Advanced Systems Lab (Fall'16) – Third Milestone

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

Grading

Section	**Points**
1	
2	
3	
4	
5	
Total	

Contents

1 System as One Unit

1.1 Data

The experimental data used in this section comes from the updated trace experiment, found in `results/trace_rep3` (short names `trace_ms*`, `trace_mw` and `trace_req` in Milestone 1). For details, see Milestone 2, Appendix A.

The first 2 minutes and last 2 minutes were dropped as warm-up and cool-down time similarly to previous milestones.

1.2 Model

In this section I create an M/M/1 model of the system. This means the following definitions and assumptions:

- The queues are defined as having infinite buffer capacity.
- The population size is infinite.
- The service discipline is FCFS.
- Interarrival times and the service times are exponentially distributed.
- We treat the SUT as a single server and as a black box.
- Arrivals are individual, so we have a birth-death process.

Parameter estimation Using the available experimental data, it is not possible to directly calculate the mean arrival rate λ and mean service rate μ so we need to estimate them somehow. I estimated both using throughput of the system: I take λ to be the *mean* throughput over 1-second windows, and μ to be the the *maximum* throughput in any 1-second window, calculated from middleware logs. I chose a 1-second window because a too small window is highly susceptible to noise whereas a too large window size drowns out useful information.

Problems The assumptions above obviously do not hold for our actual system. Especially strong is the assumption of a single server; since we actually have multiple servers, this model is likely to predict the behaviour of the system very poorly. A second problem arises from my very indirect method of estimating parameters for the model (and an arbitrary choice of time window) which introduces inaccuracies.

1.3 Comparison of model and experiments

TODO: this whole subsection

	variable	predicted	actual
1	mean_num_jobs_in_system	3.95	1.57
2	std_num_jobs_in_system	19.55	0.57
3	mean_num_jobs_in_queue	3.15	1.43
4	utilisation	0.20	0.93
5	mean_response_time	0.38	14.55
6	response_time_q50	0.27	12.00
7	response_time_q95	1.15	30.00

Table 1: Comparison of experimental results and predictions of the M/M/1 model.

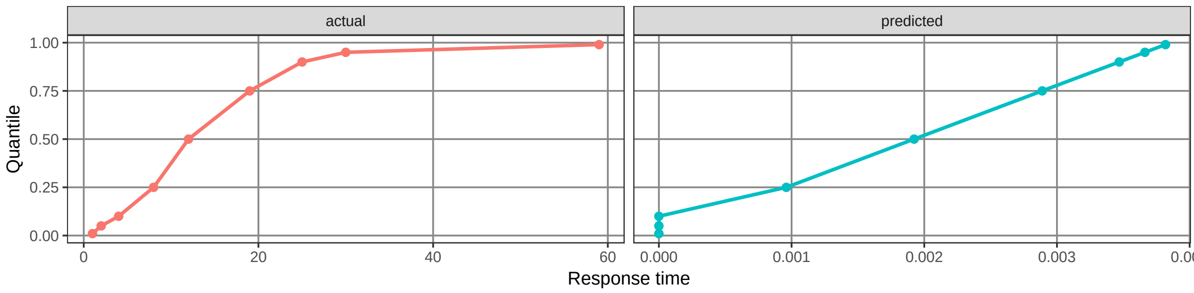


Figure 1: Quantiles of the response time distribution: experimental results and predictions of the M/M/1 model. Note the extreme difference in the response time scale.

2 Analysis of System Based on Scalability Data

2.1 Data

The experimental data used in this section comes from Milestone 2 Section 1 and can be found in `results/throughput`.

2.2 Model

The assumptions and definitions of the M/M/m model are the same as for the M/M/1 model laid out in Section 1.2 with the following modifications:

- We treat the SUT as a collection of m servers.
- All jobs waiting for service are held in one queue.
- If any server is idle, an arriving job is serviced immediately.
- If all servers are busy, an arriving job is added to the queue.

Parameters TODO: describe how I found the parameters

Problems TODO:

1. I actually have m queues (one for each server), not a single queue; each request is assigned to a server when LoadBalancer receives it.
2. I map requests to servers uniformly. M/M/m assumes that each server takes a request when it finishes with the previous one, but that is not true in my case – I take earlier

2.3 Comparison of model and experiments

TODO:

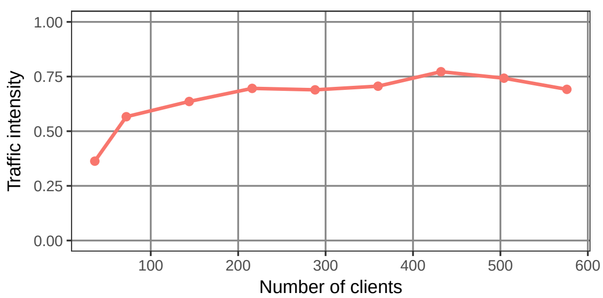


Figure 2: TODO:

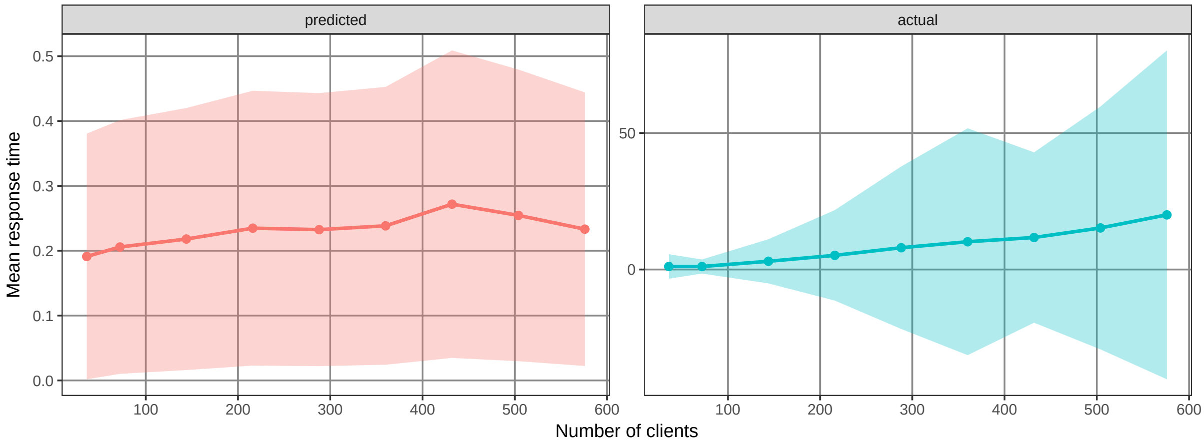


Figure 3: TODO: note difference in scale

Figure 4: TODO:

	variable	clients	predicted	actual
1	response_time_mean	36.00	0.19	1.11
2	response_time_mean	72.00	0.21	1.10
3	response_time_mean	144.00	0.22	3.01
4	response_time_mean	216.00	0.23	5.19
5	response_time_mean	288.00	0.23	7.98
6	response_time_mean	360.00	0.24	10.18
7	response_time_mean	432.00	0.27	11.72
8	response_time_mean	504.00	0.25	15.23
9	response_time_mean	576.00	0.23	20.02
10	response_time_std	36.00	0.19	4.51
11	response_time_std	72.00	0.20	2.58
12	response_time_std	144.00	0.20	8.09
13	response_time_std	216.00	0.21	16.58
14	response_time_std	288.00	0.21	29.77
15	response_time_std	360.00	0.21	41.55
16	response_time_std	432.00	0.24	31.21
17	response_time_std	504.00	0.22	44.49
18	response_time_std	576.00	0.21	60.23

Table 2: Comparison of experimental results and predictions of the M/M/m model.

3 System as Network of Queues

3.1 Guidelines

Length: 1-3 pages

Based on the outcome of the different modeling efforts from the previous sections, build a comprehensive network of queues model for the whole system. Compare it with experimental data and use the methods discussed in the lecture and the book to provide an in-depth analysis of the behavior. This includes the identification and analysis of bottlenecks in your system. Make sure to follow the model-related guidelines described in the Notes!

4 Factorial Experiment

4.1 Guidelines

Length: 1-3 pages

Design a 2^k factorial experiment and follow the best practices outlined in the book and in the lecture to analyze the results. You are free to choose the parameters for the experiment and in case you have already collected data in the second milestone that can be used as source for this experiment, you can reuse it. Otherwise, in case you need to run new experiments anyway, we recommend exploring the impact of request size on the middleware together with an other parameter.

5 Interactive Law Verification

5.1 Data

The experimental data used in this section comes from Milestone 2, Section 2 (Effect of Replication) and can be found in `results/replication` (short name `replication-S*-R*-r*` in Milestone 2).

The first 2 minutes and last 2 minutes were **not** dropped because the Interactive Response Time Law (IRTL) should hold also in warm-up and cool-down periods. Repetitions at the same configuration were considered as separate experiments.

5.2 Model

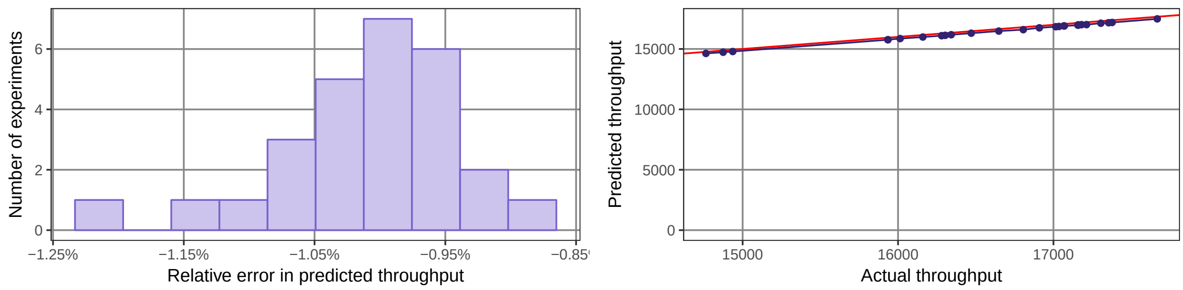
We are assuming a closed system, i.e. clients wait for a response from the server before sending another request. Under this assumption, the IRTL should hold:

$$R = \frac{N}{X} - Z$$

where R is mean response time, Z is waiting time in the client, N is the number of clients and X is throughput.

5.3 Results

Using IRTL, we can verify the validity of experiments by calculating the predicted throughput $X_{predicted}$ (given the number of clients C and mean response time R) and comparing it with actual throughput X_{actual}. This is precisely what I did for all experiments of Milestone 2, Section 2. $C = 180$ in all experiments, and both R and X_{actual} are aggregated results reported by the three memaslap instances generating load in that experiment.



(a) Histogram of the relative error of throughput predicted using IRTL, counting the number of experiments in a given error range. Note the horizontal scale does not include 0.

(b) Throughput predicted using IRTL, as a function of actual throughput calculated from experimental data. The red line shows perfect predictions (the $x = y$ line). Note the horizontal scale does not include 0.

Figure 5: Evaluation of the validity of Milestone 2 Section 2 experiments

If we assume the wait time Z to be 0, we get a mean relative prediction error of -1.01%, defined as $\frac{X_{predicted} - X_{actual}}{X_{actual}}$. The distribution of these errors is shown in Figure 5a; the distribution looks reasonably symmetric. Figure 5b plots $X_{predicted}$ against X_{actual} and shows again that the predicted throughput is very close to actual throughput, but consistently smaller in all regions of the graph.

If we assume a nonzero Z and estimate it from the experiments, we get a mean estimated wait time of -0.111ms. Clearly this is impossible: wait time must be non-negative. It could be that memaslap starts the clock for a new request before stopping the clock for the previous request – which would violate the closed system assumption – but this is very unlikely.

TODO: why the consistent -1% error? because total cycle time is higher than it should for given throughput – but why? since measuring is done by memaslap, probably the problem lies on that side

Appendix A: Template appendix