Advanced Systems Lab (Fall'16) – Third Milestone

Name: *Eduardo Pedroni*
Legi number: *16-930-596*

Grading

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

1 System as One Unit

An M/M/1 model consists of a single queue of jobs and a single server processing jobs from that queue. Arrival and service time distributions are assumed to be exponential, buffers are assumed to be unlimited and a first-come-first-served service discipline is used by default. In this section, an M/M/1 model of the complete system is built and investigated.

1.1 Model Description

In order to model a multi-queue system such as the one investigated in milestone 1 with a single-queue model, it is first necessary to determine which parts of the real system comprise the model's service. In this investigation, the modelled service consists of the work performed by the Memcached instances, including the network time for the request to reach the servers and for all responses to arrive back at the middleware. The modelled queue, therefore, should account for the internal middleware queues, but also IO buffers used by the operating system to perform network operations, the network time between the clients and the middleware and the time taken to hash and distribute keys in the middleware. Figure 1 illustrates this relationship in detail.

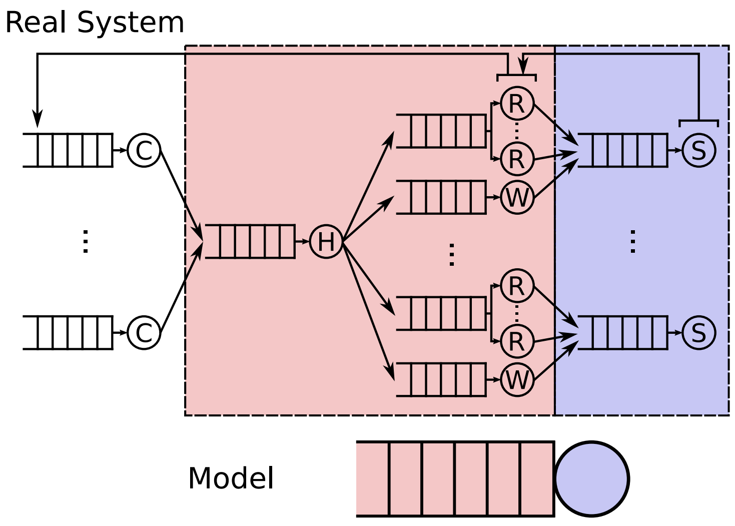

Figure 1: Colour-coded relationship between the real system and the M/M/1 model under investigation.

All modelled quantities can be calculated using two basic parameters: the mean arrival rate λ and the mean service rate μ. In this investigation, both parameters were calculated from the results obtained in the trace experiment of milestone 1.

1.1.1 Mean Arrival Rate

The arrival rate is the number of jobs arriving at the system per unit time. It can be calculated by dividing the total number of arrivals observed during a specified time by the observation time. Both values are taken from the memaslap logs of the trace experiment, yielding the following arrival rate:

$$\lambda = \frac{58168070}{3617} = 16078.15\, jobs/s \tag{1}$$

1.1.2 Mean Service Rate

The mean service rate is the number of jobs serviced by the system per unit time. It is by definition the inverse of the mean service time (the time taken to service one job, which is obtained from the middleware logs of the trace experiment). As clarified previously, only the middleware server time (T_{server}) is used as the mean service time. However, T_{server} is the time taken for a single request to be processed *by its respective worker*. The middleware's true mean service rate, therefore, is much greater, since the middleware was configured to execute with several worker threads. With 20 read threads and one write thread for each of three servers, the real mean service rate is calculated as follows:

$$\mu = \frac{1}{0.00325} \times 3\,(20+1) = 19395.67\, jobs/s \tag{2}$$

1.2 Results

Using the values for λ and μ obtained in Section 1.1, a variety of metrics can be computed, as shown in Table 1:

Metric	Modelled Value
Traffic intensity	0.829
Probability of no jobs in the system	0.171
Mean number of jobs in the system	4.85 jobs
Mean number of jobs in the queue	4.02 jobs
Mean response time	0.000301 s
Mean waiting time	0.000250 s
Mean busy period duration	0.000301 s
Mean number of jobs served in one busy period	5.85 jobs

Table 1: Values modelled with a M/M/1 based on milestone 1 trace experiment.

Traffic intensity is a useful measure of the stability of the system; a value greater than one means the arrival rate is larger than the service rate, implying that the system will either have to drop requests or buffer them, eventually running out of memory. The modelled system is stable, which is consistent with the stable throughput and response time observed in the trace experiment in milestone 1.

The modelled probability of no jobs in the system is too high to be realistic. With a total of 63 threads handling jobs from 192 connected clients, it is unlikely that the middleware spent 17% of the time handling no requests. This discrepancy is reflected in the modelled mean number of jobs in the system and queue; with 192 clients in a closed system, the number of jobs in the system should always be approximately 192 (assuming short think time).

The observed response time in the trace experiment (0.0119 s) was significantly higher than the modelled value. This discrepancy arises from the model's inherent inability to account for all of the latencies present in the real system, since the modelled queue is simply a function of arrival time and service rate. Similar discrepancies occur between the real (0.0087 s) and modelled waiting times, for the same reasons, since waiting time is a fraction of the total response time.

In any M/M/1 model, busy time is by definition equal to response time since jobs are serviced sequentially. The real system, however, processes jobs largely in parallel. The real busy time with a large number of clients approximates the total duration of the experiment, and therefore cannot be equal to the response time.

It is apparent from the obtained results that the M/M/1 model investigated does not model the system accurately. Better results may be obtained if the entire middleware and server system is modelled as the service instead.

2 Analysis of System Based on Scalability Data

The M/M/m model simulates m servers dequeuing jobs from a single queue. As in the M/M/1 model, arrival and service time distributions are assumed to be exponential, buffers are unlimited and the service discipline is FIFO. This section investigates a series of M/M/m models of the system under different workloads and with different resources, using the data collected in part 3 of milestone 2.

2.1 Model Description

As in Section 1, the relationship between the real system and the model must first be established. In this investigation, the modelled services are mapped directly to the memcached instances used in the experiments. The modelled queue corresponds to the rest of the system, from the middleware's IO buffers all the way to the output of the worker threads. Figure 2 shows this mapping in detail.

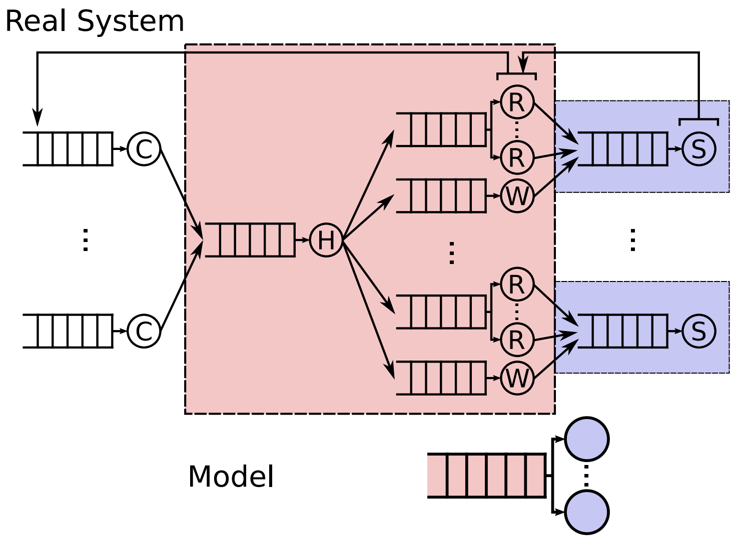

Figure 2: Colour-coded relationship between the real system and the M/M/m model under investigation.

In this section, as in Section 1, the arrival rate of each model is calculated from the total number of operations and the duration of each experiment. The service rate is taken to be the inverse of the measured server time, T_{server}; however, since incoming jobs are handled by a number of worker threads before reaching their destination server, the measured service rate must still be scaled by the total number of worker threads. Therefore, using the experimental parameters from part 3 of milestone 2:

$$\mu = \frac{1}{T_{server}} \times (20 + 1) \tag{3}$$

where T_{server} is the time taken for the job to be processed by memcached (including network time), and 20 and 1 correspond to the number of reader and writer threads per server, respectively.

In addition to mean arrival rate λ and mean service rate μ, the M/M/m model requires the number of servers as an additional input variable m. In this investigation, m is simply taken to be the number of memcached instances used in the experiment, allowing the scalability of the system to be modelled. The effect of replication is outside the scope of this section; the data used to create the models is taken from experiments with replication factor 1.

2.2 Results

2.2.1 Traffic Intensity

According to the model, traffic intensity decreases as the set workload percentage is increased, as shown in Figure 3. The decrease in traffic intensity can be interpreted as a reduction in system overload, since the system is further from instability. This is consistent with the throughput drop observed in the experiment, which implies that the system moves further from saturation as more sets are added to the workload. As explained in milestone 2, this is consistent with the design of the middleware's write threads.

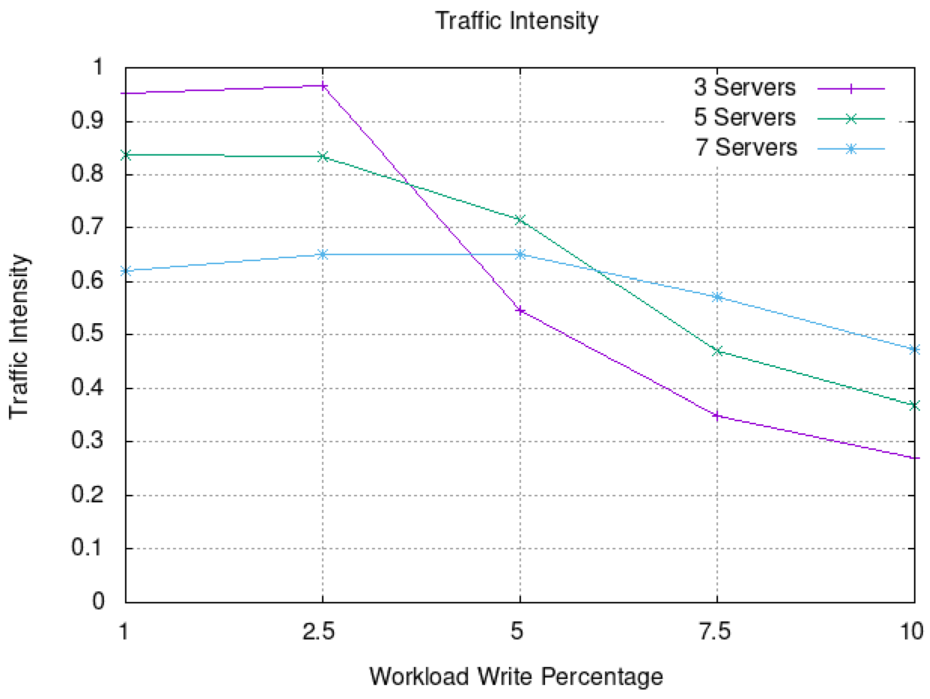

Figure 3: Modelled traffic intensity as a function of number of servers and workload.

The model suggests additionally that traffic intensity is less affected by workload as the number of servers is increased. This is not reflected in the measured throughput, but is nevertheless a reasonable result since, given additional servers, a system that scales perfectly should display increased service rate.

2.2.2 Response Time

As shown in Figure 4, modelled response time tends to decrease with increasing set workload, in contrast with the measured response time which increases given the same load change. As discussed in milestone 2, the measured response time increase in the no-replication case is caused by the synchronous behaviour of the writer threads. The model under investigation fails to take this design element into account as the service time used is T_{server}, which does not reflect the effects of synchronous writer threads.

Similarly to Section 2.2.1, the effect of workload changes on the modelled response time becomes less pronounced with a greater number of servers. Once again this is the expected behaviour for an ideal system which scales perfectly given more resources. The model also predicted flat response time with 7 servers, due to the flat measured server time.

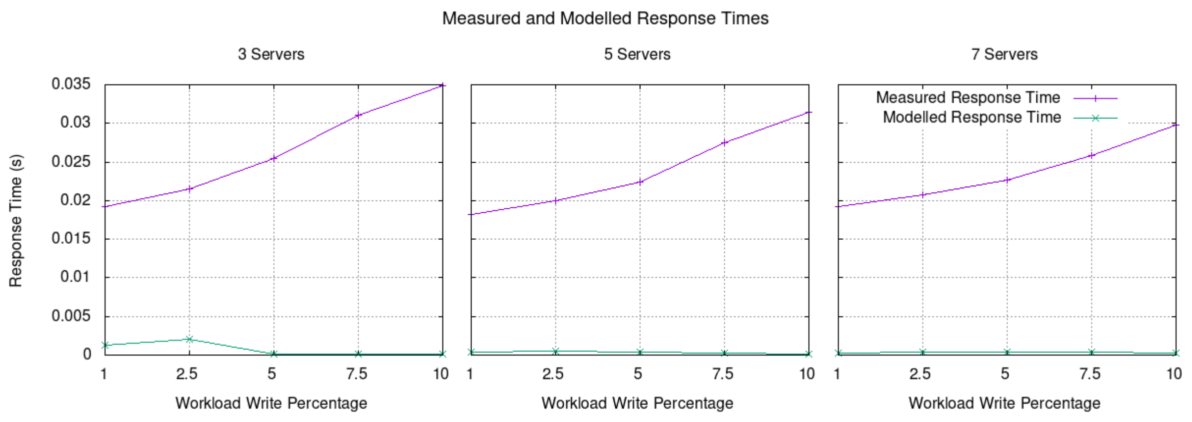

Figure 4: Modelled and measured response times as a function of number of servers and workload.

2.2.3 Jobs in the System

Figure 5 shows the number of jobs in the system predicted by the model under investigation. The number of jobs appears to stabilise given additional servers, though it never exceeds 30. As discussed in Section 1, however, the number of jobs in a closed system should always be approximately the same as the number of clients. Since the number of clients was held constant for the experiments, Little's law can be used to estimate the average number of jobs in the system across all configurations to be approximately 322 jobs. This discrepancy is explained by the fact that the model does not have enough information to predict the number of clients.

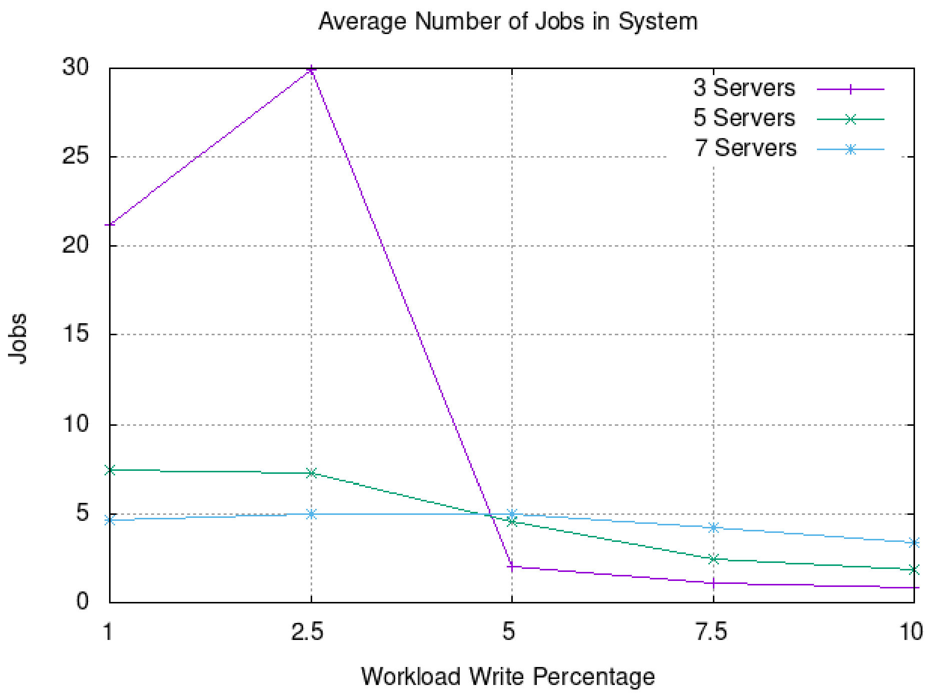

Figure 5: Modelled average number of jobs in the system as a function of number of servers and workload.

The modelled number of jobs in the queue is shown in Figure 6. Although extremely small, the modelled values are quite stable across all configurations. It is possible to estimate the real number of jobs in the queue (if the modelled single queue existed in the real system) by subtracting the number of servers from the number estimated using Little's law, yielding 319 jobs. As before, this discrepancy can be explained by the simplicity of the model and the information lost when modelling a network of queues with a single-queue model.

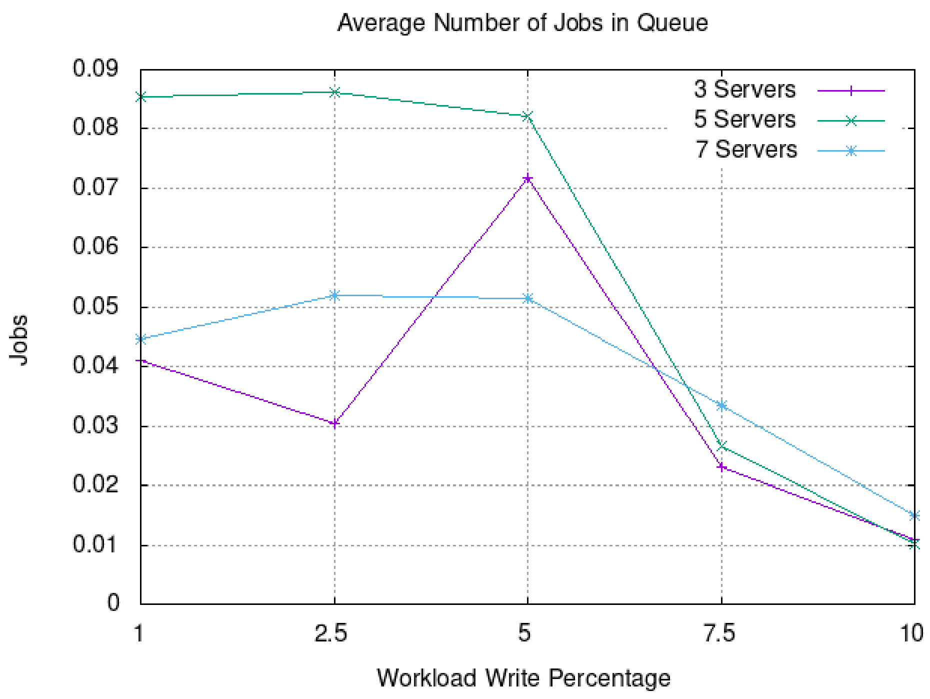

Figure 6: Modelled average number of jobs in the queue as a function of number of servers and workload.

3 System as Network of Queues

In this section, the complete system is modelled as a network of queues using experimental data from part 3 of milestone 2. The goal of this exercise is to create a set of models that approximate the measured behaviour of the system as closely as possible, and to use those models to identify bottlenecks in the system.

3.1 Model Description

A system such as the one under investigation can be modelled as a network of queues, with each queue accounting for a different part of the system. Figure 7 illustrates how each part of the system corresponds to different parts of the network of queues model investigated here.

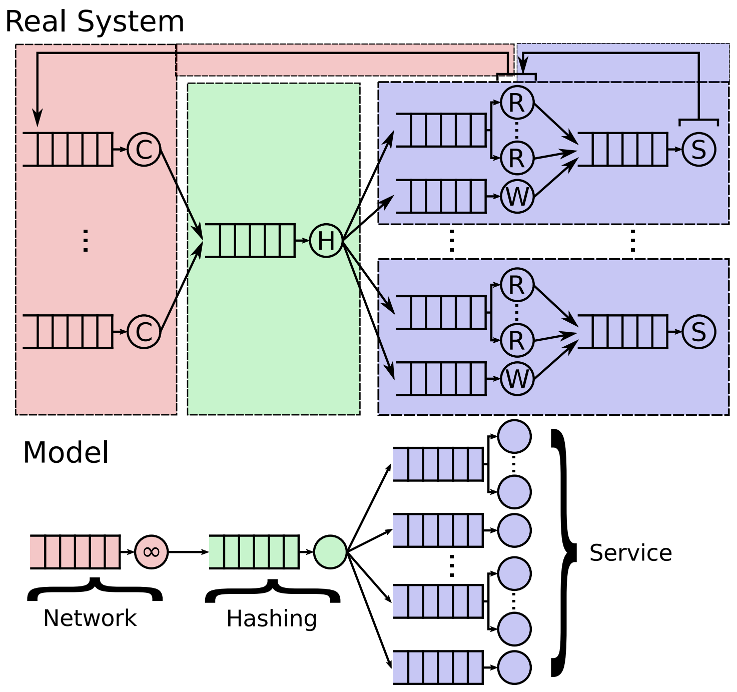

Figure 7: Colour-coded relationship between the real system and the network of queues model under investigation.

The network section of the model accounts for network times both from the client to system and from the system back to the client. This information can be obtained from the data collected in part 3 of milestone 2 by subtracting the total middleware time T_{mw} from the memaslap response time. Client think time is assumed to be negligible and not taken into account by the model. Network time is modelled as an M/M/∞ model since jobs are assumed to be serviced by the network immediately after they are queued, meaning the network does not have waiting time, but rather only service time.

The hashing section of the model corresponds to the time spent by the system analysing each request and determining where to send it. This value was not measured in the middleware, but can nevertheless be calculated as $T_{mw} - T_{queue} - T_{server}$, assuming that any other internal middleware operations are negligible in comparison to the hashing.

Each Memcached instance and its associated set of worker threads is modelled as a single M/M/1 queue for writes and a single M/M/m queue for reads, where m is equal to the number of read threads in the middleware configuration. The response times used for these queues are the separate read and write T_{server} measurements made inside the middleware during the experiment.

The model investigated here was implemented using GNU Octave[1] with the "queueing" package[2], which provides mean-value analysis (MVA) models of generic closed queueing network systems.

3.2 Results

3.2.1 Effect of Workload

To determine whether the model successfully simulates the effect of workload variations, the throughput and response time of the system were modelled with 5 servers, single replication, 1% and 10% workloads. Table 2 shows the obtained results.

Workload	Throughput (operations / s)		Response Time (s)	
	Measured	Modelled	Measured	Modelled
1%	17688.6	18515.0	0.01916	0.00952
10%	10242.6	9862.14	0.03482	0.00621

Table 2: Throughput and response time modelled with network of queues, using data from part 3 of milestone 2.

The modelled throughput is remarkably accurate, being within 5% of the measured value in both cases. Response time is not modelled as accurately, however, although it is closer to the measured value than models investigated in Sections 1 and 2.

In addition to modelling throughput and response time, the network of queues can be used to calculate the utilisation of each part of the system, as shown in Table 3:

Workload	Utilisation		
	Hashing	Write	Read
1%	0.54	0.24	0.91
10%	0.49	0.98	0.32

Table 3: Utilisation modelled with network of queues, using data from part 3 of milestone 2.

With 1% write workload, the bottleneck of the system is clearly the read thread pool. This is a reasonable result, since the majority of requests in this case are get requests. However, with a higher write proportion of 10%, the bottleneck is very clearly the write threads. This dramatic utilisation change can be attributed to the penalty imposed by the partially synchronous design of the writer threads.

3.2.2 Effect of Replication

The model can also be used to model the effect of replication on the system. This can then be compared with the measurements taken in milestone 2. Table 4 shows figures for two models using the data with 5 servers, 10% workload, single and full replications:

Replication	Throughput (operations / s)		Response Time (s)	
	Measured	Modelled	Measured	Modelled
Single	10242.6	9862.14	0.03143	0.00621
Full	7162.0	7408.59	0.04489	0.0051

Table 4: Throughput and response time modelled with network of queues, using data from part 3 of milestone 2.

[1] https://www.gnu.org/software/octave/

[2] https://octave.sourceforge.io/queueing/

Similarly to the results in Section 3.2.1, the modelled throughput is accurate to within 5% of the measured values. However, as in Section 3.2.1, modelled response time is not close to the measured values.

The same method as used in the previous section can be employed to calculate the utilisation of the modelled queues:

Replication	Utilisation		
	Hashing	Write	Read
Single	0.49	0.98	0.32
Full	0.36	0.98	0.24

Table 5: Utilisation modelled with network of queues, using data from part 3 of milestone 2.

Since both models are based on data collected with 10% write workload, the utilisation of the write threads is quite high regardless of replication factor. However, read threads are completely unaffected by replication, which confirms the expected behaviour based on the fact that get requests are not replicated.

4 Factorial Experiment

This section explores the effect of replication and write workload on system throughput for set and get requests combined, using a $2^k r$ factorial experiment.

4.1 Hypothesis

Both replication and write workload are expected to have a negative effect on the throughput of the system. Increasing the number of replications should reduce the throughput since set requests will then take longer to be carried out. Similarly, increasing the write request percentage should decrease the throughput since write requests take longer than get requests. Increasing both replications and write percentage should have an even more significant negative effect on throughput than the sum of their individual effects, since the penalty of replication actually increases if more write requests are received.

4.2 Experimental Setup

The experiment was carried out with the following parameters:

Middleware replication factor	Single, full
Workload write proportion	1%, 10%
Number of servers	3
Number of client machines	2
Total number of clients	320 (160 per machine)
Middleware read threads	20 per server
Runs	$5 \times 200s$

4.3 Results

Table 6 shows the results obtained in the experiment:

Workload	Replication	Throughput (operations / s)
1%	single	16860.2
10%	single	9222.6
1%	full	16412
10%	full	7501.6

Table 6: Average combined (gets and sets) throughput for all four combinations of factors.

4.3.1 Computation of Effects

Using the results outlined in Table 6 along with the throughput measurements from each run, a sign table can be constructed to calculate the effect of each factor on throughput. To estimate experimental errors, the 2^k factorial experimental design can be adapted to include the repetition of the experiment r times. The goal is to generate a model of the following form:

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B + e \tag{4}$$

where q represents the effects of different factors, x represents the values of the factors themselves and e is the experimental error. Table 7 is used to derive the effects q_0, q_A, q_B and q_{AB}, where A represents workload, B represents replication, and a change from -1 to 1 represents an increase in the respective factor (from single to full replication, from 1% to 10% write percentage).

The following effects can therefore be calculated:

$$q_0 = 12499.1, q_A = -4137, q_B = -542.3, q_{AB} = -318.2 \tag{5}$$

I	A	B	AB	y	Mean $\bar{y}$
1	-1	-1	1	(7502, 7573, 7576, 7239, 7618)	7501.6
1	1	-1	-1	(16443, 15623, 16502, 16841, 16651)	16412
1	-1	1	-1	(8933, 9205, 9512, 9261, 9202)	9222.6
1	1	1	1	(16477, 16905, 17886, 16446, 16587)	16860.2
49996.4	-16548	-2169.2	-1272.8		Total
12499.1	-4137	-542.3	-318.2		Total/4

Table 7: Sign table analysis of $2^2 5$ experiment.

As hypothesised, the effects of increasing workload and replication are both negative, meaning throughput decreases in both cases. When both factors are increased simultaneously, an additional negative effect (AB) is observed. As suggested, this is likely caused by the compounding effect of additional set requests when doing full replication.

4.3.2 Estimation of Experimental Errors

The experimental error for each of the $2^2 r$ observations can now be calculated using the following equation:

$$q_0 = y_{ij} - q_0 - q_A x_{Bi} - q_{AB} x_{Ai} x_{Bi} \tag{6}$$

Table 8 shows the calculated error for each measurement:

Estimated Response	Measured Responses					Errors				
$\hat{y}_i$	y_{i1}	y_{i2}	y_{i3}	y_{i4}	y_{i5}	e_{i1}	e_{i2}	e_{i3}	e_{i4}	e_{i5}
7501.6	7502	7573	7576	7239	7618	0.4	71.4	74.4	-262.6	116.4
16412	16443	15623	16502	16841	16651	31	-789	90	429	239
9222.6	8933	9205	9512	9261	9202	-289.6	-17.6	289.4	38.4	-20.6
16860.2	16477	16905	17886	16446	16587	-383.2	44.8	1025.8	-414.2	-273.2

Table 8: Computation of errors for $2^2 5$ experiment.

The sum of the squared errors (SSE) defined below is later used to calculate the confidence interval for the effects and variance of the errors:

$$\sum_{i=1}^{2^2} \sum_{j=1}^{r} e_{ij}^2 = 2,583,029 \tag{7}$$

4.3.3 Allocation of Variation

The amount of variation explained by each factor can be determined through the calculation of the *Total Sum of Squares*, comprised of the sum of squares for effects A, B, AB, and the sum of the squared errors:

$$\sum_{i,j} (y_{ij} - \bar{y}_{..})^2 = 2^2 r q_A^2 + 2^2 r q_B^2 + 2^2 r q_{AB}^2 + \sum_{i,j} e_{i,j}^2 \tag{8}$$

The results shown in Table 9 reflect the strong effect of workload variations in comparison to replication or their combination. In addition, the experimental error is shown to only account for less than 1% of the variation, giving credibility to the obtained figures.

Sum of Squares Terms	Formula	Value	Allocation of Variation
Sum of Squares *Factor A*	$2^2 r q_A^2$	342,295,380	97%
Sum of Squares *Factor B*	$2^2 r q_B^2$	5,881,785.8	1.67%
Sum of Squares *Factor AB*	$2^2 r q_{AB}^2$	2,025,024.8	0.57%
Sum of Squares *Error*	Formula (7)	2,583,029.2	0.73%

Table 9: Sum of squares for the different effects and error.

4.3.4 Confidence Intervals for Effects

The confidence interval of the effects can be found using the following equation:

$$q_i \pm t_{1-\alpha/2;2^2(r-1)} s_{qi} \tag{9}$$

where the standard deviation of errors, s_e, is necessary to compute the standard deviation of effects, s_{qi}.

Standard Deviation	Formula	Value
s_e	$\sqrt{\frac{SSE}{2^2(r-1)}}$	401.79
s_{qi}	$\frac{s_e}{\sqrt{2^2 r}}$	89.84

Table 10: Standard deviations used to calculate confidence intervals.

The t-value at 16 degrees of freedom evaluated at a 95% confidence level is 1.746. The confidence levels for the parameters $q_i \pm (1.746)(89.84) = 156.86$. Therefore:

Parameter	Confidence Interval	
q_0	12499.1 ± 156.86	(12342.24, 12655.96)
q_A	-4137 ± 156.86	(-3980.14, -4239.86)
q_B	-542.3 ± 156.86	(-385.44, -699.16)
q_{AB}	-318.2 ± 156.86	(-475.06, -161.34)

Table 11: Modelled effects with confidence intervals.

5 Interactive Law Verification

The interactive response time law describes the relationship between the response time and throughput of a system as a function of the number of requests in the system and the clients' think time. In this section, the interactive law is used to estimate the think time and verify the validity of the results obtained in part 2 of milestone 2.

5.1 Definition

The interactive response time law is defined as follows:

$$R = \frac{N}{X} - Z \tag{10}$$

where R is the measured response time of the system, N is the number of requests in the system, X is the system's measured throughput and Z is the client think time. In a closed system, clients only submit a request when the response for the previous request has been received. Therefore, the number of requests in the system can be estimated as the number of connected clients.

5.2 Validity Check

Equation 10 can be used to check the validity of experimental data given the number of clients, measured throughput and assuming zero think time. Figure 8 shows the calculated response time in relation to the measured value for all configurations in part 2 of milestone 2.

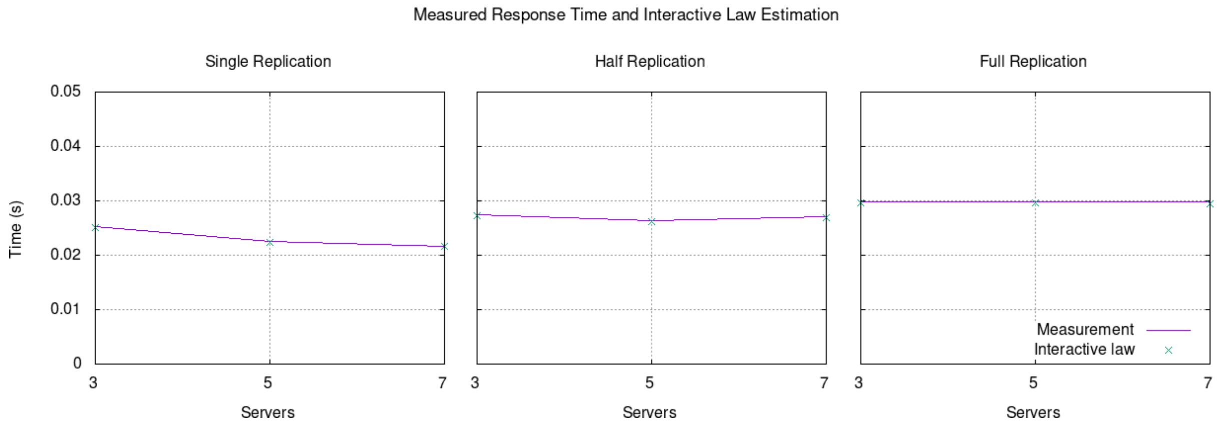

Figure 8: Measured response time and calculated response time using interactive law as a function of replications and number of servers.

As seen in Figure 8, the difference between the calculated and measured response time values is negligible, a strong indication that the experiment results are valid. However, the calculated response time, which assumes zero think time, is consistently smaller than the measured response time. This is contrary to the expected trend; since the measured response time does not include think time, the calculated value should be consistently larger.

5.3 Think Time

Think time is the time spent by each client between receiving a response and sending the next request. Assuming no overload on the client machines and normal network latency on the order of milliseconds, the think time should be a small fraction of the response time.

The average think time of the clients can be calculated using the estimated response time given by the interactive law in conjunction with the response time measured by memaslap during the experiment. Table 12 shows the calculated think time for part 2 of milestone 2:

Ignoring the negative sign, the estimated think time accounts for a small percentage of the measured response time in all cases, as shown in the Ratio column of Table 12. This is the expected behaviour as discussed above. Additionally, since the number of clients is kept constant throughout the experiments,

Servers	Replications	Response Time (s)	Think Time (s)	Ratio
3	single	0.0252	-0.0005	2.13%
3	half	0.0275	-0.0004	1.61%
3	full	0.0299	-0.0005	1.76%
5	single	0.0226	-0.0002	1.10%
5	half	0.0264	-0.0005	1.97%
5	full	0.0298	-0.0005	1.53%
7	single	0.0217	-0.0005	2.14%
7	half	0.0271	-0.0006	2.26%
7	full	0.0297	-0.0007	2.36%

Table 12: Calculated client think time in comparison to confidence interval of response time measurements.

the same think time should be observed in all cases. This is largely the case, which suggests that the negative values are in fact caused by a systematic error in the measurements.

Think time is a function of throughput, response time and number of clients. The latter can be discarded as a possible source of systematic errors as it is a configuration parameter rather than a measured output. Negative think times might therefore occur if either the throughput or response time are systematically overestimated by memaslap.

One possible cause for such an overestimation is the way memaslap measures the passage of time. If the application uses the stdlib `sleep` function to sample a global variable shared by the client threads every second, it may in fact sleep slightly longer than one second, as `sleep` does not guarantee that the actual sleep time does not exceed the specified value. This would in turn lead to an overestimation of the number of operations performed in one second, which might lead to an error in the interactive law calculation.