Latency, throughput, and availability: system design interview concepts (3 of 9)

SOFTWARE ENGINEERING FEB. 14, 2023

"Performance" is an important concept in system design, and it's also a common topic that comes up on system design interviews for tech roles.

Intuitively, we want our services to be fast, and to ensure that every user has this experience we need precise measurements of what "fast" means.

In this article, we'll go over three standard measurements of system performance (latency, throughput, and availability) and how to optimize for them in your system designs. Here's an overview of what we'll cover:

Table of contents

5. System design interview preparation

1. Latency

Latency is the amount of time in milliseconds (ms) it takes a single message to be delivered. The concept can be applied to any aspect of a system where data is being requested and transferred. You're probably already familiar with the latency of web requests as a user of the internet - some pages are really snappy and quick to use, and others take a long time to load and lag as you interact with them.

In a distributed system we might be interested in the latency of a database returning the results of a query. If the database is a substantial source of latency that could indicate we need to add an index or choose a different database model (read more about choosing the right database in our guide to databases).

As another example, we might be interested in the effects a caching layer has on latency - if a cache is placed effectively it can lower latencies by making it faster to retrieve data. But if it's placed ineffectively a cache can raise latencies by introducing extra computational steps to the response (read more about how to use caches effectively in our guide to caching).

1.1 What causes latency

- **Physical distance:** the speed of light is the fastest anything can travel, so no matter how you design your system, transferring data through space will always take some time.
- **Complex computation:** if a computation is complex, it’s going to take longer to execute and increase latency. For example, a complex relational database query with lots of joins will take longer than a simple lookup by id.
- **Congestion:** when there are many message requests coming in at once and the system doesn't have the capacity to process them all, some requests will have to wait, increasing latency. Either these requests will be dropped and sent again, or they'll sit in a queue waiting to be processed.
- **Too many nodes:** if there are too many decision points in the pathway of a request, it will increase latency because each node along the way adds time while processing the request and deciding where to route it.

There are a lot of nuances in tuning latencies that we don’t have the space to cover here.

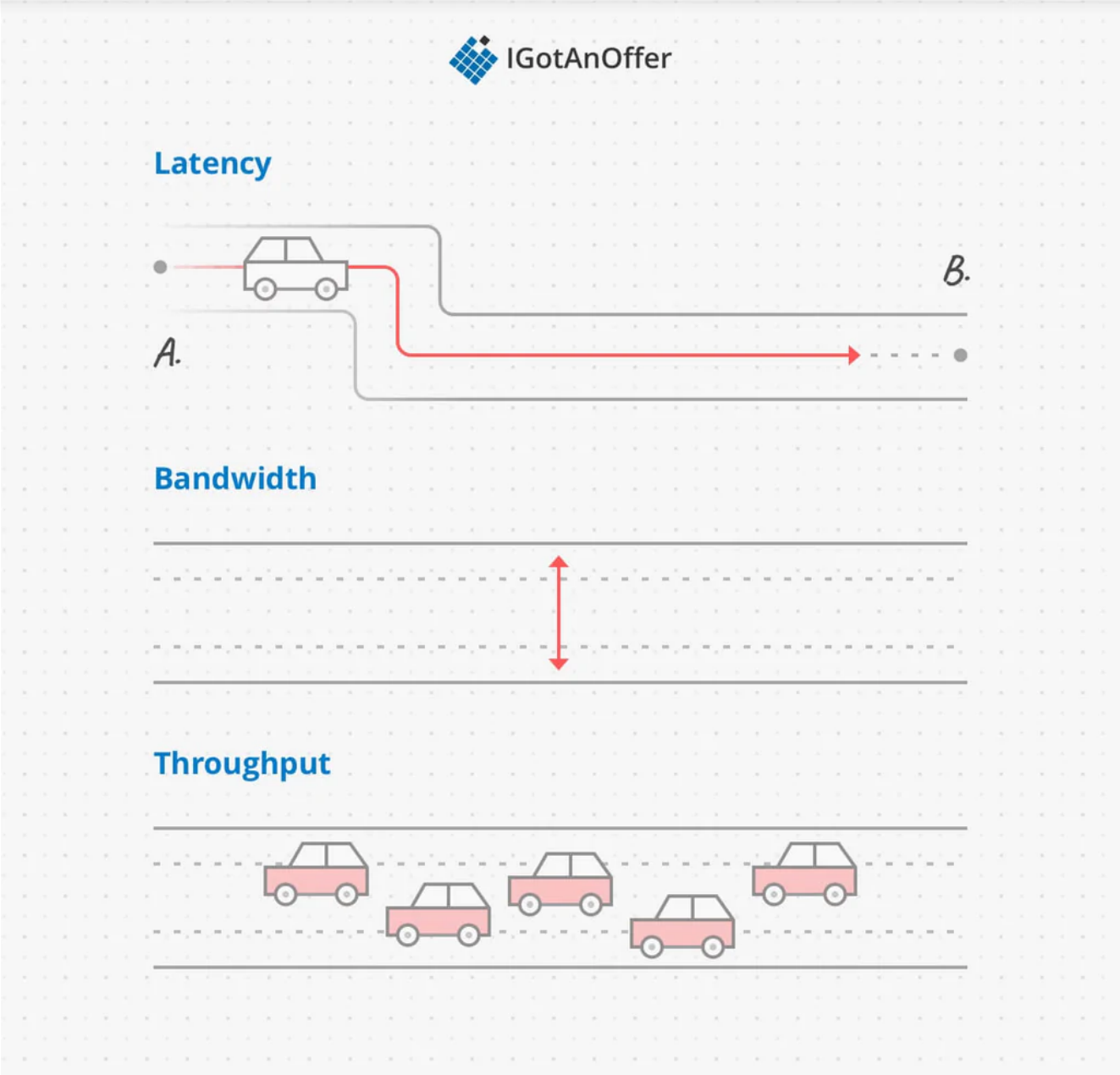
1.2 How to improve latency

- **Better paths:** minimizing the number of nodes a request has to travel through can help improve latencies.
- **Caching:** caching can dramatically improve latencies when applied correctly, by storing a copy of repeatedly accessed data for faster retrieval.
- **Protocol choice** - certain protocols, like HTTP/2, intentionally reduce the amount of protocol overhead associated with a request, and can keep latency lower. TCP also has congestion avoidance features that can help mitigate congestion-related causes of high latencies.

2. Throughput

Throughput is the amount of data that is successfully transmitted through a system in a certain amount of time, measured in bits per second (bps). Throughput is a measurement of how much is actually transmitted, and it is not the theoretical capacity (bandwidth) of the system.

Latency, throughput, and bandwidth can be easy to confuse, but it’s very similar to a system you’re probably already familiar with: traffic of cars on the road. Latency is how long it takes you to drive from A to B, bandwidth is how wide the roads are, and throughput is how many cars are on the road right now.



2.1 What causes low throughput

- **Congestion** - just like road traffic is caused by many people trying to get to the same destination, low throughput in a software system can be caused by too many requests on the same network. Essentially, the hardware can't handle the number of requests going through it.
- **Protocol overhead** - if the protocols used in message transmission require handshakes and other back-and-forth communication patterns, the network can be overloaded with overhead from just the protocols and not the message content itself.
- **Latency** - since throughput is the amount of data transmitted over a set time period, high latencies (i.e. slow data transmission speeds) will reduce the amount of data that is transmitted overall.

Table of contents

- **Increasing bandwidth** - if you improve the capacity of a system to transport data (bandwidth), then the actual amount of data transferred (throughput) will increase too. This generally means adding new hardware or upgrading the hardware at bottlenecks in the system.
- **Improving latency** - since latency limits throughput, improving latency can improve throughput.
- **Protocol choice** - TCP has congestion avoidance features that can help mitigate congestion that causes low throughput.

3. Availability

Availability is the amount of time that a system is able to respond, that is the ratio of Uptime / (Uptime + Downtime). Availability is a critical metric of performance for a service, because downtime can both harm users who rely on the systems and cause a business to lose large profits in a short amount of time. Availability is so important that infrastructure services like AWS and Google Cloud will guarantee certain uptimes in their service-level agreements (SLAs).

The gold standard for "highly available" systems is the *five nines*: 99.999% uptime. This might seem like an excessively precise number, but 90% uptime is about 17 hours of downtime a week, and 99% is about 1.7 hours of downtime a week, which is a fair amount of time for a service to be *regularly* unavailable.

Five nines availability can be misleading though if it's not balanced by other aspects of performance and user experience. Uptime doesn't matter that much if it takes a minute to respond to every user request. It's important to make sure good latency and throughput are maintained while designing a highly available system.

It's also important to think about when downtimes happen. A business will be notably affected if all downtimes occur at unfortunate high-use moments, like major shopping holidays. It can be just as advantageous to plan longer downtimes for maintenance at low-activity times and not achieve five nines availability, if it means the system will be able to function properly under peak use.

3.1 What causes low availability

Downtime happens when part of the service breaks, like a hardware component fails, or a deployment goes wrong so the software is inconsistent. Systems can fail in many ways, and here are some of the most common:

- **Hardware failure** - computer components eventually fail, and this can take down a

- **Software bugs** - when something in the code is wrong, a request can run into a bug and the code can be killed (e.g. a null pointer is dereferenced)
- **Complex architectures** - the more complex a system design is, the more points of failure there are, and the harder it gets to synchronize more computers and make them fault tolerant to other computers in the system failing.
- **Dependent service outages** - an outage of a service that the system relies on, like DNS or an authorization server, can cause the system to become unavailable even if nothing is broken internally.
- **Request overload** - if a system reaches its maximum capacity, it can start failing to respond to some requests. Too many requests can also cause the computer to shut down if it runs out of a key resource and can't process any more operations (e.g. the disk space fills up).
- **Deployment issues** - when a deployment is conducted and the changes to software or configurations don't go as expected, a number of problems can arise that could make the system unavailable. For example, deployment issues could put the servers in an inconsistent state, prevent them from starting, prevent them from talking to each other, or they might run short on resources.

3.2 How to improve availability

The solution to partial system failure is redundancy - making sure that if something goes wrong there's a copy of it so the system can continue functioning. Redundancy has many components, including:

- **Failover systems** - duplicates of any part of the system that are switched to in the case of failure. These can be either hot failovers that run in parallel for immediate switch, or cold failovers that only start when needed.
- **Clustering** - running multiple instances of a part of the system, so if one node goes down the rest can manage without it.
- **Backups** - data backups and replication ensure that if the data storage fails, for example a power outage at the data center, there's another copy that can be accessed.
- **Geographic redundancy** - physically locating systems in different parts of the world, so if something happens that affects a region, there are redundant systems that still function.
- **Automatic testing, deployment, and rollbacks** - to mitigate all the issues related to deploying new software or changing the architecture, it's helpful to automate processes that catch software bugs, prevent human deployment errors, and

4. Example latency, throughput, and availability questions

The questions asked in system design interviews tend to begin with a broad problem or goal, so it's unlikely that you'll get an interview question that's entirely about latency, throughput, or availability.

However, you may be asked to solve a problem where these topics will be important considerations. As a result, what you really need to know is WHEN you should bring them up and how you should approach them.

To help you with this, we've compiled the below list of sample system design questions, where latency, throughput, or availability considerations are relevant.

- Design Twitter timeline and search (Read the answer)
- Design Mint.com (Read the answer)
- Design DropBox (Read the answer)

5. System design interview preparation

Latency, throughput, and availability each describe one metric of a system. But to succeed on system design interviews, you'll also need to familiarize yourself with a few other concepts. And you'll need to practice how you communicate your answers.

It's best to take a systematic approach to make the most of your preparation time, and we recommend the steps below. For extra tips, take a look at our article: 19 system design interview tips from FAANG ex-interviewers.

Alternatively, if you're looking to save time and access everything you need in one place, we recommend taking The System Design Strong Hire Course. The course includes detailed lessons on system design fundamentals and how to answer system design interview questions, plus mock interview videos with FAANG ex-interviewers and more than 100 practice questions.

Otherwise, you can prepare by following the steps below.

5.1 Learn the concepts

There is a base level of knowledge required to be able to speak intelligently about system design. To help you get this foundational knowledge (or to refresh your

- Network protocols and proxies
- Databases
- Latency, throughput, and availability
- Load balancing
- Leader election
- Caching
- Sharding
- Polling, SSE, and WebSockets
- Queues and pub-sub

We'd encourage you to begin your preparation by reviewing the above concepts and by studying our system design interview prep guide, which covers a step-by-step method for answering system design questions. Once you're familiar with the basics, you should begin practicing with example questions.

5.2 Practice by yourself or with peers

Next, you'll want to get some practice with system design questions. You can start with the examples listed above, or with our list of 31 example questions.

We'd recommend that you start by interviewing yourself out loud. You should play both the role of the interviewer and the candidate, asking and answering questions. This will help you develop your communication skills and your process for breaking down questions.

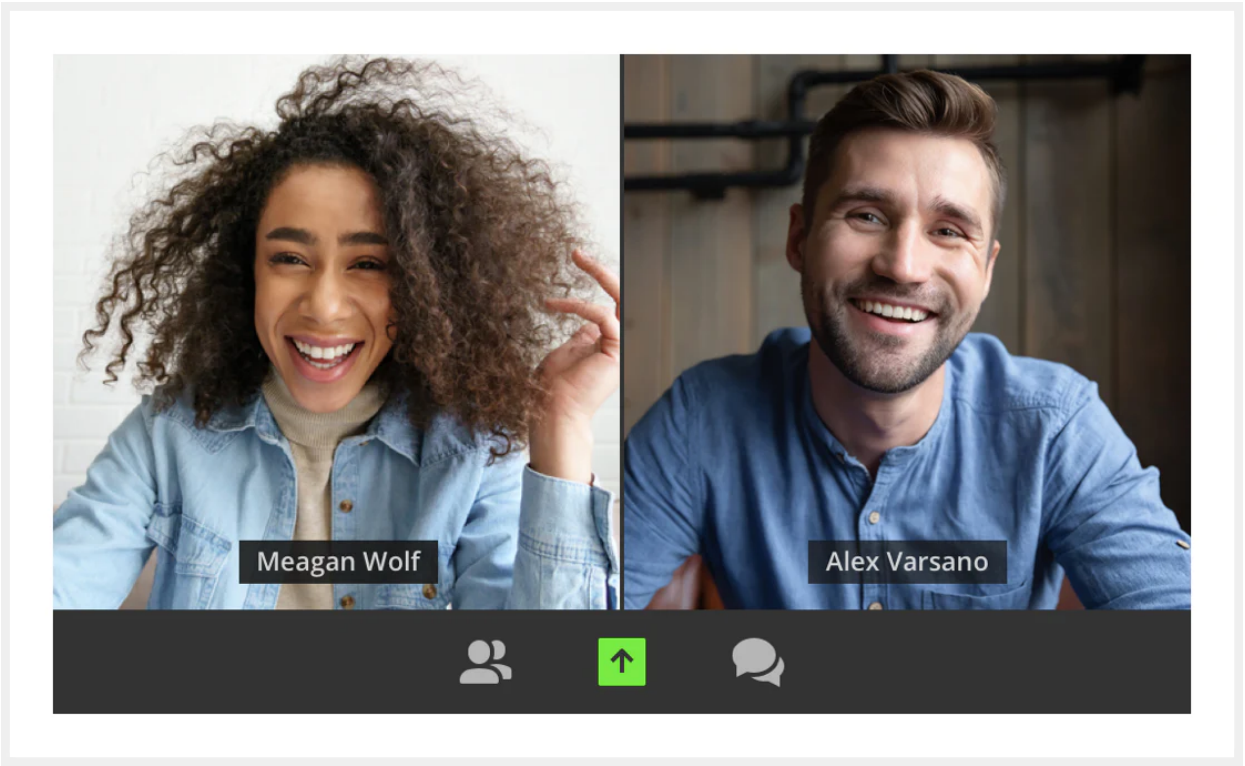
We would also strongly recommend that you practice solving system design questions with a peer interviewing you. A great place to start is to practice with friends or family if you can.

5.3 Practice with ex-interviewers

Finally, you should also try to practice system design mock interviews with expert ex-interviewers, as they'll be able to give you much more accurate feedback than friends and peers.

If you know someone who has experience running interviews at Facebook, Google, or another big tech company, then that's fantastic. But for most of us, it's tough to find the right connections to make this happen. And it might also be difficult to practice multiple

Here's the good news. We've already made the connections for you. We've created a coaching service where you can practice system design interviews 1-on-1 with ex-interviewers from leading tech companies. Learn more and start scheduling sessions today.



BROWSE FAANG EX-INTERVIEWERS

Learn more about system design interviews

This is just one of 9 concept guides that we've published about system design interviews. Check out all of our system design articles on our Tech blog.

Table of contents

ALSO ON **IGOTANOFFER TECH**

Google engineering manager interview: ...

2 years ago · 2 comments

Complete guide to Google engineering manager interviews. Learn the ...

Google ML engineer interview: the only ...

a year ago · 1 comment

Complete guide to Google machine learning engineer interviews. Learn the ...

How to crack product improvement ...

3 years ago · 8 comments

Product improvement questions are asked in PM interviews at Google, ...

Amazon s developm

2 years ago

Complete g software d manager in

Table of contents

All articles
Amazon interview coaching

INTERVIEW COURSES

PM interview course
Google PM course
Amazon PM course
Meta PM course
System design course

MOCK INTERVIEW SERVICE

Product manager mocks
System design mocks
Coding mocks
Google mocks
Amazon mocks

OTHER COACHING

Consulting interviews
Finance interviews
MBA admissions

Table of contents