P51a - Lab 3, Complex interactions Database interactions: Network and Disk

Prof Andrew W. Moore

Lent 2022/23

The goal of this lab is, by building upon the tools of Lab 2, investigate more complex interactions. The first of these are the interactions among client and server systems, and extend this to incorporate disk interactions at the server end.

Like the previous, as part of this lab you will be following the textbook and if you wish to compile your results into a convenient location; a blank booklet is made available to you in the P51 repository.

This lab will require you make extensive use of the toolbase built in the previous lab. Recall, the code base can be retrieved from `https://www.cl.cam.ac.uk/teaching/2223/P51/ucamonly/book-user-code.zip`

We will closely follow Chapter 7 detailing the measurement of a network-based database server and then extend this by comparing with a network-based database server that utilises a local disk store.

1 Before you begin

If you have completed the instructions of Lab2, use the code and directory created for that lab.
Update your pull of the P51a repository.

Remember, the textbook tools create html viewable using a browser; to extract svg files (for your lab reports) you can use the tool SVG crowbar 2 from `https://nytimes.github.io/svg-crowbar/`

2 Notes for Chapter 7 - Disk and Network Database Interactions

Follow the chapter closely; including the Exercises 7.1 to 7.5 as appropriate. Note 7.1 actually uses the outputs of several Chapter 6 experiments.

Like Chapter 6, you will require copies of programmes on each of your cluster machines; you will also want to collect the output log files into a single location for processing. The R.Pi machines can display the output .html files but once again you will get a better experience if you copy them to your local computer before display.

As last time, Gary Guo has provided a page with notes and corrections to the instructions. `https://hackmd.io/@nbdd0121/Skwn08b6s#Disk-amp-Network`

I have sumarised parts of it here for your convenience.
The programme timealign can take a wildcard arguement aka

```
./timealign *.log
```

However, do not attempt to timealign a set of files from different machines at the same time. E.g., execute timealign for each set of R.Pi's logs
Book Section 7.4 Experiment 1
A B and C represent three machines from your own Pi cluster.

```
# on l51-piAAA
./server4

# on l51-piBBB
./client4 l51-piAAA 12345 -k 1000 -seed1 write -key "aaaa" + -value "valueaaa_0000" + 10
./client4 l51-piAAA 12346 -k 1000 -seed1 write -key "bbbb" + -value "valuebbb_0000" + 10

# on l51-piCCC
./client4 l51-piAAA 12347 -k 200 -seed1 write -key "cccc" + -value "valueccc_0000" + 100
./client4 l51-piAAA 12348 -k 200 -seed1 write -key "dddd" + -value "valueddd_0000" + 100
```

You will need to collect the logs onto one machine prior to processing.

Programmes (such as `makeself`) expect to be run in the `book-user-code` directory.

```
./timealign client4_20230211_2124*piAAA*.log
# Must be aligned separately because they're from different machine

./timealign client4_20230211_2124*piBBB*.log
```

```
./dumplogfile4 "Four clients, 1MB writes" client4_20230211_2124*align.log > fourclient.js

# Mark file as unsorted to please makeself
sed -i '2 i \ "unsorted": true,' fourclient.json


# For Fig 7.6a
cat fourclient.json | ./makeself show_rpc.html > fourclient.html
# For Fig 7.6b
cat fourclient.json | LC_ALL=C sort | ./makeself show_rpc.html > fourclient.html
```

Gary noted an undefined (bug) behaviour in `server_disk.cc` suggesting changing line 673 to be

```
memset(&shareddata.lockandhist, 0, sizeof(shareddata.lockandhist));
```

When running the disk-based version of server run it creating/accessing files in your home directory; this is the SD-CARD and should provide the worst performance so the most interesting artefacts. You may wish to replicate the later experiments on the /flash and /nvme disks as time permits.

```
  # pi-0AA
mkdir diskdir
./server_disk diskdir
# pi-0BB
./client4 l51-pi0AA 12345 -k 1000 -seed1 write -key "aaaa" + -value "valueaaa_0000" + 100
# pi-0CC
./client4 l51-pi0AA 12347 -k 200 -seed1 write -key "cccc" + -value "valueccc_0000" + 1000
./client4 l51-pi0AA 12348 -k 200 -seed1 write -key "dddd" + -value "valueddd_0000" + 1000
```

Again, collect the logs onto one machine prior to processing.

```
./timealign client4_2023021*pi0AA*.log
# Once again, you must align each machine's logs separately because they each have a diff
./timealign client4_2023021*pi0BB*.log


./dumplogfile4 "Three clients server_disk" client4_20230211_2159*align.log > threedisk.js

# Mark file as unsorted to please makeself
sed -i '2 i \ "unsorted": true,' threedisk.json

cat threedisk.json | ./makeself show_rpc.html > threedisk.html
```

Experiment 3 uses the data created in the previous experiment (the data in `diskdir`).

These commands will ensure the experiment starts from a consistent state; first by purging the operating-system cache then by systematically accessing the a* and d* files prior to the experiment. This has the result of imposing the disk load overhead on the c* files only.

```
# on server
# l51-piAAA
# Purge page cache
echo 1 | sudo tee /proc/sys/vm/drop_caches

# Bring files back to RAM
cat diskdir/aaaa diskdir/ddddd > /dev/null
```

ot refreshing the cache, thereby imposing the disk overhead on all key accesses.

```
# l51-piAAA
  ./server_disk diskdir
# l51-piCCC
./client4 l51-piAAA 12345 -k 1000 -seed1 chksum -key "aaaa" +
# l51-piBBB
./client4 l51-piAAA 12347 -k 200 -seed1 chksum -key "cccc" + & \
./client4 l51-piAAA 12348 -k 200 -seed1 chksum -key "dddd" +

XXX bcaa might be a book typo?? XXX
```

After you have brought the log files back to a single machine, once again you need to process those logs.

```
./timealign client4_2023XYZ*pi045*.log
./timealign client4_2023XYZ*pi047*.log

./dumplogfile4 "Chksum x3, buffered I/O" client4_2023XYZ*align.log > chksum_buf.json

# Mark file as unsorted to please makeself
sed -i '2 i \ "unsorted": true,' chksum_buf.json
cat chksum_buf.json | ./makeself show_rpc.html > chksum_buf.html
```

Remember that the `diskdir/c*` files are uncached; starvation patterns are different from those in the text (as different hardware has been used.) Don't forget to study, understand, and explain your results (not the textbooks.)

Exercise 7.5 is easier than it seems as the Direct I/O modifications are available as an option to the server_disk executable. (there is no mystery3.cc)

```
# l51-piAAA
# Purge page cache
echo 1 | sudo tee /proc/sys/vm/drop_caches
# Bring aaaa and dddd back to RAM
cat diskdir/a* diskdir/d* > /dev/null

./server_disk diskdir -direct

# pi-CCC
./client4 l51-piAAA 12345 -k 1000 -seed1 chksum -key "aaaa" +

# pi-BBB
./client4 l51-piAAA 12347 -k 200 -seed1 chksum -key "cccc" + & \
./client4 l51-piAAA 12348 -k 200 -seed1 chksum -key "dddd" +
```

Copy log files to the same machine, and then

```
  # Must be aligned separately because they are from different machines
./timealign client4_20230212_2019*piCCC*.log
./timealign client4_20230212_2019*piBBB*.log

./dumplogfile4 "Chksum x3, direct I/O" client4_20230212_2019*align.log > chksum_direct.js

# Mark file as unsorted to please makeself
sed -i '2 i \ "unsorted": true,' chksum_direct.json
cat chksum_direct.json | ./makeself show_rpc.html > chksum_direct.html
```

Despite pre-filling the cache, the direct I/O methods will bypass the operating system cache.

Question

what effects do you see? and how does this compare with the results of the previous experiment?

3 Something extra

When you are finished and exploring the log files, in particular when you explore the server-side logfiles; the additional argument `-all` for `dumplogfile4` is required.

4 Saving Your Experiments

Make sure to back up your experiments, including (but not limited to) Jupyter notebooks, dump files and scripts. Remember that multiple teams may use the same test machines, so be careful when handling data.
All the measurements are saved under your crsid folder, so backing up the entire folder is a good idea. To copy a remote directory onto your local machine:
`sftp l51@<hostname>.nf.cl.cam.ac.uk` and `get -r <directory>`.
There are also other ways to copy a remote directory, you are welcome to use those as well. You may wish to compress results files in order to save space.
Exporting a Notebook as `.tex` will save graphs as separate files, which you can then include in your lab report.

Please do not push any changes, data or results directly to L50 repository. You can fork the repository to your own user and push changes there. If you would like to suggest a correction or an enhancement to a notebook or a script, please use pull-requests.

5 Understanding Your Measurements

A single lab report will be required for the first three labs. Instructions for the lab report will be provided with Lab 4.
The following items are intended to help you understand your results, and may 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.

You should always look for odd or surprising results, and try to explain them. Note that sometimes exceptional results indicate a problem in your setup or scripts.