ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Advanced Systems Lab

A Message Passing System Performance Analysis

(Milestone 2)

Departement of Computer Science

Andrin Jenal

December 19, 2014

Contents

1 Introduction **2**

2 Implemented System **2**
2.1 Changes 2
2.2 System Behavior Revisited 3

3 Modeling **5**
3.1 System As Queues 5
3.2 Model System As One Unit 5
3.3 Model For Individual Components 9
3.3.1 M/M/m Middleware 10
3.3.2 M/M/m Database 11
3.4 Model Of System As Queuing Network 14
3.5 Bottleneck Analysis 15

4 Interactive Response Time Law **17**

A Experiments **18**
A.1 Single Client 18
A.2 2kr Factorial Design 18
A.3 Increasing Number of Clients 18

1 Introduction

In this report the project that has been realized as part of the "Advanced Systems Lab", is will be further examined. In the following paragraphs the message passing system which was built in the first project will be modeled and analyzed accordingly.

First, a brief summary about the modifications of the implemented system will be given and the repeated experiments will be discussed. Then the system will be modeled after [1] with increasing complexity of the chosen models. This means that the system as whole, but also individual components will be modeled as queues. The resulting numbers will be analyzed and compared to the outcome of the experiments obtained in the first project. In the end findings will be explained in detail and a conclusion will be drawn.

All necessary scripts and data used to perform the analysis for the second project can be found in the `milestone2/scripts` and `milestone2/data` folder respectively.

2 Implemented System

After a few changes have been performed since milestone 1, the message passing system and its functionalities are briefly revisited and the outcome of the repeated 2^k factorial design with three factors and three replications is shown.

2.1 Changes

The major changes from the first version to the current state of the project are the way pooled connections are implemented, additional time measurements, independent instantiation of middleware components, different amount of sending and receiving messages and thinking time of clients.

Middleware Server

Each physical instance of the middleware servers provides an assigned number of open connections to the single database server. For each middleware the number of threads in the thread pool and the number of connections to the database are equal. Furthermore, is each middleware component instantiated on a independent physical server. To get a more precise time measurement for the modeling of the individual queues, times are now measured at package arrival in the selector thread, in the task queue while waiting, during the processing of the request handler and for the SQL query. Time measurements are depicted in Figure 1.

Client

For a simpler and less error-prone timing analysis, clients send and receive requests at a rate of 50%. The thinking time which contributes to the sleeping time is no longer dependent of the service time of a middleware server.

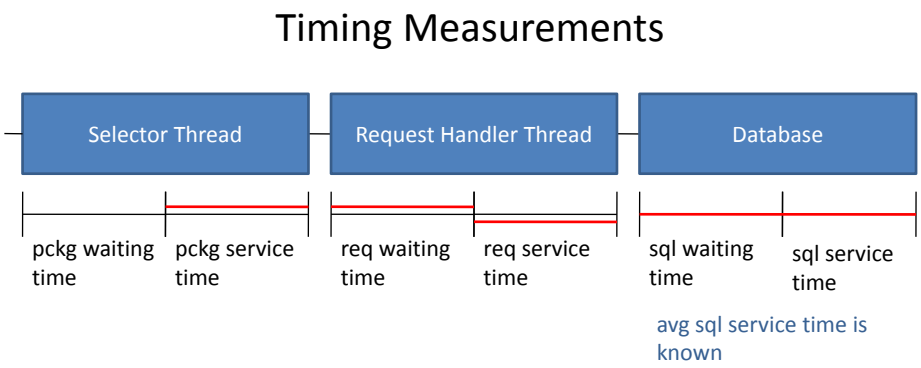

Figure 1: Different stages with the corresponding time measurements.

2.2 System Behavior Revisited

The built and modified message passing system is a closed system. This means there exist no external arrivals. A client sends a certain amount of requests per second, where a subsequent request can only be sent if the previous one has finished. All jobs in the system will therefore be passed from on queue to the next. Hence, the number of jobs in the system remains constant. Each middleware component itself is split up on physically independent machines, but all work identical. They are connected to the same database through a fixed-size connection pool. Clients are simulated by one machine as multi-threaded application. A request send from a client is first handles by a NIO Selector thread which puts the job on a task queue. The task queue is processed in a multi-threaded manner by the request handler thread. Each request handler thread sends the SQL query through a pooled connection. After the SQL query is processed by the database it is returned to the middleware and finally send back to the client.

2k Factorial Design Revisited

Due to implementation issues in the first version of the project the validity of the 2^k factorial design was not truly guaranteed. Thus, the 2^k factorial analysis (as in [1]) is repeated. The factors of focus are similar to the previous approach found in [2]. Namely, the number of middlewares, which now run on physically independent machines, the number of threads in the thread pool, which is similar as the number of connections to the database, and the message size are investigated. An overview of the primary factors for the $2^k r$ factorial design is shown in table 1. To assure statistical significance the experiments are replicated three times with all combinations, which leads to 24 experiments in total.

Since, different daytimes may influence measurements differently, some experiments are run multiple times a day. Finally, the mean latency (total service time) of the different settings are compared with each other.

The results, represented in table 3, indicate that the number of middlewares (24.19%) and message size (26.56%) influenced the computation the most. It is interesting to see that having more threads and more instances (A & B together) has a higher impact on the performance (26.25%) than increasing the number of threads itself (16.79%).

A	**B**	**C**
Number Of Middleware Instances	Number Of Threads	Message Size
1	10	200
2	50	2000

Table 1: An overview of the chosen primary factors.

From the experimental results in table 2 it can be derived that the service time in the middleware remains more or less constant, and the only numbers that change are waiting time in the middleware and waiting & service time in the database. The isolated service time in the database can not be measured. It is assumed, that especially the waiting time in the database grows with the number of connections made to the database.

Configuration	Waiting Time Middleware	Service Time Middleware	Waiting & Service Time DB
1 inst 10 threads 200 chars	3.52 ms	0.98 ms	1.71 ms
2 inst 10 threads 200 chars	0.17 ms	0.99 ms	1.7 ms
1 inst 50 threads 200 chars	0.06 ms	0.96 ms	1.7 ms
2 inst 50 threads 200 chars	0.02 ms	1.09 ms	2.02 ms

Table 2: Different timings resulting from the 2^k factorial design experiments.

I	A	B	C	AB	AC	BC	ABC	y	$\bar{y}_{mean}$
1	-1	-1	-1	1	1	1	-1	(6.75, 7.01, 6.89)	6.88
1	-1	-1	1	1	-1	-1	1	(10.82, 7.78, 10.3)	9.63
1	-1	1	-1	-1	1	-1	1	(3.3, 3.31, 3.29)	3.3
1	-1	1	1	-1	-1	1	-1	(5.07, 5.57, 5.59)	5.41
1	1	-1	-1	-1	-1	1	1	(3.19, 3.04, 3.09)	3.11
1	1	-1	1	-1	1	-1	-1	(4.57, 5.24, 4.93)	4.92
1	1	1	-1	1	-1	-1	-1	(3.39, 3.51, 3.37)	3.42
1	1	1	1	1	1	1	1	(5.76, 5.46, 5.18)	5.47
42.18	-8.31	-6.93	8.71	8.66	-0.4	-1.01	0.87		Total
5.27	-1.03	-0.86	1.08	1.08	-0.12	-0.05	0.1		Total/8
SSE	SSA	SSB	SSC	SSAB	SSAC	SSBC	SSABC		SST
5.9	25.95	18.02	28.5	28.16	0.38	0.06	0.28		107.28
Percentage of Variation									
5.5%	24.19%	16.79%	26.56%	26.25%	0.05%	0.35%	0.26%		

Table 3: All relevant numbers for the 2^k factorial design. The faction of variation are at the bottom.

3 Modeling

The previous tasks involved mainly implementing a system and analyzing its performance by running different experiments. With the gained knowledge we are able to make rough assumptions and predictions how the built system behaves in different situations. To extend the comprehension of our system, another approach is considered and included in the performance analysis - namely, modeling. One crucial analytical modeling technique is queuing theory, which will be the focus of the remaining chapters. The model used for the queuing theory analysis will be illustrated and explained briefly in this section.

First, the system will be modeled as a single queue. Different experimental settings will be applied to this model and compared with the practical outcomes described in milestone 1. Then the models will be extended to more complex queues of individual components and a whole queuing network. In the end, the experimental data will be examined applying the interactive law, and a conclusion will be drawn.

3.1 System As Queues

The message passing system designed and implemented in the previous project modeled as a network of queues is depicted in Figure 2. Clients are depicted as "Terminals" and the middleware and the database are concatenations of different types of queues. This model is way to complex to be analyzed in detail. It will be broken down into simpler models and individual components, for the analysis. It is assumed that the network times are much smaller than the influencing numbers arising from the middleware and database. Thus, the network queues will be neglected for further analysis. Also the measured NIO Selector times contribute to little to the relevant timings (range of microseconds) and are therefore discarded for further analysis. The simplified system is illustrated in the Figure 3.

3.2 Model System As One Unit

The simplest approximation is to model the whole system as single-queue system, as depicted in Figure 4. For this M/M/1/∞/∞/FCFS system, it is assumed that the arrivals and service times are exponentially distributed, there is one server, the buffer and serving capacity are infinity and the service discipline is first come, first served. The way this message passing system is implemented, it is expected, that this approximation is to simple, since it models no multi-threading and should therefore not be a plausible model.

Initially, the calculation was done with the numbers from the trace experiment. However, since there are multiple clients served concurrently, the stability condition is not guaranteed, with this simple model, as demonstrated in table 4.

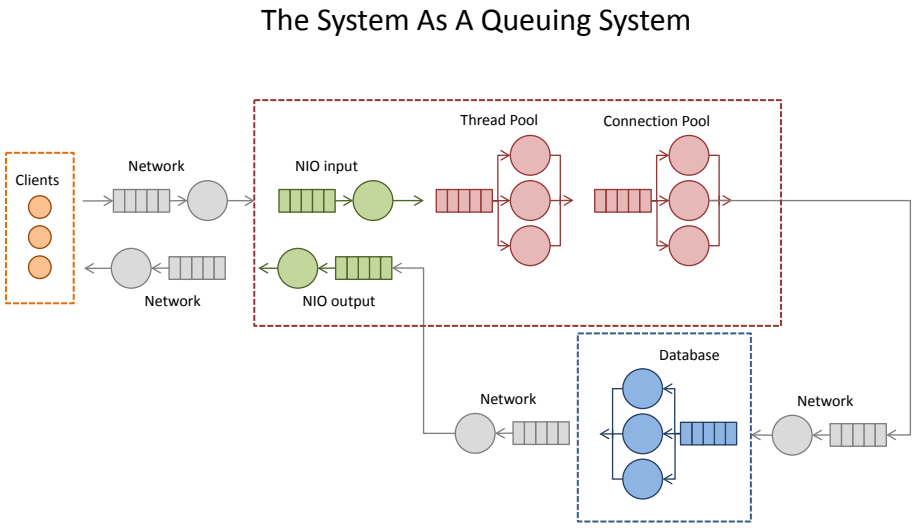

Figure 2: Deatiled model of the message passing system as queues.

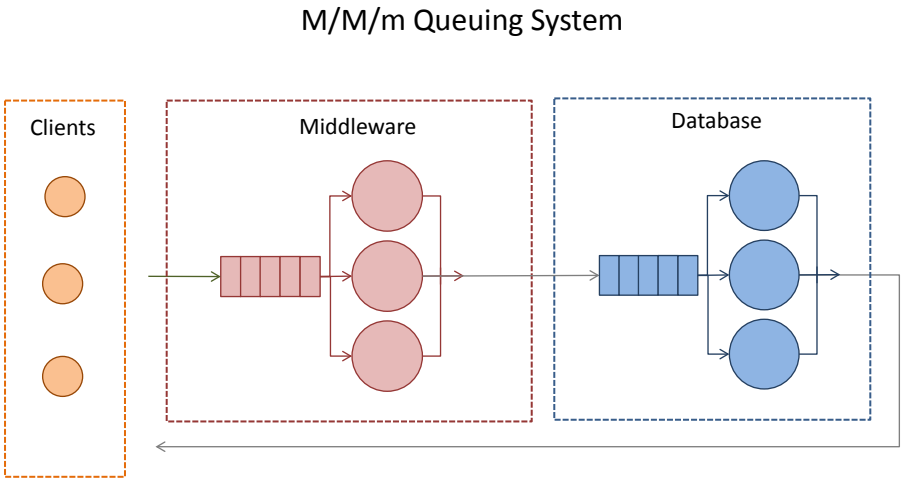

Figure 3: The system modeled as a network of two M/M/m queues.

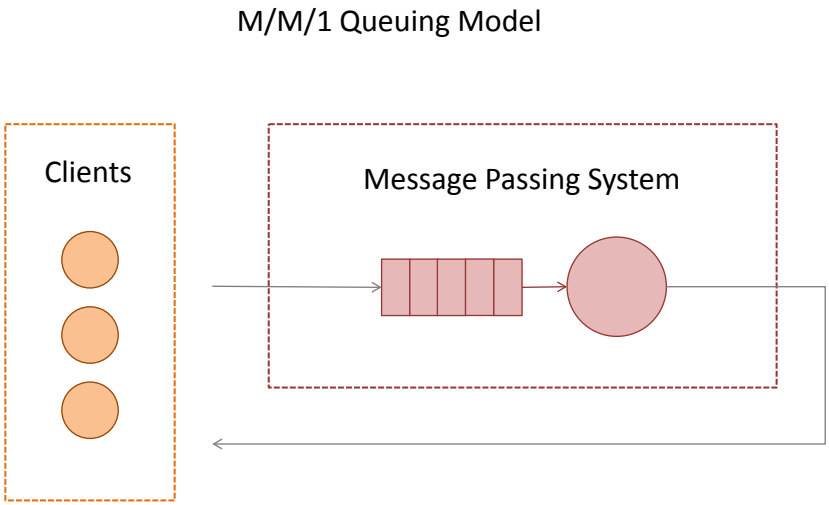

Figure 4: The simplified system modeled as a M/M/1 queue.

λ	0.592 $\frac{reqs}{msec}$
μ	0.334 $\frac{reqs}{msec}$
Trafic intensity ρ:	1.77

Table 4: M/M/1 queue numbers based on the trace experiment.

To get meaningful numbers for the M/M/1 queuing model an additional experiment has been run. The measured times are used for most of the M/M/1 models in this queuing analysis, if not mentioned otherwise. To approximate the whole system as a single queue, which it is not in reality, one client only sends requests to measure the different timings without interfering of other threads. The system is treated as a closed system and the results are shown in table 5. For the M/M/1 model it is now assumed that the arrival rate equals the throughput and the service time equals $avg\_service\_time + avg\_time\_db = 1.929\ ms$ from the single client experiment.

avg_waiting_time_mw	avg_service_time_mw	avg_time_db	avg_lateny	avg_throughput_per_second
0.021 ms	0.707 ms	1.222 ms	2.188 ms	436.711

Table 5: Single client experiment: The system is treated as a closed system.

Modeling the message passing system as a M/M/1 queue as described above with the single client experiment numbers, the following results are computed.

λ	0.436 $\frac{reqs}{msec}$
μ	0.518 $\frac{reqs}{msec}$
Stability condition:	OK
Probability of zero jobs in the system:	0.158
Mean number of jobs in the system:	5.291
Variance of number of jobs in the system:	33.286
Mean number of jobs in the queue:	4.45
Variance of the number of jobs in the queue:	31.737
Mean response time [ms]:	12.135
Variance of the response time [ms]:	147.268
90-Percentile of the response time [ms]:	27.911
Mean waiting time [ms]:	10.206
Variance of waiting time [ms]:	143.547
90-Percentile of the response time [ms]:	25.842
Mean number of jobs served in one busy period [ms]:	6.291
Mean busy period duration [ms]:	12.135

Table 6: Calculated numbers for the M/M/1 queuing model for the single client experiment.

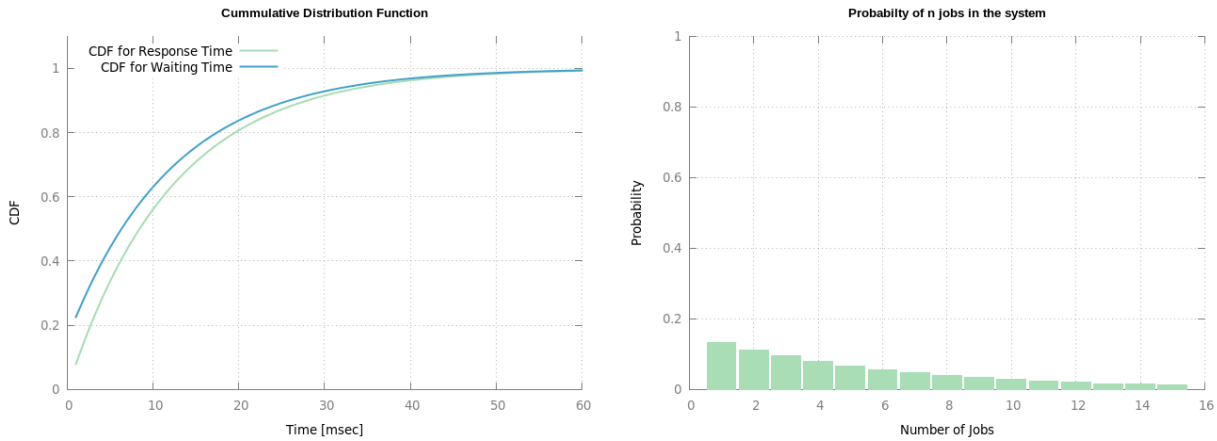

Figure 5: For the single client experiment: Cumulative distribution function of the response time and waiting time on the left. And on the right the probability of n jobs in the system

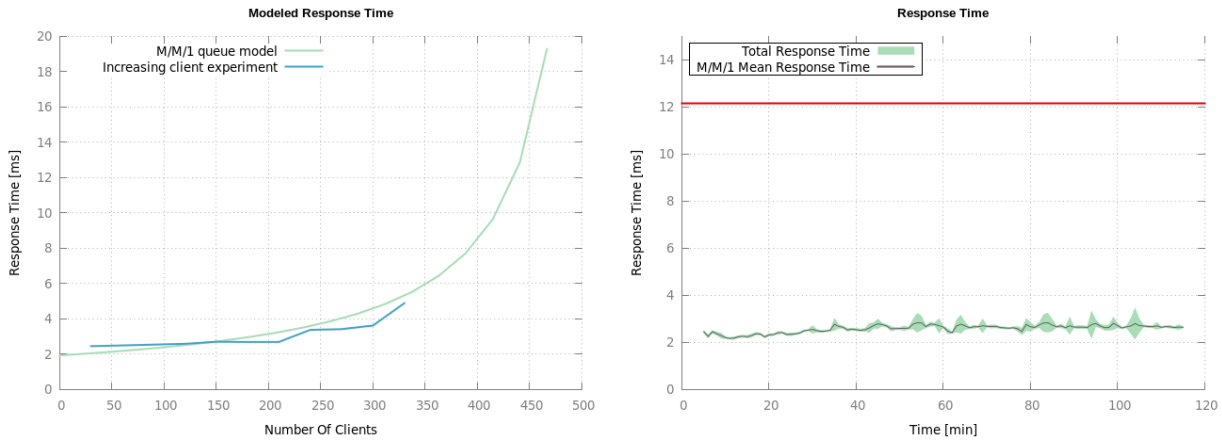

Figure 6: On the left side, the single client model is compared to the increasing number of clients experiment. On the right, the mean response time from the M/M/1 model is compared to the experiment data.

For this analysis it is not important to distinguish between the number of clients interacting with the system and the number of requests made to the system. This means that 60 clients sending 10 requests per second is equal to 600 clients sending 1 request per second. In Figure 6 (left) the limit of the M/M/1 model is at approximately 460 clients or 460 requests per second. In the increasing number of clients experiment the load is actually 20x more than as depicted in the figure 6. This means the experimental data in reality is shifted far to the right. However, it is nice to see, that they have the same trend. In the trace experiment of milestone 1 the average requests per second is about 600 requests per second. This can not be modeled with the simple M/M/1 approximation. It is obvious that the calculated mean response time of 12.135 ms (see Figure 6 (right)) is far above the measured 2.99 ms.

For further analysis only M/M/m models will be considered. The M/M/1 model is a too rough approximation to give reasonable results.

3.3 Model For Individual Components

The core focus of the individual component modeling is the middleware and the database. Both components will be modeled as individual M/M/m/∞/∞/FCFS queues, as depicted in Figure 3, but independent of each other. For the M/M/m models the 2^k factorial as well as the increasing number of client experiment numbers will be taken to calculate the arrival and service rate, respectively. The first experiment (2^k factorial) was run with an average load of 4104 jobs per second. The service time for the middleware is 0.968 ms and the service time for the database is 1.704 ms. The latter experiment (increasing number of clients) was run with an average load of 3307 jobs per second. The service time for the middleware is 0.055 ms and the service time for the database amounts to 3.265 ms. In the two experiments the number of threads in the middleware as well as the number of pooled connections is 50. The model therefore assumes $m = 50$ servers. This number has to be interpreted with caution, because the machines do not have 50 cores. The number of parallel executions in reality lies between 4-12 cores with hyper-threading.

3.3.1 M/M/m Middleware

λ	4.104 $\frac{reqs}{msec}$
μ	1.033 $\frac{reqs}{msec}$
m	50
Stability condition:	OK
Average server utilization:	0.079
Probability of queuing:	0.000
Probability of zero jobs in the system:	0.018
Mean number of jobs in the system:	3.972
Mean number of jobs in the queue:	0.000
Mean response time [ms]:	0.968
Variance of the response time [ms]:	0.937
Mean waiting time [ms]:	0.000

Table 7: Middleware: Calculated numbers of the M/M/m queuing model for the 2^k factorial experiment.

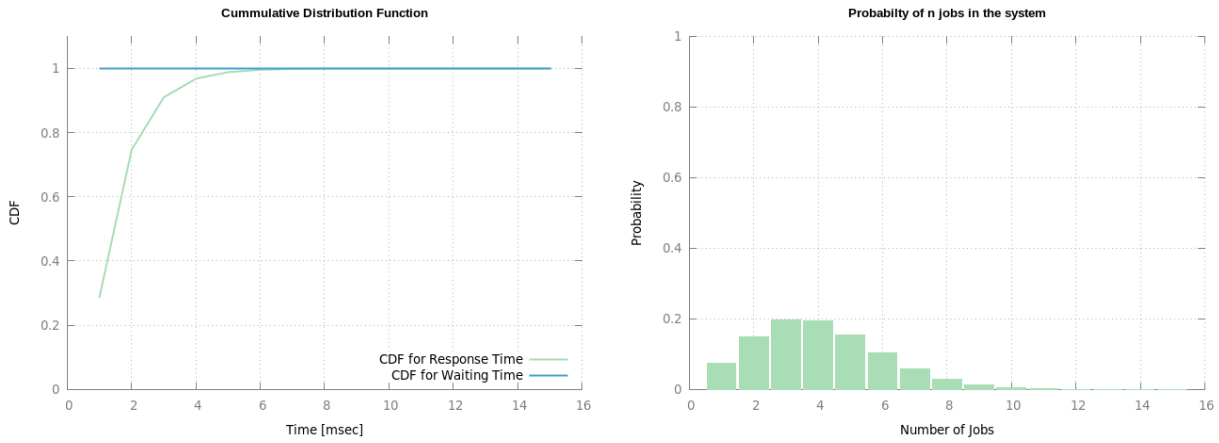

Figure 7: For the 2^k factorial experiment: Cumulative distribution function of the response and waiting time on the left. Depicted on the right is the probability of n jobs in the system.

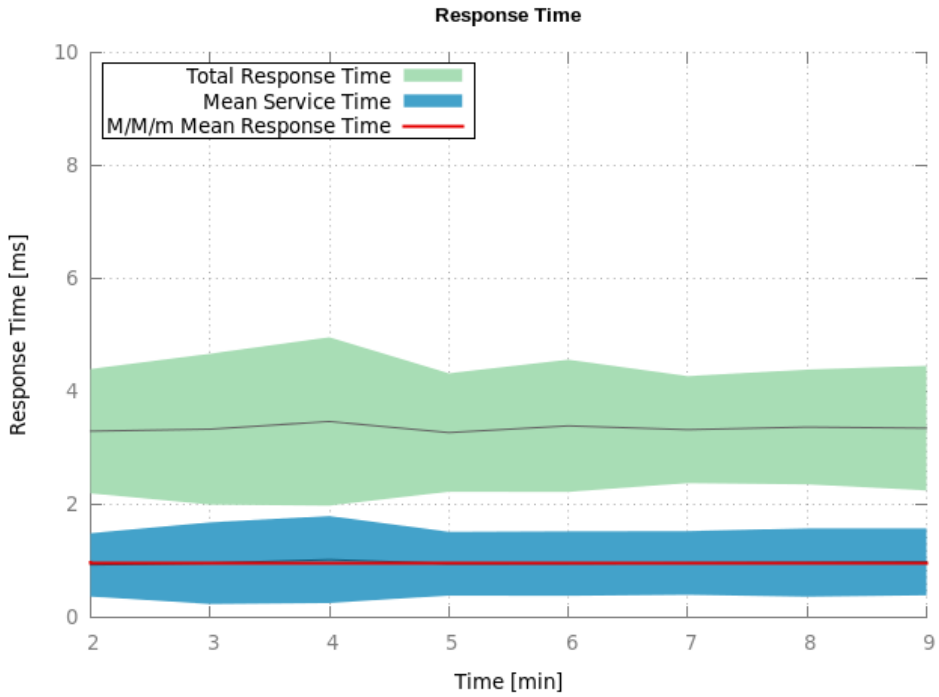

Figure 8: The calculated mean response time value for the service time of the middleware compared to the measured values. The data is taken from the 2^k factorial experiment.

As illustrated in Figure 8 the calculated mean response time(red line) exactly matches the average service time of the 2^k factorial experiment.

3.3.2 M/M/m Database

λ	4.104 $\frac{reqs}{msec}$
μ	0.586 $\frac{reqs}{msec}$
m	50
Stability condition:	OK
Average server utilization:	0.139
Probability of queuing:	0.000
Probability of zero jobs in the system:	0.000
Mean number of jobs in the system:	6.993
Mean number of jobs in the queue:	0.000
Mean response time [ms]:	1.704
Variance of the response time [ms]:	2.903
Mean waiting time [ms]:	0.000

Table 8: Database: Calculated numbers of the M/M/m queuing model for the 2^k factorial experiment.

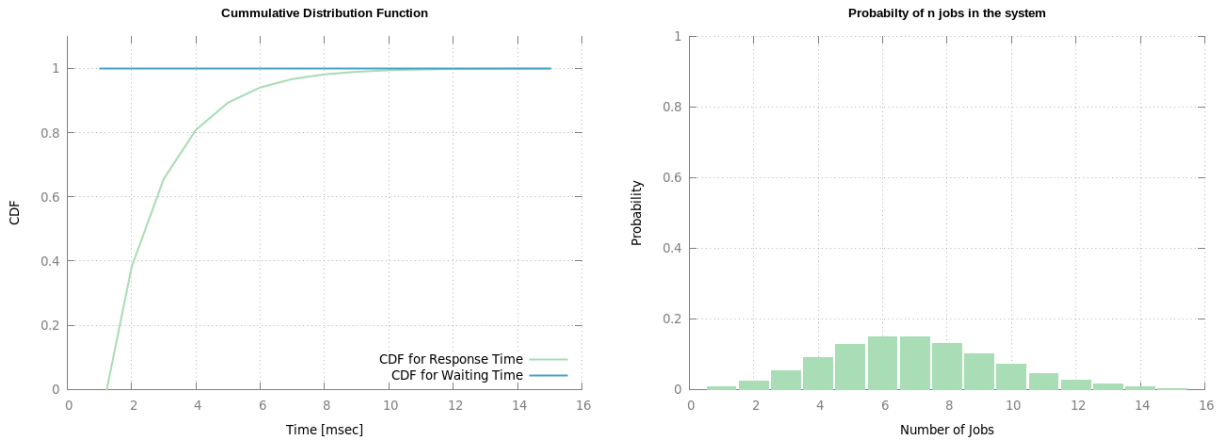

Figure 9: Depicted on the left is the cumulative distribution function of the response and waiting time. Depicted on the right is the probability of n jobs in the system.

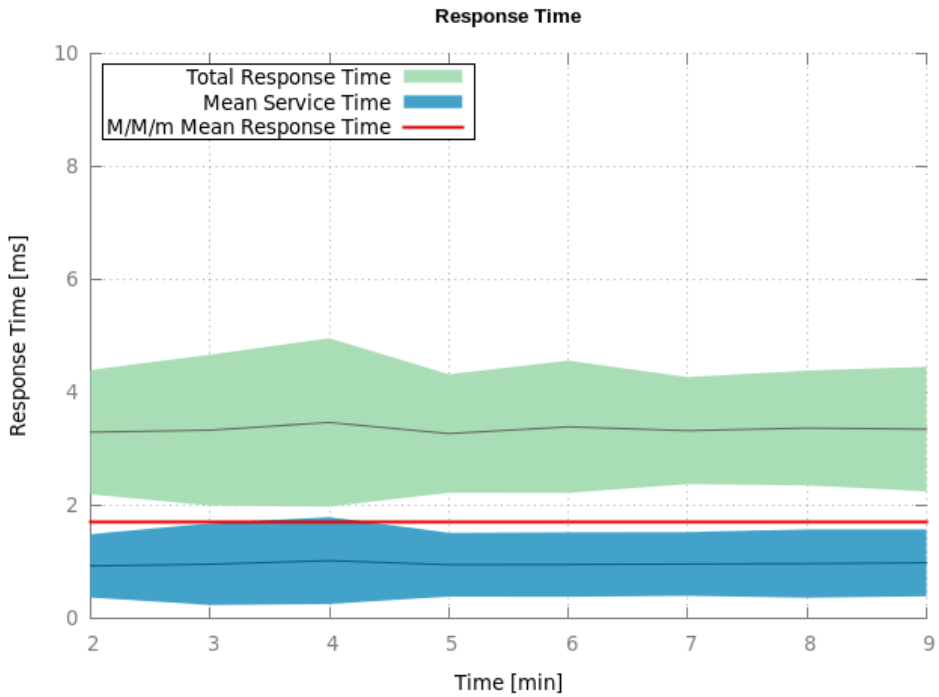

Figure 10: The calculated mean response time value for the service time of the middleware compared to the measured values. The data is taken from the 2^k factorial experiment.

In the Figure 10 the calculated mean response time is visualized (red line) and compared to the average service time of the database. The mean service time of the middleware amounts to 3.306 ms and the mean service time of the database to 1.704 ms, what is equal the calculated numbers.

For the increasing number of clients experiment only the M/M/m model for the database is considered. The calculated numbers are shown in table 9.

λ	3.307 $\frac{reqs}{msec}$
μ	0.306 $\frac{reqs}{msec}$
m	50
Stability condition:	OK
Average server utilization:	0.215
Probability of queuing:	0.000
Probability of zero jobs in the system:	0.000
Mean number of jobs in the system:	10.797
Variance of jobs in the system:	10.797
Mean number of jobs in the queue:	0.000
Mean response time [ms]:	3.265
Variance of the response time [ms]:	10.66
Mean waiting time [ms]:	0.000
Variance of the waiting time [ms]:	0.000
90-Percentile of the waiting time [ms]:	0.000

Table 9: Database: Calculated numbers of the M/M/m queuing model for the increasing clients experiment.

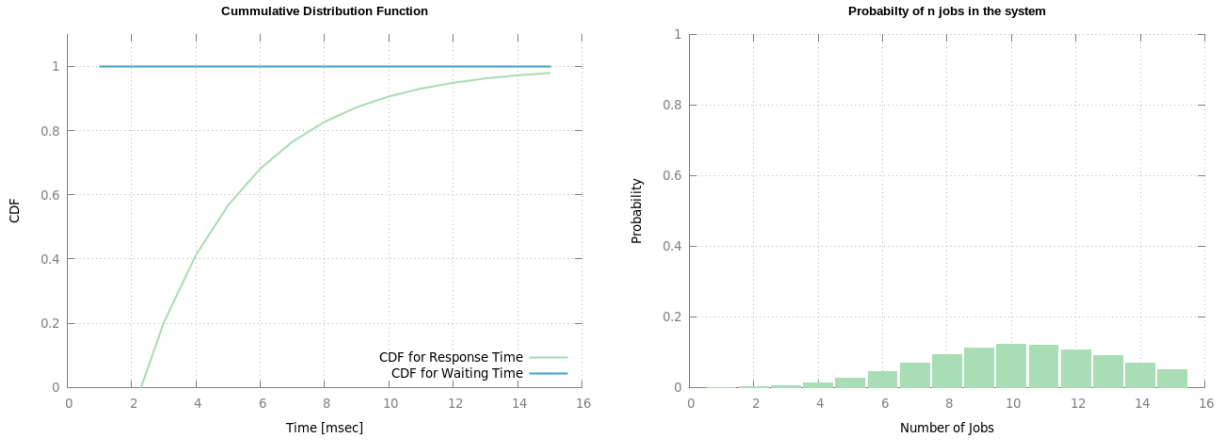

Figure 11: Depicted on the left is the cumulative distribution function of the response and waiting time and on the right the probability of n jobs in the system is shown for the increasing load experiment.

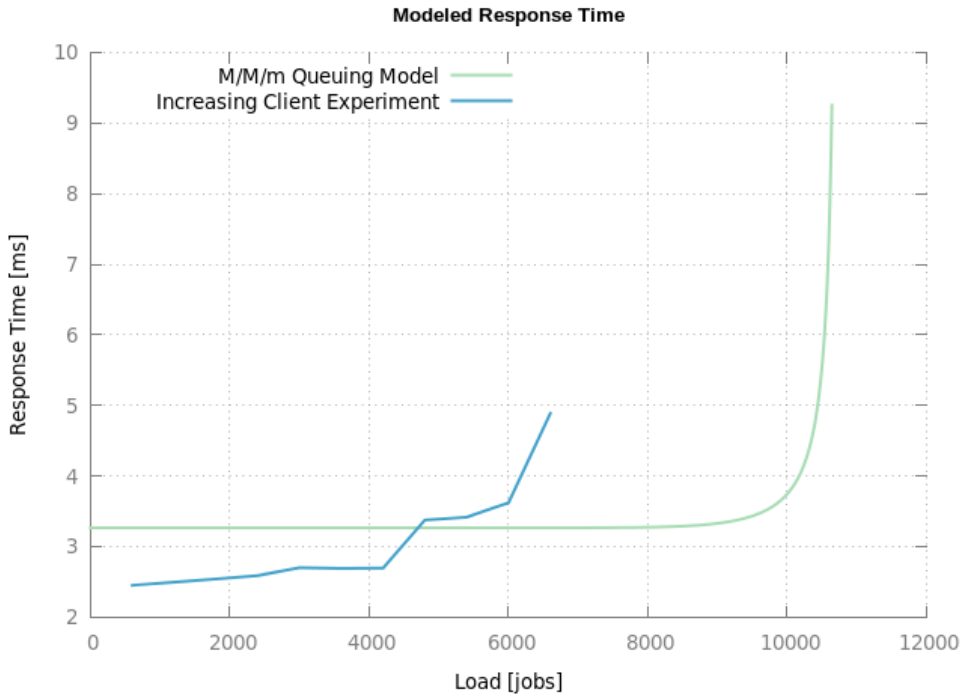

Figure 12: Load of the increasing clients experiment compared with the modeled M/M/m queue.

Comparing the M/M/m model result to the increasing clients experiment data already gives a nice approximation. Important to notice is that otherwise in the M/M/1 analysis (see Figure 6) the computed load is actually the measured load and not scale shifted anymore. Another approximation which is not encountered in the model is the fact, that one job can be in only one queue. This is actually not the case for the message passing system. After processing the job in the request handler, the job is handed to the database. While the jobs is in the database, the thread has to wait for the answer to hand it back to the NIO selector. However, in the model it is assumed that after passing the job to the database the middleware can process the next one in the queue and the database sends back the request directly to the terminal (clients) after the query execution, as depicted in Figure 3.

3.4 Model Of System As Queuing Network

A further refinement of the model is to see the individual components as a connected network of queues as depicted in Figure 3. According to [1], one approach to examine closed queuing networks is the mean-value analysis. For this analysis, it is assumed that the queuing network consists of only fixed-capacity service centers, which is actually a reasonable assumption for the middleware. However, for the database, it is likely that the service time of the database is influenced by the number of connections to it. This was shown to be the bottleneck in milestone 1. Therefore it might be better consider it a load-dependent service center.

The different devices for the MVA with the corresponding visit ratio and service times are

listed in the table 10. The number of users is increased up to $N = 600$. The total number of devices is selector thread, request handler thread and database, therefore $M = 3$. The think time is the inverse of the number of requests sent $Z = 0.05$. Regarding the fact that neither the middleware nor the database machine can run 50 jobs in parallel, it was assumed that the visit ratio is inverse the number of cores including hyper-threads. For the middleware this amounts to 8 and for the database to 16. The details of the increasing number of clients experiments are listed in the appendix.

Device	Service Time	Visit Ratio
Selector	0.0044 ms	1
Request Handler	1.1221 ms	0.125
Database	2.5505 ms	0.0625

Table 10: Overview of the input parameters for the MVA algorithm.

Applying the mean-value analysis leads to the output depicted in Figure 13. The bound of the throughput is approximated close to the measured value from the experiment. However, the response time increases linearly with the number of clients. This is because, we assume that for each additional time the queue in front of the device increases by one, as we have a closed system. The model still is not precise enough to fit the experimental data close. One aspect, that still is discarded is that the measured service time of the database actually corresponds to the waiting and service time of the database. For an approximation the mean query execution time of the database is computed. This amount only to approximately 0.03 ms, what is way to less for the entire service time, since concurrent update of the database is not included in this number.

3.5 Bottleneck Analysis

Based on the previous computation a bottleneck analysis can be performed. The bottleneck is the device with the highest utilization and hence the highest demand. An overview of the devices and their corresponding throughput, utilization and demand is shown in table 12. The computation is based on the throughput formula: $X = N/(R+Z)$, utilization formula: $U_i = X_i S_i$ and the demand formula: $D_i = V_i S_i$.

Device	Service Time	Visit Ratio	Throughput	Utilization	Demand
Selector	0.0044 ms	1	6.273	0.027	0.0044
Request Handler	1.1221 ms	0.125	0.7841	0.8799	0.1402
Database	2.5505 ms	0.0625	0.392	1	0.1594

Table 11: Overview of the devices.

From this table arises that the database is the device with the highest demand and thus the bottleneck device. This was already found to be the bottleneck in the previous

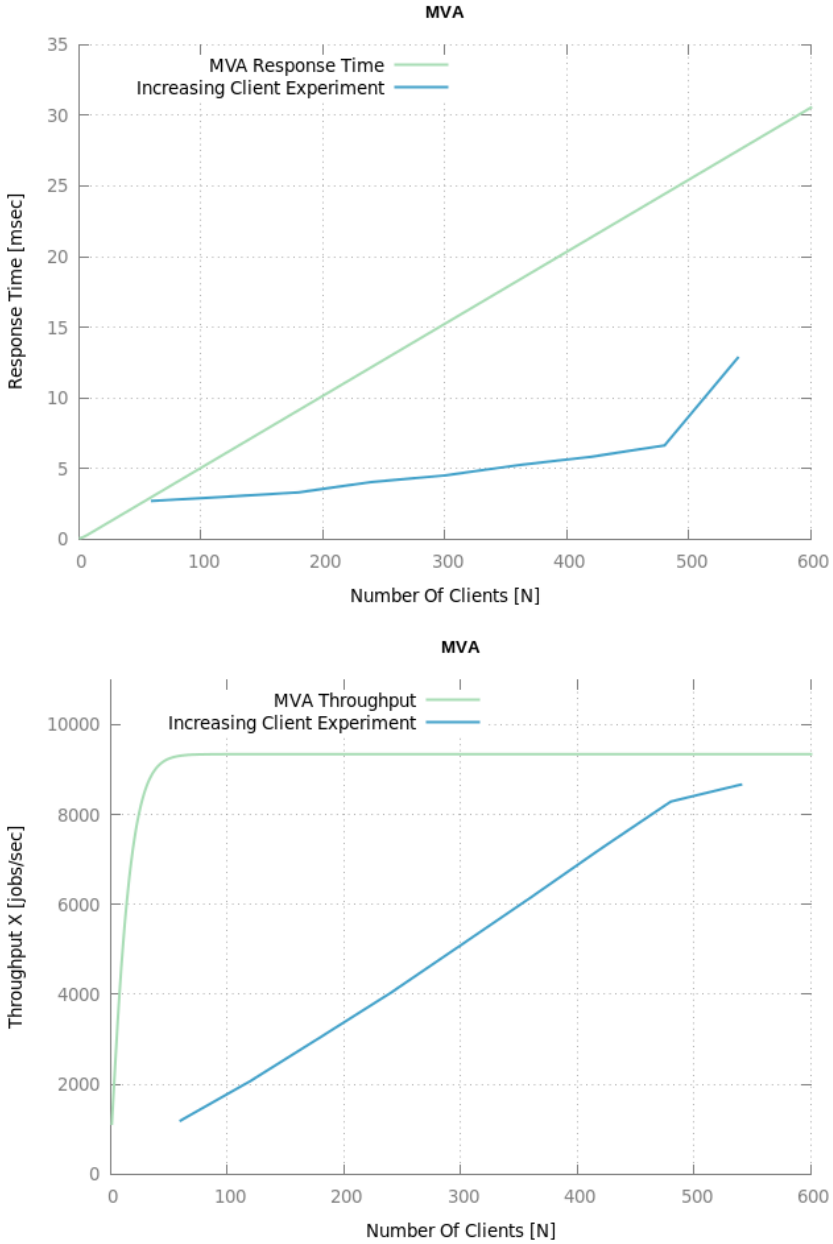

Figure 13: Depicted on the left is the cumulative distribution function of the response and waiting time and on the right the probability of n jobs in the system is shown for the increasing load experiment.

milestone. The MVA algorithm is the most fine-grained modeling analysis and fits best compared to the M/M/1 and M/M/m models.

4 Interactive Response Time Law

In the following section the experiments from the milestone 1 will be examined using operational laws. It is assumed that the forced flow law can be applied, which assumes that there are equal number of arrivals as completions $A_i = C_i$. From the interactive response time law we get the throughput x as: $X = \frac{N}{R+Z}$, where N denotes the number of users, R the response time and Z the thinking time.

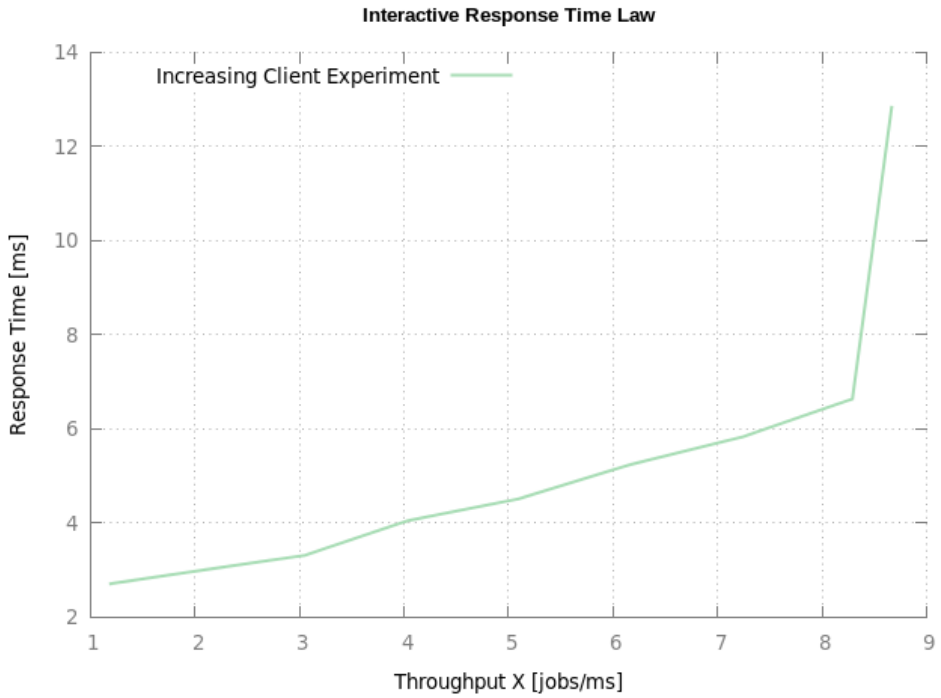

Figure 14: Increasing throughput plotted against the response time. The input data is taken from the increasing number of clients experiment.

Trace Experiment

	Throughput X	Response Time R	Thinking Time
Measured	592	2.99 ms	0.1 ms
Computed	194	0.913 ms	-

Table 12: Increasing clients experiment: Plot shows influence of throughput to response time.

A Experiments

A.1 Single Client

avg_service_time_selector	avg_waiting_time_mw	avg_service_time_mw	avg_time_db	avg_lateny	avg_throughput_per_second	sd_throughput_per_second
0.001495	0.020088 ms	0.707048 ms	1.222178 ms	2.188380 ms	436.711864	33.423123

Table 13: Single client experiment - Exact numbers.

A.2 2kr Factorial Design

The $2^k * r$ factorial design is found in the repository. A replication of three was chosen for a statistical significance.

A.3 Increasing Number of Clients

To get more detailed time measurements, this experiment is rerun. The increasing number of clients can also be seen as increasing load. Each clients send 20 requests per second.

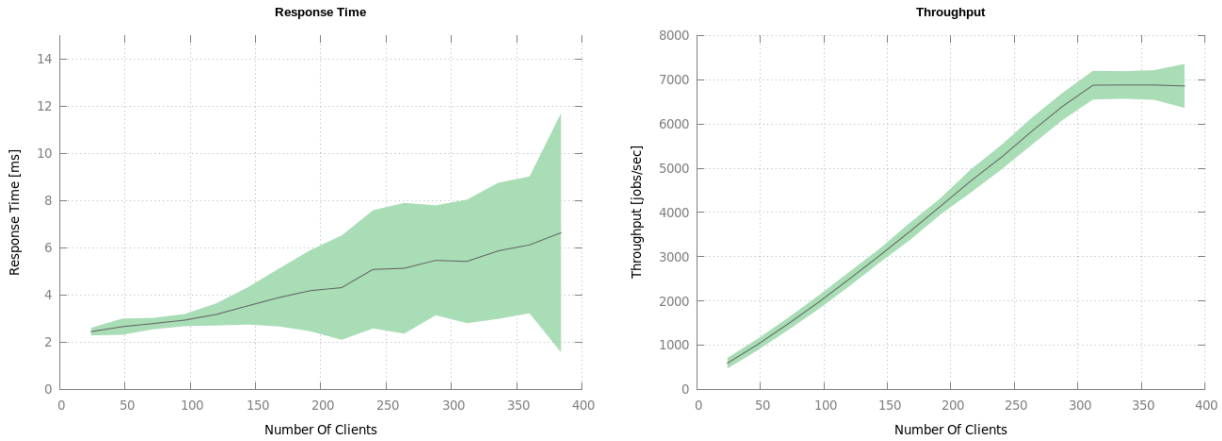

Figure 15: Response Time and Throughput of the increasing number of clients/load. Increase until maximum clients 400.

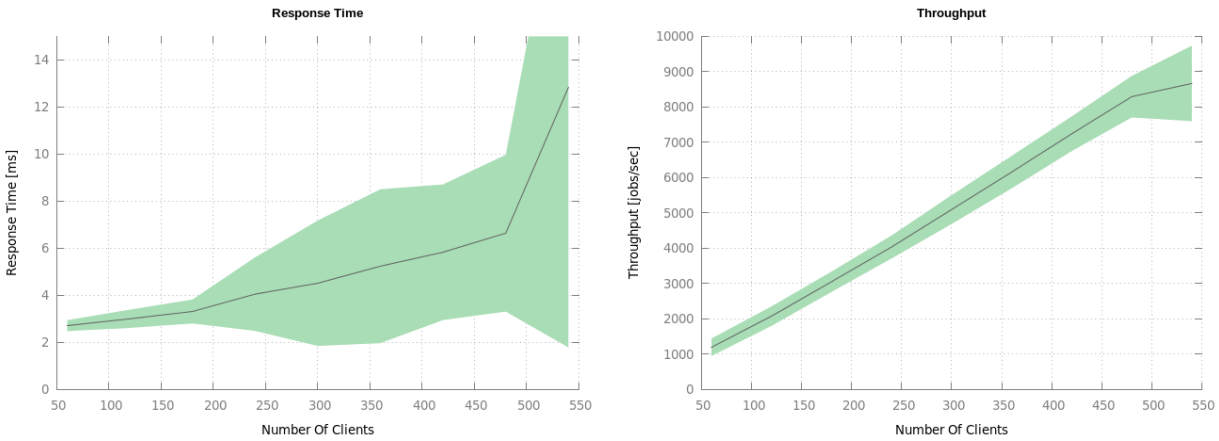

Figure 16: Response Time and Throughput of the increasing number of clients/load. Increase until maximum clients 600.

References

[1] R. Jain. The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling. *Book*, 1991.

[2] A. Jenal. A message passing system performance analysis. *Report*, 2014.