Sam Anderson

COEN 241

HW 1

10/11/2021

Homework 1 Report

Virtualizing Computers vs. Virtualizing Operating Systems

Setup and system environment

1. For this project, all tests and systems were executed within a Windows 10 host which has access to plenty of compute power. The total system has an 8-core, 16-thread Intel 11700K processor, 16GB of memory and for file tests, a WD550 500GB M.2 SSD.
2. Once QEMU was successfully installed using the documentation provided, for control QEMU tests the following command was used so control tests have 4Gb of memory and 4 threads.

```
qemu-system-x86_64 -hda ubuntu.img -boot cd -cdrom Downloads/ubuntu-20.04.3-live-server-amd64.iso -m 4096 -L "C:\Program Files\qemu" -smp 4
```

3. Sysbench 1.0.18 was installed using the provided documentation.
4. Docker was then installed but, rather than using the provided image, a custom ubuntu container was made because the container provided used a different Sysbench version. A fresh ubuntu container allowed for easier testing because it has set testing length and similar reporting with events per second.
5. With Docker installed, a custom Docker image was created by running the following command:

```
docker run -i -t ubuntu:latest /bin/bash
```

6. If we wanted to stop a docker container to ensure there was only 1 running container for testing, we will run the following commands.

a. To get running containers

```
docker container ls
```

b. Then stop the container by using the string under the “NAMES” column

```
docker stop <container name>
```

7. After that, we can install Sysbench in the same way within the QEMU VM.
8. With the scripts, then manually entering and the raw data into the provided excel spreadsheet, we can get our results.
9. Finally, additional tests can run and, their commands will be provided.

CPU and Fileio Tests

1. **Control** Test:
   a. This will be our control value to compare experiments.
   b. Docker start command:

```
docker run -it  --memory="4g" ubuntu
```

   c. QEMU start command:

```
qemu-system-x86_64 -hda ubuntu.img -boot cd -cdrom Downloads/ubuntu-20.04.3-live-server-amd64.iso -m 4024 -L "C:\Program Files\qemu" -smp 1
```

   d. Sysbench CPU test command:

```
sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
```

   e. Sysbench fileio Command

```
sysbench --test=fileio --file-block-size=1G --file-num=5 --time=30 –file-test-mode=rndrw
```

2. **Additional Threads** (From 1 thread to 4 threads)
    a. We will see hoe both docker and QEMU scale as a task is allowed to use more threads.
    b. Docker start command:

```
docker run -it  --memory="1g" ubuntu
```

    c. QEMU start command:

```
qemu-system-x86_64 -hda ubuntu.img -boot cd -cdrom Downloads/ubuntu-20.04.3-live-server-amd64.iso -m 1024 -L "C:\Program Files\qemu" -smp 4
```

    d. Sysbench run command:

```
sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
```

    e. Sysbench fileio Command: N/A

3. **Limited Memory** (4GB to 1GB in Docker and QEMU)
    a. See how both environments react to limitations in memory.
    b. Docker start command:

```
docker run -it  --memory="1g" ubuntu
```

    c. QEMU start command:

```
qemu-system-x86_64 -hda ubuntu.img -boot cd -cdrom Downloads/ubuntu-20.04.3-live-server-amd64.iso -m 1024 -L "C:\Program Files\qemu" -smp 4
```

    d. Sysbench run command:

```
sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
```

e. Sysbench fileio Command: N/A

4. **QEMU** with -accel whpx
   a. This is expected to lead to significant improvements given the host environment allows for shortcuts which allow for more direct CPU access, leading to less overhead calculations and additional user-end performance gains.
   b. Docker start command: [Not available with docker]
   c. QEMU start command:

```
qemu-system-x86_64 -hda ubuntu.img -boot cd -cdrom Downloads/ubuntu-20.04.3-live-server-amd64.iso -m 4096 -L "C:\Program Files\qemu" -smp 4 -accel whpx
```

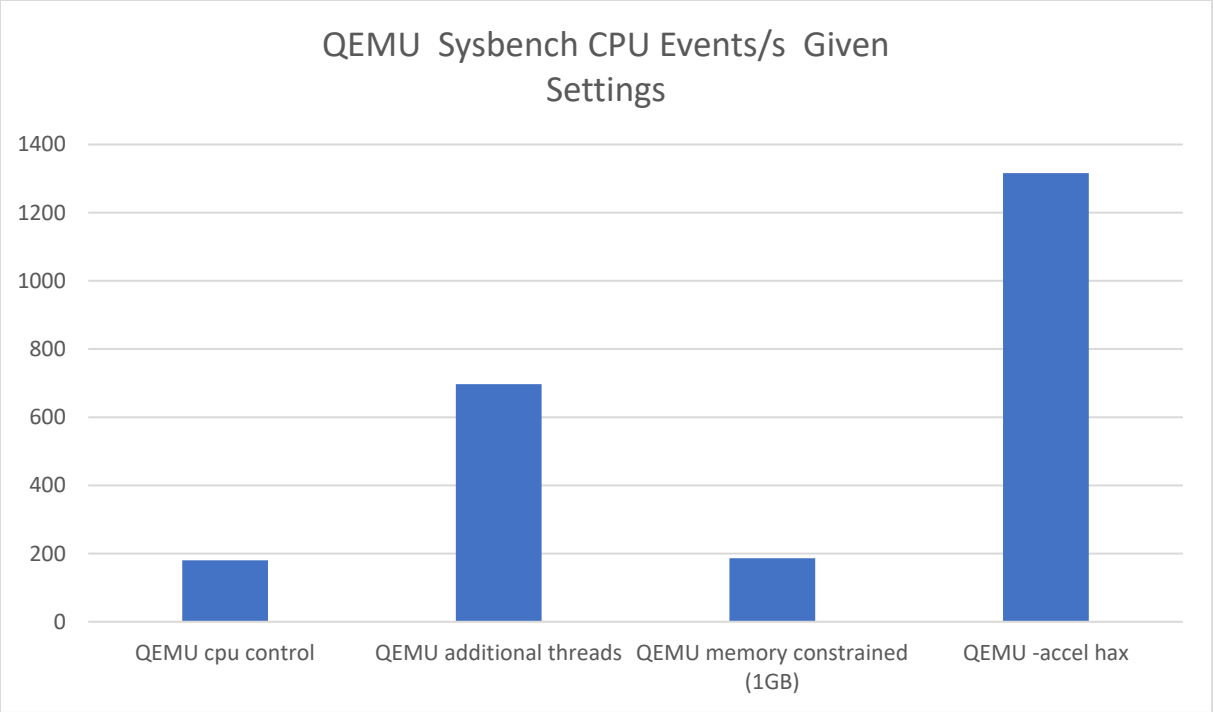
   d. Sysbench run command:

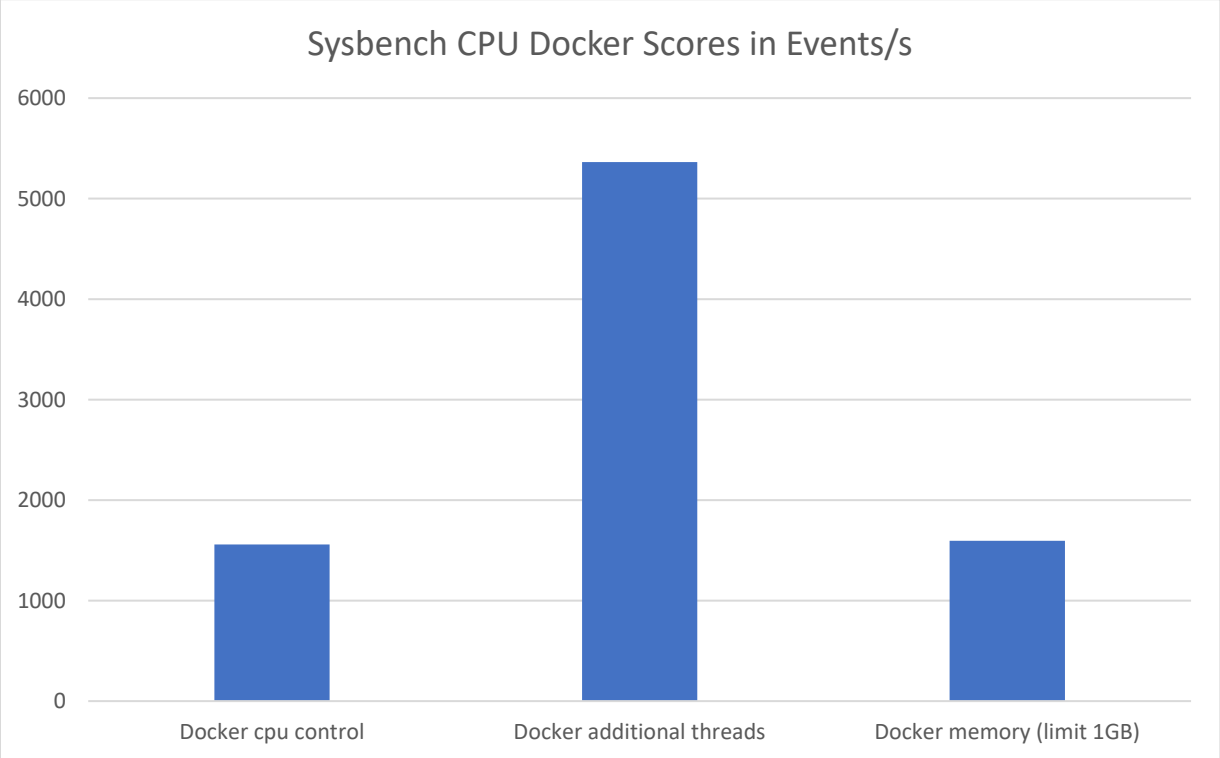
```
sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
```

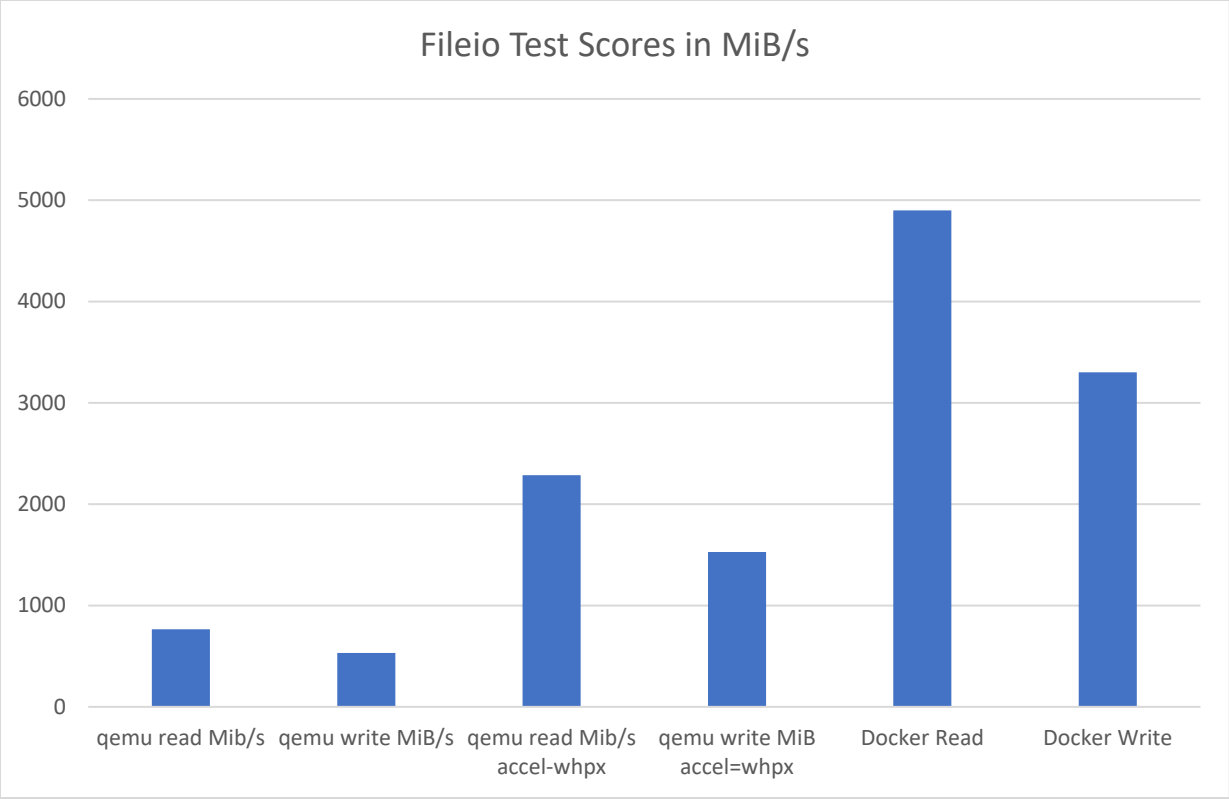
   e. Sysbench fileio Command

```
sysbench --test=fileio --file-block-size=1G --file-num=5 --time=30 –file-test-mode=rndrw
```

Charts


QEMU Sysbench CPU Events/s Given Settings
1400
1200
1000
800
600
400
200
0
QEMU cpu control
QEMU additional threads
QEMU memory constrained (1GB)
QEMU -accel hax


Sysbench CPU Docker Scores in Events/s
6000
5000
4000
3000
2000
1000
0
Docker cpu control
Docker additional threads
Docker memory (limit 1GB)



Conclusions

Starting with the CPU tests, there are clear differences between the control group as the docker container outpaces QEMU speeds by be nearly a factor of 8. The QEMU virtual machine scored an average of ~180 events/s while the docker container scored ~1559 events/s. This was partially expected since there is a lot of overhead emulating an entire Unix system, rather than just a Unix OS. This means memory calls, and all CPU instructions all needed to be translated so the windows host can understand how the program runs.

Next, when we gave the Sysbench test 4 threads. Given this can multiply performance with parallelism, scores were expected for both tests with an observed a ~3.5-3.8x improvement with additional threads. This was expected given some overhead with managing threads. Additionally, it important to note, there is similar scalability when a host OS is managing many threads running intensive workloads.

After that, we tested QEMU and the docker container with a limited 1GB of memory. Surprisingly both systems were slightly faster. This would be possible if the container and guest OS in QEMU were never threatened with a memory bottleneck. In which case, the host OS will have additional resources to run the container or emulation.

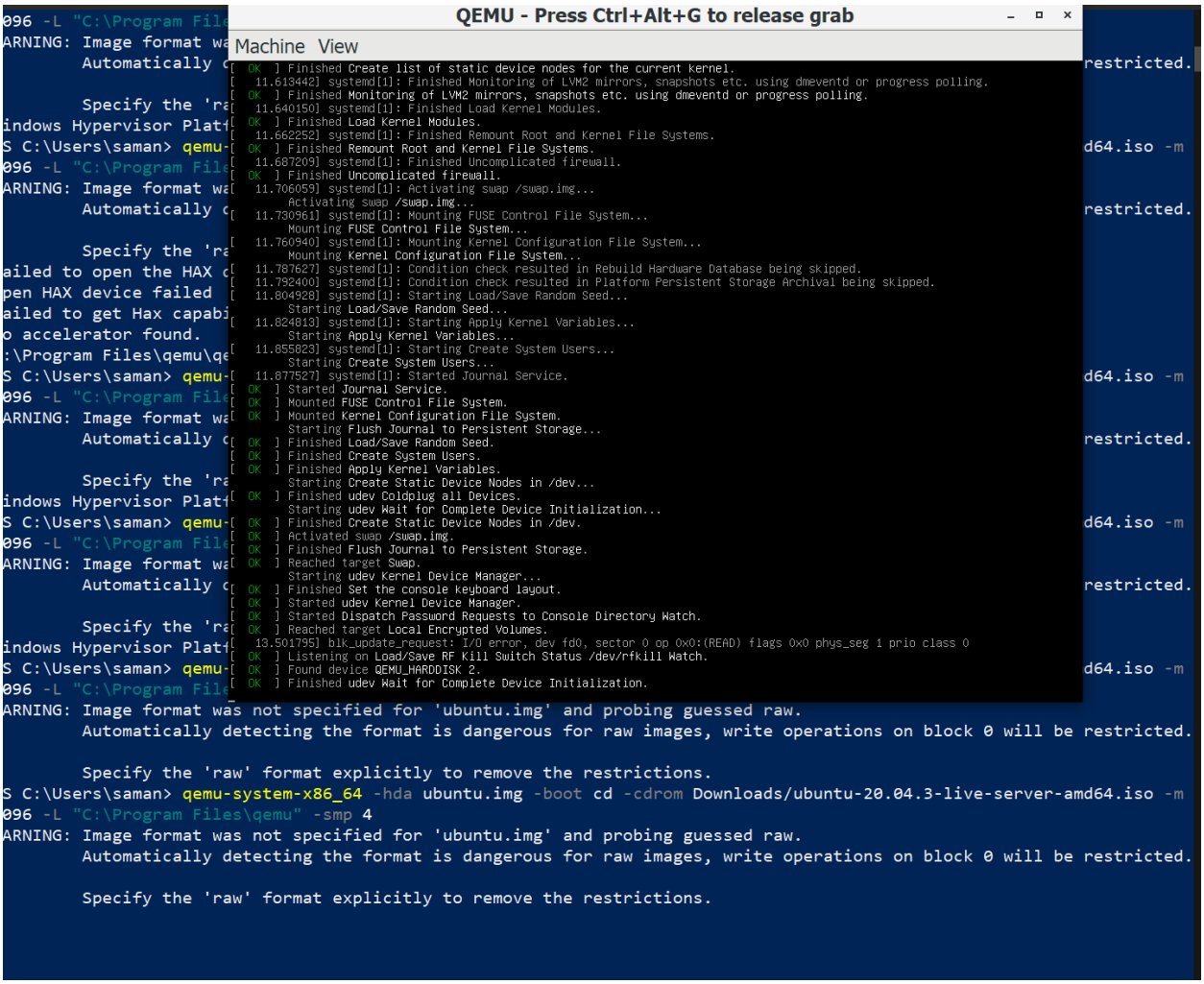
Finally, when we tested the QEMU system with hardware acceleration from Microsoft. There were significant performance boosts where the QEMU test jumped to ~1316 events/s, which is much more comparable, yet still lagging, to docker's performance benchmarks. Although containers are still faster, in certain applications that benefit from hardware level separation, will still be a strong solution due to the reduced computation overhead of a virtual machine. These results show that hardware acceleration is superior to allocating additional threads to a workload because 4 parallel threads were beaten by a single thread using hardware acceleration.

For the Fileio test, as an important note, **all tests appeared to exclusively use RAM for file storage and access** so many metrics will exceed the rated speed of my SSD yet produce expected results with memory access speeds. For every Fileio test, I noticed disc activity remained low, yet memory use and activity were high. With that noted, there are similar speed differences between regular QEMU, QEMU with whpx enabled, and Docker when referencing differences in the CPU test. The one metric that whpx was unable to significantly improve was access latency. While docker was at ~35ms average latency, yet all QEMU tests ware getting 200-400ms latency.

Gallery

```
TEST 9
WARNING: the --test option is deprecated. You can pass a script name or path on the
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:   186.63

General statistics:
    total time:                          30.0028s
    total number of events:              5600

Latency (ms):
         min:                                    4.87
         avg:                                    5.35
         max:                                   27.92
         95th percentile:                        5.57
         sum:                                29937.02

Threads fairness:
    events (avg/stddev):           5600.0000/0.00
    execution time (avg/stddev):   29.9370/0.00

cloud@cloud:~/scripts/HW1$ _
```

```
# ./cpu_test.sh
what command do we use?
sysbench --test=cpu --cpu-max-prime=20000 --time=30 run
TEST 0
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  1569.65

General statistics:
    total time:                          30.0005s
    total number of events:              47092

Latency (ms):
         min:                                    0.61
         avg:                                    0.64
         max:                                    1.43
         95th percentile:                        0.68
         sum:                                29977.83

Threads fairness:
    events (avg/stddev):           47092.0000/0.00
    execution time (avg/stddev):   29.9778/0.00

TEST 1
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!
```


QEMU - Press Ctrl+Alt+G to release grab
Machine View
[OK] Finished Create list of static device nodes for the current kernel.
[11.613442] systemd[1]: Finished Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling.
[OK] Finished Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling.
[11.640150] systemd[1]: Finished Load Kernel Modules.
[OK] Finished Load Kernel Modules.
[11.662252] systemd[1]: Finished Remount Root and Kernel File Systems.
[OK] Finished Remount Root and Kernel File Systems.
[11.687209] systemd[1]: Finished Uncomplicated firewall.
[OK] Finished Uncomplicated firewall.
[11.706059] systemd[1]: Activating swap /swap.img...
Activating swap /swap.img...
[11.730961] systemd[1]: Mounting FUSE Control File System...
Mounting FUSE Control File System...
[11.760940] systemd[1]: Mounting Kernel Configuration File System...
Mounting Kernel Configuration File System...
[11.787627] systemd[1]: Condition check resulted in Rebuild Hardware Database being skipped.
[11.792400] systemd[1]: Condition check resulted in Platform Persistent Storage Archival being skipped.
[11.804928] systemd[1]: Starting Load/Save Random Seed...
Starting Load/Save Random Seed...
[11.824813] systemd[1]: Starting Apply Kernel Variables...
Starting Apply Kernel Variables...
[11.855823] systemd[1]: Starting Create System Users...
Starting Create System Users...
[11.877527] systemd[1]: Started Journal Service.
[OK] Started Journal Service.
[OK] Mounted FUSE Control File System.
[OK] Mounted Kernel Configuration File System.
Starting Flush Journal to Persistent Storage...
[OK] Finished Load/Save Random Seed.
[OK] Finished Create System Users.
[OK] Finished Apply Kernel Variables.
Starting Create Static Device Nodes in /dev...
[OK] Finished udev Coldplug all Devices.
Starting udev Wait for Complete Device Initialization...
[OK] Finished Create Static Device Nodes in /dev.
[OK] Activated swap /swap.img.
[OK] Finished Flush Journal to Persistent Storage.
[OK] Reached target Swap.
Starting udev Kernel Device Manager...
[OK] Finished Set the console keyboard layout.
[OK] Started udev Kernel Device Manager.
[OK] Started Dispatch Password Requests to Console Directory Watch.
[OK] Reached target Local Encrypted Volumes.
[13.501795] blk_update_request: I/O error, dev fd0, sector 0 op 0x0:(READ) flags 0x0 phys_seg 1 prio class 0
[OK] Listening on Load/Save RF Kill Switch Status /dev/rfkill Watch.
[OK] Found device QEMU_HARDDISK 2.
[OK] Finished udev Wait for Complete Device Initialization.
ARNING: Image format was not specified for 'ubuntu.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
S C:\Users\saman> qemu-system-x86_64 -hda ubuntu.img -boot cd -cdrom Downloads/ubuntu-20.04.3-live-server-amd64.iso -m
096 -L "C:\Program Files\qemu" -smp 4
ARNING: Image format was not specified for 'ubuntu.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.

```
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!

CPU speed:
    events per second:  1299.73

General statistics:
    total time:                          30.0011s
    total number of events:              38995

Latency (ms):
         min:                                    0.65
         avg:                                    0.74
         max:                                    4.81
         95th percentile:                        0.80
         sum:                                28923.07

Threads fairness:
    events (avg/stddev):           38995.0000/0.00
    execution time (avg/stddev):   28.9231/0.00

TEST 4
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
WARNING: --max-time is deprecated, use --time instead
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Prime numbers limit: 20000

Initializing worker threads...

Threads started!
```

```
Removing test files...
is there another test to run? [y/n]
y
Add additional flags
--file-test-mode=rndwr
sysbench --test=fileio --file-block-size=1G --file-num=5 --time=30 --file-test-mode=rndwr
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

5 files, 419430Kb each, 2047Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
```

```
File operations:
    reads/s:                      16.20
    writes/s:                     10.79
    fsyncs/s:                     1.49

Throughput:
    read, MiB/s:                  4991.73
    written, MiB/s:               3264.35

General statistics:
    total time:                          30.1132s
    total number of events:              853

Latency (ms):
         min:                                    0.16
         avg:                                   35.21
         max:                                   71.19
         95th percentile:                       63.32
         sum:                                30037.04

Threads fairness:
    events (avg/stddev):           853.0000/0.00
    execution time (avg/stddev):   30.0370/0.00

TEST 2
WARNING: the --test option is deprecated. You can pass a script name or path on the command line without any options.
sysbench 1.0.18 (using system LuaJIT 2.1.0-beta3)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Extra file open flags: (none)
5 files, 409.6MiB each
2GiB total file size
Block size 1GiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!
```