Advanced Systems Lab (Fall'16) – Second Milestone

Name: *Taivo Pungas*
Legi number: *15-928-336*

Grading

Section	**Points**
1	
2	
3	
Total	

Notes on writing the report **(remove this page for submission)**

Before starting to work on this milestone please remember:

- The prerequisite to a successful completion to this milestone is to have a stable system and also the necessary logging functionalities in place.
- Depending on the workload and goal of the experiment, you might need to change the sampling rate from the default level. Make sure to indicate when doing so.
- The choice of experiment length and repetitions is up to you to decide, please make sure that you do not include warm-up and cool-down phases in the measurements. There are many experiments to run in this milestone, try to make a tradeoff.
- We recommend that you have scripts in place to deploy and run experiments.
- All experiments have to be executed on the Microsoft Azure cloud.
- When plotting graphs include errors or measures of accuracy whenever possible.
- Keep the report compact and concise! The total length should not exceed 20 pages. Log listings are not counted in this length, but all text, figures and tables are. If you have many logs, compress them by experiment and reference the archive instead of the independent files.

In this milestone we expect to see the different experiments you ran to exercise the system, and with each experiment we expect a clear description of the system configuration used, the hypothesis on behavior and the explanation of the behavior observed (in terms of the different design decisions taken beforehand) – *missing either of these for an experiment might make you lose all points for that given experiment!*

Keep in mind that for a good explanation of the results of an experiment you might have to use one or more methods of data analysis presented in the lecture and in the book. You might have to combine measurements taken in the middleware with the ones at the clients to be able to provide a full picture.

Please feel free to structure the three sections of this report as it makes most sense for your experiments and explanations, but please respect the goal of each section. Also, similarly to the first milestone, include tables and descriptions about your experimental setup before each set of experiments.

Modifications to the middleware

In the last milestone submission, my middleware implemented all functionality as necessary. However, the resource usage was extremely wasteful: each read thread took up nearly 100% of the resources allocated to them and never went to a sleeping state. This caused more than 10-fold drops in performance when going from $T = 1$ to $T = 4$ (for $S = 5$), and would have made the maximum throughput experiment useless. The changes can be seen on GitLab.

To verify that the system is still stable, I re-ran the trace experiment. The throughput and response time are shown in Figures 1 and 2, and are confirmed to be stable (and throughput is roughly 30% higher). For explanations of the figures, see Milestone 1 report.

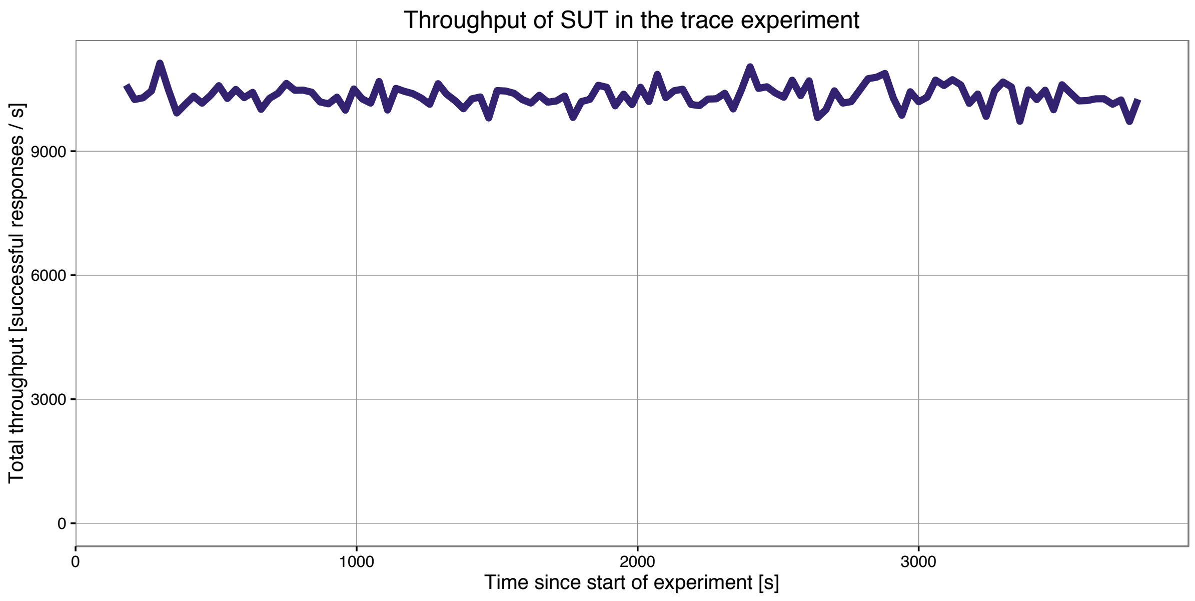

Figure 1: Throughput trace of the middleware.

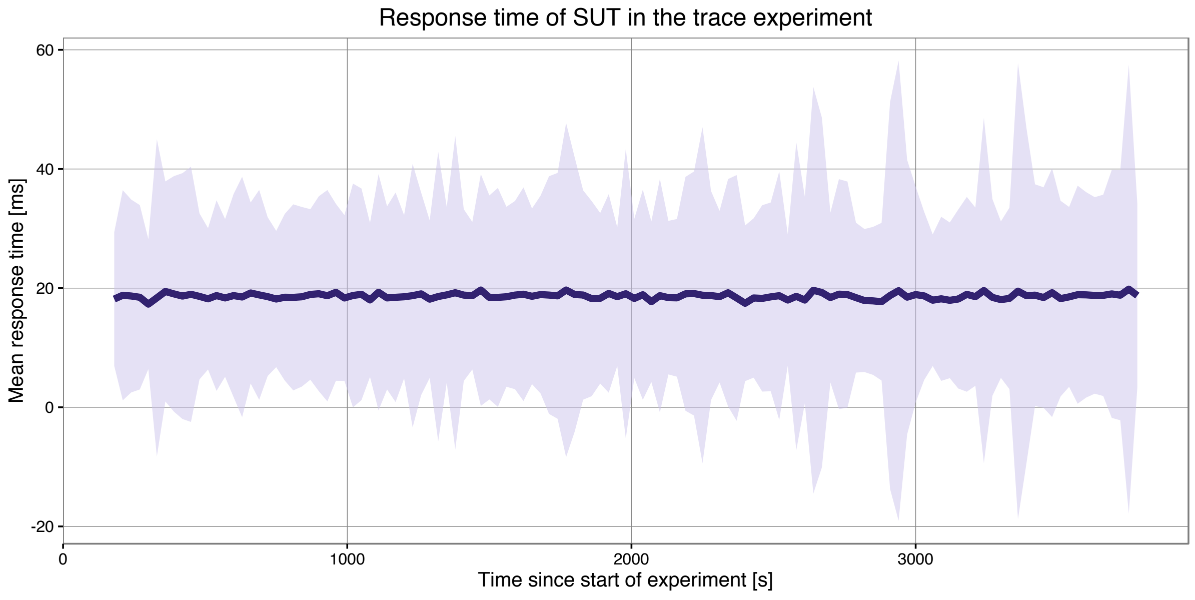

Figure 2: Response time trace of the middleware as measured by memaslap.

1 Maximum Throughput

Find the highest throughput of your system for 5 servers with no replication and a read-only workload configuration. What is the minimum number of threads and clients (rounded to multiple of 10) that together achieve this throughput? Explain why the system reaches its maximum throughput at these points and show how the performance changes around these configurations.

Provide a detailed breakdown of the time spent in the middleware for each operation type.

1.1 Experimental question

The system under test (SUT), in this section and hereafter, is the middleware together with the memcached servers running on virtual machines in the Azure cloud.

In this section, I will run experiments to find out a) the maximum throughput of the SUT, b) the number of read threads (T) in the middleware that achieves this c) the number of virtual clients (C) that achieves this.

To this end, I will measure throughput (the number of requests the SUT successfully responds to, per unit of time) as a function of T and C. I will find the maximum sustained throughput of the SUT, i.e. the throughput with which the response time does not increase rapidly with additional clients.

1.2 Hypothesis

I predict the following.

Number of threads Throughput will be maximised at a very low value of $T = 1$, after which the throughput will start to decrease rapidly.

The reason for that lies in my implementation: since each thread runs through an infinite while-loop even if there are no requests to process, the threads almost never wait or sleep and use 100% of the resources allocated to them by the scheduler. For this reason, once T is high enough that not all threads can run concurrently, threads need to wait to be allocated time, and this waiting decreases throughput significantly. The total number of threads used by the system is given as $S * (T + 1) + 1$; for $S = 5$ and $T = 1$, the system uses 11 threads. This is more than what's available in a Basic A4 machine, so increasing T will only decrease throughput.

Note that the middleware can be easily changed to change this non-optimal behaviour by making the threads sleep for a short period if there is nothing to do.

Number of clients Throughput will be maximised at roughly 110 virtual clients per memcached server, so 550 virtual clients in total. This is based on the fact that in the Milestone 1 baseline experiment, the throughput of a single memcached server without middleware saturated at around 110 virtual clients. It is likely to be an overestimate since the SUT here has an additional part (the middleware), but 550 clients is a reasonable upper bound.

Throughput Since in the trace experiment, the throughput was roughly 7500 requests per second, we have a lower bound for the expected throughput. Naively assuming that the throughput of GET requests scales linearly with the number of servers S would yield an expected throughput of $\frac{5}{3} \cdot 7500 = 12500$ requests per second. However, this does not take into account our different experimental setup compared to the trace experiment (the number of writes). Thus I expect the maximum throughput to be definitely more than 7500 and near 12500 requests per second.

- Draw graphs with expected results

- Even try to predict variance and statistical properties
- Make bullet points with explanations
- Use modeling to make hypothesis
- Formulate a number of questions on what you expect to see, develop a hypothesis around the behavior that is expected, explain the hypothesis, and write it all down

1.3 Experiments

Number of servers	5
Number of client machines	3
Virtual clients / machine	TODO:
Workload	Key 16B, Value 128B, Writes 0% TODO:
Middleware: replication factor	1
Middleware: read threads	TODO:
Runtime x repetitions	60s x 5 TODO:
Log files	TODO:

1.4 Results

Reporting experiment results. Comparison of hypothesis and experiment results.

2 Effect of Replication

Explore how the behavior of your system changes for a 5%-write workload with S=3,5 and 7 server backends and the following three replication factors:

- Write to 1 (no replication)
- Write to $\lceil \frac{S}{2} \rceil$ (half)
- Write to all

Answer at least the following questions: Are `get` and `set` requests impacted the same way by different setups? If yes/no, why? Which operations become more expensive inside the middleware as the configuration changes? How does the scalability of your system compare to that of an ideal implementation? Provide the graphs and tables necessary to support your claims.

2.1 Experimental question

- Define the system(s) under test
- Define what to measure and understand why

2.2 Hypothesis

- Draw graphs with expected results
- Even try to predict variance and statistical properties
- Make bullet points with explanations
- Use modeling to make hypothesis
- Formulate a number of questions on what you expect to see, develop a hypothesis around the behavior that is expected, explain the hypothesis, and write it all down

2.3 Experiments

Number of servers	$\in \{3, 5, 7\}$
Number of client machines	3
Virtual clients / machine	TODO:
Workload	Key 16B, Value 128B, Writes 5% TODO:
Middleware: replication factor	$\in \{1, ceil(S/2), S\}$
Middleware: read threads	TODO:
Runtime x repetitions	60s x 5 TODO:
Log files	TODO:

2.4 Results

Reporting experiment results. Comparison of hypothesis and experiment results.

3 Effect of Writes

In this section, you should study the changes in throughput and response time of your system as the percentage of write operations increases. Use a combination of 3 to 7 servers and vary the number of writes between 1% and 10% (e.g. 1%, 5% and 10%). The experiments need to be carried out for the replication factors R=1 and R=all.

For what number of servers do you see the biggest impact (relative to base case) on performance? Investigate the main reason for the reduced performance and provide a detailed explanation of the behavior of the system. Provide the graphs and tables necessary to support your claims.

3.1 Experimental question

- Define the system(s) under test
- Define what to measure and understand why

3.2 Hypothesis

- Draw graphs with expected results
- Even try to predict variance and statistical properties
- Make bullet points with explanations
- Use modeling to make hypothesis
- Formulate a number of questions on what you expect to see, develop a hypothesis around the behavior that is expected, explain the hypothesis, and write it all down

3.3 Experiments

Number of servers	$\in \{3, 4, 5, 6, 7\}$
Number of client machines	3
Virtual clients / machine	TODO:
Workload	Key 16B, Value 128B, Writes $\in \{1\%, 5\%, 10\%\}$ TODO:
Middleware: replication factor	$\in \{1, S\}$
Middleware: read threads	TODO:
Runtime x repetitions	60s x 5 TODO:
Log files	TODO:

3.4 Results

Reporting experiment results. Comparison of hypothesis and experiment results.

Logfile listing

Short name	Location
baseline-m*-c*-r*	gitlab.inf.ethz.ch/.../results/baseline/baseline_memaslap*_conc*_rep*.out
trace-ms4	gitlab.inf.ethz.ch/.../results/trace_rep3/memaslap4.out