



# Examples of Resource Estimation

Try your hand at some of the back-of-the-envelope numbers.

## We'll cover the following



- Introduction
- Number of servers required
  - Peak capacity
    - Improving the RPS of a server
    - Improving over the peak load assumption
  - Cost of servers
- Storage requirements
- Bandwidth requirements

## Introduction

Now that we've laid the foundation for resource estimation, let's make use of the knowledge we gained in the previous lesson to estimate resources like servers, storage, and bandwidth. Below, we consider the Twitter service, make assumptions, and based on those assumptions, we make estimations. Let's jump right in!

## Number of servers required

Let's make the following assumptions about a Twitter-like service.

**Assumptions:**



- There are 500 million (M) daily active users (DAU).
- A single user makes 20 requests per day on average.
- We know that a single server (with 64 cores) can handle 64000 RPS.

>

## Estimating the Number of Servers

Daily active users (DAU)	500	Million
Requests on average / user / day	20	
Total requests / day	<i>f</i> 10	Billion
Total requests / second	<i>f</i> 115	K
Total servers required	<i>f</i> 2	

 Show Detailed Calculation

Points to ponder

Question

Can you identify a hidden assumption in our calculations above?

?

Tt





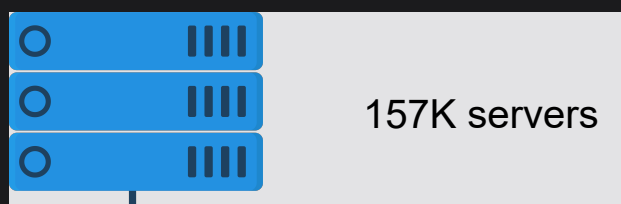
**Plausibility test:** For all BOTECs, we need to judge if our numbers seem reasonable. For example, if the estimate we obtained was two servers for a large service with millions of DAUs, that number can be a lower bound but seems far from reality.

## Peak capacity

Often, large services need to be ready for any flash crowds. We can get an estimate for the peak capacity. We assume that there's a specific second in the day when all the requests of all the users arrive at the service simultaneously. We use it to get the capacity estimation for a peak load. To do better, we'll need request and response distributions, which might be available at the prototyping level. We might assume that requests follow a particular type of distribution, for example, the **Poisson distribution**↗.

By using DAU as a proxy for peak load for a *specific second*, we've avoided difficulties finding the distributions of requests. Therefore, the DAU will then become the number of requests per second. So the number of servers at peak load can be calculated as follows:

$$\text{Servers needed at peak load} = \frac{\text{Number of requests/second}}{\text{RPS of server}} = \frac{10 \text{ Billion}}{64,000} = 157K$$



If our assumption is correct that all of the workloads can show up simultaneously in a specific second and each of our servers can handle 64,000 requests per second, we'll need the astronomical count of 157K servers! If that's not feasible, then we have two potential paths forward now, as explained below.

## Improving the RPS of a server

First, if we think our assumption for the peak load is correct, we can work out how many servers at max we can commission. Let's assume we can employ 100,000 servers at most. That implies:

$$\frac{\text{Number of requests/second}}{\text{Max servers we can employ}} = \frac{10 \text{ Billion}}{100,000} = 100,000 \text{ Requests per second per server}$$

We'll need extensive engineering to bump the RPS we can extract from a server from 64,000 to 100,000!

There are many examples where organizations relied on extensive engineering to improve the RPS of servers.

**First example:** WhatsApp reported in 2012 that they can manage 2 million concurrent TCP connections on one server. A report in 2017 mentioned that WhatsApp uses ~ 700 servers from IBM for its service. It's not clear what the specific specification of a server was.

**Second example:** A research system that was extensively optimized for IO won multiple sorting competitions in 2010. They sorted one trillion data records in 172 minutes using just a quarter of the computing resources of the other record holder, which means they improved the RPS three times more than the other record holder.



The examples above highlight that improving RPS for specific use cases is possible, though at the expense of focused R&D efforts and related dollar expenses.



## Improving over the peak load assumption

The second choice for us is to change our assumption for the peak load. Using the Pareto principle, also known as the 80/20 rule, estimating peak traffic can be a reasonable approach in many cases. The Pareto principle suggests that approximately 80% of the effects come from 20% of the causes. In the context of estimating peak traffic, we can make the assumption that 80% of our peak traffic occurs within 20% of the time (a 4.8-hour window in 24 hours).

$$\frac{\text{Number of requests/second}}{\text{RPS of server}} = \frac{\frac{0.8 \times 10 \text{ Billion}}{4.8 \times 60 \times 60 \text{ Seconds}}}{64,000} \approx 8 \text{ Servers}$$

Once again, we've assumed that requests are equally distributed in the 4.8-hour window. The examples above show us that it makes a huge difference if the requests show up concurrently versus requests spread out over time. When systems are built on such assumptions, monitoring systems are in effect to make sure such assumptions are never violated. If the load gets higher than we predict, techniques like load-shedding, circuit-breakers, and throttling can be employed. Dealing with an unexpected traffic peak is a difficult problem.

Graceful degradation

### Question

Let's consider a service hosting the dynamic and personalized website of a large news organization. Due to some unexpected events, such as 9/11,



flash crowds are coming to the website to find out updates. It might be a situation where all the DAUs come in simultaneously.

Such a situation will clearly break our usual load assumptions. Can you think of some way to gracefully degrade the service to meet such an unexpected load?

Show Answer ▾

## Cost of servers

We picked an EC2 instance type called m7i.16xlarge with a 64-core processor and 256 GB of RAM to get a handle on the dollar cost of servers. It's powered by 4th-Generation Intel Xeon Scalable processors. The hourly cost of one such instance is \$3.54816 with a 1-year contract plan.

An approximate cost of an AWS instance



We've taken an [EC2 instance from AWS](#) with the following specifications:

# EC2 Instance Specifications

Instance Size	vCPU	Memory (GiB)	Instance Storage (GB)	Network Bandwidth (Gbps)
m7i.16xlarge	64	256	EBS-Only	25

The following table shows the cost of m7i.16xlarge instances for two, eight, and 157K servers. The cost can quickly pile up, as we can see for the peak load case. In real projects, the dollar budget available for specific items, such as servers, is a hard constraint that the engineering team needs to meet.

## Cost of servers

Low bound server cost per hour	Cost under 80–20 assumptions per hour	Peak load cost per hour
$2 * \$3.548 = \$7.096$	$8 * \$3.548 = \$28.38$	$157K * \$3.548 = \$554,086$

## Storage requirements

In this section, we attempt to understand how storage estimation is done by using Twitter as an example. We estimate the amount of storage space required by Twitter for new tweets in a year. Let's make the following assumptions to begin with:

- We have a total of 500M daily active users.
- Each user posts three tweets in a day.
- Ten percent of the tweets contain images, whereas five percent of the tweets contain a video. Any tweet containing a video won't contain an image, and

vice versa.

- An image is 200 KB and a video is 3 MB in size, on average.
- The tweet text and its metadata require a total of 250 bytes of storage in the database.

Then, the following storage space will be required per day:

### Estimating Storage Requirements

Daily active users (DAU)	500	M	
Daily tweets	3		
Total requests / day	<i>f</i> 1500	M	
Storage required per tweet	250	B	
Storage required per image	200	KB	
Storage required per video	3	MB	
Storage for tweets	<i>f</i> 375	GB	
Storage for images	<i>f</i> 30	TB	?
Storage for videos	<i>f</i> 225	TB	Tt
Total storage	<i>f</i> 255	TB	⚙



💡 Show Detailed Calculation

>

- Total storage required for one day =  
 $0.375TB + 30TB + 225TB \approx 255 TB$ .
- Storage required for one year =  $365 \times 255 TB = 93.08 PB$ .



The total storage required by Twitter in a year

## Bandwidth requirements

In order to estimate the bandwidth requirements for a service, we use the following steps:

1. Estimate the daily amount of incoming data to the service.
2. Estimate the daily amount of outgoing data from the service.
3. Estimate the bandwidth in Gbps (Gigabits per second) by dividing the incoming and outgoing data by the number of seconds in a day.

**Incoming traffic:** Let's continue from our previous example of Twitter, which requires 255 TBs of storage each day. Therefore, the incoming traffic should support the following bandwidth per second:

$$\frac{255 \times 10^{12}}{86400} \times 8 \approx 24Gbps$$

?

Tt

⚙️

**Note:** We multiply by 8 in order to convert bytes (B) into bits (b) because bandwidth is measured in bits per second.



**Outgoing traffic:** Assume that a single user views 50 tweets in a day. Considering the same ratio of five percent and 10 percent for videos and images, respectively,


 **educative**

the following estimations:

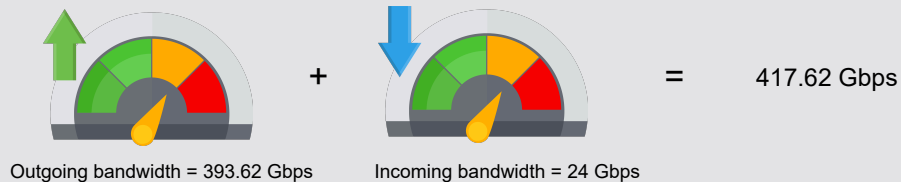
### Estimating Bandwidth Requirements

Daily active users (DAU)	500	M
Daily tweets viewed	50	per user
Tweets viewed / second	<i>f</i> 289	K
Bandwidth required for tweets	<i>f</i> 0.58	Gbps
Bandwidth required for images	<i>f</i> 46.24	Gbps
Bandwidth required for videos	<i>f</i> 346.8	Gbps
Total bandwidth	<i>f</i> 393.62	Gbps



 **Show Detailed Calculations**

Twitter will need a total of 24 *Gbps* of incoming traffic and 393.62 *Gbps* of outgoing, assuming that the uploaded content is not compressed. Total bandwidth requirements =  $24 + 393.62 = 417.62$  *Gbps*.



The total bandwidth required by Twitter

Once again, the calculations above depend on our specific assumption about the traffic mix (text versus images versus video) and read/write mix.

### Points to ponder

#### Question 1

We came up with the number of 93 PB for storage needs per year. Is this number plausible?

Show Answer ▾

1 of 2



> This lesson is a template for resource estimations in the rest of the course. We'll use numbers from this lesson throughout the course. BOTECS enable us to show the system's feasibility under a specific design. During interviews, the ability to do such calculations exhibits a candidate's problem-solving skills in the face of the unknown. There's only so much we can do in an interview. In the real world, however, organizations rely on real measurements for their specific workload and input to their back-of-the-envelope calculations.

← Back

Put Back-of-the-envelope Numbers in ...

✓ Completed

Introduction to Building Blocks for Mod...

Next →

