Advanced Systems Lab (Fall'16) – First Milestone

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

Grading

Section	Points
1.1	
1.2	
1.3	
1.4	
2.1	
2.2	
3.1	
3.2	
3.3	
Total	

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

The report for first milestone not need to be extensive but it must be concise, complete, and correct. Conciseness is important in terms of content and explanations, focusing on what has been done and explanations of the results. A long report is not necessarily a better report, especially if there are aspects of the design or the experiments that remain unexplained. Completeness implies that the report should give a comprehensive idea of what has been done by mentioning all key aspects of the design, experiments, and analysis. Aspects of the system, be it of its design or of its behavior, that remain unexplained detract from the credibility of the report. Correctness is expected in terms of the explanations being logical and correlate with the numbers in the experiments and the design.

Remember that this is a report about the system you have designed and built, about the experiments you have performed, and about how you interpret the results of the experiments and map them to your design and implementation. Please do not contact us seeking confirmation and assurances about, e.g., whether the report is sufficient, your interpretation of the data, validation of concrete aspects of your design, or whether you have done enough experiments. Making those decisions is your job and part of what the course will evaluate.

The report will be graded together with the code and data submitted. **The maximum number of points is 200 for each of the milestones and you will need at least 100 to pass. Keep in mind that to pass the project you need to collect at least 400 points from the three milestones.** You might be called for a meeting in person to clarify aspects of the report or the system and to make a short presentation of the work done. By submitting the report, the code, and the data, you confirm that you have done the work on your own, the code has been developed by yourself, the data submitted comes from experiments your have done, you have written the report on your own, and you have not copied neither code nor text nor data from other sources.

A passing grade for the milestone requires at the very minimum:

- Conforming to this template
- A working system
- Consistent experimental results of the entire system
- Internal measurements of the middleware
- Solid and credible explanations of the design, experimental results and behavior of the implemented system

Formatting guidelines

We expect you to use this template for the report, but in case you want to use Word or an other text processor, keep in mind the following:

- We expect you to submit **a single PDF that has the same section structure as this template, and answers all points we outline here**. If you use this file, you should remove this page with notes, and the short description provided by us at the beginning of sections.
- Keep the same cover page as on this document and **fill out your name and legi number**. Leave the grading table empty.
- The main text should be in **single-column format with 11pt font on A4 paper**. In case you don't start with one of the files provided by us, **for margins use 2.54 cm (1 inch) on all sides**.

1 System Description

1.1 Overall Architecture

Length: at most 1 page

Explain how the abstract architecture we provided for you has been implemented in terms of classes and shortly outline the main design decisions. Mark on the figure where are the points that you instrumented the architecture (see Section 2.3 of the Project Description) and give the different timestamps a name that you **will use throughout the three milestones** whenever referencing measurements (e.g., $T_{requestreceived}$, $T_{responsesent}$).

Reference throughout the report all relevant java source files, result files, etc. by providing the gitlab link in a footnote, for instance[1]. An exception to this rule is the referencing of log files belonging to experiments. These should be referenced by an ID, or short name, and there has to be a table at the end of the report mapping these to files in the git repository.

1.2 Load Balancing and Hashing

Length: at most 1 page

Explain what hash function you use for load balancing and how you implement the selection of servers. Give a short reasoning on why the chosen scheme should uniformly distribute load (assuming no skew on the client side).

1.3 Write Operations and Replication

Length: at most 1 page

Provide a short description of how the writes are handled in the middleware. Explain how the replicated case differs from the simple "write one" scenario.

Give an estimate of the latencies the writing operation will incur, and generalize it to the replicated case. What do you expect will limit the rate at which writes can be carried out in the system (if anything)?

1.4 Read Operations and Thread Pool

Length: at most 1 page

How are reads handled in the system? How does the middleware make sure that the queue between the "main receiving" thread and the read handlers is not accessed in unsafe concurrent manner? Explain what is the relation between threads in the thread pool and connections to servers.

2 Memcached Baselines

This section will report experimental results. All such parts will start with a short description of the experimental setup. The log files should be identified by a short name, or number, which will be explicitly listed at the end of the document (see Logfile Listing at the end). **If this table is missing or the logfiles listed can't be found in your repository the experiment could be considered invalid, and no points will be awarded!**

Number of servers	1
Number of client machines	1 to 2
Virtual clients / machine	1 to 64
Middleware	Not present
Runtime x repetitions	30s x 5
Log files	microbench1, microbench2, ...

[1] `https://gitlab.inf.ethz.ch/zistvan/asl-fall16-project/blob/master/src/ch/ethz/SomeClass.java`

For baseline measurement of memcached provide **two** graphs (Section 2.1 and 2.2), one with aggregated throughput and one with average response time and standard deviation as a function of number of virtual clients. Increase these in steps from 1 to 128. Give a short explanation of memcache's behavior and find the number of virtual clients that saturate the server.

2.1 Throughput

See previous explanation.

2.2 Response time

See previous explanation.

3 Stability Trace

In this section you will have to show that the middleware is functional and it can handle a long-running workload without crashing or degrading in performance. For this you will run it without replication for one hour connected to three memcache instances and three load generator machines. The experiment has to be repeated for the case when writes are replicated to 1+2 nodes, that is to all three of them.

Number of servers	3
Number of client machines	3
Virtual clients / machine	64 (explain if chosen otherwise)
Middleware	No replication, Replicate to all
Runtime x repetitions	1h x 1
Log files	trace1, ...

You will have to provide two graphs each depicting two separate measurements of the middleware (with and without replication). The x-axis is time and the y-axis is either throughput or response time. Include standard deviation whenever applicable.

3.1 Throughput

See previous explanation.

3.2 Response time

See previous explanation.

3.3 Overhead of middleware

Compare the performance you expect based on the baselines and the one you observe in the trace and quantify the overheads introduced by the middleware (if any), Look at both response time and achievable throughput when making the comparison. Provide an overview of the overheads in a table form.

Logfile listing

Short name	Location
microbench1	`https://gitlab.inf.ethz.ch/.../baseline/logfile.log`
microbench2	`https://gitlab.inf.ethz.ch/.../baseline/logfile2.log`
trace1	`https://gitlab.inf.ethz.ch/.../baseline/logfile.log`
...	...