TIPS AND TRICKS

WORKING IN DISTRIBUTED SYSTEMS

WHAT IS A DISTRIBUTED SYSTEM?

"APPLICATION THAT EXECUTES A COLLECTION OF PROTOCOLS TO COORDINATE THE ACTIONS OF MULTIPLE PROCESSES ON A NETWORK, SUCH THAT ALL COMPONENTS COOPERATE TOGETHER TO PERFORM A SINGLE OR SMALL SET OF RELATED TASKS."

WHY BUILD A DISTRIBUTED SYSTEM?

"A DISTRIBUTED SYSTEM CAN BE MUCH LARGER AND MORE POWERFUL GIVEN THE COMBINED CAPABILITIES OF THE DISTRIBUTED COMPONENTS, THAN COMBINATIONS OF STAND-ALONE SYSTEMS. BUT IT'S NOT EASY - FOR A DISTRIBUTED SYSTEM TO BE USEFUL, IT MUST BE RELIABLE."

TO BE TRULY RELIABLE, A DISTRIBUTED SYSTEM MUST HAVE THE FOLLOWING CHARACTERISTICS

- **Fault-Tolerant:** It can recover from component failures
- **Highly Available:** It can restore operations, permitting it to resume providing services even when some components have failed.
- **Recoverable:** Failed components can restart themselves and rejoin the system, after the cause of failure has been repaired.
- **Consistent:** The system can coordinate actions by multiple components often in the presence of concurrency and failure.
- **Scalable:** It can operate correctly even as some aspect of the system is scaled to a larger size. We might increase the number of users or servers, or overall load on the system. In a scalable system, this should not have a significant effect.
- **Predictable Performance:** The ability to provide desired responsiveness in a timely manner.
- **Secure**: The system authenticates access to data and services

TIP #1

DESIGN FOR FAILURE

DISTRIBUTED DESIGN PRINCIPLES

DESIGN FOR FAILURE

- Explicitly define failure scenarios and identify how likely each one might occur. Make sure you prepared for the most likely ones.
- Both clients and servers must be able to deal with unresponsive senders/receivers.
- If a process stores information that can't be reconstructed, then problems arise. One possible question is, "Are you now a single point of failure?"

TIP #2

COORDINATION IS VERY HARD

TIP #3

DISTRIBUTED LOCKS

TIP #4

"IT'S SLOW" IS THE HARDEST PROBLEM YOU'LL EVER DEBUG.

DISTRIBUTED DESIGN PRINCIPLES

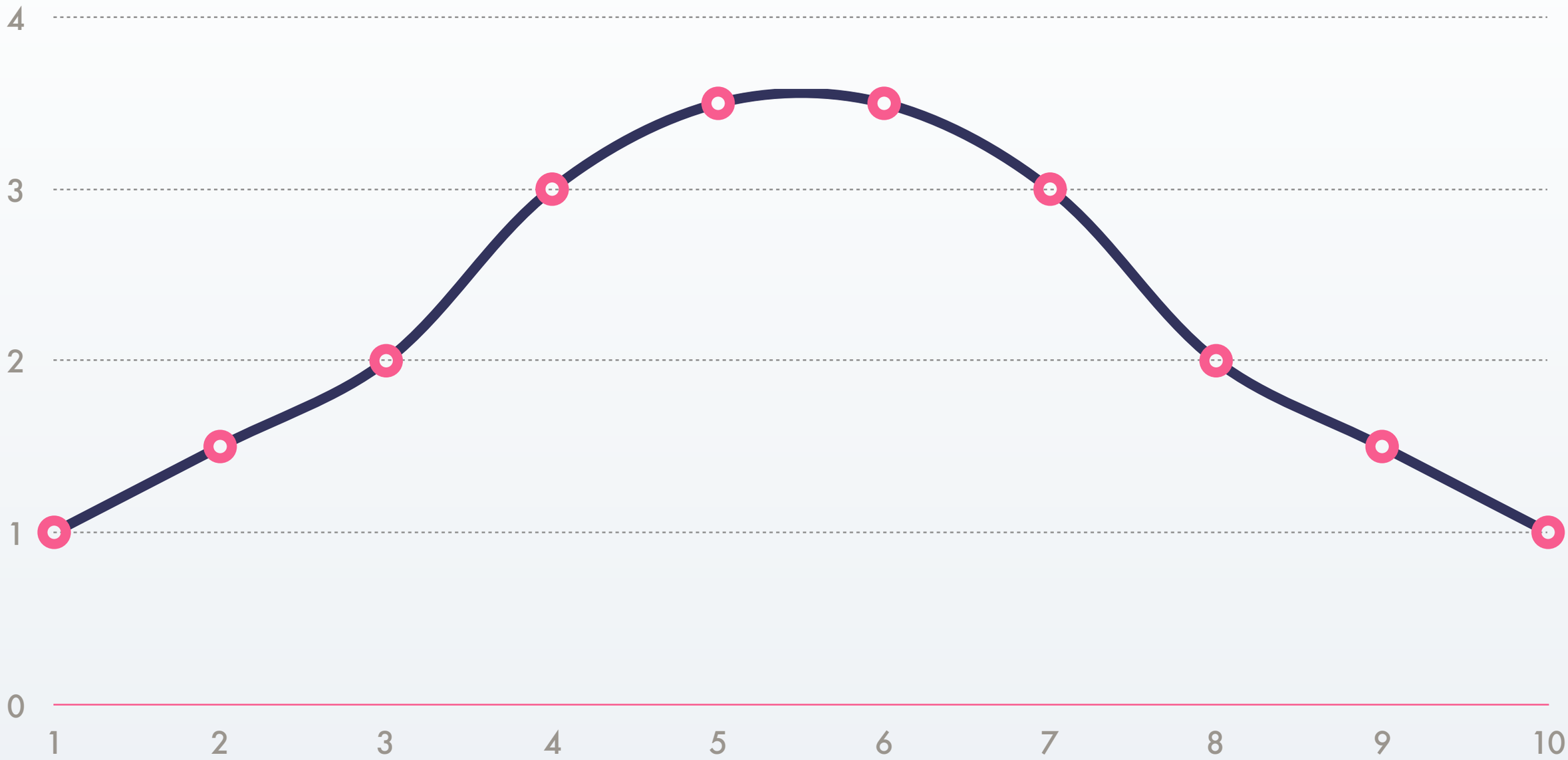
"IT'S SLOW"

- Be sensitive to speed and performance. Take time to determine which parts of your system can have a significant impact on performance: Where are the bottlenecks and why? Devise small tests you can do to evaluate alternatives. Profile and measure to learn more. Talk about these alternatives and your results, and decide on the best solution.

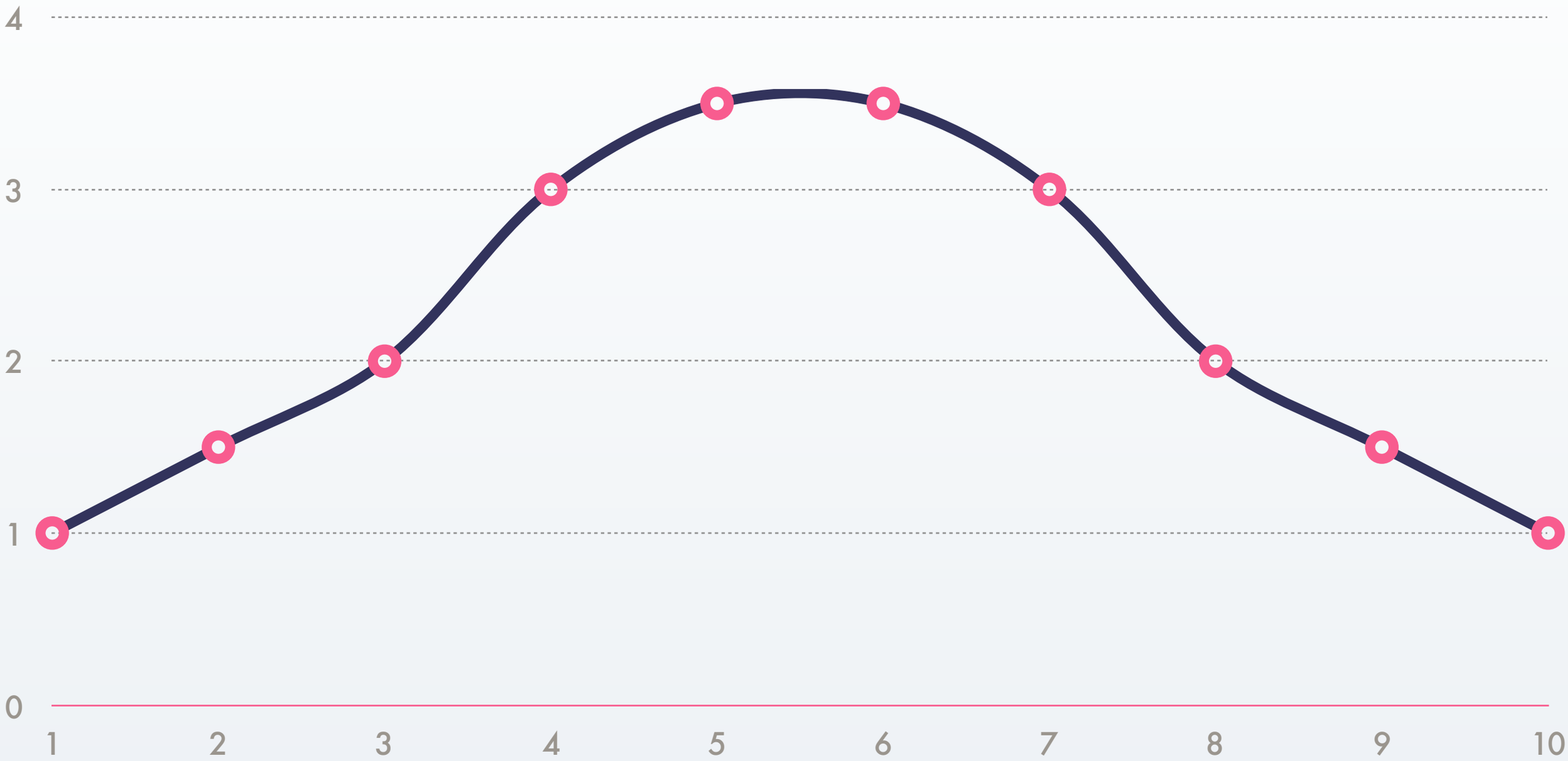
- Latency is the time between initiating a request for data and the beginning of the actual data transfer. Minimizing latency sometimes comes down to a question of whether you should make many little calls/data transfers or one big call/data transfer. The way to make this decision is to experiment. Do small tests to identify the best compromise.

TIP #5

USE PERCENTILES, NOT AVERAGES.



Average Latency = 2

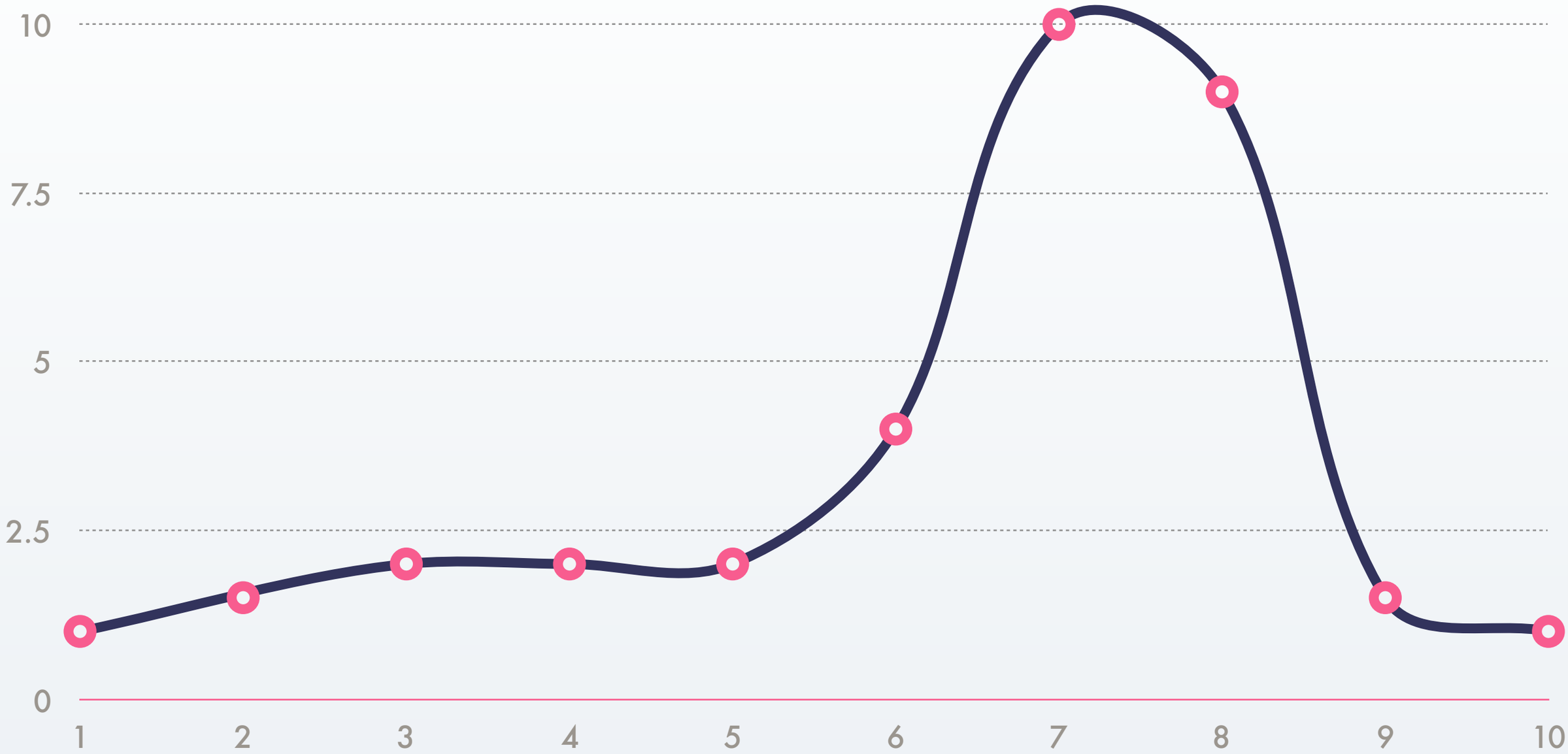


50th = 2

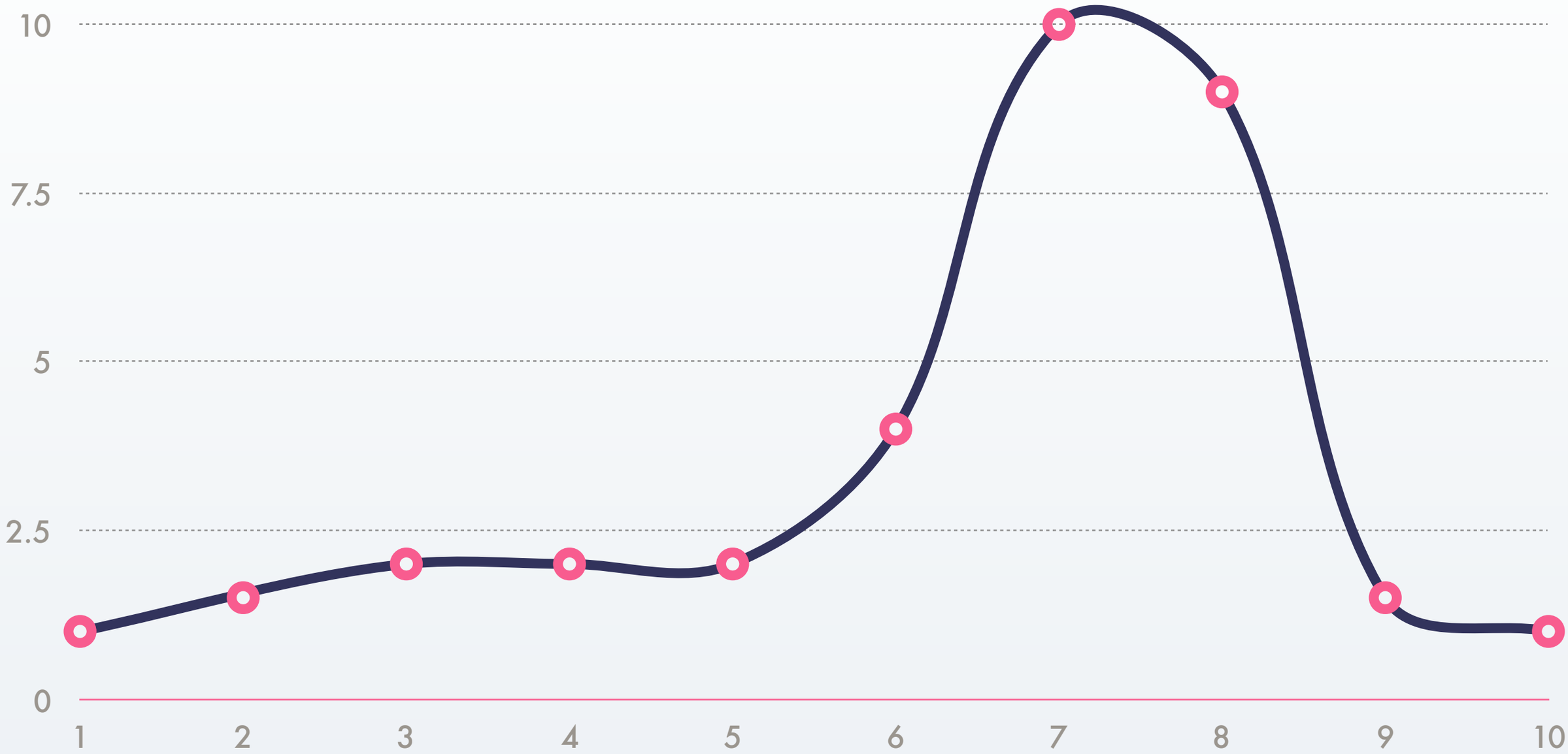
90th = 3.5

99th = 3.5

Average Latency = 2



Average Latency = 3.4



50th = 2

90th = 9.1

99th = 9.9

Average Latency = 3.4

TIP #6

IMPLEMENT BACK PRESSURE THROUGHOUT THE SYSTEM.

TIP #7

FIND WAYS TO BE PARTIALLY AVAILABLE.

TIP #8

EXPOSE METRICS

TIP #9

WRITING CACHED DATA BACK TO PERSISTENT STORAGE IS BAD.

TIP #10

USE THE CAP THEOREM TO CRITIQUE SYSTEMS.

DISTRIBUTED DESIGN PRINCIPLES

USE THE CAP THEOREM TO CRITIQUE SYSTEMS

- **Consistency** - A read is guaranteed to return the most recent write for a given client.
- **Availability** - A non-failing node will return a reasonable response within a reasonable amount of time (no error or timeout).
- **Partition Tolerance** - The system will continue to function when network partitions occur.

TIP #11

DON'T FORGET THE NETWORK

DISTRIBUTED DESIGN PRINCIPLES

DON'T FORGET THE NETWORK

- Think carefully about how much data you send over the network. Minimize traffic as much as possible.

- Don't assume that data sent across a network is the same data when it arrives. If you must be sure, do checksums or validity checks on data to verify that the data has not changed.

- It will fail. Plan accordingly (*see designing for failure*)

TIP #12

EXTRACT SERVICES.