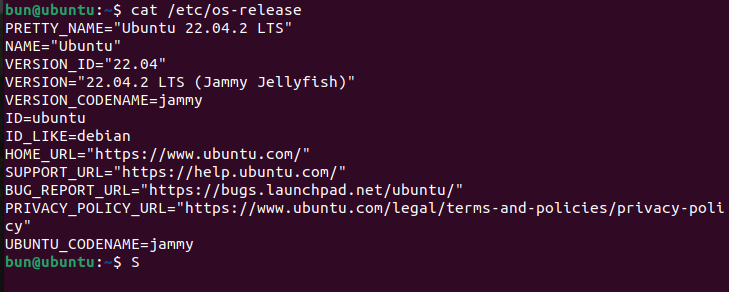


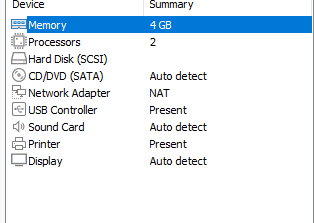
Санкт-Петербург 2023

**Задание:**

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

**Усложение:** Модифицировать существующий планировщик на уровне ядра

**Name: Ubuntu**

****

1. **Простое задание:**

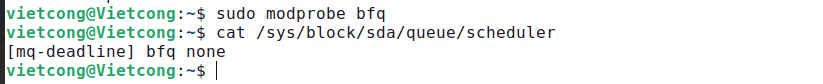
**First, we check which schedulers are in our system:**



* The **cat /sys/block/sda/queue/scheduler** command in Linux Ubuntu displays the current I/O scheduler used for the **sda** block device. When we execute this command, the output will display a list of available I/O schedulers enclosed in square brackets, with the currently active scheduler marked by parentheses.
* The **mq-deadline** is an I/O scheduler that can be used to optimize the performance of storage devices. It is a part of the multiqueue block layer in the Linux kernel that manages I/O operations to storage devices such as hard disks and solid-state drives. The **mq-deadline** scheduler aims to minimize the average response time of I/O requests by setting deadlines for each request. It accomplishes this by giving higher priority to I/O requests that are closer to their deadline, which helps to ensure that important I/O operations are completed in a timely manner.
* none: Implements a first-in-first-serve (FIFO) scheduling algorithm. It aggregates requests at a common block level via a simple last access cache.

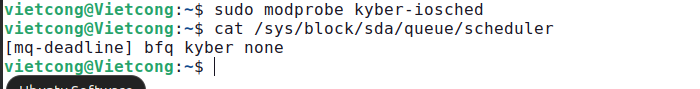
**Install scheduler bfq:**

BFQ (Budget Fair Queuing) is an I/O scheduler that is designed to improve the responsiveness and interactivity of storage devices, particularly for desktop and multimedia workloads. BFQ implements a fair queuing algorithm that divides I/O requests into multiple queues based on the priority of the process that issued the request. This allows the scheduler to give higher priority to I/O requests from interactive processes, such as those related to user input or multimedia playback, over those from background processes.



**Install scheduler kyber:**

Kyber is an I/O scheduler that is designed to improve the latency and predictability of storage devices, particularly for flash-based storage devices. Kyber is based on the concept of earliest deadline first (EDF) scheduling, which prioritizes I/O requests based on their deadline rather than their position in the queue. This allows Kyber to optimize the completion order of I/O requests to reduce latency and improve throughput.



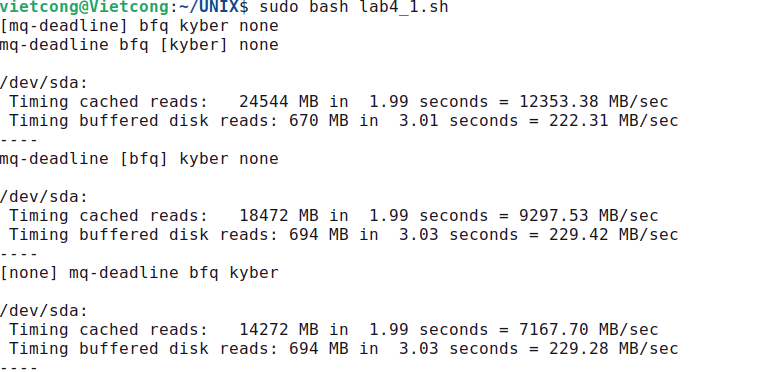
Code (Linux I/O Scheduler): this script is testing the performance of the block device specified by the **DISC** variable with different I/O schedulers (**kyber**, **bfq**, and **none**) and displaying the results of each test. The purpose of the script is to help determine which I/O scheduler provides the best performance for the block device under the current workload.

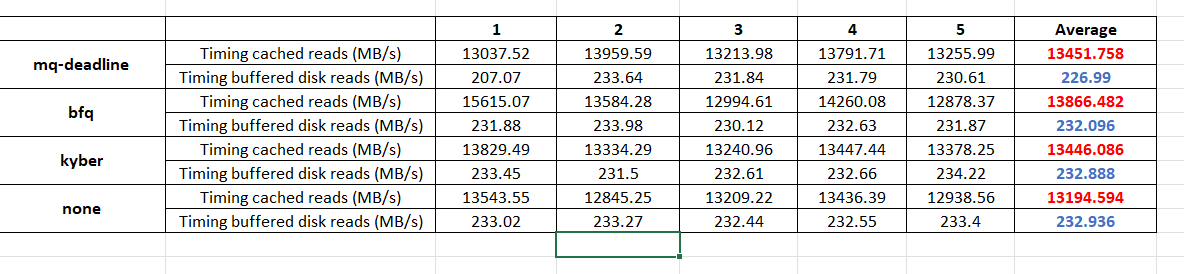
1. Sets the **DISC** variable to "sda".
2. Displays the current I/O scheduler for the block device specified by the **DISC** variable using the **cat** command and the **/sys/block/$DISC/queue/scheduler** file.
3. Enters a loop that tests the performance of the block device with different I/O schedulers.
4. For each iteration of the loop, the script sets the I/O scheduler for the block device to **kyber**, **bfq**, and then **none** using the **echo** command and the **/sys/block/$DISC/queue/scheduler** file.
5. After setting the I/O scheduler for the block device, the script displays the new scheduler using the **cat** command and the **/sys/block/$DISC/queue/scheduler** file.
6. The script then performs a disk read test using the **hdparm** command with the **-tT** options on the block device specified by the **DISC** variable.
7. After the test is complete, the script displays a separator line using the **echo** command.
8. The script waits for 15 seconds using the **sleep** command before starting the next iteration of the loop.

Text

Description automatically generated

**Run the code:**

****

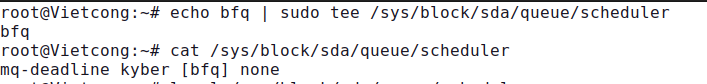
According to the results of 5 try-outs:

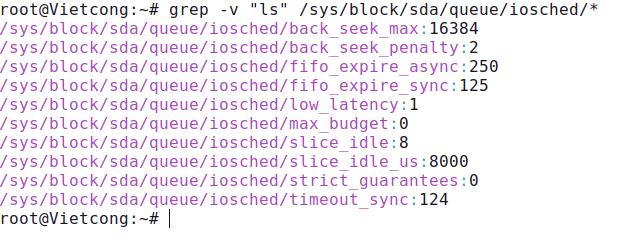
+ Timing cached reads: bfq > mq-deadline > kyber > none

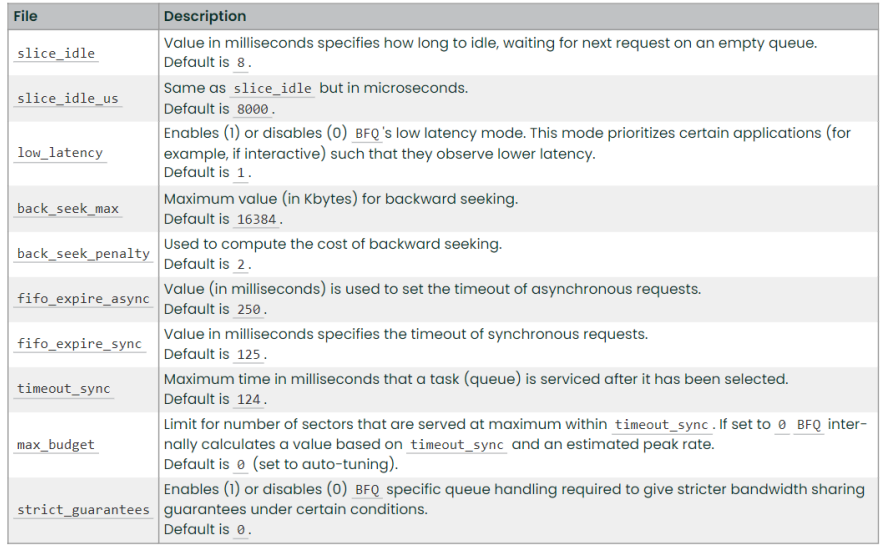
+ Timing buffered disk reads: none > kyber > bfq > mq-deadline

1. **Усложнение: Модифицировать существующий планировщик на уровне ядра.**

Для планировщика “bfq”:





\*\* test.sh scripts:

Text

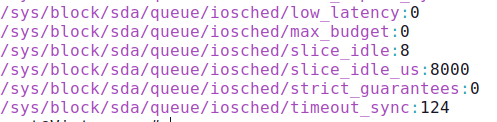
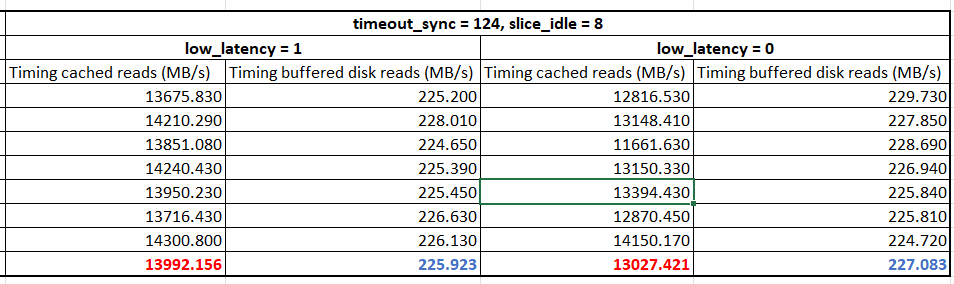
Description automatically generated

We will change 3 parameters: low\_latency, timeout\_sync, slice\_idle

* **low\_latency**: When this parameter is enabled (set to 1), the BFQ scheduler gives more priority to interactive processes, which can improve responsiveness for tasks such as desktop applications and web browsing. However, enabling this parameter may decrease overall throughput for background tasks such as file transfers.
* **timeout\_sync**: This parameter controls the maximum amount of time the BFQ scheduler waits for a synchronous I/O operation to complete before switching to another task. By default, this parameter is set to a high value (2000 milliseconds), which can cause slow responsiveness for interactive tasks. We can reduce this value to improve responsiveness, it may increase the CPU overhead of the scheduler.
* **slice\_idle**: This parameter controls the time slice given to idle processes before switching to another task. By default, this parameter is set to a high value (5000 milliseconds), which can cause slow responsiveness for interactive tasks. We can reduce this value to improve responsiveness, but it may increase the CPU overhead of the scheduler.

A screenshot of a computer

Description automatically generated with low confidence



Chart, line chart

Description automatically generated

Chart, line chart

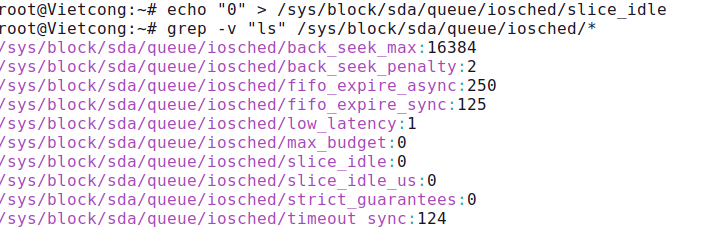
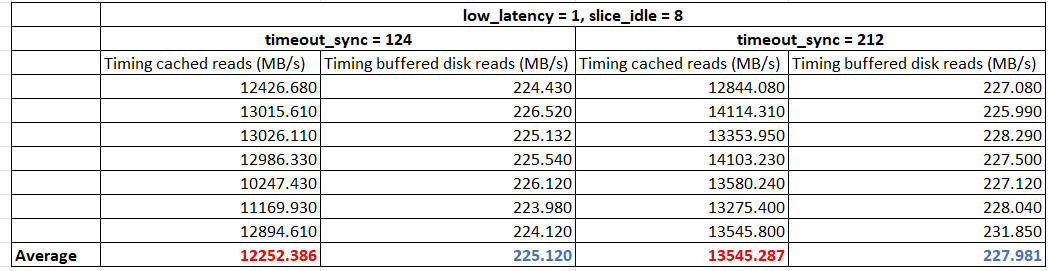
Description automatically generated

We can see that decreasing "low\_latency" may decrease the timing of cached reads, as the scheduler may prioritize requests with high throughput requirements over those with low latency requirements, which may include cached reads. On the other hand, it may slightly increase the timing of buffered disk reads, as the scheduler may prioritize these requests more quickly, resulting in faster disk I/O throughput.

\*\*

Text

Description automatically generated



Chart, line chart

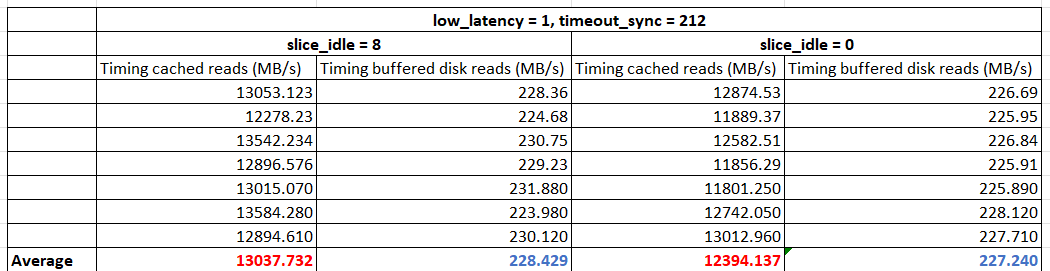
Description automatically generated

Chart, line chart

Description automatically generated

From the results, increasing the "timeout\_sync" from 124 to 214 may increase the timing of buffered disk reads, as the scheduler may wait for a longer period of time before dispatching synchronous I/O. Also, it may slightly increase the timing of cached reads, as the scheduler may wait longer before dispatching these requests in favor of other requests.

\*\*

Text, letter

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

From the results, decreasing the "slice\_idle" from 8 to 0 may decrease the timing of cached reads, as the scheduler may check for new I/O requests more frequently and prioritize them over cached reads. Also, it may decrease the timing of buffered disk reads, as the scheduler may prioritize these requests more quickly.