Advanced Operating Systems
Assignment Week 10

Paul Lödige
Student ID: 37-229753

December 26, 2022

1 Assignment **1**

2 Code **2**
2.1 Custom Python Code 2
2.1.1 Output 3
2.2 Using Existing Programs 4
2.2.1 Output 4

1 Assignment

Create a user-space program (any language is acceptable) that differentiates the performance of the Linux Kernel with the PREEMPT_ RT patches from that without the PREEMPT_ RT patches. Compile and run the program. Submit the source code and the screen shots of the standard output and the kernel log message, respectively, when the program is being executed.

2 Code

2.1 Custom Python Code

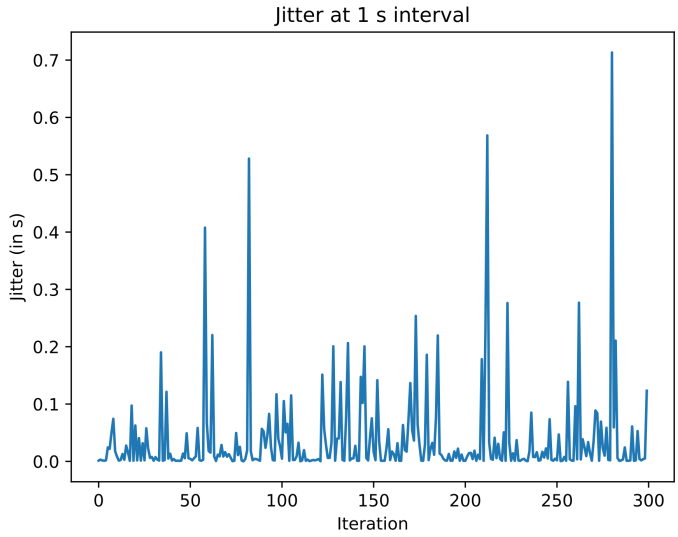
The following code was written by me with the intend to measure the jitter in a python code. It uses the *schedtool* program provided with the PREEMPT_RT patches to schedule itself.

```
0  import subprocess
1  import time
2  import matplotlib.pyplot as plt
3
4  ITERATIONS = 300
5  ITERATION_DURATION = 1 #second(s)
6
7  def set_rt_priority():
8     subprocess.run(["schedtool", "-F", "-p", "99"], check=True)
9
10 def rt_task():
11     new_time = time.time()
12     jitter_record = []
13
14     for i in range(ITERATIONS + 1):
15         old_time = new_time
16         new_time = time.time()
17
18         print("------------------------------------------------")
19         print(f"system time:\t{new_time} s")
20         print(f"time diff:\t{new_time - old_time} s")
21
22         # jitter
23         jitter = new_time - old_time - ITERATION_DURATION
24         jitter_record.append(jitter)
25         print(f"jitter:\t{jitter} s")
26
27         # sleep for the rest of one second
28         time.sleep(new_time + ITERATION_DURATION - time.time())
29
30     #remove the first measurement
31     jitter_record.pop(0)
32
33     #plot and store the jitter record
34     plt.plot(jitter_record)
35     plt.title(f"Jitter at {ITERATION_DURATION} s interval")
36     plt.xlabel("Iteration")
37     plt.ylabel("Jitter (in s)")
38     plt.savefig("out.svg")
39
40
41 if __name__ == "__main__":
42     set_rt_priority()
43     rt_task()
```

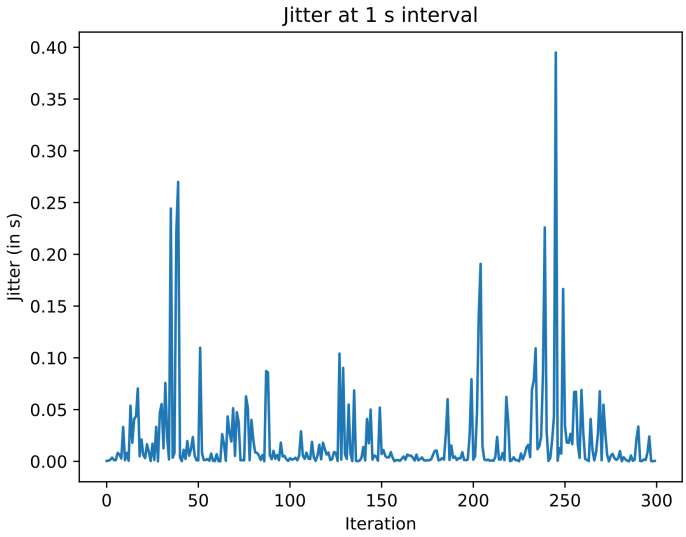
2.1.1 Output

Unfortunately, as can be seen by the following output, the jitter isn't really affected by the activation or deactivation of the PREEMPT_RT patches.

with PREEMPT_RT



without PREEMPT_RT



2.2 Using Existing Programs

The following code is just a shell script that makes use of two programs provided by the `rt-tests` package (Arch Linux). It uses the `hackbench` program to put stress on the system and then evaluates the maximum latency with the `cyclictest` program. This provides a much more reliable method of testing the real-time capabilites of the system than any self-developed code I was able to come up with.

```
0 #!/bin/bash
1 echo "================================================" >> output.txt
2 uname -a >> output.txt
3 echo "------------------------------------------------" >> output.txt
4 hackbench -l 40000 >> output.txt &
5 sleep 1
6 echo "------------------------------------------------" >> output.txt
7 sudo cyclictest -q --mlockall --smp --priority=80 --interval=200 --distance
      =0 -D 1m >> output.txt
8 echo "------------------------------------------------" >> output.txt
```

The code is based on the following two sources:
`https://wiki.archlinux.org/title/Realtime_kernel_patchset`
`https://shuhaowu.com/blog/2022/02-linux-rt-appdev-part2.html`

2.2.1 Output

The following output shows that while the PREEMPT_RT patch doesn't really affect the average latency it drastically reduces the maximum latency.

with PREEMPT_RT

```
================================================
Linux advancedoperatingsystemsvm 6.0.5-2-rt14-MANJARO #1 SMP PREEMPT_RT Sun Nov 6 15:25:56 CET 2022 x86_64 GNU/Linux
------------------------------------------------
Running in process mode with 10 groups using 40 file descriptors each (== 400 tasks)
Each sender will pass 40000 messages of 100 bytes
------------------------------------------------
# /dev/cpu_dma_latency set to 0us
T: 0 ( 2872) P:80 I:200 C: 297927 Min:      2 Act:   51 Avg:   37 Max:    1756
T: 1 ( 2873) P:80 I:200 C: 297834 Min:      2 Act:   29 Avg:   37 Max:    1726
T: 2 ( 2874) P:80 I:200 C: 297550 Min:      3 Act:   31 Avg:   47 Max:    1815
T: 3 ( 2875) P:80 I:200 C: 297519 Min:      3 Act:   34 Avg:   30 Max:    1743
------------------------------------------------
Time: 153.851
```

without PREEMPT_RT

```
================================================
Linux advancedoperatingsystemsvm 6.0.11-1-MANJARO #1 SMP PREEMPT_DYNAMIC Fri Dec 2 21:23:52 UTC 2022 x86_64 GNU/Linux
------------------------------------------------
Running in process mode with 10 groups using 40 file descriptors each (== 400 tasks)
Each sender will pass 40000 messages of 100 bytes
------------------------------------------------
# /dev/cpu_dma_latency set to 0us
T: 0 ( 2846) P:80 I:200 C: 297642 Min:      3 Act:    8 Avg:   41 Max:   34750
T: 1 ( 2847) P:80 I:200 C: 297348 Min:      3 Act:   17 Avg:   35 Max:   34712
T: 2 ( 2848) P:80 I:200 C: 297864 Min:      3 Act:   58 Avg:   60 Max:   34786
T: 3 ( 2849) P:80 I:200 C: 298013 Min:      3 Act:   50 Avg:   35 Max:   34711
------------------------------------------------
Time: 89.334
```