Distributed Systems
Autumn Semester 2020
Dr. Valerio Schiavoni, Prof. Peter Kropf
Assistant: Isabelly Rocha

Project 2 Part II — Benchmarking in-memory key-value stores static and dynamic network conditions

Part II - Due: December 1st, 23:59

1 Requirements and prerequisites

This project can be executed and completed on the dedicated server machine that each of you was provided on our cluster. *If you use your own laptop to complete Part I, you can use it as well to complete Part II.* If you have not received instructions to access it, please contact immediately the instructors and the assistant. This work is individual, *i.e.*, no pairs, no groups. There won't be any deadline extensions.
You must have successfully and correctly completed all the tasks (from 1 to 7) described in Part I before working on the tasks described in this document. Also, you must have read and understood the THUNDERSTORM paper [1].

Task 8 - RedisBenchmark

Task 8.1

With the topology defined and deployed in Task 7, execute the `redis-benchmark`[*], a tool to execute a large set of benchmarks for Redis. Note that it was installed at the same time as the `redis-cli` (see Part I, Listing 12, Line 2). By default, `redis-benchmark` executes a large number of benchmarks. For testing purposes, it is possible to execute only a subset of them:

```
$ redis-benchmark -t set,lpush -n 100000 -q -h ...
SET: 74239.05 requests per second
LPUSH: 79239.30 requests per second
```

In this example, we only execute `SET` and `LPUSH`, in quiet mode (see the `-q` switch).

Collect the results of the benchmark execution in a file (`redis-benchmark -h` to see the various options). Once the file is complete, you must copy it back into the physical host (*e.g.*, your dedicated server). You can do this using the `docker cp` command. Read the documentation at:
`https://docs.docker.com/engine/reference/commandline/cp/`.

[*] `https://redis.io/topics/benchmarks`

Task 8.2
Next, modify the underlying network topology. Specifically, increase the bandwidth available in the client-to-switch and switch-to-server links (up to 200 Mb/s), as shown in Figure 1.

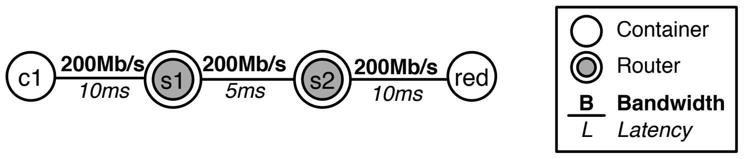

Figure 1: Topology to use in Task 8.2

Remember that for every change in the topology, you must generate stop the current experiments, generate new deployment descriptors and restart the experiment. Compare the results between the two executions. What do you observe? Comparare the performance of at least 4 different commands in a plot or table that clearly show how the change in the underlyng network is affecting the results of the benchmark.

Task 8.3
Modify the underlying network topology once again. In this new variant, add a new client node `client2` directly attached to the `s2`, as shown in Figure 2.

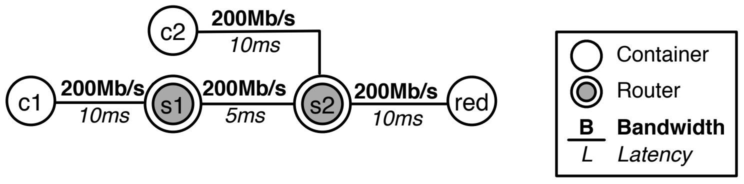

Figure 2: Topology to use in Task 8.3

Execute the `redis-benchmark` suite of benchmarks *at the same time* from the two clients, and demonstrate, by means of experimental evidence, that their execution leads to interferences between the results gathered by the two clients.

Task 8.4
What is the effect of a link drop rate on the applications deployed over a given topology? In this task, we will study this behaviour. In particular, using the same topology of Task 8.3, modify the drop rate of link `c2-s2`. A link's given packet drop rate can be specified with the following XML syntax:

Listing 1: Syntax for link packet drop rate

```xml
<link origin="client1" dest="s1" latency="5"
    upload="10Mbps" download="10Mbps"
    drop="0.001"
    network="test_overlay"/>
```

Evaluate the effect of different packet drop rates on the results for the `set` benchmark (see Task 8.1). Report the results in a plot. On the x-axis the different packet drop rates tested, on the y-axis the achieved requests per seconds issued by `c2`.

Expected Task 8.1/8.2/8.3/8.4] results.

- the .xml file describing this topology
- the generated .yaml file
- the output froom the `docker stack ...` command used to deploy the topology.
- plots, commands executed to run the benchmarks, and short textual descriptions of your results.

Task 9 - Dynamic Topologies

So far, the underlying topologies deployed by your experiments using THUNDERSTORM are static, *e.g.*, once deployed there are no changes. One of the most novel features of THUNDERSTORM is the ability to change the topology along several dimensions, *e.g.*, adding or removing nodes, changing the properties of the links, and so on. In this task, we will try to observe how the network latency changes affect the perceived throughput of Redis. Take one more look at the example given in Task 1, Listing 1 (lines 27-32). Those `<dynamic>` events are actually changing the deployed topology. It is possible to change the latency of a link between two nodes `a` and `b` to 10ms after 20 seconds after the experiment has started using the following syntax:

Listing 2: Dynamically changing the link latency (similar for changes to bandiwidth).

```
<schedule origin="a" dest="b" time="20.0" latency="10"/>
```

In this task, you are asked to deploy a simple point-to-point topology with one client and one Redis server, a single link with an initial 100 Mb/s bandwidth and 5ms latency. We assume the experiment starts at time `t0`. Then, the following events must follow:

- at `t0+10s`, the link latency drops to 50ms
- at `t0+30s`, the link bandwidth drops to 50 Mb/s
- at `t0+60s`, the link bandwidth increases back to 150 Mbs
- at `t0+90s`, the link latency improves and decreases to 10ms
- at `t0+120s`, the experiment ends.

While the experiment runs, a client must continuously executes the `redis-benchmark` with the `set` test. Produce a plot that allows us to observe how the throughput changes in time, reflecting changes to the underlying topology.

Expected Task 9 results.

- the .xml file describing this topology
- the generated .yaml file
- the output froom the `docker stack ...` command used to deploy the topology.
- plots, commands executed to run the benchmarks, and short textual descriptions of your results.

Submission

The following documents must be uploaded on ILIAS:

1. The work is individual. No exceptions.
2. A pdf version of your report. Each student has to implement his/her own version of the report. This report should contain all your interesting results (in table and/or plot formats), as well as a pertinent analysis and evaluation.
3. A zip file containing all the result files (text format) produced by the program that you used to draw plots of your report.

Please respect the following point:

- Make sure that your compressed document is of reasonable size (no need for images of high-quality) and upload it on ILIAS by strictly following the naming conventions.

Note that the fact that you strictly followed all the instructions concerning the way to provide your report and results (formatting, type of file, deadline,...) will be taken into account to establish your mark.

References

[1] Luca Liechti, Paulo Gouveia, João Neves, Peter Kropf, Miguel Matos, Valerio Schiavoni. *THUNDERSTORM: a tool to evaluate dynamic network topologies on distributed systems*. Proceedings of IEEE SRDS 2019 (38th IEEE International Symposium on Reliable Distributed Systems). Lyon, France, October 2019.