CS60038 - Advances in Operating Systems

Assignment 1 Part-A Report

20CS10010 - Ashish Rekhani
20CS30063 - Aman Sharma

Objective: In this assignment, we needed to configure, build, and install a Linux kernel from the source. We needed to make the following configurations using menuconfig, and then build and install the compiled kernel over the Ubuntu 20.04 LTS Desktop version:

1. Remove NUMA memory allocation, scheduler, and emulation
2. Remove Kyber I/O Scheduler
3. Include multipath TCP (MPTCP)

NUMA Memory Allocation, Scheduler, and Emulation

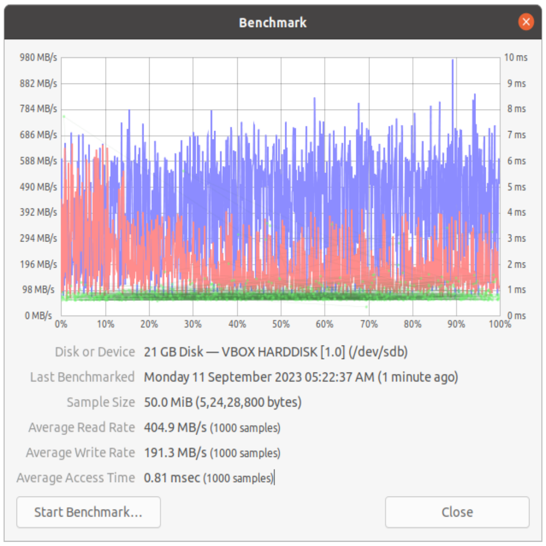
NUMA Memory Allocation, Scheduler Support, and Emulation were turned off by navigating to '*Processor type and features* -> *NUMA Memory Allocation and Scheduler Support'*

NUMA stands for non-uniform memory architecture (or access). It is a type of physical memory design used in shared memory processing (or symmetric multiprocessing)architectures. This allows the processor to have faster access to its local memories in comparison to the shared or non-local memory.
Enabling NUMA in the Linux kernel optimizes memory allocation by favoring the local memory node of the requesting CPU and employs NUMA-aware process scheduling for improved performance on multi-CPU, multi-memory-node systems keeping in mind memory locality.
In contrast, disabling NUMA results in more uniform memory allocation and non-NUMA-aware scheduling, which can lead to higher memory access latencies and suboptimal performance in NUMA architectures.
However, on a single-CPU system, NUMA settings have minimal impact, as there's only one CPU and memory node, making these optimizations less relevant. So no significant difference in performance can be found.

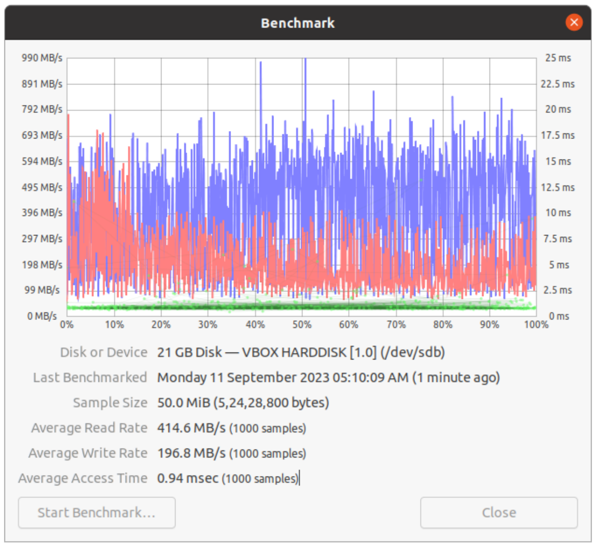
KYBER I/O Scheduler

Kyber I/O Scheduler was turned off by navigating to '*IO Schedulers -> Kyber I/O Scheduler*'

In the VM provided to us, kernel 5.15.0-82 Kyber Scheduler is set. While in the kernel 5.10.191 config file Kyber is not set and so BFQ scheduler is set.
BFQ: This scheduler prioritizes latency rather than maximum throughput. The BFQ disk scheduler is recommended for desktop or interactive tasks and traditional HDD storage.
KYBER: This scheduler tunes itself by analyzing I/O requests. A calculation is done with each IO test to determine if the I/O can be satisfied with the least amount of latency. Kyber is recommended for high performance storage like SSDs and NVMe drives.

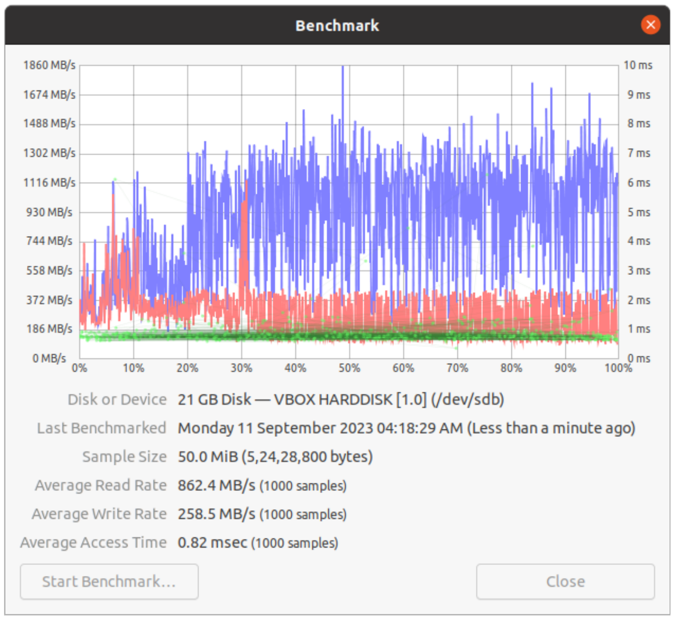


Disk or Device 21 GB Disk — VBOX HARDDISK [1.0] (/dev/sdb)
Last Benchmarked Monday 11 September 2023 05:22:37 AM (1 minute ago)
Sample Size 50.0 MiB (5,24,28,800 bytes)
Average Read Rate 404.9 MB/s (1000 samples)
Average Write Rate 191.3 MB/s (1000 samples)
Average Access Time 0.81 msec (1000 samples)



Disk or Device 21 GB Disk — VBOX HARDDISK [1.0] (/dev/sdb)
Last Benchmarked Monday 11 September 2023 05:10:09 AM (1 minute ago)
Sample Size 50.0 MiB (5,24,28,800 bytes)
Average Read Rate 414.6 MB/s (1000 samples)
Average Write Rate 196.8 MB/s (1000 samples)
Average Access Time 0.94 msec (1000 samples)

Given above are the benchmarking results of a kernel with KYBER I/O scheduler (right) and BFQ scheduler (left). As we can note, KYBER scheduler prioritizes throughput over access time while BFQ scheduler does the opposite and prioritizes latency. These numbers are not as indicative of real world values as we would like them to be since we are running these benchmarks on a VM and there is another layer of scheduling happening between the VM and hardware. However the results are enough to show the difference between the two and highlight the importance the 2 schedulers give to throughput v/s latency.



Disk or Device 21 GB Disk — VBOX HARDDISK [1.0] (/dev/sdb)
Last Benchmarked Monday 11 September 2023 04:18:29 AM (Less than a minute ago)
Sample Size 50.0 MiB (5,24,28,800 bytes)
Average Read Rate 862.4 MB/s (1000 samples)
Average Write Rate 258.5 MB/s (1000 samples)
Average Access Time 0.82 msec (1000 samples)

The above benchmarking result is from NO-OP scheduler which essentially does scheduling in a FIFO queue and thus no major scheduling decisions are being taken, it is very fast for our case which is accessing an SSD. This is also indicative of real world performance, if not by numbers but by comparison, where NO-OP or no scheduler for SSD gives the best performance as scheduling aims to reduce the seek time of using a HDD, which is not the case for an SSD. Thus the scheduler of choice would depend on your hardware.

Multipath TCP (MPTCP)

Multipath TCP was turned on by navigating to '*Networking Support -> Networking Options -> MPTCP: Multipath TCP*'

The Multipath TCP (MPTCP) protocol allows for simultaneous usage of multiple paths between connection endpoints. The protocol design improves connection stability and also brings other benefits compared to the single-path TCP.
In order to facilitate its deployment, Multipath TCP presents the same socket interface as TCP. This implies that any standard TCP application can be used above Multipath TCP while in fact spreading data across several subflows.
Enabling Multipath TCP (MPTCP) in the Linux kernel can enhance network performance, resilience, and bandwidth utilization by allowing a single network connection to utilize multiple paths simultaneously. Disabling MPTCP reverts to the traditional single-path TCP behavior, which may not take full advantage of available network resources. However, on machines with a single network interface, these differences are negligible, as MPTCP's benefits are more pronounced in multipath network environments where it can distribute the traffic over multiple available paths and achieve an effective bandwidth utilization close to the sum of the individual bandwidths.
Link: Paper that compares performance benefit of MPTCP over single flow TCP