L50 - Lab 5, Kernel Tracing 2 & Performance with Antagonists

Professor Andrew W. Moore and Dr Gary Guo

Lent, 2022/2023

The aim of this lab is to study application performance using the KUtrace system alongside the application integrated logging used in previous labs. To extend this work by studying the impact of an antagonistic workload upon the previous lab workloads.

The approach for this lab will be to rerun a number of experiments from Lab 3 / Chapter 7 – specifically the server_disk experiments of Section 7.7 (Experiments 3 and 4) and then to study this application using the KUtrace system.

1 Tracing `server_disk`

Running the server_disk write example under KUtrace

Where required, refer to the lab4 handout for instructions on loading the KUtrace module and setting the size of the trace buffer.

Refer to the lab3 handout for instructions on operating experiments 3 and 4 from that laboratory.

Where necessary load the trace with a large trace buffer, e.g.

```
sudo insmod kutrace_mod.ko tracemb=40
```

Beware the resulting html file will be extremely large.

These traces can be quite long, by running KUtrace without instruction counter information, the trace will be more memory efficient, e.g.

```
~/KUtrace/control/control/kutrace_control
Entering kutrace_control
control>  gowrap  <<<- this command starts the trace in flight-recorder mode
```

Remember to run KUtrace on the server host.

When running the `server_disk` chksum example, a 20MB of trace is sufficient.

For each of these experiments, identify regions of particular, peculiar, unusual, or unexpected patterns.

2 Operating with an Antagonist

For these experiments repeat Lab3's experiment 3 and 4 again but this time with multiple copies of an antagonist. Our antagonist will be a cpuhog in `~/KUtrace/bookcode/aw_files/book-user-co`

Code for the cpuhog is in `~/KUtrace/bookcode/aw_files/book-user-code/cpuhog_highissue.cc` can be recompiled using the command:

```
g++ cpuhog_highissue.cc -o cpuhog
```

`cpuhog` runs for about 4 minutes then exits.

You may uncover interesting behaviours as you run experiments with varying numbers of simultaneous cpuhog executables.

An example command to run three cpuhog processes with the server.

```
~/KUtrace/bookcode/aw_files/book-user-code/cpuhog & \
~/KUtrace/bookcode/aw_files/book-user-code/cpuhog & \
~/KUtrace/bookcode/aw_files/book-user-code/cpuhog & \
~/KUtrace/bookcode/aw_files/book-user-code/server_disk diskdir
```

For each set of experiments, recall the commands to purge and reload the operating system block cache

```
# Purge page cache
echo 1 | sudo tee /proc/sys/vm/drop_caches
# Bring aaaa and dddd back to RAM - suitable for experiment 3
cat diskdir/a* diskdir/d* > /dev/null
```

Check the lab3 handout for details.

Further cpu pressure can be achieved by increasing the number of cpuhog processes and by changing the scheduling priority. This is an example using nice to make increase run priority for three cpuhog processes.

```
cat diskdir/a* diskdir/c* diskdir/d* > /dev/null
sudo nice --10 ~/KUtrace/bookcode/aw_files/book-user-code/cpuhog & \
sudo nice --10 ~/KUtrace/bookcode/aw_files/book-user-code/cpuhog & \
sudo nice --10 ~/KUtrace/bookcode/aw_files/book-user-code/cpuhog & \
~/KUtrace/bookcode/aw_files/book-user-code/server_disk diskdir
```

3 Understanding Your Experiments

A single lab report will be required for the final two labs. Instructions for the final lab report will be provided shortly.
The following prompts are intended to help you understand your results, and provide supporting evidence for your report. However, they are just suggestions — feel free to approach the data differently!

- Discuss the methodology of the measurement tools.
- Explain how the limitations of the methodology are mitigated in this lab.
- Explore the limitations of the experiments conducted in this lab, and explain where the quality of the experiment (e.g., setup, methodology) could have been improved or altered.

Remember to start each experiment with a model of your understanding, then run the experiment and you can either validate or improve, develop, and refine your understanding.
You should always look for odd or surprising results (such as time spent in unexpected ways), and try to explain them. Note that sometimes exceptional results indicate a problem in your setup or scripts.