Advanced Systems Lab (Fall'16) – Third Milestone

Name: *Fabio M. Banfi*
Legi number: *09-917-972*

Grading

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

1 System as One Unit

In this section the middleware is modeled as a single M/M/1 queuing system, which is tested against the experimental data from the baseline experiment carried out in the second[1] milestone. In that experiment 3 client machines executing 64 virtual clients each were connected to the middleware which was attached to 3 back-end servers, and ran with 16 threads in the reading pools (see Table 3 on the report of the first milestone for the complete experimental setup). The M/M/1 model is a queuing system consisting of a single queue, which is assumed to comprise the connections from the clients to the middleware and the middleware itself, and a single service server, which is assumed to comprise the connections from the middleware to the servers and the servers themselves, where both arrival and service rate are exponentially distributed. This means that the arrival times are assumed not to depend on previous arrivals, and therefore they are *memoryless* (thus the 'M'). Moreover, buffer and population size are assumed to be infinite, and server discipline is First Come, First Served (FCFS). Therefore, in full Kendall notation, the model is a M/M/1/∞/∞/FCFS queuing system.

This of course is a very rough abstraction of the middleware, since internally it actually has much more than a single queue. Moreover, when retrieving the parameters for the model, one has to account for the fact that the middleware is a multi-threaded system, and is connected to multiple server machines, not just one as the model assumes. Specifically, for the experiment being analyzed, the number of back-end servers is 3, and for each server the middleware spawns 16 threads for reading and 1 thread for writing.

The M/M/1 model is entirely specified by the following two parameters:

- *Mean arrival rate* $\lambda \doteq \frac{1}{\mathbb{E}[\tau]}$, where $\mathbb{E}[\tau]$ is the *expected interarrival time.*
- *Mean service rate* $\mu \doteq \frac{1}{\mathbb{E}[s]}$, where $\mathbb{E}[s]$ is the *mean service time per job.*

Since `memaslap` waits to send new requests until it receives responses to the last sent, the system can be assumed to be closed, and so the *Job Flow Balance* condition, which states that the number of arrivals equals the number of departures, can be assumed to hold. This in particular implies that the mean arrival rate λ can be set to be equal to the throughput X. For this reason, λ has been computed from the `memaslap` outputs of the baseline experiment as the mean throughput. For the mean service rate, the logs of the middleware have been used. More precisely, the average of the server time T_{srv} has been computed, and then it has been divided by the total number of threads running in parallel in the middleware, that is, according to the above reasoning, $3 \cdot 17 = 51$. This clearly gives a *lower-bound* on the mean service time $\mathbb{E}[s]$, since this way it is assumed that all the threads are busy 100 % of the time. In turn, this corresponds to an *upper-bound* on the mean service rate μ.

The script at `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/scripts/analyses/m3/1_MM1` extracts λ and μ from the log files as outlined above, and then computes the relevant information for the analysis of the M/M/1 model of the middleware according to Box 31.1 from [1]. The results are summarized in Table 1.

Hypothesis 1. *The derived quantities of the M/M/1 model of the middleware should be not too accurate, that is, they are expected to differ significantly from the measured quantities, as the model does not take in consideration multi-treading, multiple queues, and multiple servers.*

Analysis. From Table 1, it emerges that the model is stable ($\rho < 1$) and that it has a quite large utilization, namely $\rho = 86.89\,\%$. Another interesting derived quantity is the mean number of jobs in the system, $\mathbb{E}[n] \approx 6$. This number is considerably low; because in the real setting

[1] Originally this experiment was performed in the first milestone, but it was re-run for the second milestone (for 10 minutes instead of 1 hour) after some important changes in the behavior of the middleware were made (see relative report for more details on the modifications).

Parameters		Value
Name	Symbol	
Arrival rate	λ	18 062
Service rate	μ	20 788
Derived quantities		**Value**
Name	Symbol	
Traffic intensity	$\rho \doteq \frac{\lambda}{\mu}$	86.89 %
Probability of zero jobs	$p_0 = 1 - \rho$	13.11 %
Probability of n jobs	$p_n = (1-\rho)\rho^n$	$0.1311 \cdot 0.8689^n$
Mean # of jobs	$\mathbb{E}[n] = \frac{\rho}{1-\rho}$	6.63
Variance of # of jobs	$\mathbb{V}[n] = \frac{\rho}{(1-\rho)^2}$	50.53
Mean # of jobs in queue	$\mathbb{E}[n_q] = \frac{\rho^2}{1-\rho}$	5.76
Variance of # of jobs in queue	$\mathbb{V}[n_q] = \frac{\rho^2(1+\rho-\rho^2)}{(1-\rho)^2}$	48.90
Mean response time	$\mathbb{E}[r] = \frac{1/\mu}{1-\rho}$	$366.839\,\mu s$
Variance of response time	$\mathbb{V}[r] = \frac{1/\mu^2}{(1-\rho)^2}$	$0.135\,\mu s$
Mean waiting time	$\mathbb{E}[w] = \rho\frac{1/\mu}{1-\rho}$	$318.734\,\mu s$
Variance of waiting time	$\mathbb{V}[w] = \frac{(2-\rho)\rho}{\mu^2(1-\rho)^2}$	$0.132\,\mu s$

Table 1: Parameters and derived quantities of the M/M/1 queuing system model.

there are 192 clients, and since the system is closed and each client sends a job at the time, an upper-bound for the number of jobs in the system at any time is 192. Therefore $\mathbb{E}[n] \approx 6$ is a clear indicator that the M/M/1 model is extremely unsuitable for the middleware. Such a low number is explainable by the fact that the model is not multi-threaded, and therefore not too many jobs are needed simultaneously in the system in order to achieve the same model parameters.

That the model is not an accurate abstraction of the system at hand, is also clear from the derived mean response time $\mathbb{E}[r] = 366.839\,\mu s$. The script cited above also computes the effective mean response time as reported by `memaslap`, which is of $10\,622.7\,\mu s$. The middleware does not consist of a single queue and is not attached to a single server, and so the fact that the effective mean response time is only a tiny 3.45 % fraction of the one derived from the models is explainable by the fact that the model needs to be much faster than the real system in order to compensate the missing parallelism. One way to compare the expected response time $\mathbb{E}[r]$ with the one measured, might be to multiply $\mathbb{E}[r]$ by the total number of threads in the middleware (in order to compensate the fact that the service rate parameter was divided by this quantity). This gives $18\,666\,\mu s$, which is clearly larger than the observed response time, but nevertheless much closer to it than $\mathbb{E}[r]$. ♢

Remark 1. *Another possible way to estimate μ is to set it directly to the maximum throughput from the experiment, as measured by `memaslap`. This approach, taken in the next sections, clearly gives a lower-bound on the service rate μ, since it also includes the middleware-clients connections, originally excluded in the above model. The script used for the above analysis can be configured to use the maximum throughput for μ instead of the mean server time from the middleware logs. This results in very similar derived quantities. For instance, the expected response time amounts to $335.796\,\mu s$ (cf. $366.839\,\mu s$), with a similarly close rate with the measured mean response time of 3.16 % (cf. 3.45 %).*

2 Analysis of System Based on Scalability Data

In this section a more accurate modeling of the system is performed, more precisely, M/M/m models are investigated (more detailedly, referring to the previous section, the models considered here are M/M/m/∞/∞/FCFS queuing systems). As before, the queue is assumed to comprise the clients-middleware connections along with the middleware itself, while the service server is assumed to comprise the middleware-servers connections along with the servers themselves.

The models are tested against the data retrieved in the replication effect experiment from the previous milestone. There, 3 client machines executing 70 virtual clients each were connected to a varying number of servers $S = 3, 5, 7$, and replication by writes in the middleware was also varied, from 1 (none) through $\lceil S/2 \rceil$ (half) to S (full). For all different configurations, the number of users is always $N = 3 \cdot 70 = 210$ (see Table 4 on the report of the second milestone for the complete experimental setup). Therefore, for each of the nine resulting configurations, a separate model has been built, and the derived quantities compared to those measured for the specific configuration.

Analogously to the previous section, the arrival rates λ of each model are set to be equal to the mean throughput as measured by `memaslap` for the specific configuration, since it is assumed that the *Job Flow Balance* condition holds. On the other hand, for the service rate μ a different approach has been taken. As outlined in Remark 1, here μ is set to be equal to the maximum throughput registered from `memaslap` at each different configuration. This clearly represents a lower-bound for the actual service rate.

The models for each configuration are computed by the script `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/scripts/analyses/m3/2_MMm`, which also produces the graph from Figures 2 and 1, as well as the values from Tables 2 and 3. According to Box 31.2 from [1], for M/M/m queuing models, given the parameters λ, μ, and m, one can compute the *mean response time* of the model according to

$$\mathbb{E}[r] = \frac{1}{\mu}\left(1 + \frac{\varrho}{m(1-\rho)}\right), \tag{1}$$

where

$$\varrho = \frac{(m\rho)^m}{m!(1-\rho)} p_0$$

is the *probability of queuing*, and

$$p_0 = \left(1 + \frac{(m\rho)^m}{m!(1-\rho)} + \sum_{n=1}^{m-1} \frac{(m\rho)^n}{n!}\right)^{-1}$$

is the *probability of zero jobs in the system*. The *variance of the response time* for the model is computed as

$$\mathbb{V}[r] = \frac{1}{\mu^2}\left(1 + \frac{\varrho(2-\varrho)}{m^2(1-\rho)^2}\right).$$

The points of the modeled mean response time of the graphs in Figures 2 and 1 have been obtained using (1) (they also report the standard deviation, but this is too small to be noticed), and the changes in Tables 2 and 3 have been calculated from those points. Also note that in order to make the measured mean response time comparable to the one obtained from the model, the latter has been multiplied by a factor 10 in the graphs.

In the next sections, the trends of the models are compared to those measured for each configuration, first by keeping the number of servers fixed, then by keeping the replication factor fixed. Thus, in particular, first the trend of the performance when varying the replication factor is investigated, and then the trend of the performance when varying the number of servers is investigated.

2.1 Trend of Performance by Varying the Replication Factor

In this section the trend (or evolution) of the performance of the middleware in terms of response time as a function of the replication factor is investigated for fixed number of servers. Thus, the measured trend is compared to the trend of the response time obtained from the models.

Hypothesis 2. *The trend of the performance of the model as a function of the replication factor should match the one obtained from the measurements.*

Analysis. Figure 1 clearly contradicts the hypothesis. In fact, it is clear that the measured mean response time grows as the replication factor is increased for all numbers of servers, while the modeled mean response time decreases. This is also supported by Table 2, where the changes between replication factors are reported for each number of server. This mismatch between model and real system is explained by the fact that the actual middleware is at its optimal configuration, which coincides with a saturation point. Therefore, even though the model suggests that increasing the replication factor should result in a better performance, the performance of the actual system degrades.

Note that from Table 2 it emerges that the growth of the measured performance is not substantial (always between 101 % and 102 %), but for $S = 3$ servers, the modeled performance has a sharply larger decrease than for $S = 5, 7$, that is, around 87 % against 93 %–97 %. ♢

S	Measured		Modeled	
	$R : 1 \to \lceil S/2 \rceil$	$R : \lceil S/2 \rceil \to S$	$R : 1 \to \lceil S/2 \rceil$	$R : \lceil S/2 \rceil \to S$
3	102.42 %	102.88 %	84.26 %	84.74 %
5	100.96 %	102.75 %	96.42 %	92.18 %
7	101.15 %	102.66 %	97.18 %	94.54 %

Table 2: Changes in response time by varying the replication factor for fixed number of servers.

2.2 Trend of Performance by Varying the Number of Servers

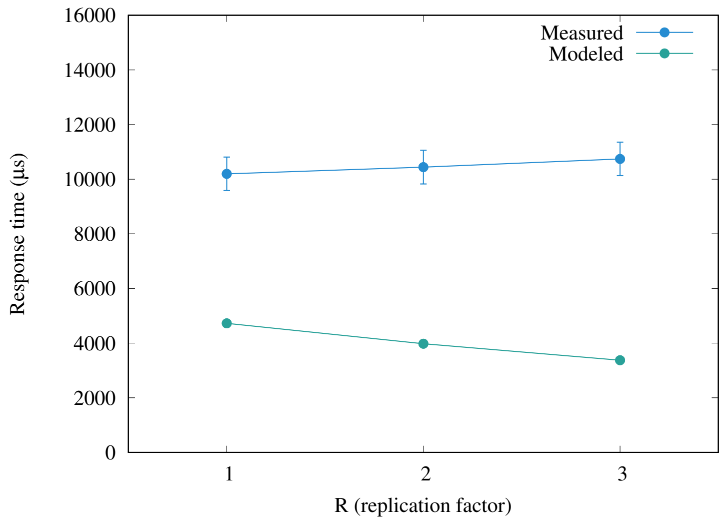
In this section the trend (or evolution) of the performance of the middleware in terms of response time as a function of the number of servers is investigated for fixed replication factor. Thus, the measured trend is compared to the trend of the response time obtained from the models.

Hypothesis 3. *The trend of the performance of the model as a function of the number of servers should match the one obtained from the measurements.*

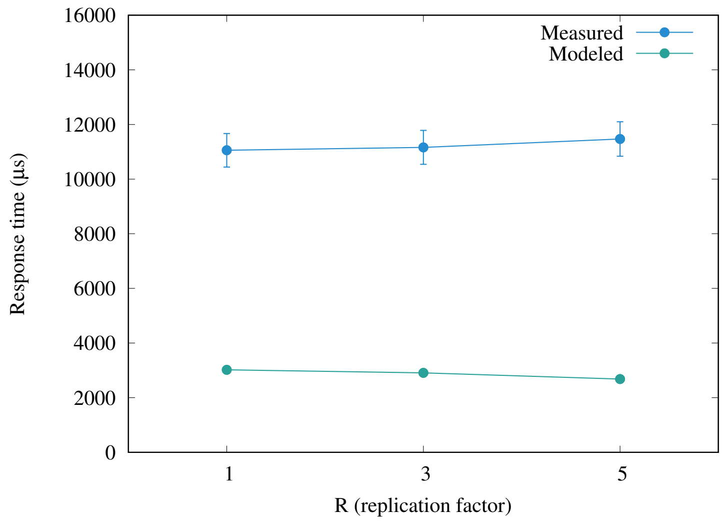
Analysis. As for Hypothesis 2, both Table 3 and Figure 2 contradict this hypothesis for the same reasons. It should be noted however, that Table 3 indicates much higher increases in the measured mean response time than above. This is because the effect of increasing the number of servers is much higher than the effect of increasing the replication factor, as will be showed detailedly in Section 4. It is interesting to note that this holds also for the model. ♢

R	Measured		Modeled	
	$S : 3 \to 5$	$S : 5 \to 7$	$S : 3 \to 5$	$S : 5 \to 7$
1	108.43 %	107.42 %	63.91 %	79.22 %
$\lceil S/2 \rceil$	106.88 %	107.62 %	73.13 %	79.84 %
S	106.75 %	107.53 %	79.55 %	81.89 %

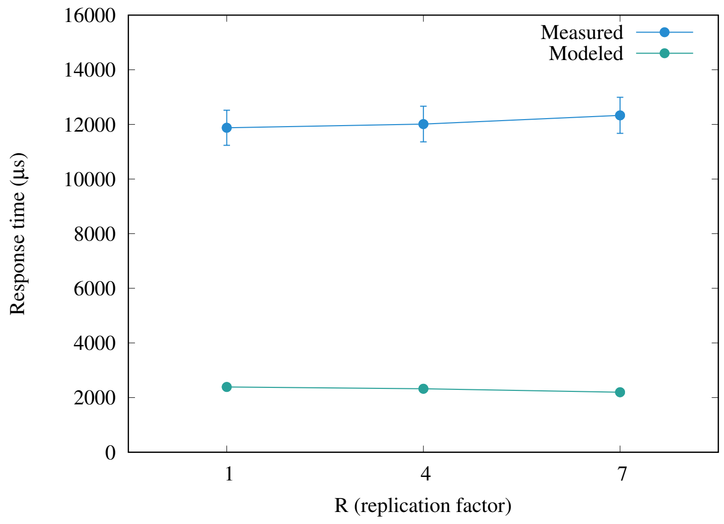
Table 3: Changes in response time by varying the number of servers for fixed replication factor.



(a) $S = 3$ servers.

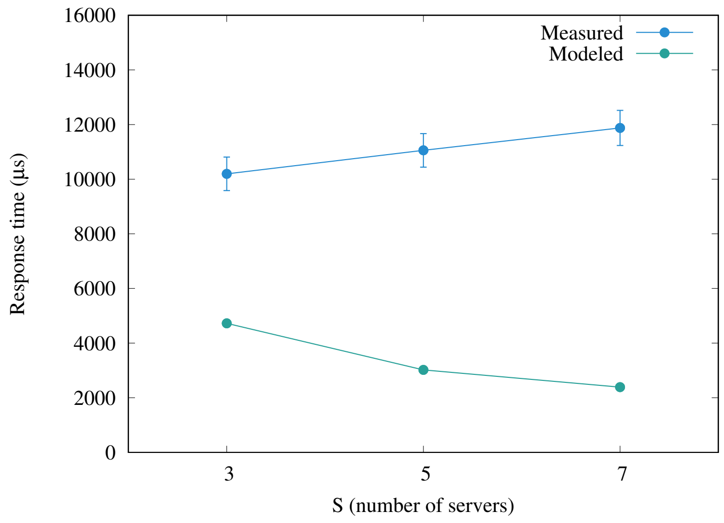


(b) $S = 5$ servers.

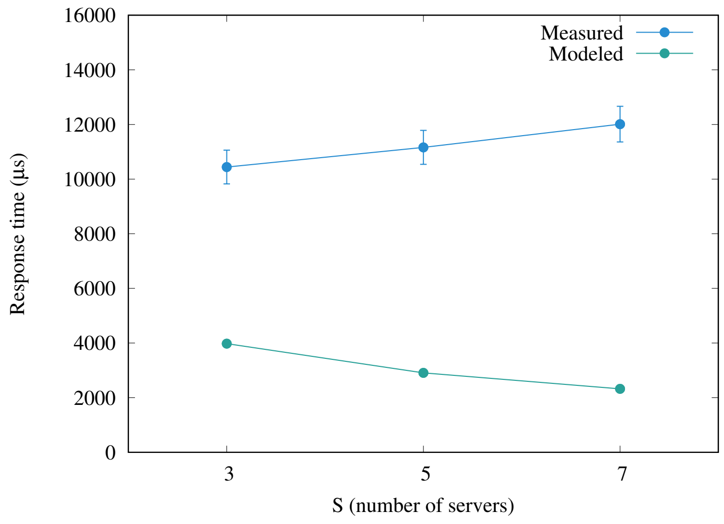


(c) $S = 7$ servers.

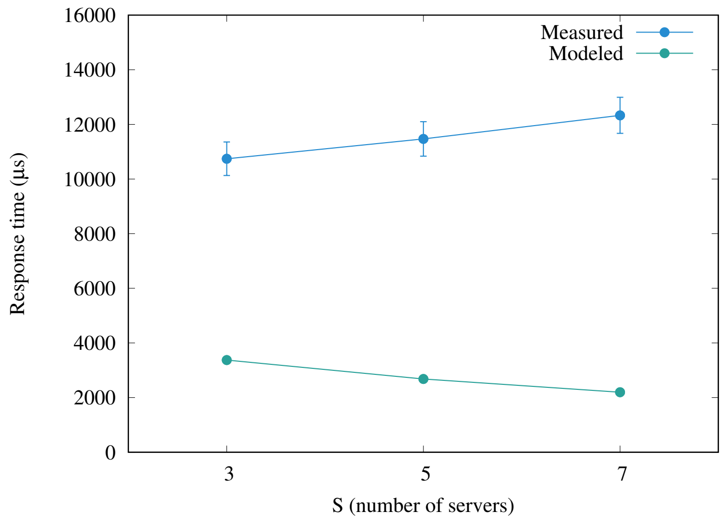
Figure 1: Trends of measured and modeled mean response time by varying replication factor.



(a) $R = 1$ (no replication).



(b) $R = \lceil S/2 \rceil$ (half replication).



(c) $R = S$ (full replication).

Figure 2: Trends of measured and modeled mean response time by varying number of servers.

3 System as Network of Queues

In this section the whole system, which comprises the middleware, the clients, and the servers, is modeled as a network of queues. The configuration with $S = 3$ `memcached` servers and replication factor $R = 1$ (no replication) from the replication effect experiment of the second milestone has been used as source of data for specifying the parameters of the model as well as for testing it against the measured data. There the reading thread pools have size $m = 16$, and a total of 210 virtual `memaslap` clients are connecting to the middleware. Therefore, since the system is closed, the total number of jobs in the network at any time is assumed to be 210. For the complete setup of the experiment, see Table 1 of the second milestone's report. The network is detailedly depicted in Figure 3.

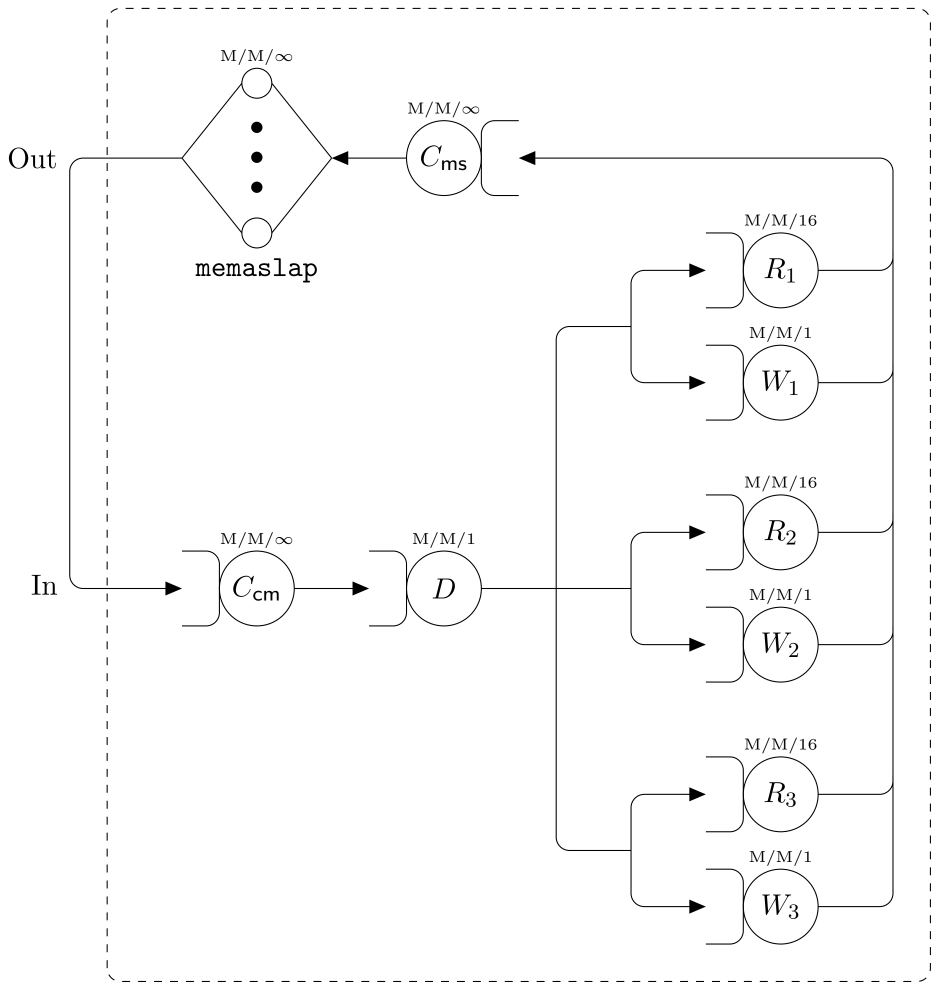


Figure 3: The system modeled as a network of queues.

The queuing devices of which the network is composed are:

- `memaslap`: client terminals, modeled as *delay centers*, that is, queuing systems with infinite number of servers, as they are supposed *not* to implement any kind of queuing.
- C_{cm}, C_{ms}: Clients-middleware and middleware-clients *connections*, respectively. Those are also modeled as *delay centers*, as they are supposed *not* to implement any kind of queuing.
- D: *Dispatcher*, that is, the portion of the middleware which accepts clients connections, processes them, and forwards them to the right reader/writer. This is modeled as a *fixed-capacity service center*, that is, a queuing system with a single server, as it is implemented by a single asynchronous thread.

- R_i: *Readers*, that is, the portions of the middleware dedicated to forwarding `GET` requests to the i-th `memcached` server plus the i-th server itself. Those are modeled as *load-dependent service centers*, that is, queuing systems with a finite number of servers, as they are implemented by a thread pool of size $m = 16$.
- W_i: *Writers*, that is, the portions of the middleware dedicated to forwarding `SET` requests to the i-th `memcached` server plus the i-th server itself. Those are modeled as *fixed-capacity service centers*, as they are implemented by a single asynchronous thread.

Note that this network of queues is *closed*. Moreover, jobs belong to two distinct classes, namely, they are either `GET` or `SET` requests. According to Table 4 of the second milestone's report, `GET`s amount to 95 % of the jobs, while `SET`s amount to 5 %. Since there are in total 210 `memaslap` virtual clients, and each generates a new job after completing the previous, it can be assumed that there are 200 `GET` jobs and 10 `SET` jobs. It is also assumed that the think time Z of the clients is very small, and thus it is set to 0. The script `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/scripts/analyses/m3/3_queues_network` computes the service times of the devices, which are reported in Table 4 (in seconds). For the device D, the average time[2] $T_{\text{pro}} \doteq T_{\text{tot}} - (T_{\text{que}} + T_{\text{srv}})$ from the stable phase (middle third) has been used, and for the devices R_i/W_i, the average time T_{srv} from the stable phase has been used. Note that the entries of the logs of the middleware are separated for `GET`s and `SET`s, and therefore those average times were taken separately for each class of request.

μ	D	R_i/W_i
`GET`	0.000039028	0.001934950
`SET`	0.000025099	0.001983290

Table 4: Service rates for devices D, R_i, and W_i for both `GET` and `SET` requests.

In order to find the throughput and the bottleneck device of the closed network at hand, the standard practice is to employ *mean value analysis* (MVA). For this, the tool `JMT`[3] has been used, which can be employed to run the MVA algorithm including load-dependent centers as described in Box 36.1 from [1]. Since the tool does not allow to specify through which devices a specific class of jobs goes, and which not, the analysis has been run twice, for `GET` requests using the configuration `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/tools/GETs.jmva` and for `SET` requests using the configuration `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/tools/SETs.jmva`. In particular, for the analysis of `GET` requests, the network from Figure 3 has been stripped of the devices W_i, while for the analysis of `SET` requests, the devices R_i have been removed. The MVA algorithm has been given as input the service times of the devices D, R_i, and W_i from Table 4, and the visit rates for each device; for C_{cm}, C_{ms}, and D, this was set to 1, while for R_i/W_i this was set to 1/3 each, since it is assumed that the hashing operation in the middleware results in a perfect uniform distribution of the requests across the servers. The reported outputs of the MVA algorithm are throughput (global and of each device), utilization (of each device), mean number of jobs (on each device), and response time (of the whole system). The results of the analyses for `GET`s and `SET`s are summarized in Table 5.

Hypothesis 4. *The MVA algorithm should return a throughput and a response time which match the ones measured, and it should also identify as bottleneck the same devices which were already identified in the previous milestone, namely the servers.*

Analysis. First, a few words on the model of the system as represented in Figure 3 are necessary. Of course, this model is a very rough abstraction of the actual system. The main difference

[2] See previous milestones for precise definition of such times.
[3] `http://jmt.sourceforge.net`

Device	GET requests			SET requests		
	Throughput	Utilization	# of jobs	Throughput	Utilization	# of jobs
C_{cm}	24 463.73	0.0000	0	1259.78	0.000	0
D	24 463.73	0.9548	20	1 259.78	0.0316	0.03
R_1/W_1	8 154.58	0.9999	60	419.93	0.8328	3.32
R_2/W_2	8 154.58	0.9999	60	419.93	0.8328	3.32
R_3/W_3	8 154.58	0.9999	60	419.93	0.8328	3.32
C_{ms}	24 463.73	0.0000	0	1 259.78	0.000	0
System	24 463.73	–	200	1 259.78	–	10
Resp. time	8 175.37 μs			7 937.90 μs		

Table 5: Results of the MVA algorithm using the `JMT` tool for both `GET` and `SET` requests.

is that the portion of the middleware dedicated to retrieve requests from the internal queues and forwarding them to the servers, the middleware-servers connections, and the servers, are all represented as a single device (R_i/W_i). This was necessary, as the time from dequeuing to forwarding the request is not included in the logs of the middleware. In fact, this time is effectively negligible, as the two operations are performed sequentially in the middleware, and the timestamps t_{deq} and t_{sent} are taken one after the other (cf. e.g. lines 88-95 of `ServerReader.java`). So it can be concluded that the devices R_i/W_i effectively represent just the `memcached` servers plus the middleware-server connections.

From Table 5 it is possible to retrieve the modeled total throughput and mean response time. Those are

$$24\,463.73 + 1\,259.78 = 25\,723.51\,\text{requests}/s$$

and

$$8\,175.37\,\mu s \cdot 95\,\% + 7\,937.90\,\mu s \cdot 5\,\% = 8\,163.50\,\mu s,$$

respectively. The same script from above also reports throughput and response time as measured by `memaslap`, which are reported in Table 6. There those values are compared to those from the model found using `JMT`, and the increase/decrease from measurement to model is also reported.

	Measured	Modeled	Ratio
Throughput (r/s)	21 060.8	25 723.51	+ 22.14 %
Resp. time (μs)	10 195.6	8 163.50	− 24.89 %

Table 6: Comparison of throughput and response time between measurements and model.

The throughput and response time values obtained with the MVA algorithm seem to match the one measured by `memaslap` pretty closely, as hypothesized. Nevertheless, the model is more optimistic, in the sense that a higher performance is expected from the middleware, than the one actually measured. This gap is explained by the fact that the model assumes that the service servers of the queuing devices in the network are constantly processing data in parallel. In reality, this is not true; in fact, in practice the service servers are instantiated by threads, which usually do not correspond to physical cores. The machine on which the middleware was run for this experiment had only 4 physical cores, and therefore it is unrealistic to expect that the $(16 + 1) \cdot 3 = 51$ total threads all work constantly at 100 % in parallel, because in reality they are scheduled.

Finally, the "Utilization" column of Table 5 confirms that, as hypothesized, the devices with the highest utilization are the R_i/W_i. Therefore, the bottleneck of the whole system are the `memcached` servers, as already observed in the previous milestone. ◇

4 Factorial Experiment

In this section a $2^k \cdot r$ factorial experiment with $k = 2$ factors and $r = 4$ repetitions is performed on part of the data collected in the replication effect experiment from the previous milestone. The experimental setup is found on Table 4 of the corresponding report, and the log files are available at `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/log/experiments/04_replication_effect_16-11-20_13:16:14.zip`. The two investigated factors are *number of servers* (A) and *replication factor* (B). The levels for the number of servers have been chosen to be $S = 3$ and $S = 7$, and for the replication factor the levels are *no replication* ($R = 1$) and *full replication* ($R = S$).

As it emerged in the previous milestone, both increasing the number of servers and increasing the replication factor have an impact on the performance of the middleware. The aim of this factorial experiment is to understand which of the two factors has the largest impact, and how the two interact. The data used for the investigation are the middleware logs, and the performance is measured in terms of response time.

4.1 Effects and Errors Computation

For this $2^2 \cdot 4$ factorial experiment, the guidelines given in Chapter 18 of [1] have been closely followed. Since in the experiment the changes in the response time were investigated, the *response variable* y (performance measure) is the response time as well. The non-linear regression model for y is defined as

$$y = q_0 + q_A \cdot x_A + q_B \cdot x_B + q_{AB} \cdot x_A \cdot x_B + e,$$

where q_0, q_A, q_B, q_{AB} are the *effects*, e is the *experimental error*, and

$$x_A \doteq \begin{cases} -1 & \text{if } S = 3 \text{ servers,} \\ +1 & \text{if } S = 7 \text{ servers,} \end{cases} \qquad x_B \doteq \begin{cases} -1 & \text{if } R = 1 \text{ (no replicaitons),} \\ +1 & \text{if } R = S \text{ (full replication).} \end{cases}$$

The effects are computed via the *contrast expressions* as

$$\begin{aligned} q_0 &= \frac{1}{4}(\bar{y}_1 + \bar{y}_2 + \bar{y}_3 + \bar{y}_4) = 3\,747.20, \\ q_A &= \frac{1}{4}(-\bar{y}_1 + \bar{y}_2 - \bar{y}_3 + \bar{y}_4) = -913.51, \\ q_B &= \frac{1}{4}(-\bar{y}_1 - \bar{y}_2 + \bar{y}_3 + \bar{y}_4) = 30.88, \\ q_{AB} &= \frac{1}{4}(\bar{y}_1 - \bar{y}_2 - \bar{y}_3 + \bar{y}_4) = 155.75, \end{aligned}$$

where $\bar{y}_1, \bar{y}_2, \bar{y}_3, \bar{y}_4$ are the means of each of the four configurations, and are reported in Table 7 (computed by the script `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/scripts/analyses/m3/4_factorial_experiment`), in which the errors are also reported according to

$$e_{ij} = y_{ij} - \hat{y}_i = y_{ij} - q_0 - q_A \cdot x_{Ai} - q_B \cdot x_{Bi} - q_{AB} \cdot x_{Ai} \cdot x_{Bi},$$

where $j = 1, 2, 3, 4$ (repetitions), and

$$\hat{y}_i \doteq q_0 + q_A \cdot x_{Ai} + q_B \cdot x_{Bi} + q_{AB} \cdot x_{Ai} \cdot x_{Bi}$$

is the *estimated response*.

i	Effect				Measured				Mean	Estim.	Errors			
	I	A	B	AB	y_{i1}	y_{i2}	y_{i3}	y_{i4}	$\bar{y}_i$	$\hat{y}_i$	e_{i1}	e_{i2}	e_{i3}	e_{i4}
1	1	-1	-1	1	4 442.8	4 627.8	4 929.1	5 142.5	4 785.6	4 785.6	-342.8	-157.8	143.5	357.0
2	1	1	-1	-1	2 730.7	2 839.0	2 590.6	2 428.0	2 647.1	2 647.1	83.6	191.9	-56.5	-219.1
3	1	-1	1	-1	4 549.6	4 411.3	4 679.6	4 502.8	4 535.8	4 535.8	13.8	-124.5	143.7	-33.1
4	1	1	1	1	3 150.8	3 122.8	3 048.1	2 759.6	3 020.3	3 020.3	130.5	102.5	27.7	-260.7

Table 7: Computation of errors for the $2^2 \cdot 4$ factorial experiment.

4.2 Allocation of Variation

With the effects and errors at hand, it is now possible to allocate the variation to the factors, to their iteration, and to the experimental error. The *total variation* or *total sum of squares* is defined as

$$\mathrm{SST} \doteq \sum_{i=1}^{2^2}\sum_{j=1}^{r}(y_{ij}-\bar{y}_{..})^2 = \underbrace{2^2 r q_A^2}_{\doteq\,\mathrm{SSA}} + \underbrace{2^2 r q_B^2}_{\doteq\,\mathrm{SSB}} + \underbrace{2^2 r q_{AB}^2}_{\doteq\,\mathrm{SSAB}} + \underbrace{\sum_{i=1}^{2^2}\sum_{j=1}^{r} e_{ij}^2}_{\doteq\,\mathrm{SSE}}, \tag{2}$$

where the last step is explained detailedly in Derivation 18.1 from [1], and

$$\bar{y}_{..} \doteq \frac{1}{2^2 r}\sum_{i=1}^{2^2}\sum_{j=1}^{r} y_{ij} = q_0$$

is the *mean of responses from all replications*. The four term in the right-hand side of (2) indicate the impact of the factors (SSA and SSB), of their interaction (SSAB), and of the error (SSE). Finally, Table 8, computed with the same script as above, reports those values and for each its percentage of the SST.

Factor	Value	Percentage
SSA	13 352 008.32	93.54 %
SSB	15 255.21	0.11 %
SSAB	388 104.08	2.72 %
SSE	519 150.30	3.64 %
SST	14 274 517.91	100.00 %

Table 8: Allocation of variation for the $2^2 \cdot 4$ factorial experiment.

Hypothesis 5. *Between number of servers and replication factor, the largest influence on performance (measured in terms of response time) should be caused by the number of servers, but their interaction should also significantly matter.*

Analysis. The magnitude of q_A is much larger than that of q_B and q_{AB}, and this clearly is already in line with the first part of the hypothesis. Moreover, Table 8 confirms this: a 93.54% fraction of variation is due to the number of servers, while to the replication factor, just a tiny 0.11% fraction is allocated. This seems to suggest a very positive property of the middleware: *a large replication factor does not slow down the system significantly, independently of the number of servers.* This means that the writer thread implements an almost perfect asynchronous mechanisms, even though, as discussed in the previous milestone (Section 2.2.3), this is not entirely true.

In support of the second part of the hypothesis, from Table 8 it also emerges that the interaction of the two factors plays a more important role than just the replication factor alone. This is in line with intuition and the reasoning above, since assuming that the writer thread is perfectly asynchronous, the increasing replication only affects the performance if also the number of servers in increased. ◇

4.3 Confidence Intervals for Effects

Finally, after having allocated the variation, what one can still do is to compute the variance of the errors and of the effects. The following formulas are also taken from [1]. The variance of the errors is

$$s_e^2 = \frac{\text{SSE}}{2^2(r-1)} = 43\,262.52,$$

while the variance of the effects is

$$s_{q_0}^2 = s_{q_A}^2 = s_{q_B}^2 = s_{q_{AB}}^2 = \frac{s_e^2}{2^2 r} = 2\,703.91.$$

One can then compute the confidence interval for the effects as

$$q_i \mp t_{[1-\alpha/2;\, 2^2(r-1)]} \cdot s_{q_i}.$$

The t-value at $2^2(r-1) = 12$ degrees of freedom and 90% confidence is 1.78. Therefore the confidence intervals for the parameters are

$$q_i \mp 1.78 \cdot \sqrt{2\,703.91} = q_i \mp 92.56.$$

The concrete intervals for each effect are summarized in Table 9. Note that the interval for q_B (marked in red) is the only one to include zero, which means that the effect of replication is very small (close to zero) at this confidence level. Even experimental error and interaction play a more significant role. This is clearly in line with Hypothesis 5.

Effect	90 % confidence interval
q_0	(3 654.64, 3 839.76)
q_A	(-1 006.07, -820.95)
q_B	(-61.68, 123.44)
q_{AB}	(63.19, 248.31)

Table 9: 90 % confidence intervals for the effects of the $2^2 \cdot 4$ factorial experiment.

5 Interactive Law Verification

For an interactive system with N users, where each of them submits the next request after some think time Z, the *interactive response time law* (as defined in Section 33.5 of [1]) states that throughput X and response time R are related by the equation

$$X = \frac{N}{R + Z}, \tag{3}$$

or, equivalently, by

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

In this section the validity of such law is inspected for the replication effect experiment from the previous milestone (Section 2). There, 3 client machines executing 70 virtual clients each were connected to a varying number of servers ($S = 3, 5, 7$), and replication by writes in the middleware was also varied (from 1 through $\lceil S/2 \rceil$ to S). For all different configurations, the number of users is always $N = 3 \cdot 70 = 210$.

The analysis is carried out from the `memaslap` outputs, where the average throughput with standard deviation is computed as in the previous milestone, and the expected throughput is calculated from the average response time, according to (3). Since the think time Z is assumed to be very small, it is set to 0. The analysis is carried out by the script `https://gitlab.inf.ethz.ch/fbanfi/asl-fall16-project/blob/master/scripts/analyses/m3/5_interactive_law`, which produces the graphs from Figure 4 and the values in Table 10.

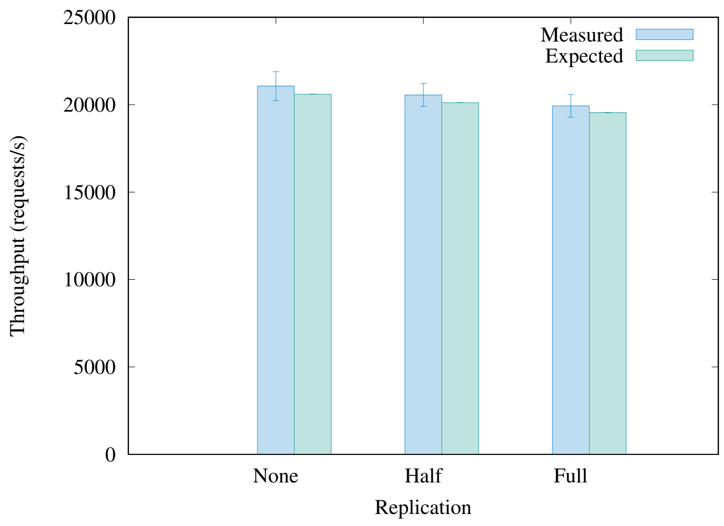
Hypothesis 6. *The interactive response time law should hold for all the results of the replication effect experiment.*

Analysis. Figures 4a to 4c clearly indicate that expected and measured throughput are very close, and in a constant fashion. This is also supported by Table 10, which shows that the ratio between the expected throughput obtained by (3) and the measured (mean) throughput obtained experimentally is of about 98% for all configurations. But it is important to note that contrary to what would be expected, the expected throughput computed with (3) is *lower* than the observed one. This would imply a negative think time Z, which clearly is impossible.

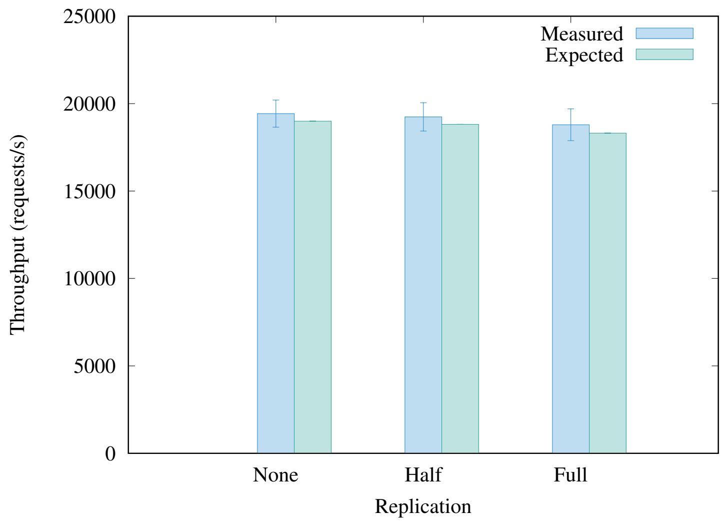
It is therefore necessary to explain this gap. The source of imprecision is clearly due to the fact, that the measured throughput is just a *mean* value. According to Figure 4 of the previous milestone, the distribution of the response time is one-tailed, with a very large peak to the right. This implies that the mean response time is greatly affected by outliers, and therefore it is not a good statistics. In fact, having a smaller response time would increase the expected throughput via (3), thus allowing positive think time Z. ◇

$S = 3$			$S = 5$			$S = 7$		
1	$\lceil S/2 \rceil$	S	1	$\lceil S/2 \rceil$	S	1	$\lceil S/2 \rceil$	S
97.80 %	97.86 %	98.07 %	97.77 %	97.78 %	97.46 %	98.04 %	98.08 %	98.01 %

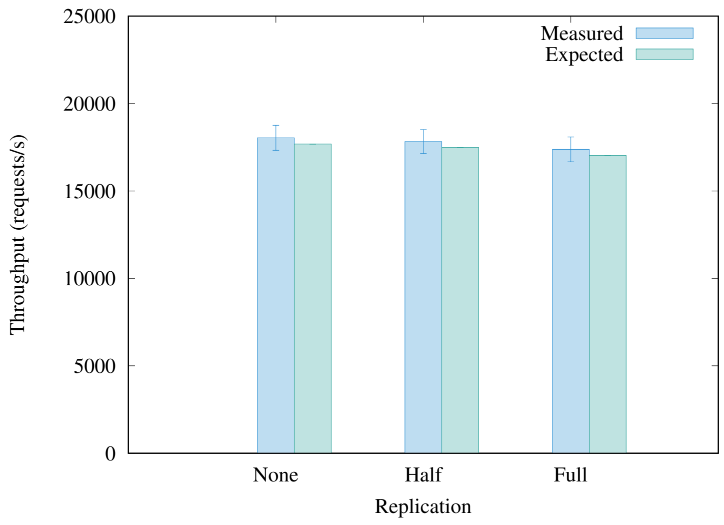
Table 10: Ratio between expected and measured throughput.



(a) Throughput for $S = 3$ servers.



(b) Throughput for $S = 5$ servers.



(c) Throughput for $S = 7$ servers.

Figure 4: Interactive law analysis for replication effect experiment.

References

[1] Raj Jain, *The Art of Computer Systems Performance Analysis* (1991).