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.

TODO: mention that I do all calculations for the system WITHOUT memaslap–middleware network times!

TODO: also that throughput statistics are not affected by the network (even though when calculating ser

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

Explain the characteristics and behavior of the model built, and compare it with the experimental data (collected both outside and inside the middleware). Map the similarities and differences to aspects of the design or the experiments.

Table 1 shows a comparison of the predictions of the M/M/1 model with actual results from the trace experiment.

TODO: this whole subsection

TODO: mention that IRTL doesn't hold for the model

TODO: mention that number of jobs is calculated using Little's law

	metric	predicted	actual
1	response_time_mean	0.38	14.55
2	response_time_std	0.38	17.62
3	response_time_quantile50	0.27	12.00
4	response_time_quantile95	1.15	30.00
5	waiting_time_mean	0.31	13.22
6	waiting_time_std	0.38	17.06
7	utilisation	0.80	
8	num_jobs_in_system_mean	3.95	149.79
9	num_jobs_in_system_std	4.42	
10	num_jobs_in_queue_mean	3.15	136.11
11	num_jobs_in_queue_std	4.26	
12	num_jobs_served_in_busy_period_mean	4.95	
13	num_jobs_served_in_busy_period_std	13.19	
14	busy_period_duration_mean	0.38	
15	busy_period_duration_std	0.76	

Table 1: Comparison of experimental results ('actual') and predictions of the M/M/1 model ('predicted') for different metrics. Where the 'actual' column is empty, experimental data was not detailed enough to calculate the desired metric. All time units are milliseconds.

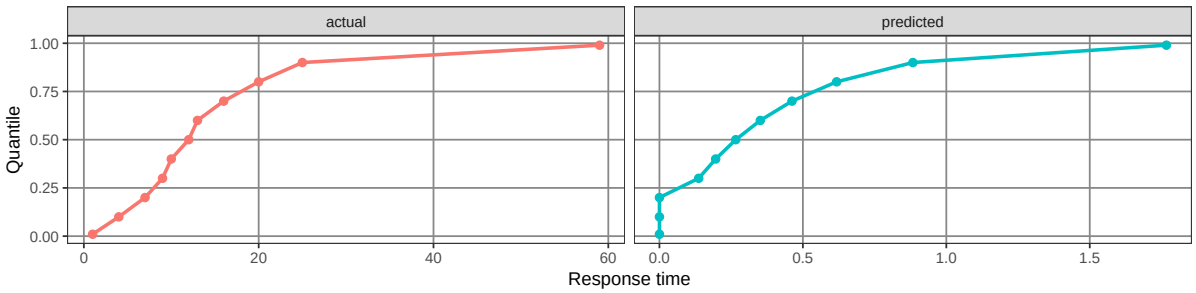


Figure 1: Quantiles of the response time distribution: experimental results and predictions of the M/M/1 model. Note the extreme difference in the response time scale.

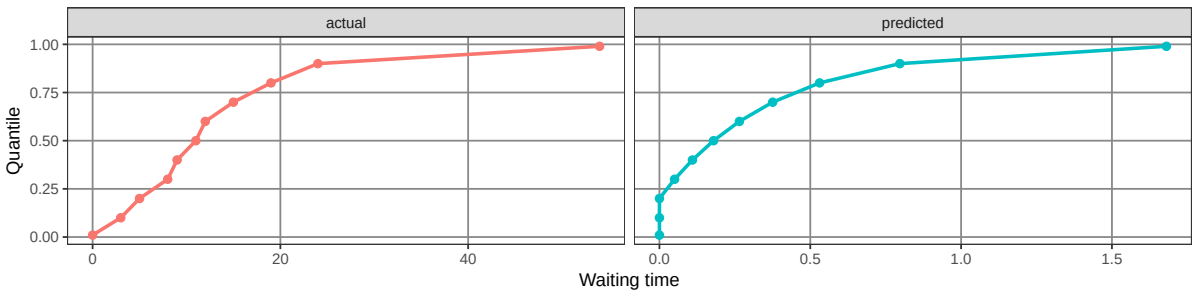


Figure 2: Quantiles of the waiting time distribution: experimental results and predictions of the M/M/1 model. Note the extreme difference in the queue time scale.

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:

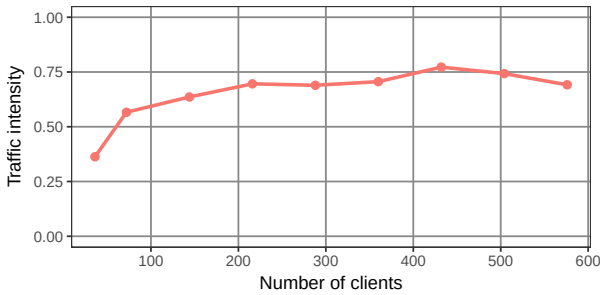


Figure 3: TODO:

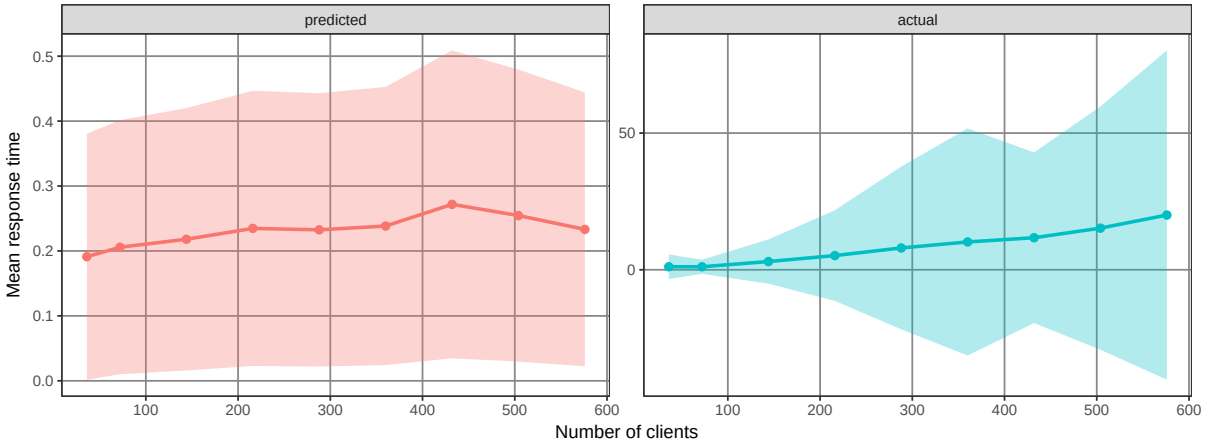


Figure 4: TODO: note difference in scale

	variable	clients	predicted	actual
1	response_time_mean	36	0.19	1.11
2	response_time_mean	72	0.21	1.10
3	response_time_mean	144	0.22	3.01
4	response_time_mean	216	0.23	5.19
5	response_time_mean	288	0.23	7.98
6	response_time_mean	360	0.24	10.18
7	response_time_mean	432	0.27	11.72
8	response_time_mean	504	0.25	15.23
9	response_time_mean	576	0.23	20.02
10	response_time_std	36	0.19	4.51
11	response_time_std	72	0.20	2.58
12	response_time_std	144	0.20	8.09
13	response_time_std	216	0.21	16.58
14	response_time_std	288	0.21	29.77
15	response_time_std	360	0.21	41.55
16	response_time_std	432	0.24	31.21
17	response_time_std	504	0.22	44.49
18	response_time_std	576	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 Data

TODO:

3.2 Model

TODO:

We have a closed queueing network: the total number of jobs in the system is constant. This number is equal to the concurrency parameter C we give to memaslap since each concurrent job in memaslap sends one request and waits for a response before sending the next one.

MVA was performed using the Octave package queueing.

3.3 Comparison of model and experiments

TODO:

performance of writeworker *depends on queue length* because if there are no elements in queue then we check memcached responses every 1ms, whereas if there are elements in queue then we check always after dequeueing an element. This dependence violates TODO: what assumption?

if there is nothing in the queue and no responses from memcached then we will wait for 2ms!

4 Factorial Experiment

4.1 Experimental question and experiment design

The goal of this section is to find out the factors that influence throughput of the system. In particular, I will investigate the effect of $k = 3$ factors – S (number of servers), R (replication level) and W (percentage of writes in workload) – on total throughput of the system. Everything else will be kept fixed: the number of threads $T = 32$ and the number of clients $C = 180$.

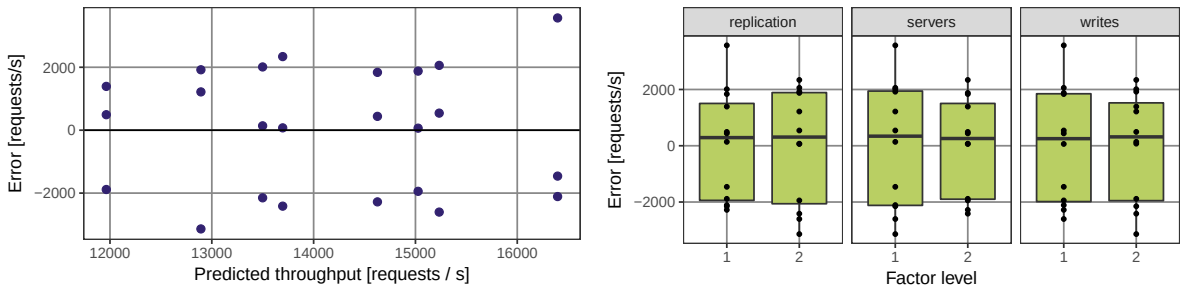
For each factor in $\{S, R, W\}$ we need to pick two levels. Since I expect throughput to monotonically increase with S, decrease with R, and decrease with W, we can use the minimum and maximum level for each factor in our 2^k experiment: $S \in \{3, 7\}$, $R \in \{1, S\}$, and $W \in \{1, 10\}$.

4.2 Data

The experimental data used in this section comes from Milestone 2 Section 3 and can be found in `results/writes` (short name `writes-S*-R*-W*-r*` in Milestone 2). For each combination of S, R, and W we have $r = 3$ repetitions. This means data from $2^k \cdot r = 24$ distinct runs are used in this section.

The data table is available in Appendix A.

4.3 Results



(a) Error of the 2^k model as a function of predicted throughput. Each point is one repetition of a configuration. Note the throughput axis does not include zero.

(b) Boxplot of the error distribution for all levels of all factors. The box shows the inter-quartile range (between 25th and 75th percentiles, IQR), with the median shown as a horizontal line. The top and bottom whisker show the highest and lowest value within $1.5IQR$ of the median. Experiment results are also plotted as single points. Note that each subplot contains all 24 data points.

Figure 5: Evaluation of the 2^k model.

4.3.1 Checking assumptions

Before delving into the predictions of model we need to check whether assumptions we made in modelling actually hold.

Independent errors The model assumes that errors are independently and identically distributed (IID). Figure 5a shows that error does not depend on the predicted throughput. Furthermore, errors do not depend on factor levels as shown in Figure 5b: the median and 25% and 75% percentiles of the error distribution are independent of the factor and the level.

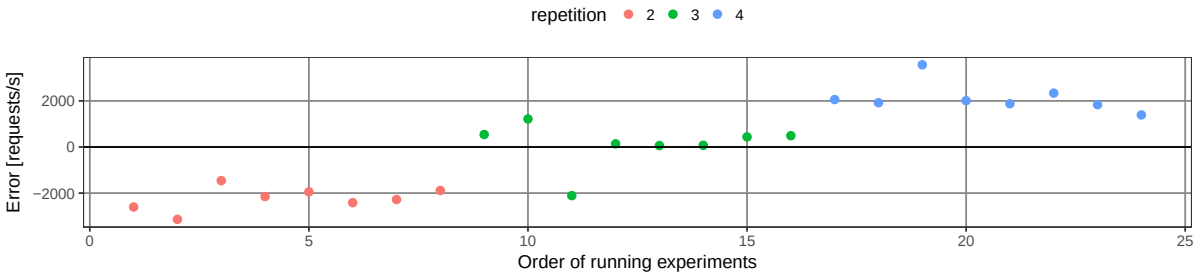


Figure 6: Error of the 2^k model as a function of the order in which experiments were run. Color of the points shows the repetition number.

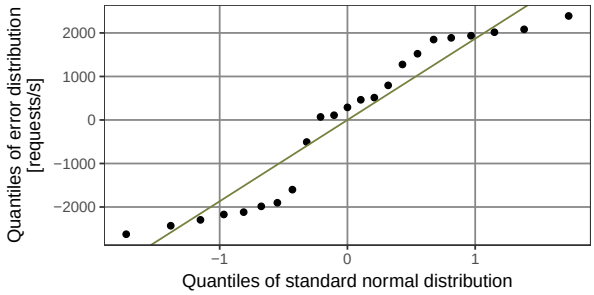


Figure 7: Quantiles of the residual distribution plotted against quantiles of the standard normal distribution. The straight line shows the best fit of a linear trendline through the points.

Figure 6 shows that errors clearly depend on repetition. An obvious hypothesis here is that each successive repetition improved the throughput for some reason. However, repetition 2 was run on Nov 20, and both repetitions 3 and 4 were run on Nov 23 with a 6-hour difference; the resource group was hibernated and redeployed before each experiment. For this reason we can't accept the hypothesis that repetitions somehow affected each other.

Thus, the large variation in throughput between experiments can be explained in two ways: it could a) be caused by differing conditions in Azure between deployments, or b) be somehow inherent to the SUT. If b) were true, we would also see large variance *within* each deployment, which is not the case. This leaves us with option a): the conditions on Azure differed between employments – possibly due to different server or network allocation, or the total load in Azure, or some other factor.

Normally distributed errors The model assumes that errors are normally distributed. The quantile-quantile plot in Figure 7 shows that this is clearly not the case – the lowest quantiles are too low and medium-to-high quantiles too high for the distribution to be normal. This, however, is caused by the trimodality of the error distribution: the throughputs (and thus, errors) of each repetition individually are distributed much more closely to a normal distributions, but when we concatenate the repetitions, the result is trimodal.

Constant standard deviation As Figure 5b shows, the distribution looks similar at all factor levels, so the model assumption of a constant standard deviation holds.

4.3.2 Checking model fit

TODO: analyse how good our model fit is – how big is error etc

Book: "The residuals are an order of magnitude smaller than the responses, and there does not appear to be any definite trend in the mean or spread of the residuals."

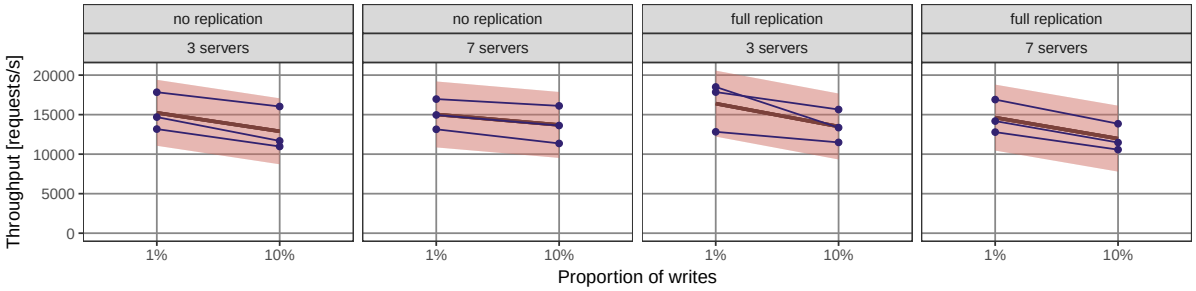


Figure 8: Throughput as a function of W. Each line with points shows the results of one repetition. The red line shows the throughput predicted by the 2^k model; the light red ribbon shows the standard deviation that we would expect to see according to the model, if we were to run 3 repetitions.

multiplicative model is not necessary because: a) range of values covered is small, residuals are small compared to actual values and the spread of residuals is independent of whether actual values are small or large TODO: check if this is true

additive model =¿ mean error 1665; multiplicative =¿ mean error 1667

	variable	coefficient_value	variation
1	x0_constant	14166.7	
2	xa_servers	-338.6	0.024
3	xb_replication	-45.4	0.000
4	xc_writes	-1153.8	0.276
5	x_ab	-487.6	0.049
6	x_bc	-236.3	0.012
7	x_ac	155.5	0.005
8	x_abc	-97.1	0.002
9	error		0.672

Table 3: Values of coefficients and allocation of variation for all variables in the 2^k model.

4.3.3 Analysis of the model

TODO:

noise is obv biggest, writes is most significant, x_ab also a tiny bit. others are all an order of magnitude smaller than writes.

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). This includes a total of 27 experiments in 9 different configurations.

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

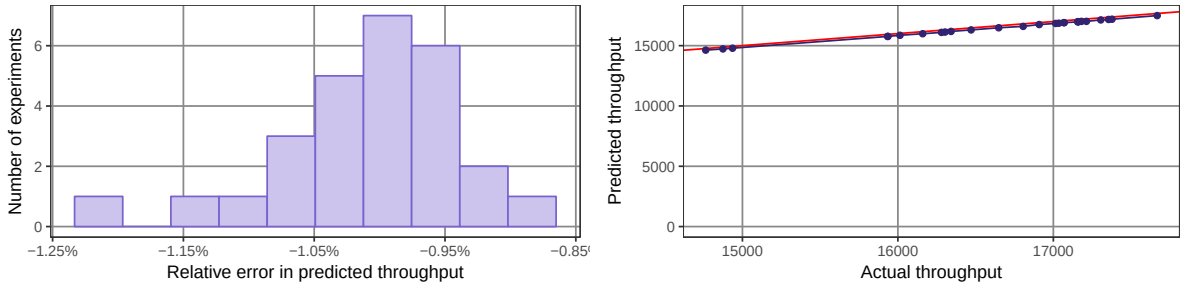
We are assuming a closed system, i.e. clients wait for a response from the server before sending another request. Under this assumption, the IRTL should hold:

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

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

Figure 9: 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 9a; the distribution looks reasonably symmetric. Figure 9b 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.

To explain these results, we need to answer the question: why is the actual throughput lower than the actual throughput? 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 as it would be a major design flaw in memaslap.

A more plausible hypothesis is that the effective value of N is slightly lower than the concurrency I set using the relevant command line flags – because the number of cores in the memaslap machine is much lower than the concurrency I'm using.

Regardless of the exact reason of the deviation, the IRTL holds to a reasonably high accuracy. Perfect accuracy is impossible even without the middleware: in the baseline experiments (Milestone 1 Section 2), relative prediction error is 0.2%-0.5% in a selection of values of N that I checked.

TODO: if time think about what could be causing the -1%, but not a priority

Appendix A: Data for the factorial experiment

	servers	replication	writes	repetition_id	throughput
1	3	none	1	2	17836
2	3	none	1	3	14690
3	3	none	1	4	13173
4	3	none	10	2	16032
5	3	none	10	3	11676
6	3	none	10	4	10972
7	3	full	1	2	17855
8	3	full	1	3	18505
9	3	full	1	4	12828
10	3	full	10	2	15649
11	3	full	10	3	13360
12	3	full	10	4	11488
13	7	none	1	2	16969
14	7	none	1	3	14962
15	7	none	1	4	13147
16	7	none	10	2	16111
17	7	none	10	3	13622
18	7	none	10	4	11356
19	7	full	1	2	16905
20	7	full	1	3	14186
21	7	full	1	4	12789
22	7	full	10	2	13849
23	7	full	10	3	11471
24	7	full	10	4	10571

Table 4: Data used in building the 2^k model in Section 4.