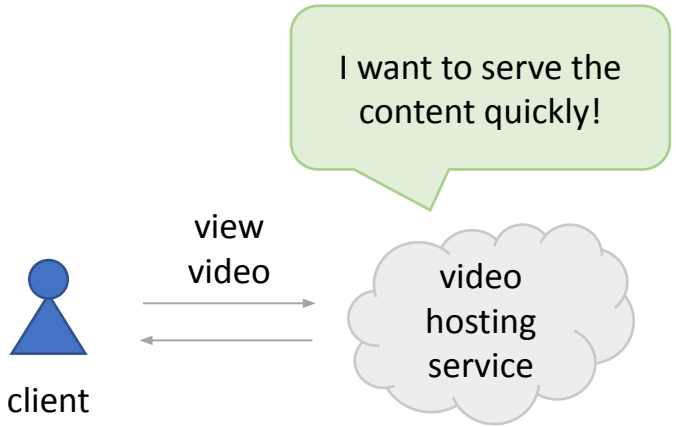
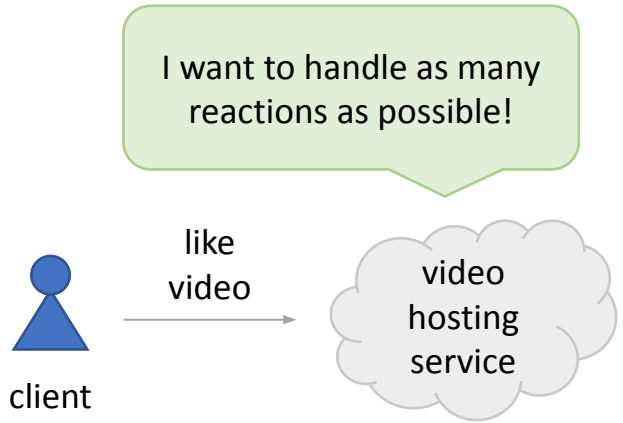
Non-functional requirements - Performance

time required to process something = performance = **rate** at which something is processed

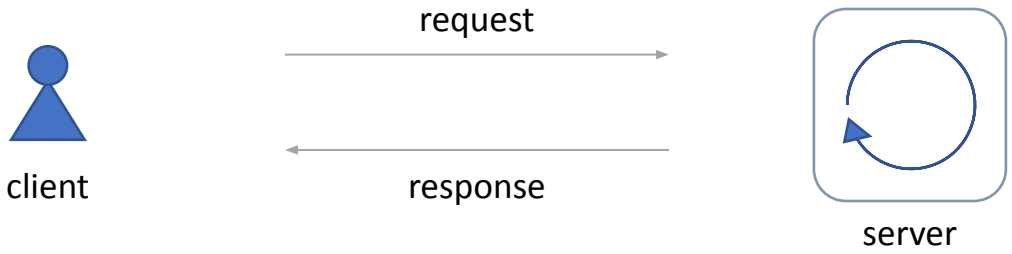


latency



throughput

Non-functional requirements - Performance



response time = network delay + service time

total time it took to respond to the request

time the request spent “on the wire” in both directions

time it took for the server to process the request

client-side latency

network latency

server-side latency

Non-functional requirements - Performance

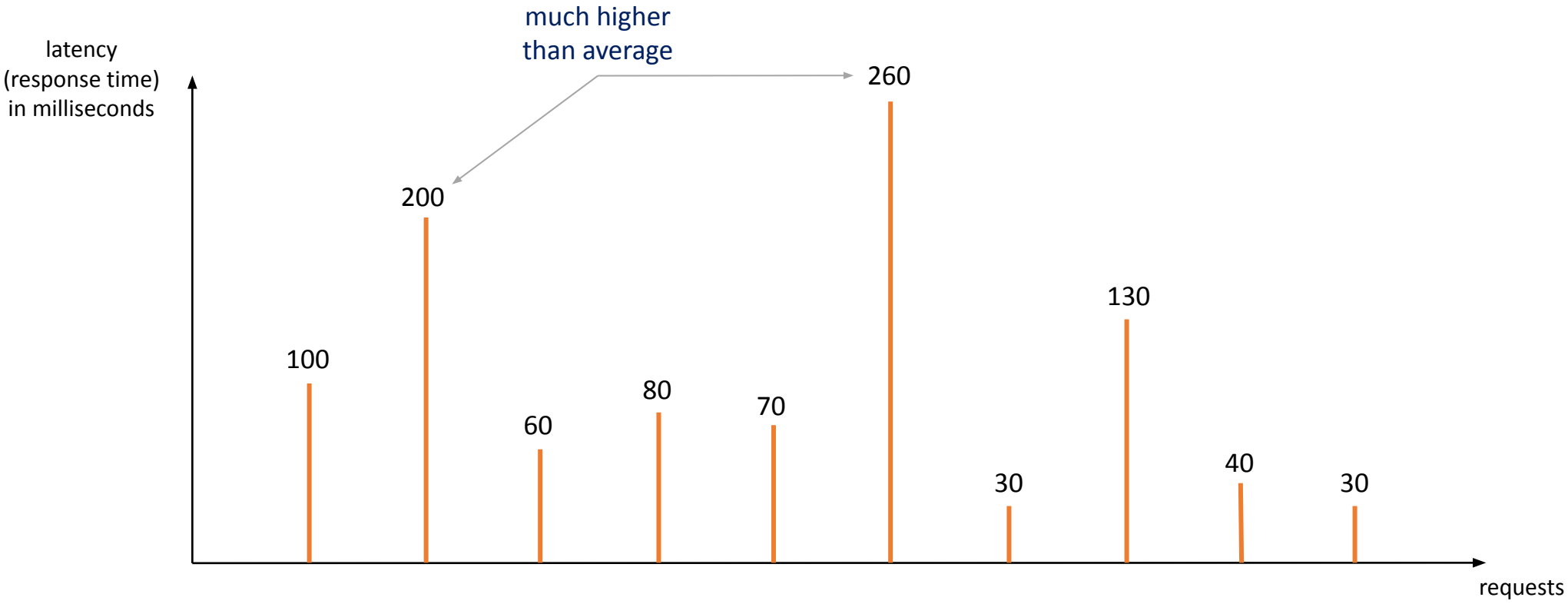
I'd rather clarify what exactly she means by latency.

software engineer

Because…

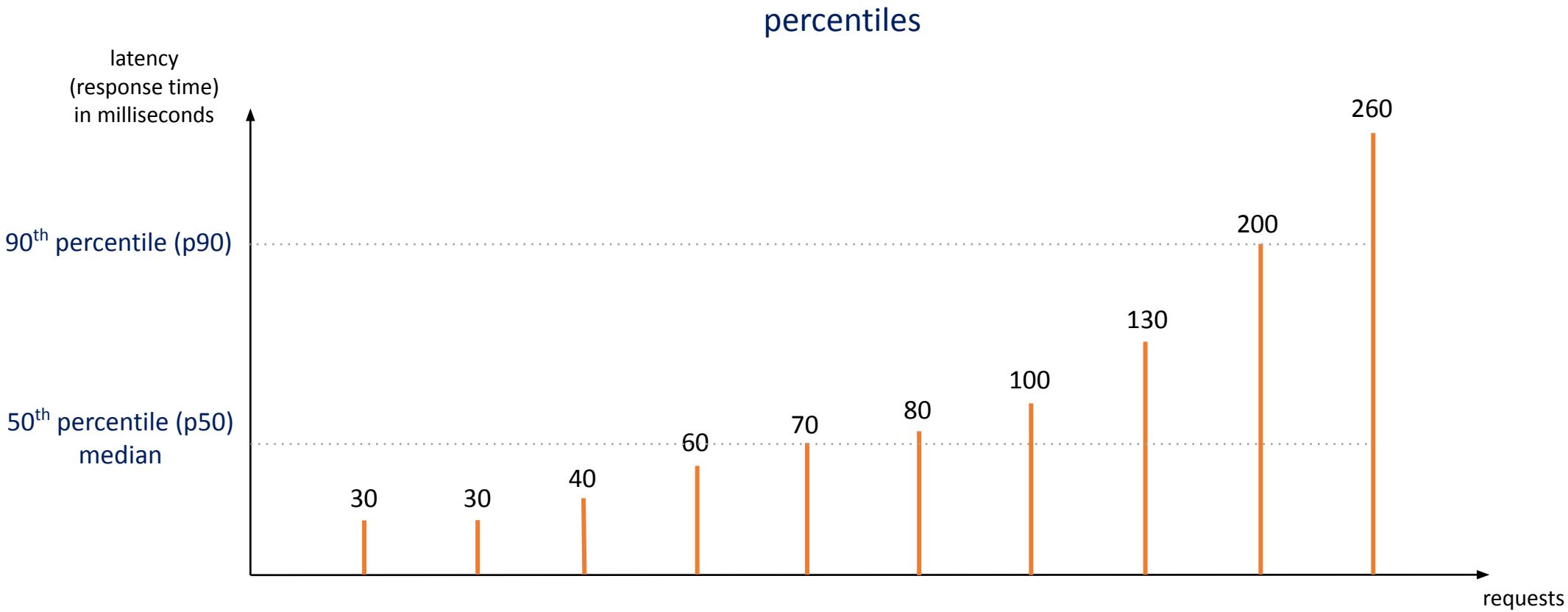
if it is…	she probably wants to hear about…
network latency	OSI model network protocols …
server-side latency	faster algorithms memory versus disk trade-offs thread pools and parallel processing local cache …
client-side latency	blocking versus non-blocking I/O, message formats data compression content delivery network external cache …

Non-functional requirements - Performance



$$\text{average latency} = \frac{\text{sum of latencies}}{\text{total number of requests}} = \frac{100 + 200 + 60 + 80 + 70 + 260 + 30 + 130 + 40 + 30}{10} = 100 \text{ ms}$$

Non-functional requirements - Performance



p99 of 3 seconds means
1% of requests took 3 seconds or more

software engineer

I should try to investigate why tail latencies (high percentiles) are so high.

Non-functional requirements - Performance

throughput

rate at which something is processed

requests
per second

database queries
per minute

network packets
per hour

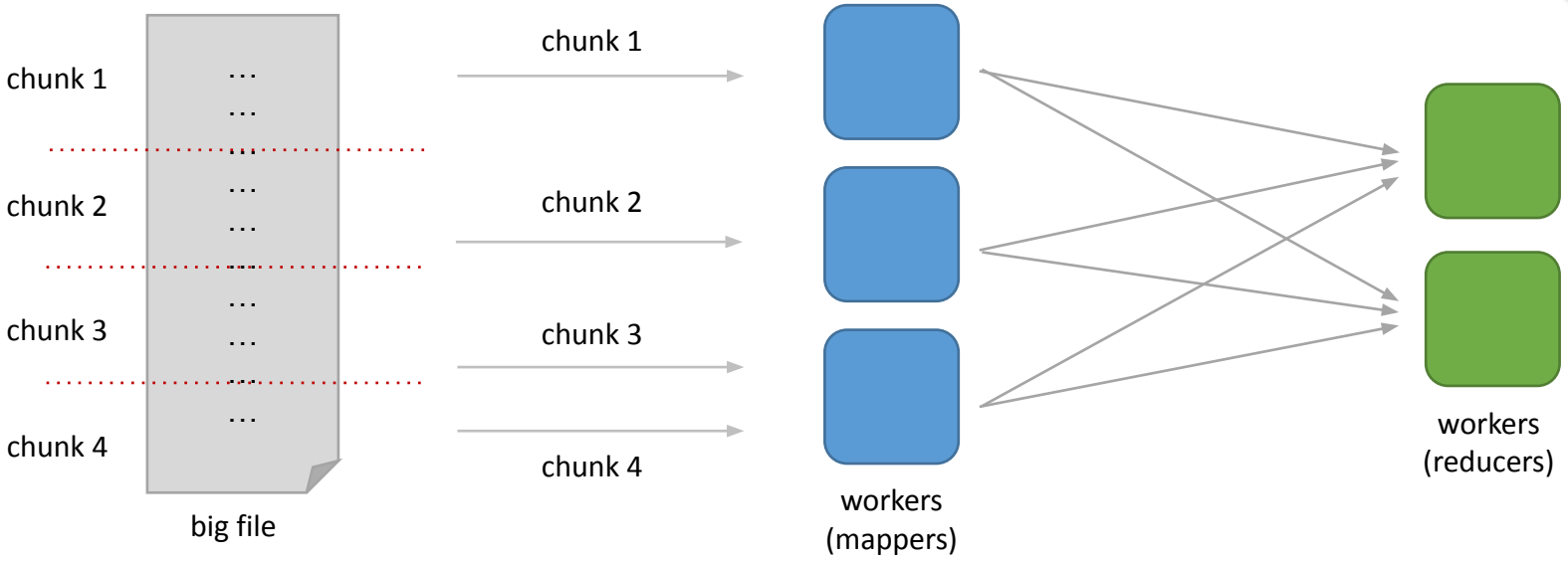
higher the throughput, better the performance

How to increase throughput?

- decrease latency
- scale up and/or out

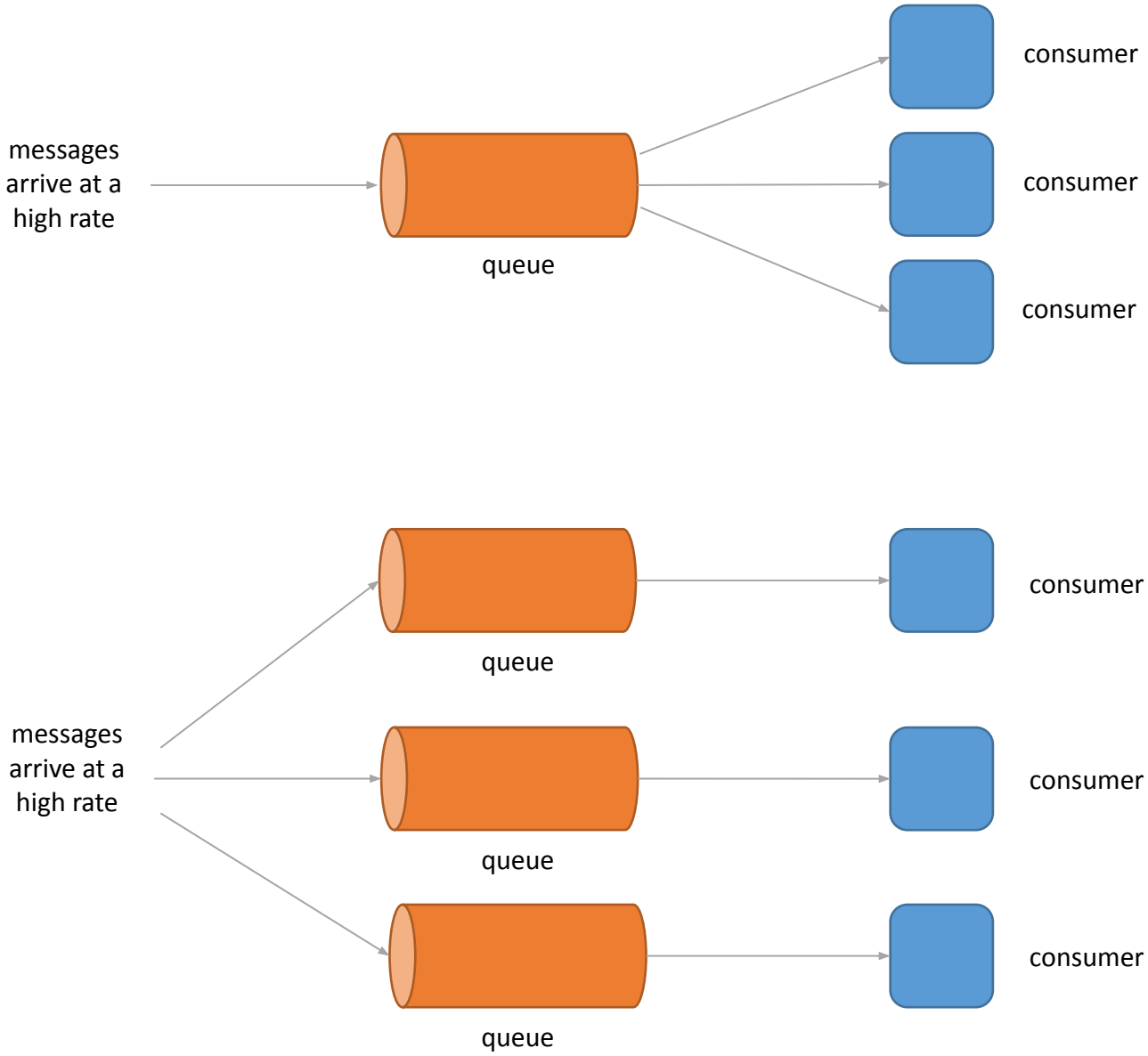
Non-functional requirements - Performance

idea behind
batch processing
(specifically, MapReduce)



we increase throughput by
splitting a larger **task** into smaller ones
and **running** tasks **in parallel**

Non-functional requirements - Performance



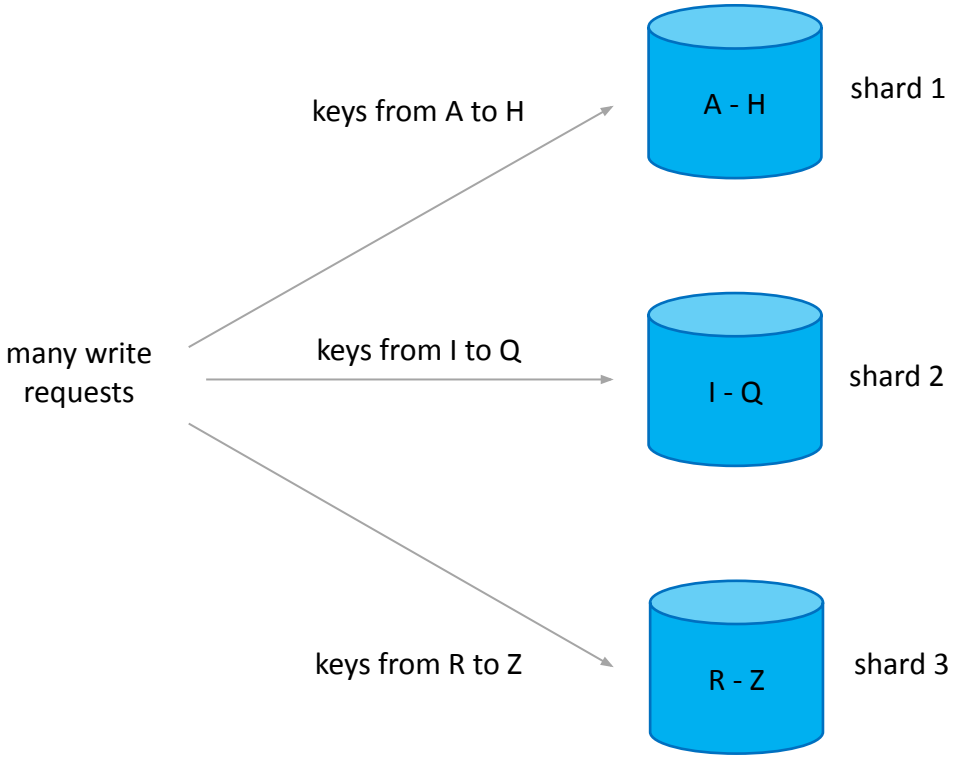
we increase throughput by
scaling out message consumers

stream processing patterns

we increase throughput by
scaling out message queues

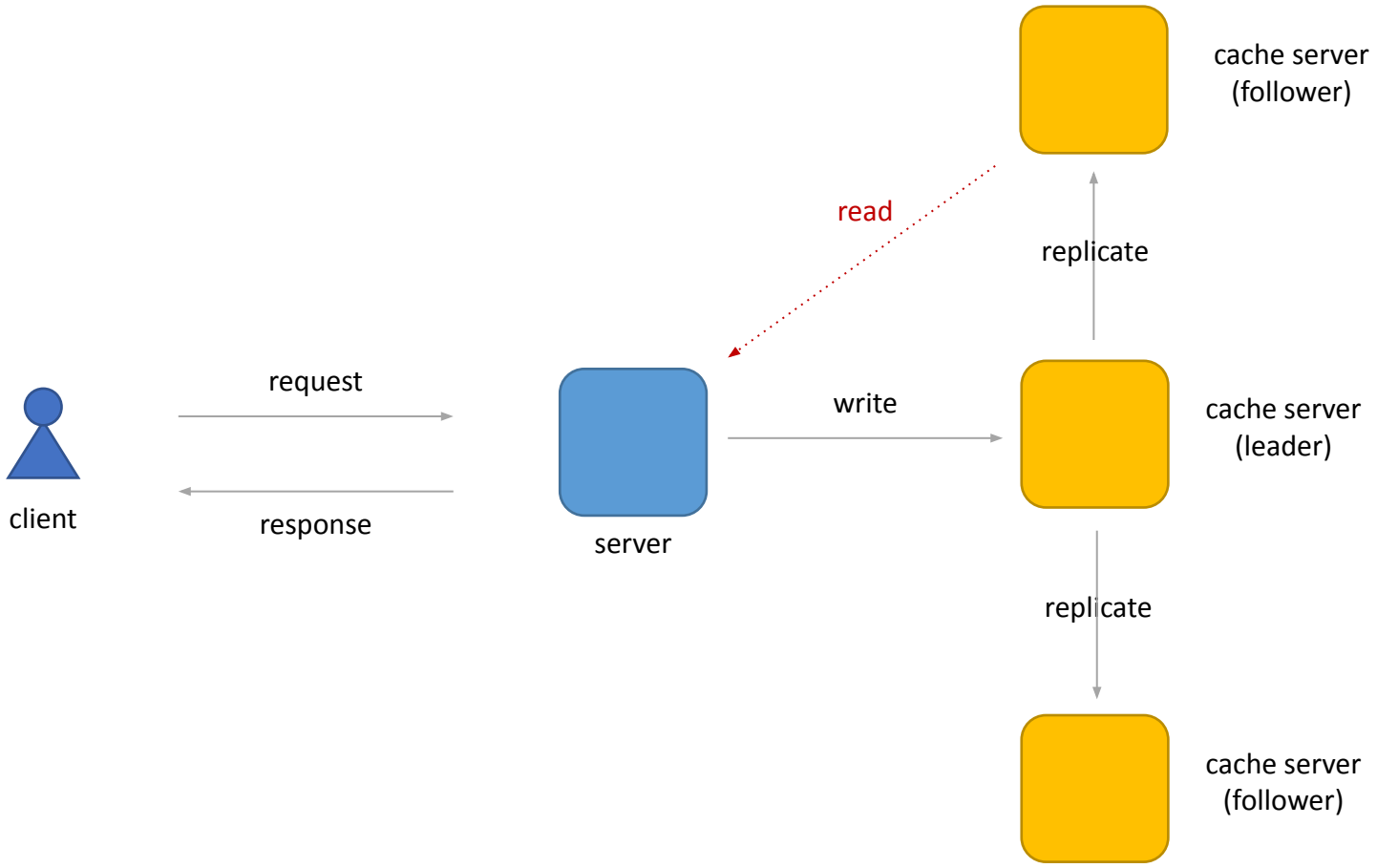
Non-functional requirements - Performance

idea behind scaling
data store writes



we increase write throughput using
sharding (**partitioning**)

Non-functional requirements - Performance



idea behind scaling
data store reads

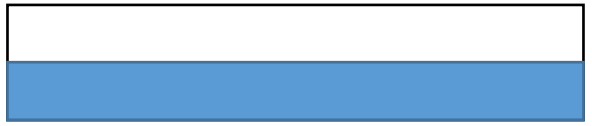
we increase read throughput using
replication

Non-functional requirements - Performance

bandwidth

maximum rate of data transfer across a given path
(bits per second)

throughput = bandwidth



throughput = ½ bandwidth