How to trade off server utilization and tail latency

SRECon 2019, Singapore

Julius Plenz <plenz@google.com>

Who am I?

- SRE at Google Sydney
- I currently work on Google Cloud Datastore and Firestore
- Worked on other Google-internal storage systems before
- I try to make visualizations to understand things!

- If you find any of this interesting, come talk to me afterwards
- Slide deck is published on the conference web site including speaker notes with additional information

Disclaimer

Queuing Theory is its own subfield in Computer Science / Statistics.

I'll gloss over many subtleties and won't use any formulas.

Instead, we'll look at graphs of numerical simulations only.

Queues are everywhere, let's talk about them!

Image: Google Datacenter Gallery



Queues are everywhere:

- Web server
- Network card
- Hard drive
- Memcache
- Database
- Restaurants
- etc etc

Definitions for this talk

Server: A binary that handles requests.

Example: web server, database server, memcache server

Utilization: Fraction of time that the server is busy in the bottleneck dimension, usually CPU.

Example: Over a 60s interval, the CPU is busy 30s. The utilization is 50%.

Tail latency: 99%ile latency of a set of requests.

Example: Out of 100 requests, the second slowest exhibits the tail latency.

The simplest queue



Queue time

Time spent waiting in the queue

+

Service time

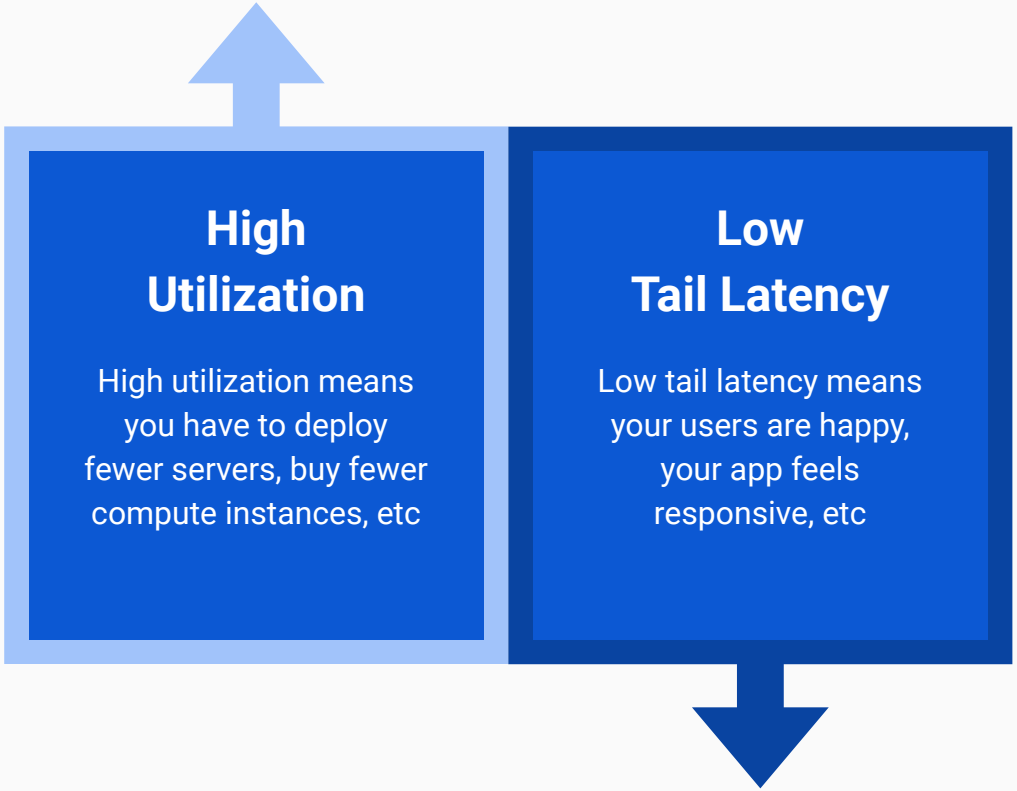
Time spent servicing the request (no waiting)

=

Request latency

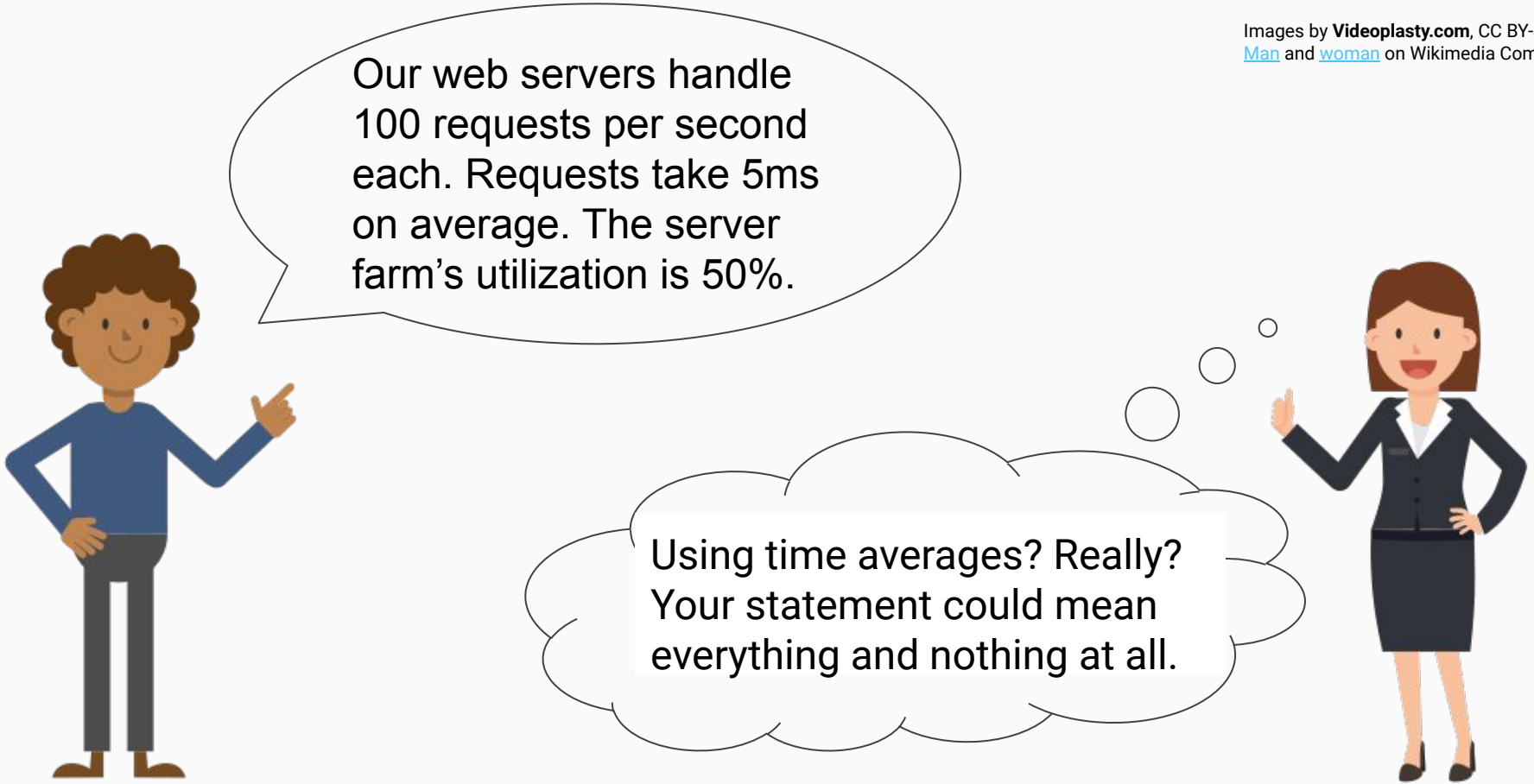
User-perceived time: how long it takes from sending a request to receiving the reply

We want both of these, but they are at odds

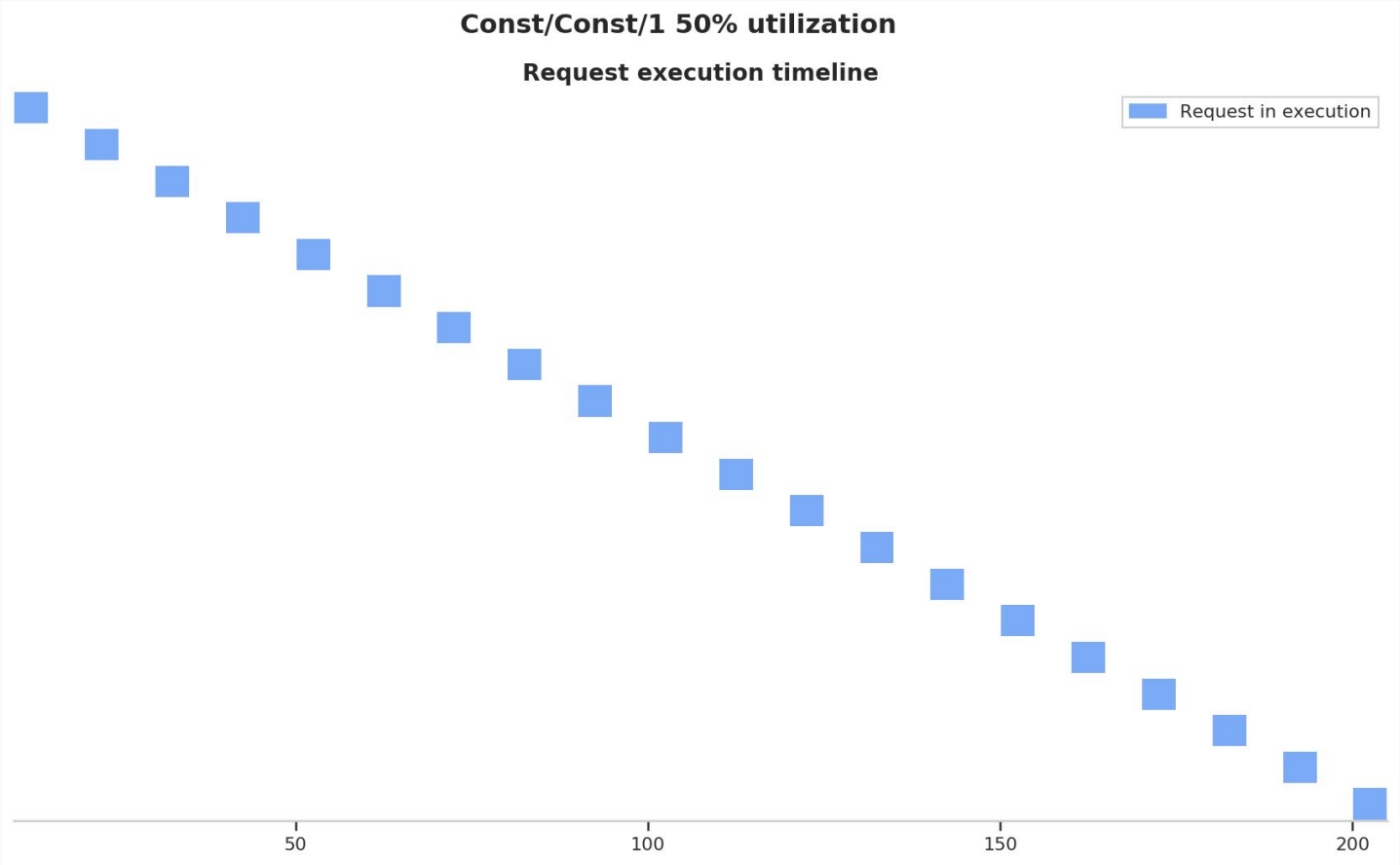


Talking about utilization is often not precise

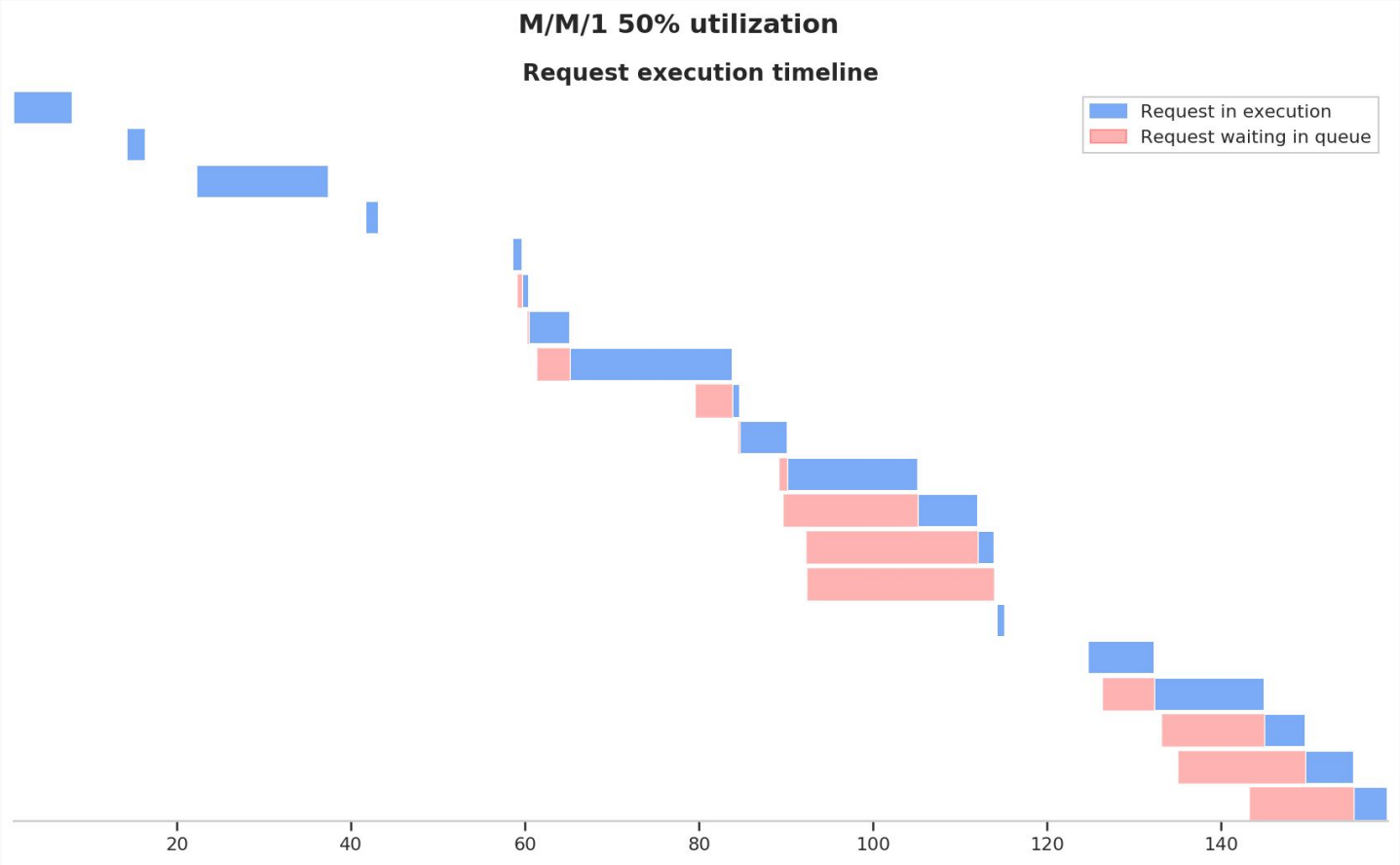
Images by **Videoplasty.com**, CC BY-SA 4.0
Man and woman on Wikimedia Commons



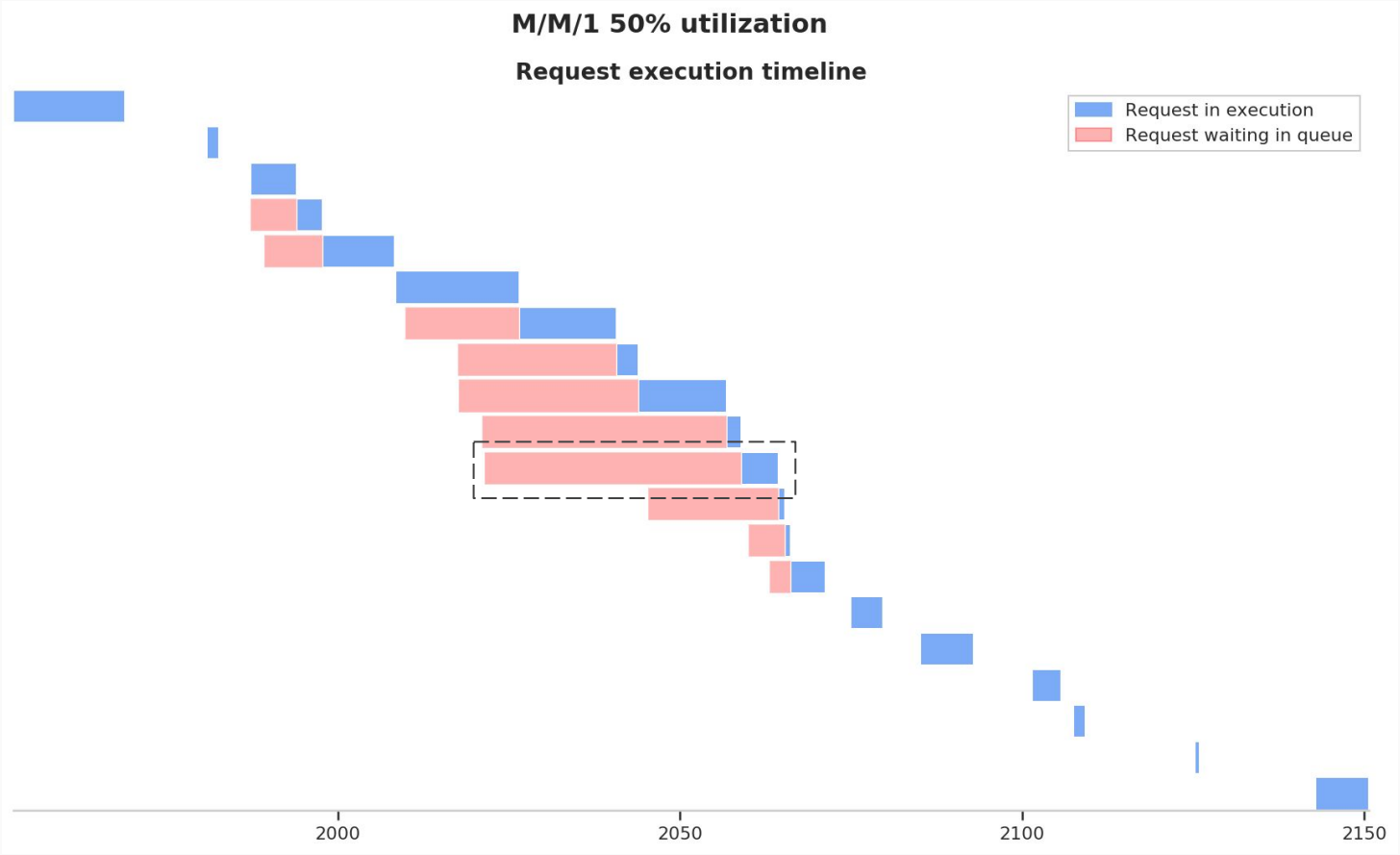
This picture of “50% utilization” is very likely misleading



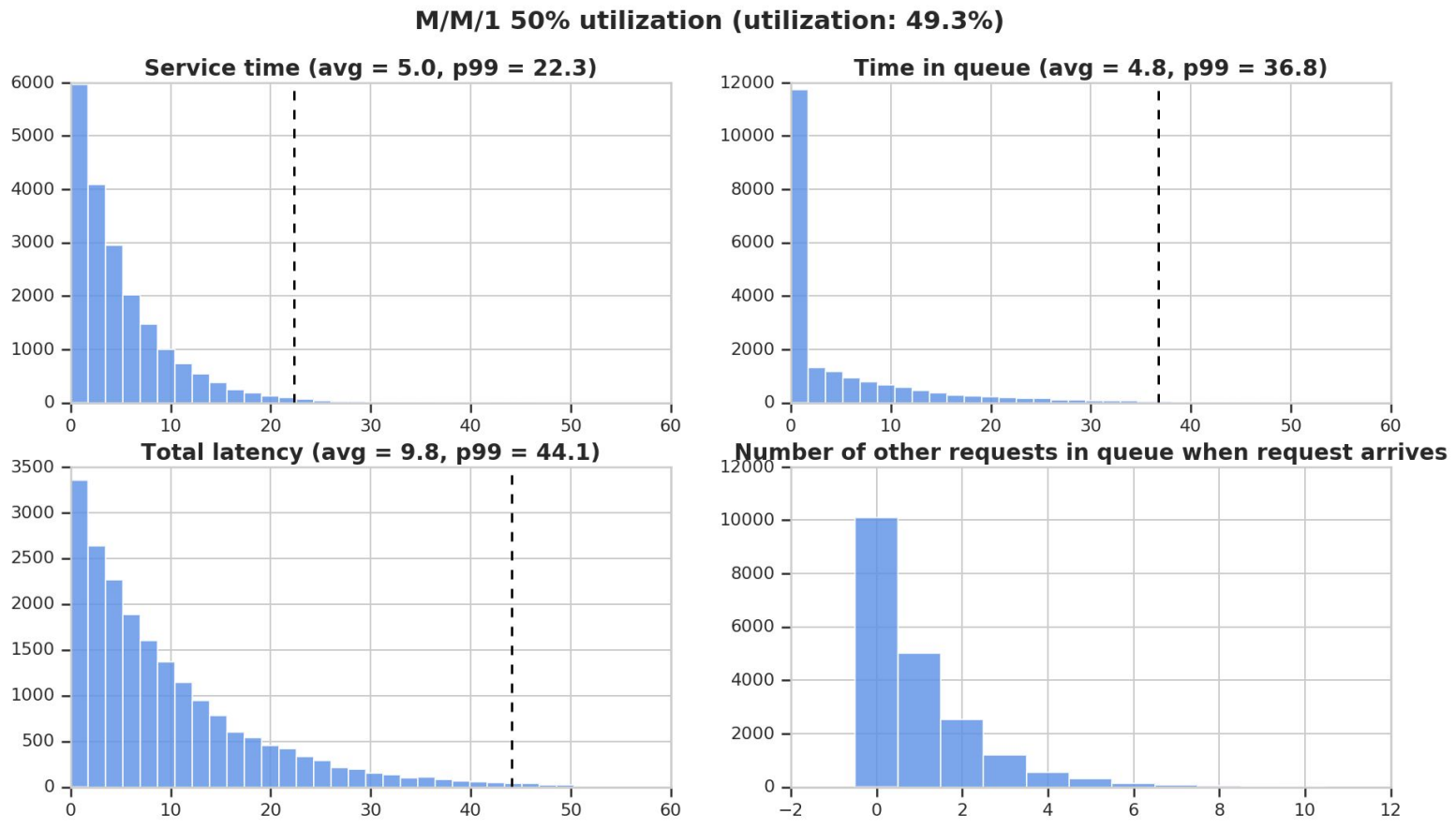
This image is much closer to reality: Variable interarrival and service times



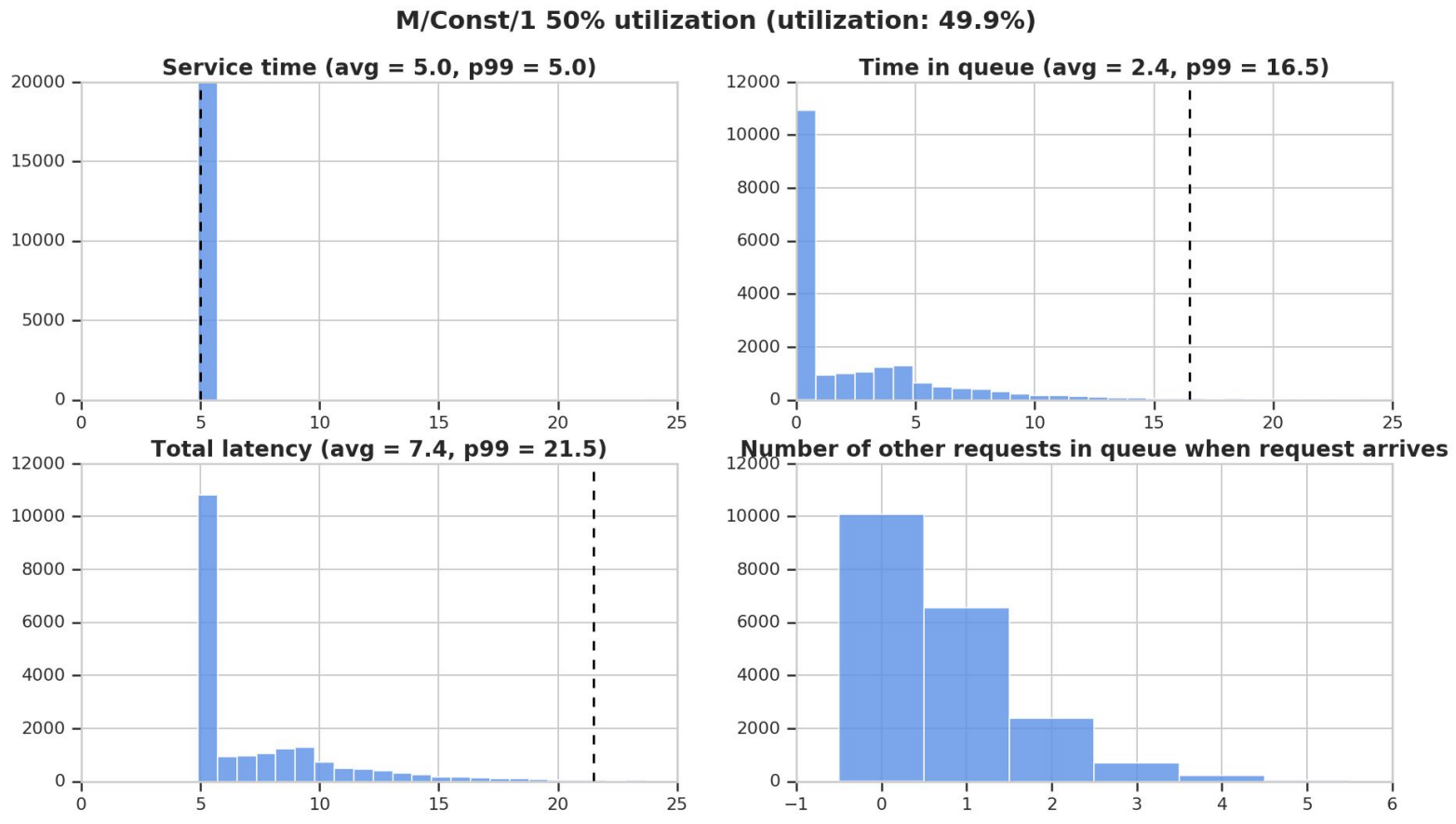
This is a snapshot of a point in time when a request exhibited the 99%ile latency



We can relate the distributions of queue and service time, and total request latency



Similar example: With constant service time, queueing effects actually dominate



Takeaway #1:

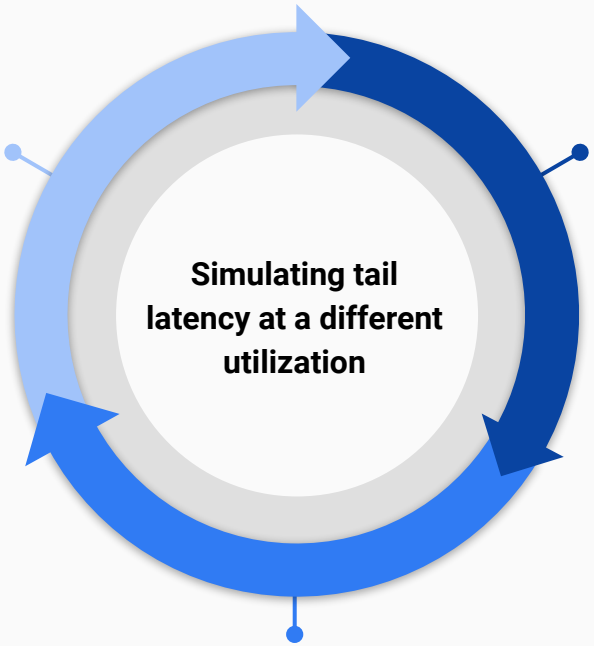
A significant amount of tail latency is due to queueing effects

From textbook examples to real data – your service here!



(Lots more details in the speaker notes.)

Simulating tail latency at different utilizations



Step 1

Compute desired utilization over current utilization (e.g. +10% → *x* = 1.1)

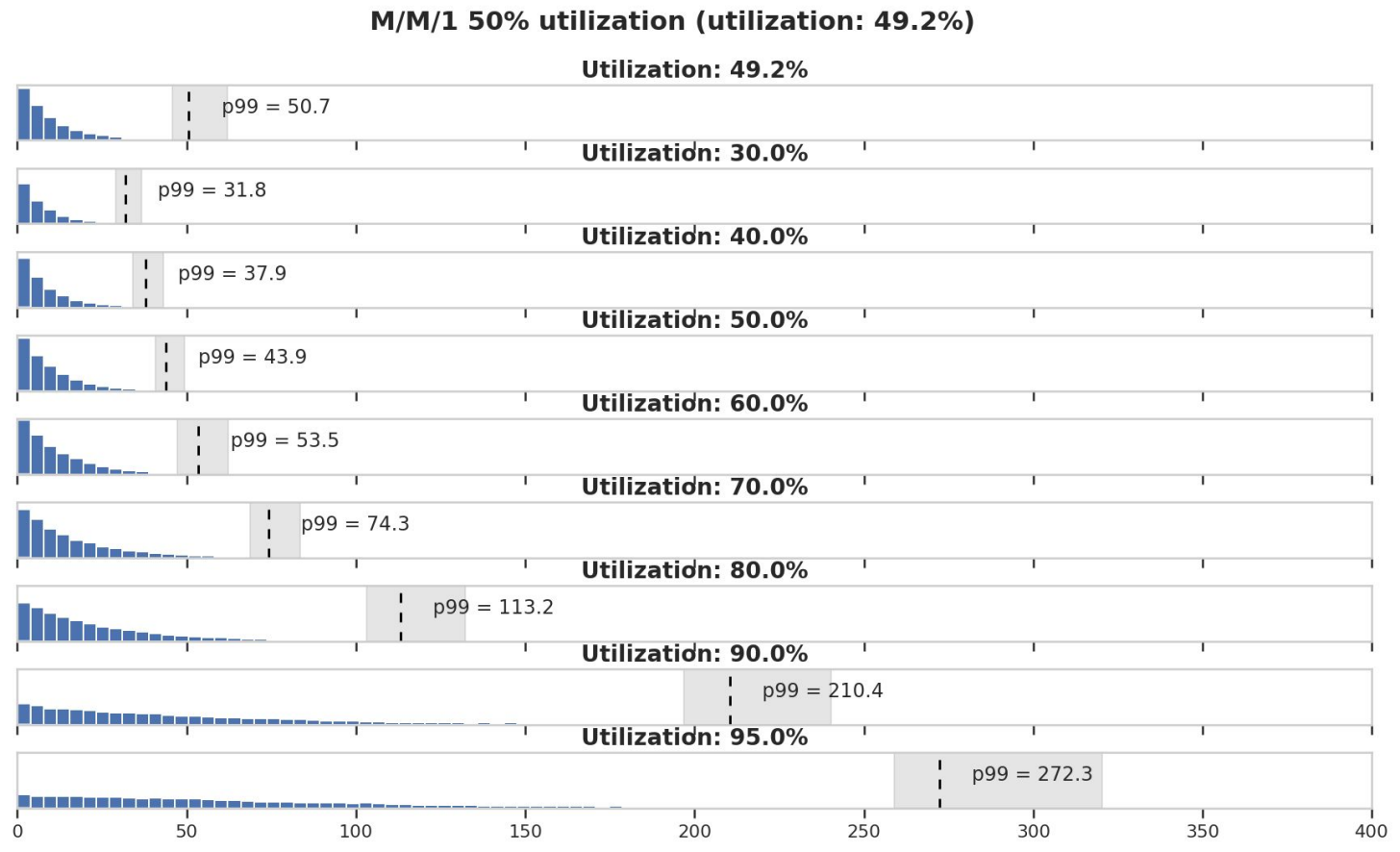
Step 2

Draw samples from the interarrival time data set. Divide by *x* to space the arrivals appropriately (closer together / farther apart).

Step 3

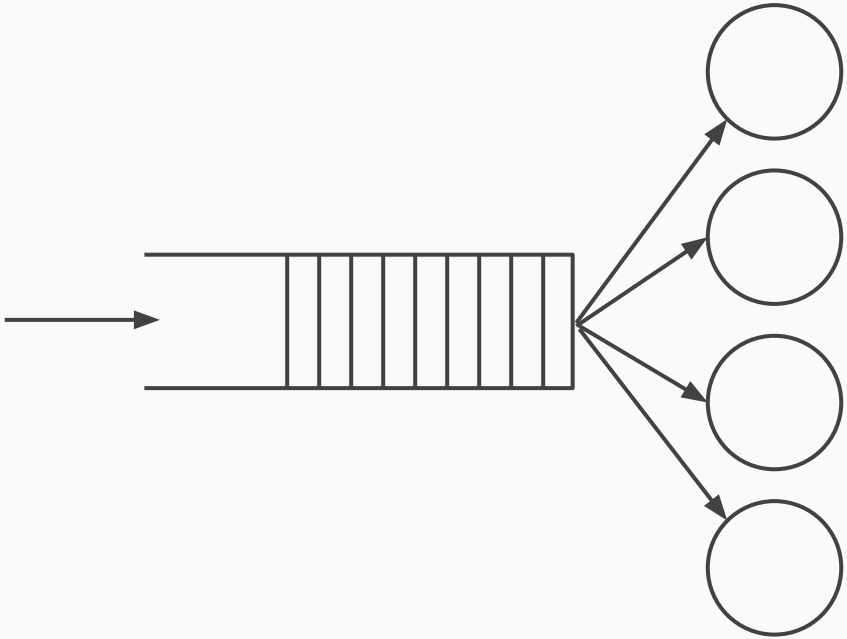
Keep the service times unchanged. Run requests through server with new interarrival times.

The dire result

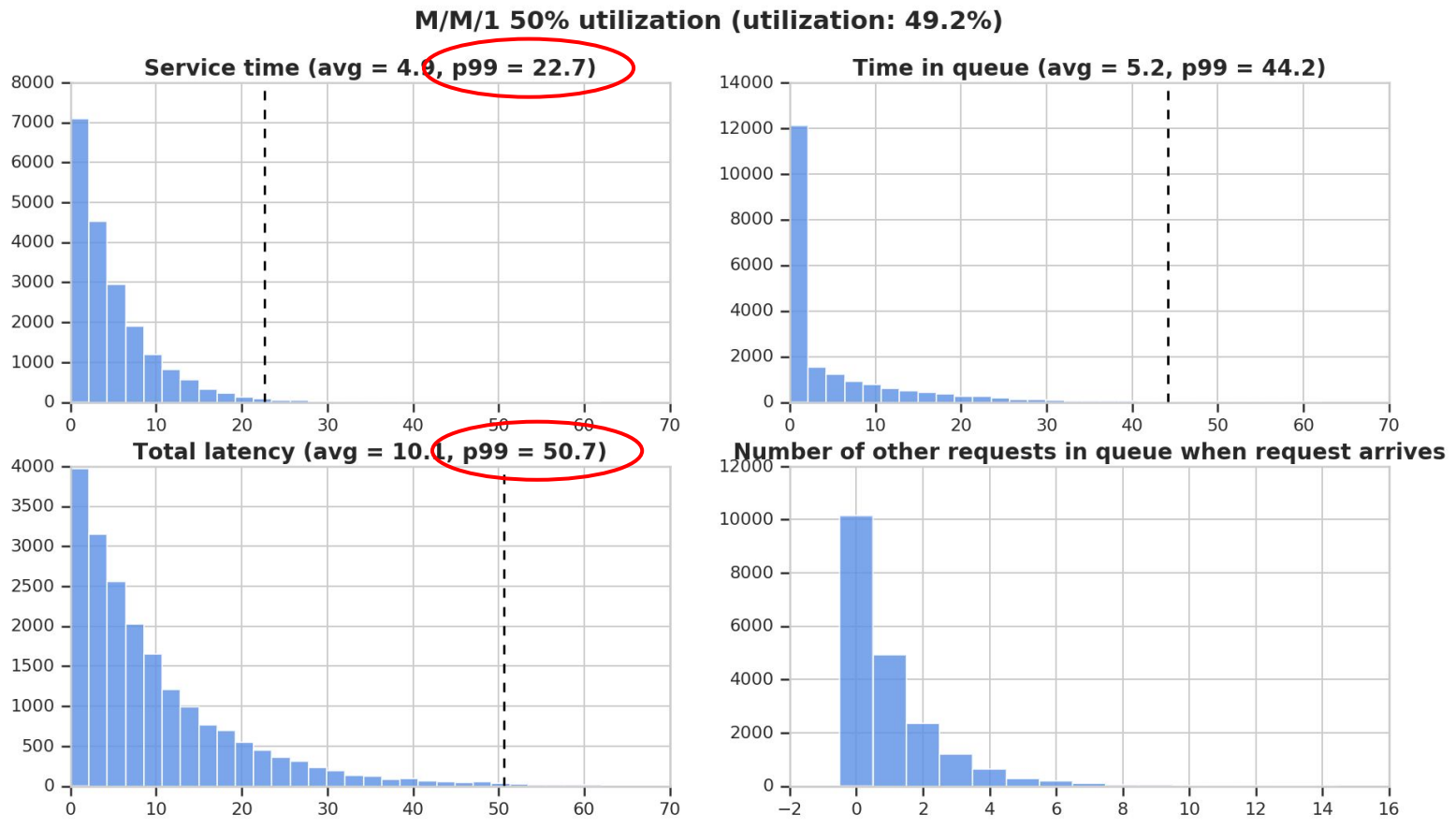


Takeaway #2:
Queueing effects increase superlinearly with utilization

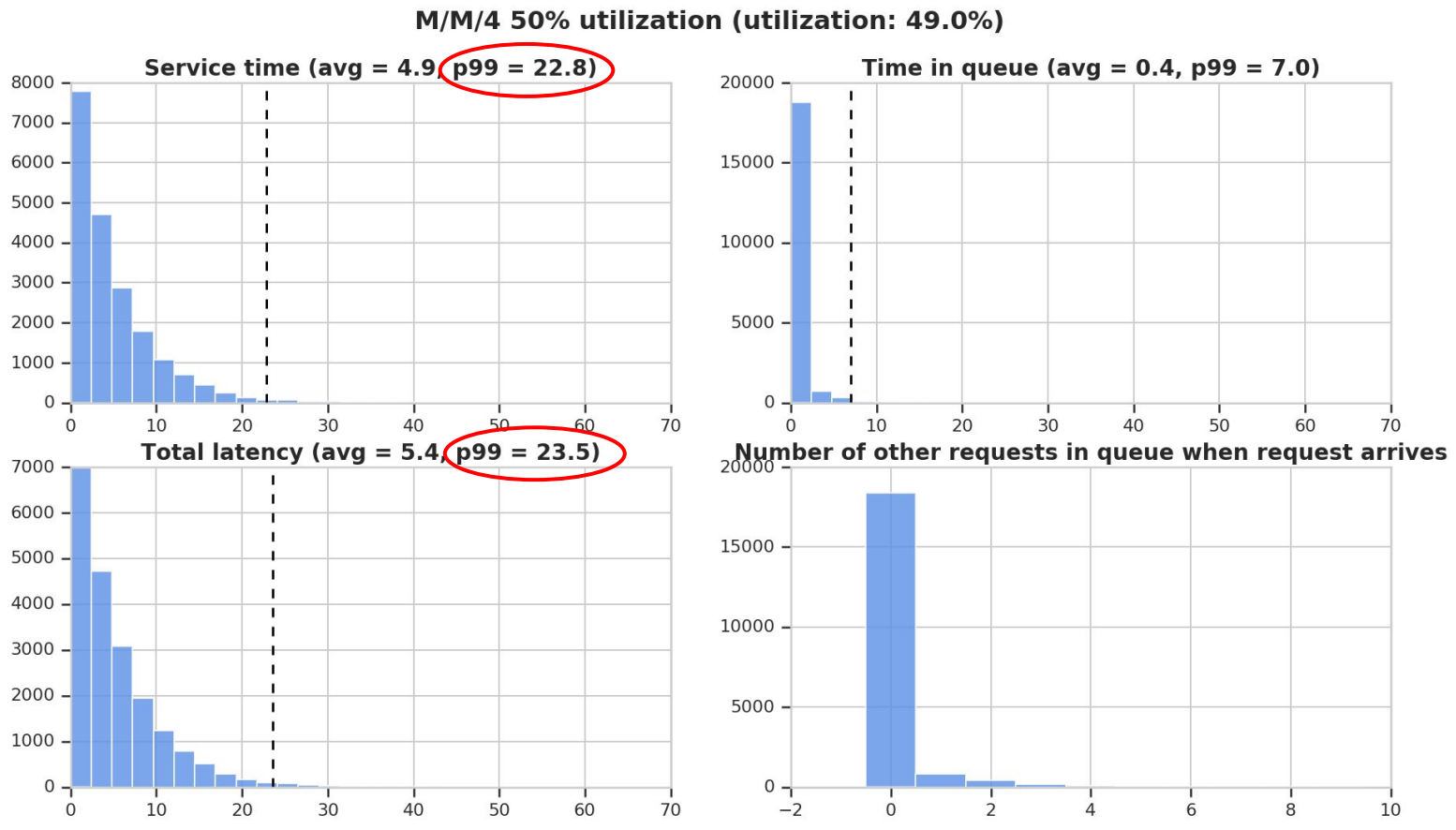
A single queue, but four cores (your system probably looks more like this)



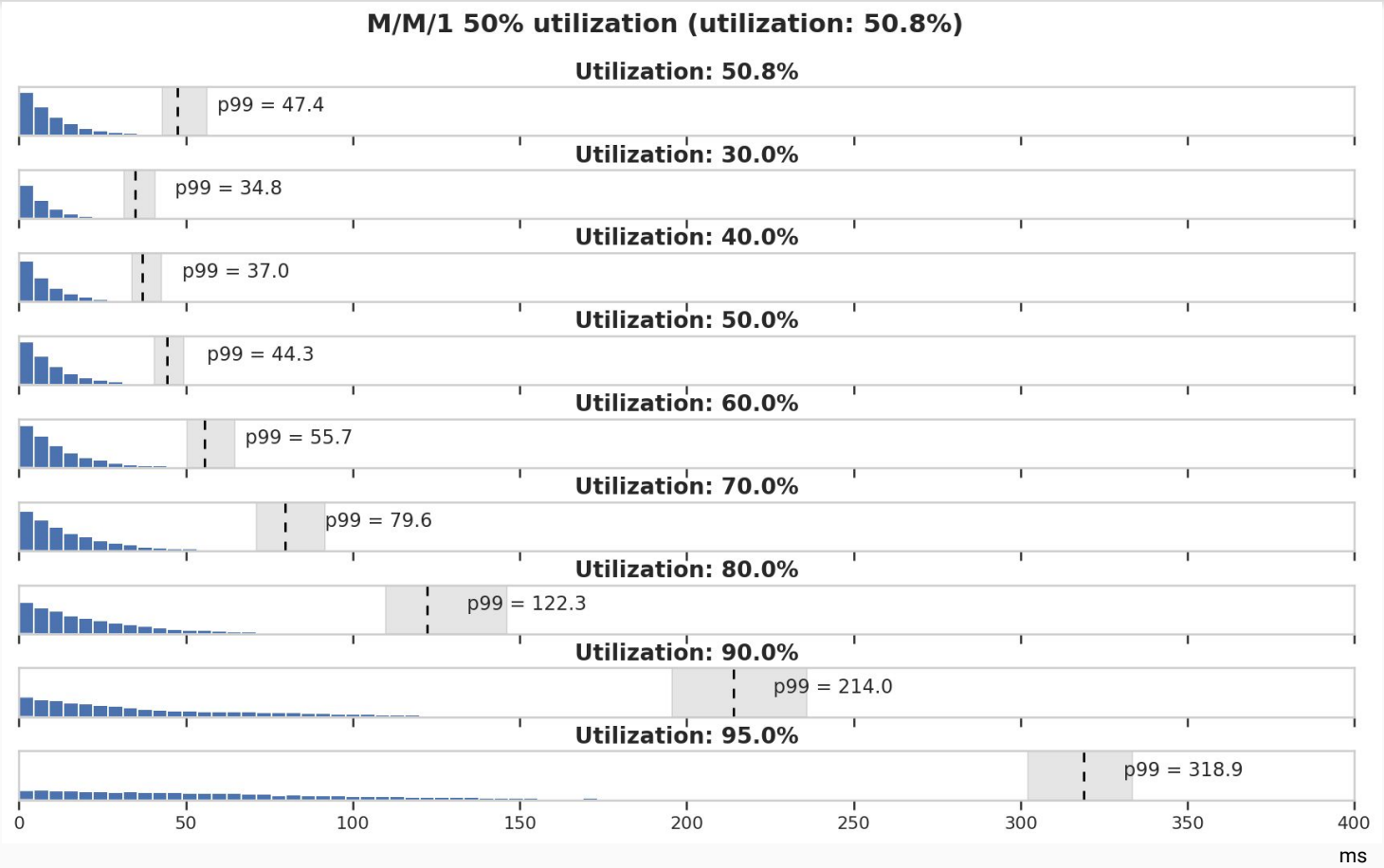
Previously: 50% utilization with a single core



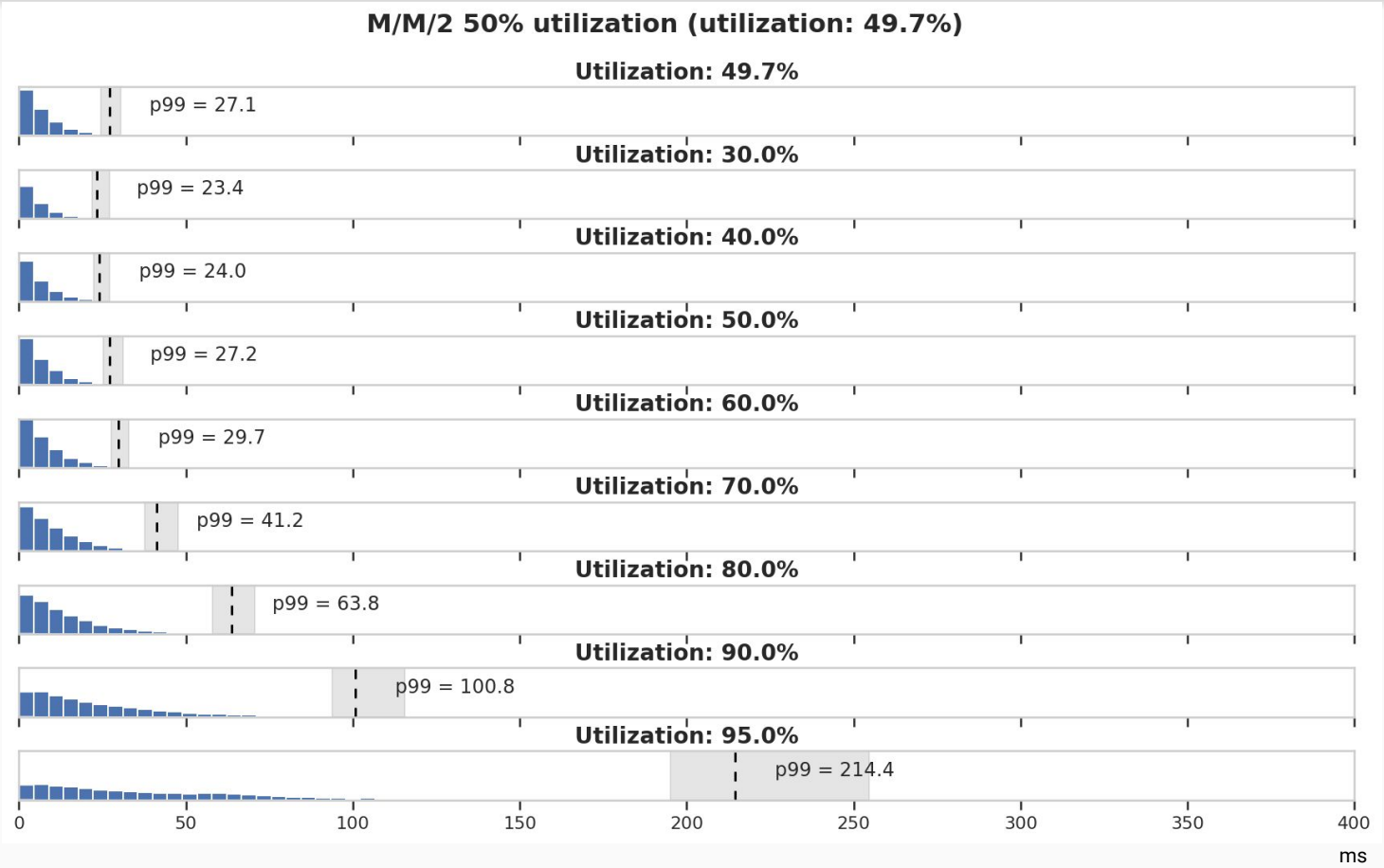
Now: 50% utilization with four cores: Barely any queueing effects



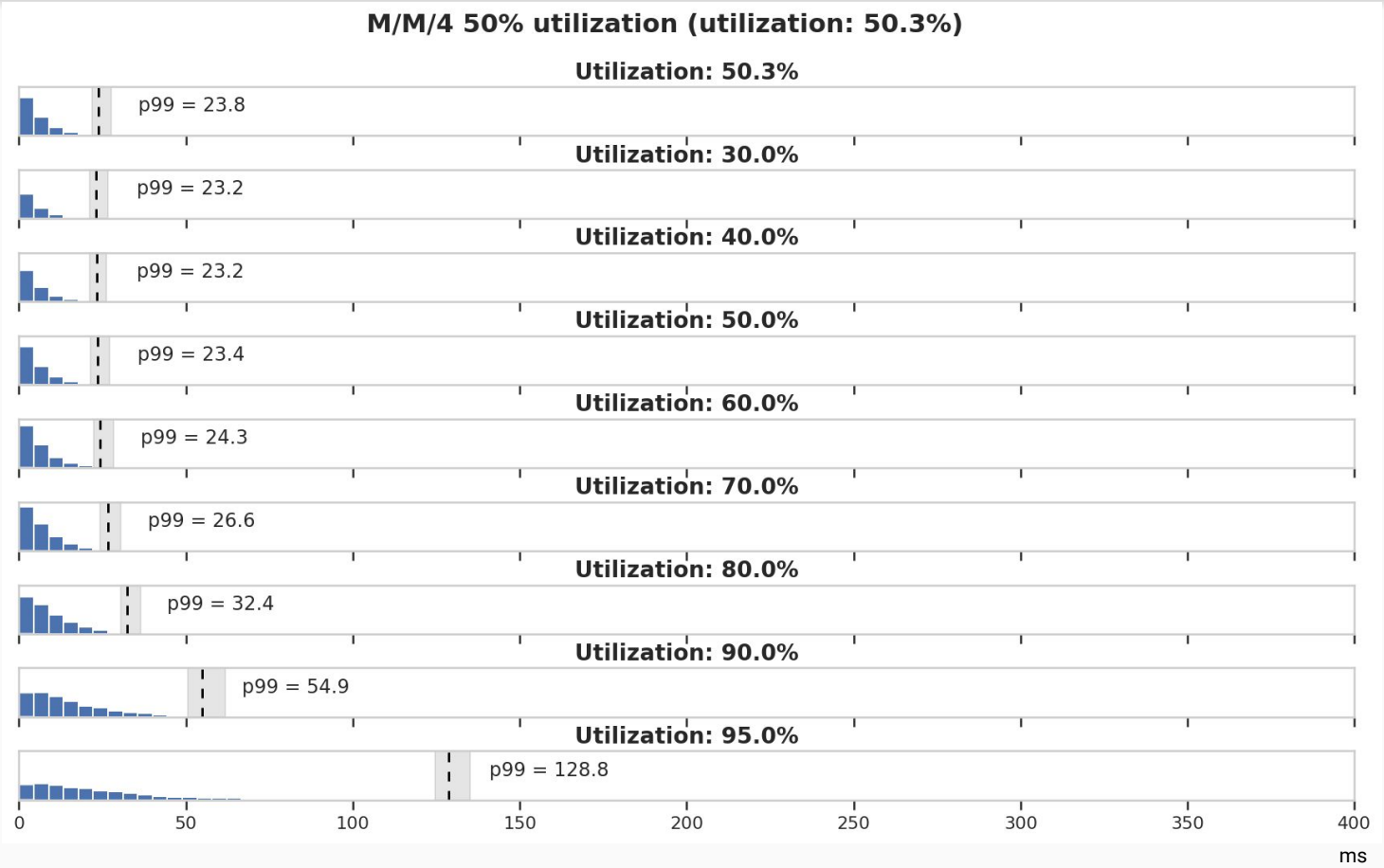
Different utilizations with $K = 1$



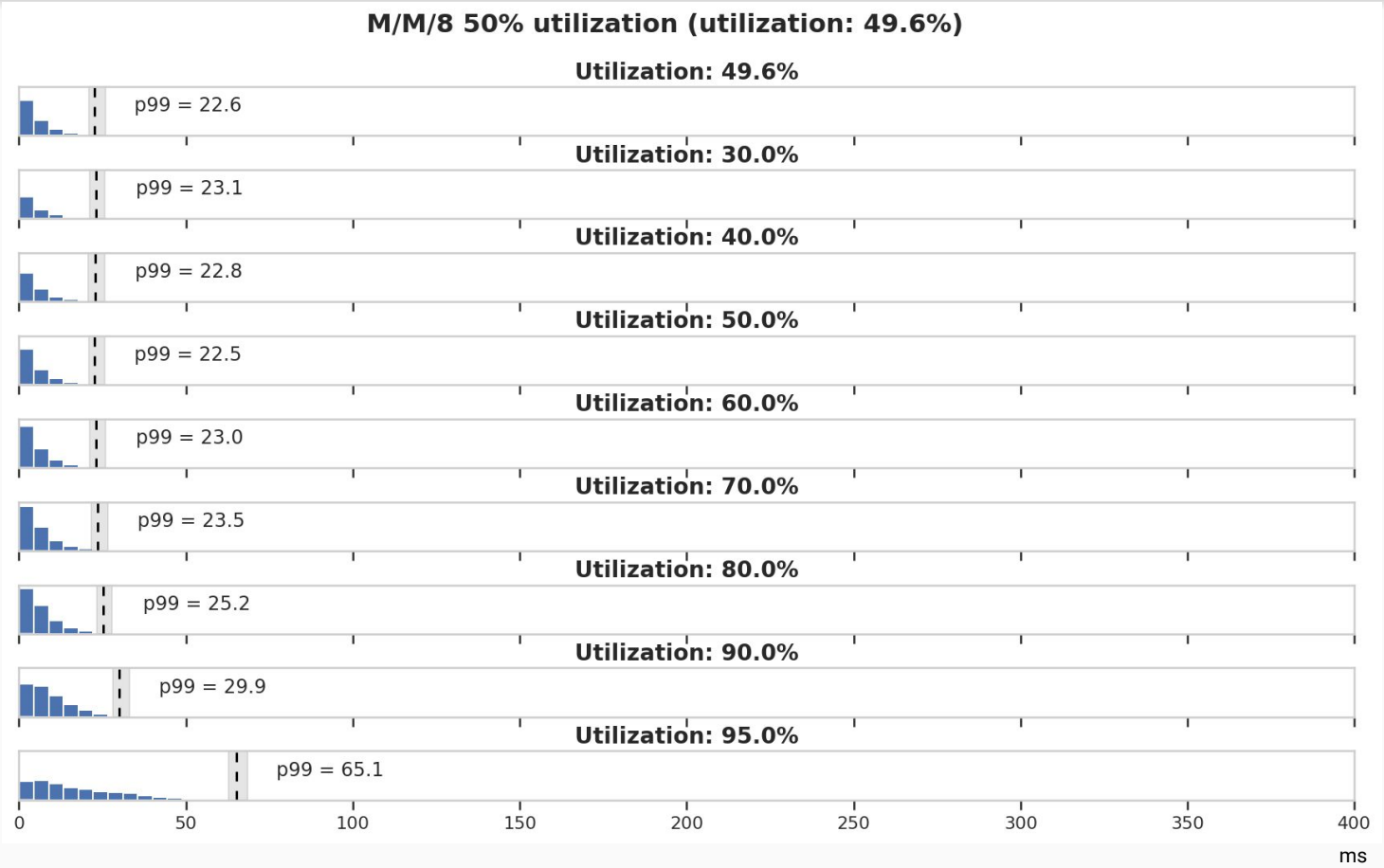
Different utilizations with $K = 2$



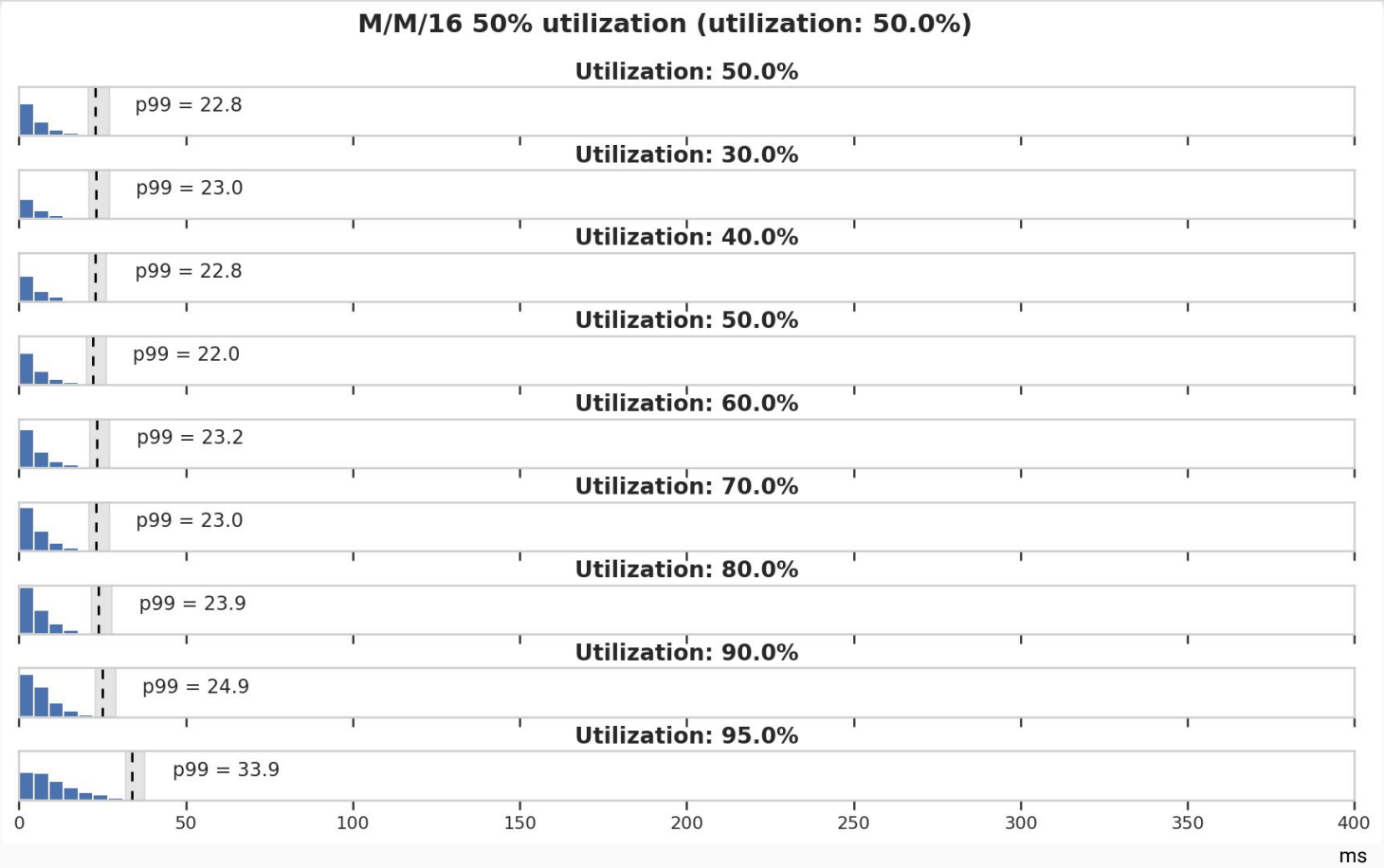
Different utilizations with $K = 4$



Different utilizations with $K = 8$



Different utilizations with $K = 16$

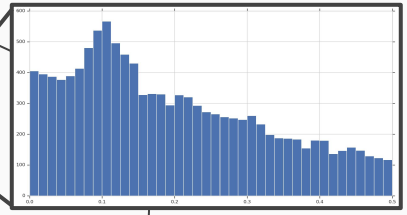


Takeaway #3:

Increasing the parallelism in your server will pull in the tail latency(*)

How our conversations about utilization *should* go

We run our web servers at **50% utilization**, with this request interarrival time. To save money, I want to target **65% utilization**. What do you think?





Sounds great. But let's invest a few weeks into rewriting some of the core data structures to be lockless, and then run the service on **larger** 16-vCPU nodes instead, and at **80% utilization**.

That way, we save even more money and **our users remain happy**!



Images by **Videoplasty.com**, CC BY-SA 4.0
Man and woman on Wikimedia Commons

Recap of the three takeaways:

#1 – A significant amount of tail latency can be due to queueing effects.

#2 – Queueing effects increase superlinearly with utilization.

#3 – Increasing the parallelism in your server will pull in the tail latency.

Thank you!

Q&A