UNIVERSITÀ DEGLI STUDI
DI GENOVA

Distributed Computing 2024-2025

Scheduling Simulator

Prof. Matteo Dell'Amico

Mohammadamin Rezaei Sepehr (5918498)

Mikhail Pilipov (6139355)

M/M/1 queue

The M/M/1 queue simulates a single-server system in which service times have an exponential distribution with rate μ and job arrivals follow a Poisson process with rate λ. The notation M/M/1 is used to describe this queueing system, where "M" stands for memoryless arrivals, "2" for memoryless service times, and "1" for a single server.

M/M/n queue supermarket model

In a multi-server environment, incoming jobs must be assigned effectively to lower average wait times. This assignment is improved by the Supermarket model, which selects the shortest queue from a subset of d queues out of all n servers.

```
class Arrival(Event):

def process(self, sim: Queues):
        sample_queues = sample(range(sim.n), sim.d) # randomly select d queues

        queue_index = min(sample_queues, key=sim.queue_len) # pick shortest queue
```

M/M/n queue supermarket model plots

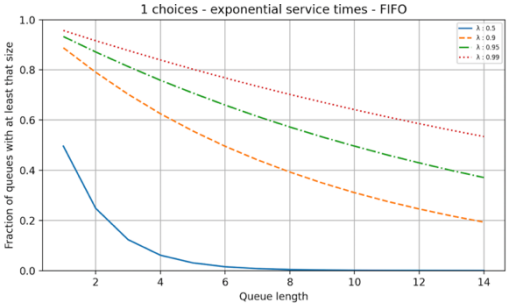
System prefered parameters

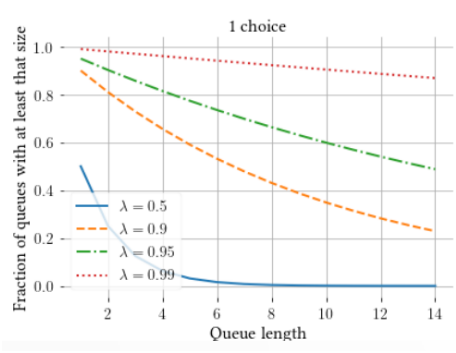
- Servers: n = 1000
- Max simulation time (T): 1000
- Queues sampled (d): 1, 2, 5, 10
- Arrival rate (lambd): 0.5, 0.9, 0.95, 0.99
- Service rate (mu): 1

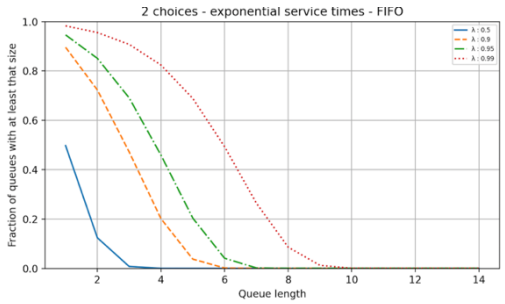
Running the simulation

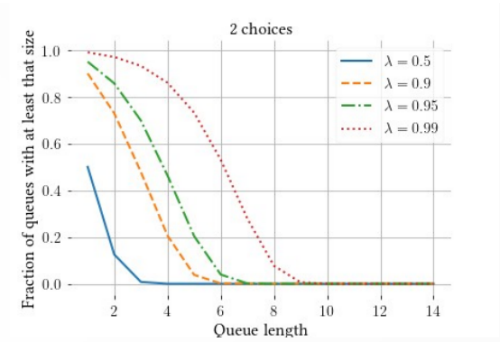
Use the following command to run the simulation with the exponential distribution:
python queue_sim.py

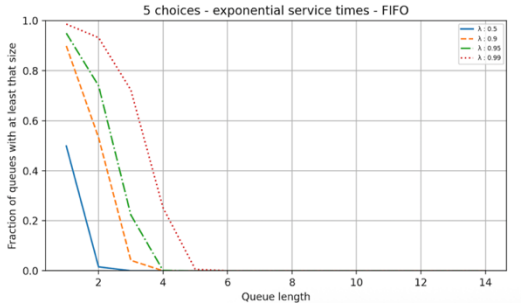
Results:

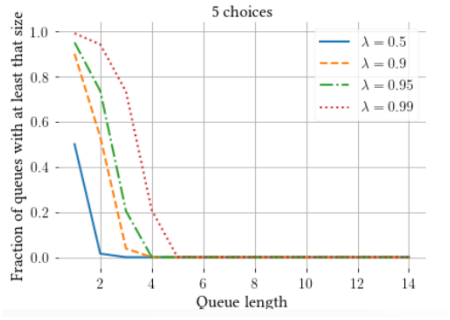


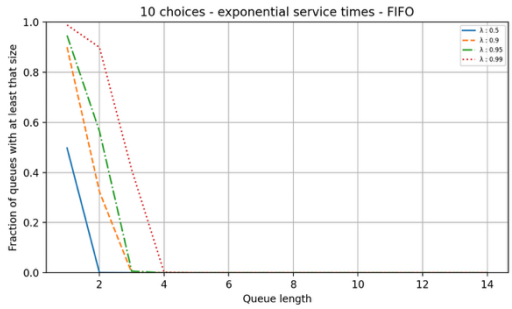


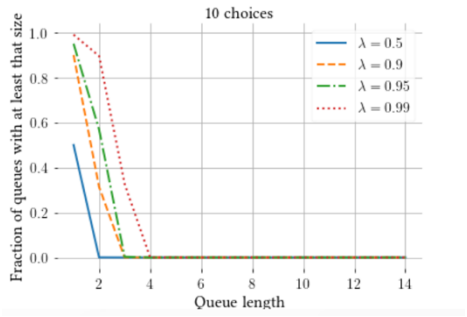












Conclusion:

Some ideas for the comparasion between different runnings taken from ChatGPT.

1. Effect of sampled queues on waiting time: increasing the number of sampled queues reduces average waiting time.
2. Greater impact at higher arrival rates: the benefit of multiple choices becomes more visible as the system load increases.
3. Less improvements with higher d: while increasing d improves performance, the rate of improvement decreases.

(See Appendix A for numerical examples.)

Weibull distribution FIFO

The Weibull distribution is a probability distribution that can model various scenarios through its shape parameter. In our queue simulation, we use Weibull to model both job arrival times and service times. This is more realistic than using exponential distributions because jobs often have varying sizes and processing times.

```
self.arrival_gen = weibull_generator(shape, 1/self.arrival_rate)
self.service_gen = weibull_generator(shape, 1/mu)
```

Running the simulation

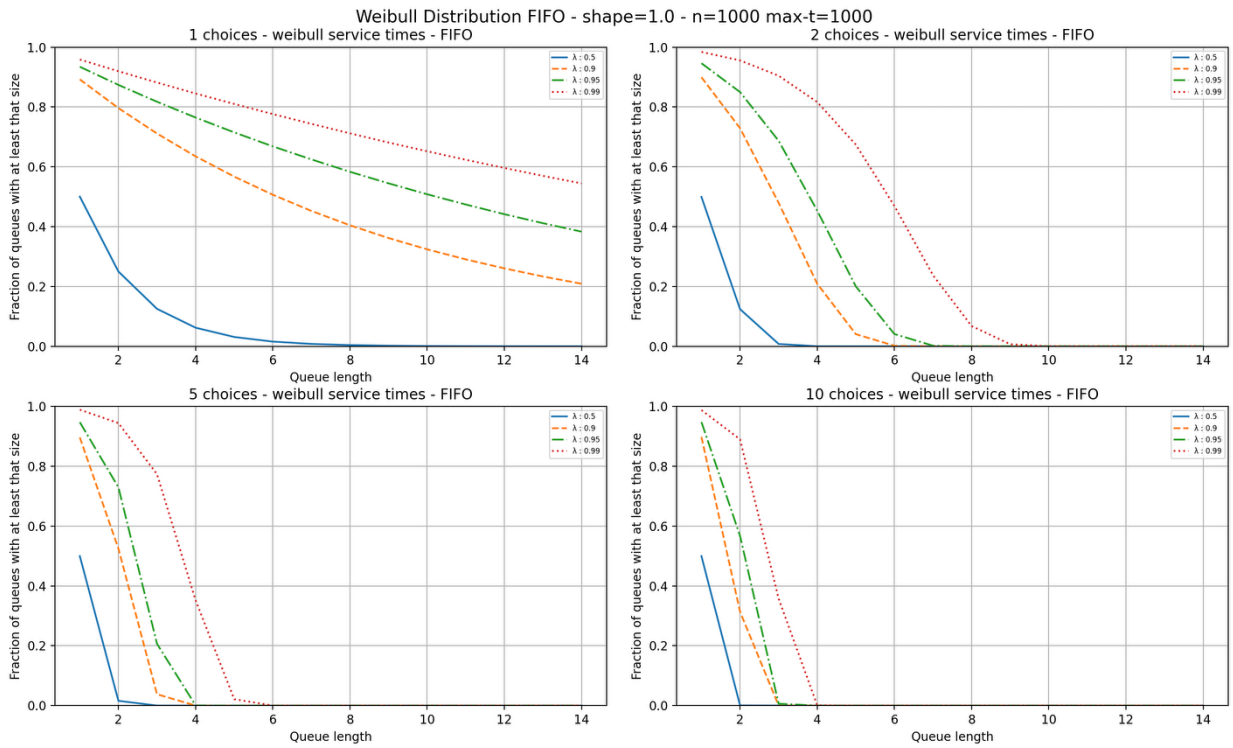
Use the following command to run the simulation with the weibull distribution and FIFO policy:

python queue_sim.py --distribution weibull --shape 1

exponential (Shape = 1):

This simulation uses Weibull distribution with shape=1 for both arrival times and service times.

Results:

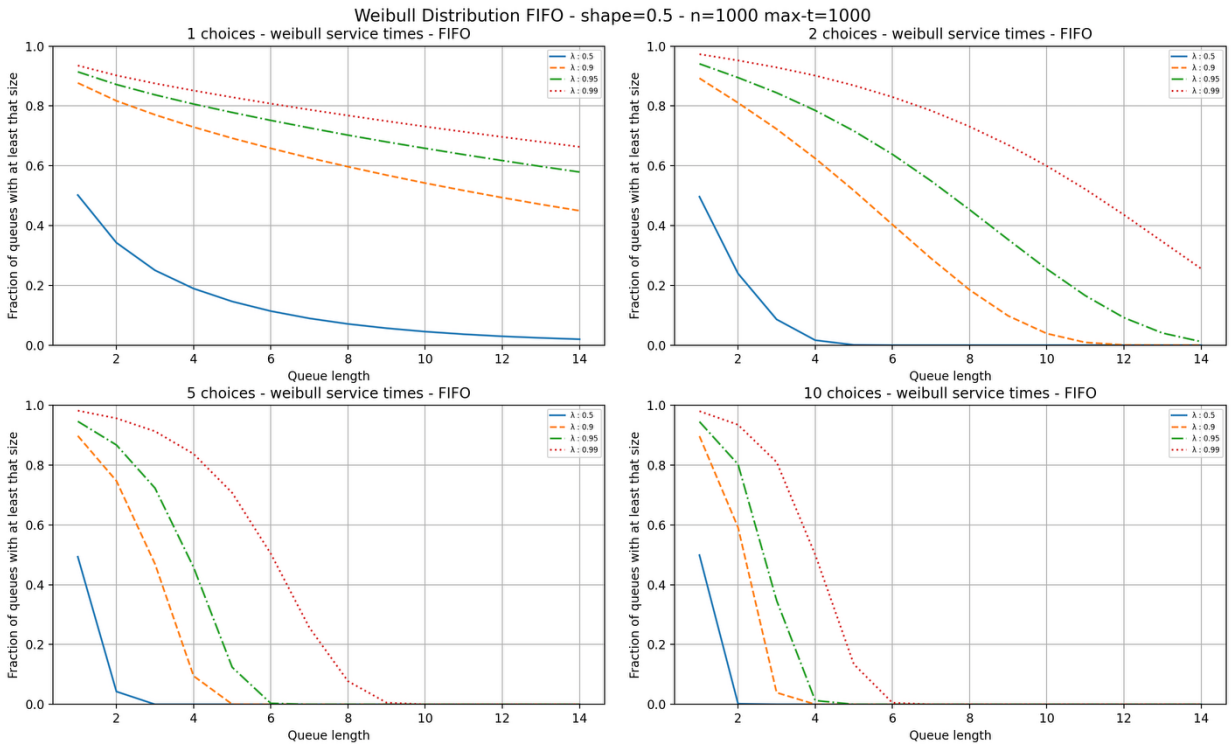


Conclusion:
Using a Weibull distribution with shape = 1 gives results similar to the exponential distribution. This shows that the simulator works correctly and matches what we expect from theory.

heavy-tailed (Shape < 1):

The simulation uses Weibull distribution with shape=0.5 as sample, which creates a heavy-tailed distribution where there are many small jobs and few very large jobs. This is different from the exponential distribution (shape=1) we saw earlier.

Results:



Conclusion:

Some ideas for the comparasion between different runnings taken from ChatGPT.

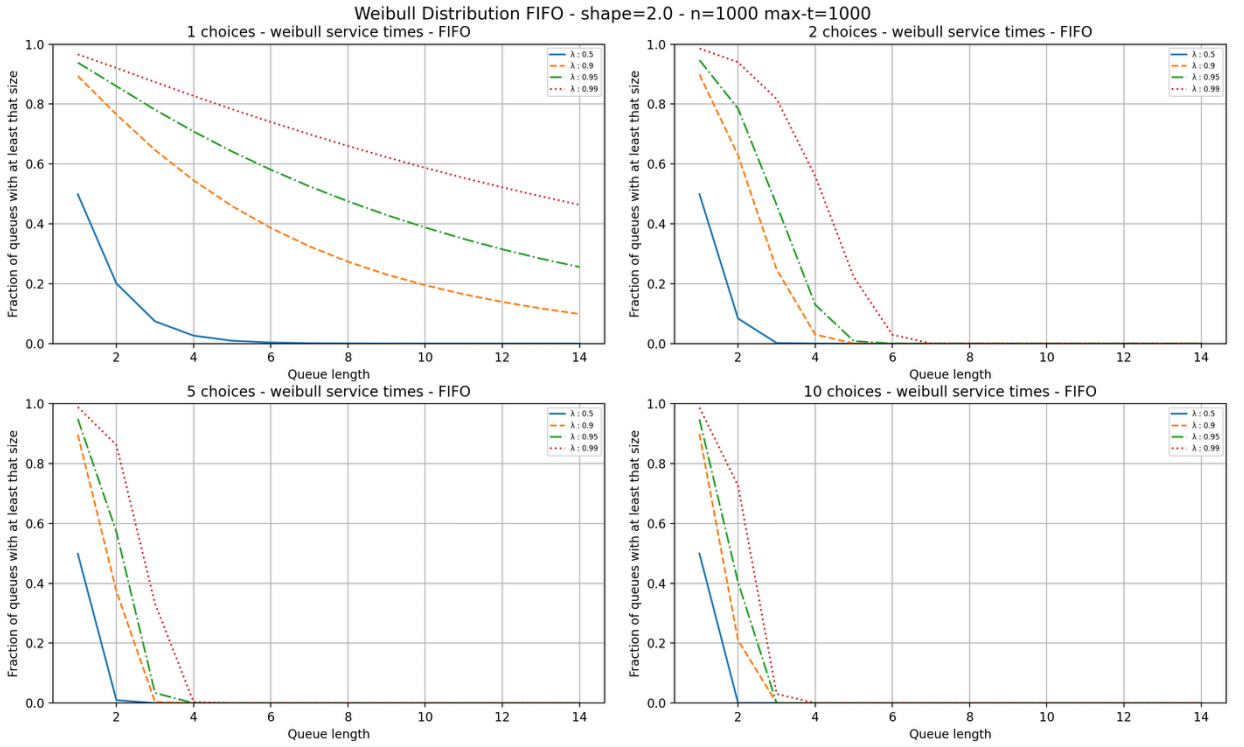
1. Longer waiting times compared to exponential (shape = 1).
2. Multiple choices (d) are more effective compared to exponential (shape = 1)

(See Appendix A for numerical examples.)

uniform (Shape > 1):

The simulation uses Weibull distribution with shape=2.0 as sample, which creates a more uniform distribution.

Results:



Conclusion:

Some ideas for the comparasion between different runnings taken from ChatGPT.

1. System performance improvments: shorter wait times compared to other shapes.
2. Impact of increasing multiple choices on wait times: increasing the number of queues sampled (d) reduces wait times but less visible as it grows.
3. Impact of high load: better performance at high arrival rates.

(See Appendix A for numerical examples.)

What can we understand?

- Shape=1.0 (exponential) gives balanced performance that fits well with theoretical models.
- Shape=0.5 (heavy-tailed) has the longest wait times, but it also benefits the most from adding more sampled queues, so it's important to make multiple choices.
- Shape=2.0 (uniform) gives the best and most consistent performance with less need to sample a lot of queues.

Weibull distribution LIFO

LIFO is a queue policy that serves the most recent job first and can stop jobs that are already running.

How LIFO Works in the Simulation:

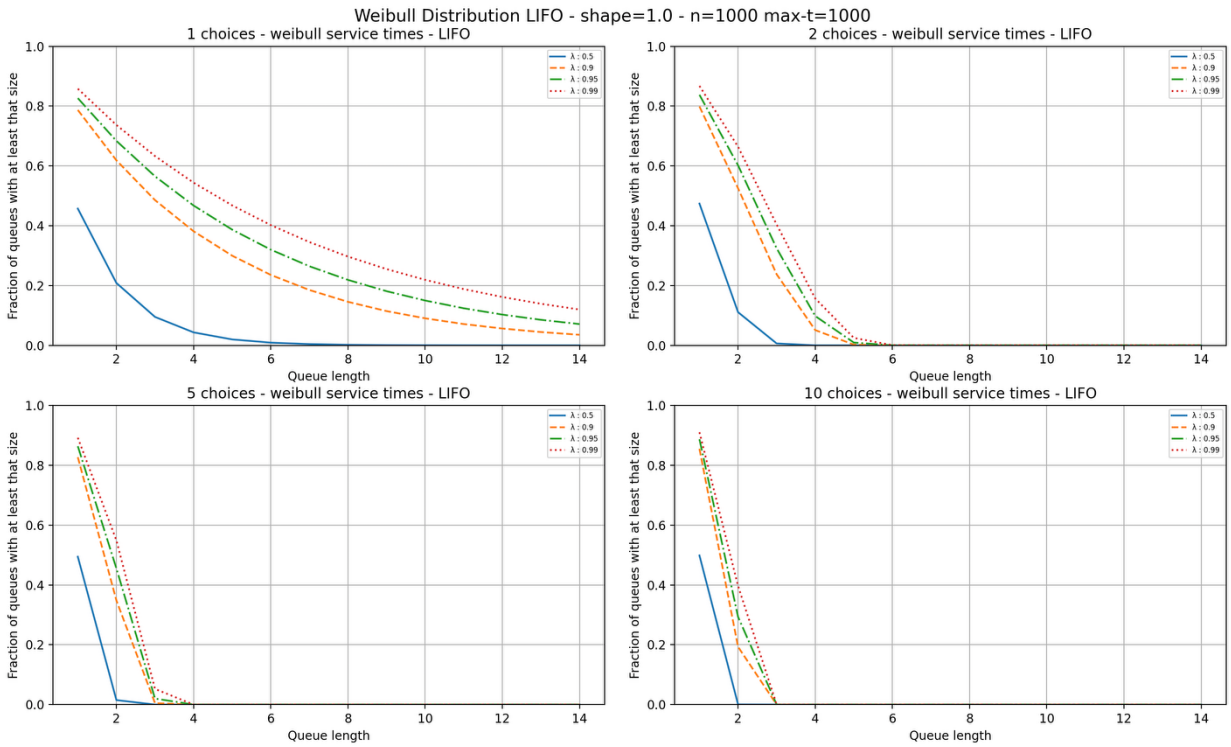
1. Job Arrival: When a new job shows up, it looks at a few random queues and picks the one with the fewest jobs waiting.
2. Preemption: If the chosen queue is already busy, the job that's working gets paused. The paused job isn't thrown away. it's put at the front of the line, so it will be the next one to run after the new job is done (or if it gets paused again).
3. Immediate Service: The new job starts working right away, taking over the server.
4. Tracking Leftover Time: If a job is paused, the program remembers how much work it still needs to finish. When that job gets another turn, it just finishes what's left.
5. Queue Structure: Each queue is a special kind of list (called a deque) that makes it quick and easy to add or remove jobs from either end. In LIFO mode, paused jobs go to the front, so the newest jobs always get to run first.

Running the simulation

Use the following command to run the simulation with the weibull distribution and LIFO policy:

python queue_sim.py --distribution weibull –queue-policy lifo --shape 1

Results:



Conclusion:
Some ideas for the comparasion between different runnings taken from ChatGPT.

1. LIFO has shorter average waiting times compared to FIFO across all shape parameters.
2. Increasing the number of queues sampled (d) improved performance for both policies, but it was more visible in LIFO.

Appendix

M/M/n queue supermarket model:

1. Effect of sampled queues on waiting time: increasing the number of sampled queues reduces average waiting time. At λ = 0.99, the average waiting time decreases from 20.89 (d=1) to 2.18 (d=10).
2. Greater impact at higher arrival rates: the benefit of multiple choices becomes more visible as the system load increases. At λ = 0.99, increasing d from 1 to 2 reduces waiting time from 20.89 to 5.30.
3. Less improvements with higher d: while increasing d improves performance, the rate of improvement decreases. At λ = 0.99, increasing d from 5 to 10 only reduces waiting time from 2.91 to 2.18.

Weibull distribution heavy-tailed (Shape < 1):

1. Longer waiting times compared to exponential (shape = 1).
   At λ = 0.90 and d = 1, wait time is 20.79 vs. 9.08 in shape = 1 (129% increase).
2. Multiple choices (d) are more effective compared to exponential (shape = 1)
   At λ = 0.99, increasing d from 1 to 10 reduces W from 34.27 to 3.27 (over 90% drop).

Weibull distribution uniform (Shape > 1):

1. System performance improvments: shorter wait times compared to other shapes.
At λ = 0.90, d = 1, wait time is 6.43, much less than 9.08 (shape=1) and 20.79 (shape=0.5).

2. Impact of increasing multiple choices on wait times: increasing the number of queues sampled (d) reduces wait times but less visible as it grows.
At λ = 0.99, wait time drops from 16.67 (d=1) to 1.77 (d=10), with only a 17% improvement from d=5 to d=10.

4. Impact of high load: better performance at high arrival rates.
At λ = 0.99 and d = 10, wait time is 1.77 (shape=2.0), compared to 2.27 (shape=1) and 3.27 (shape=0.5).