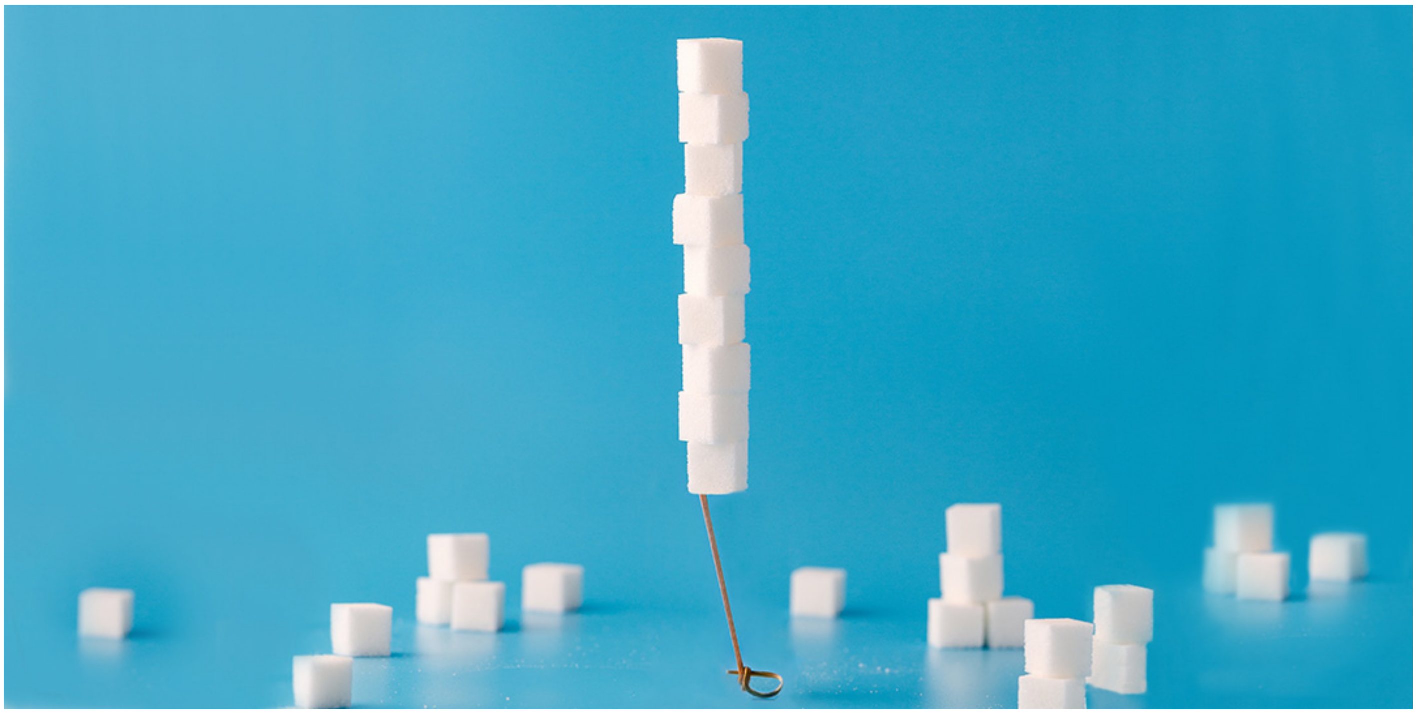
Autoscaling explained: Why scaling your site is so hard

Web Performance *Online Retail*

Scaling up is a natural first step to prepare for heavy website traffic. But if you disregard the difficulty of autoscaling, you risk failure during your busiest and most business-critical sales. Using an infographic, everyday language & plenty of analogies, here's an explainer of why scaling a website is anything but easy.

Published: 08 Nov 2019 *Updated: 26 Aug 2021*



"Just scale up!" says the ecommerce manager. "It can't be *that* hard."

You're probably here to help explain to management why "just scaling up" is harder than it seems—or to understand it better yourself.

In an ideal world, websites could scale infinitely and on demand, accommodating whatever traffic the internet throws their way. Trendy buzzwords like "autoscaling" makes this sound simple and automatic.

But scaling servers is no trivial task. And scaling your web application is even harder.

Using everyday language, plenty of analogies, and a sweet infographic, here is our take on why scaling your website is so darn difficult.

1. What do scalability, server scaling, and autoscaling mean?

First things first:

Behind every website is a physical server (or group of servers, known as a server farm). Servers provide computing power to serve data to website visitors when they open your website in their browsers.

Scalability describes the ability of a system to grow while managing the increased use that comes with growth. To serve more visitors, web applications need increased computing power from the servers and need all the applications' components to scale as well.

Server scaling describes adjusting the computing power of servers, usually to increase power by "scaling up". This can be done either by scaling vertically or horizontally. Vertical scaling involves replacing the server with a larger, more powerful one. Horizontal scaling (sometimes called "scaling out") means adding several servers and combining their totaled computing power.

Autoscaling is the strategy of automatically adjusting the computing power based on the current resource load. More website visitors place more load, or strain, on your servers. Leveraging cloud computing like Amazon AWS or Microsoft Azure, autoscaling attempts to horizontally scale up servers when there is more demand and scale down servers when demand is less.

2. Why is scaling a website so hard? An analogy

When you scale up your servers, you expect website performance to stay stable. The point of scaling up is to keep the website functioning just as well with 10 visitors per minute as 100 visitors per minute.

But the devil is in the details—scaling servers doesn't mean all parts of the website can scale equally well. The mechanics behind scaling your website make it difficult and complex to maintain performance under increased demand.

This explains why Urs Hölzle, Google's first vice president of engineering, says:

> "At scale, everything breaks."

According to Paul King, a data scientist who's worked at Facebook, Quora, and Uber, that's because your web application runs into **four main problems** when you try to scale it:

1. The search problem
2. The concurrency problem
3. The consistency problem
4. The speed problem

To illustrate this example, we'll take the analogy of using a phone book to find someone's phone number.

Let's start with the phone numbers of the people who live in your building. There might be 10-20 names listed, each with an accompanying phone number.

You can quickly find the name and phone number of whoever you need to reach. And all the information fits on a single sheet of paper that can be easily distributed in the building.

The search problem

The search problem describes the difficulty of finding the information you need in a sea of data. The bigger the data set, the more intensive it is to find what you're looking for.

So, imagine we're now growing our phone book to cover 1 million city residents. It would take hours to scan through an unorganized list of names and phone numbers.

Organizing the names in some way—alphabetically, geographically, etc.—is a strategy to deal with the search problem. But if you've ever tried this, you'd know it would still take longer than scanning a single page list of names.

The concurrency problem

The concurrency problem describes the difficulty of making data available to several people, programs, or resources simultaneously.

In our analogy, if all 1 million people wanted to find someone's phone number at the same time from 1 phone book, there would be a massive line. You'd never get to make your call!

Creating extra copies of the phone book (called "replication") could help ease the concurrency problem. As could delivering these extra copies to city residents (called "distribution"). But as you'll see, these lead to issues with the remaining two problems.

The consistency problem

The consistency problem describes the difficulty of dealing with constantly updated data whose updates need to be reflected to the people, programs, or resources using that data.

Now what happens when people change phone numbers? The printed, distributed phone books are out of date as soon as they're printed. Now you need to update all 1 million copies.

If you collect all the phone books, it will take time to replace them, and then you'll have an availability problem. The data won't be consistently accessible. You could change them out one at a time. But then data will be inconsistent across the city. You could publish addendums of all recent changes (called "change logs"). But then every time anyone wants to check a phone number, they must scan the addendum(s) first, adding time to the entire process.

The speed problem

The speed problem describes the increasing difficulty of handling more and more requests or transactions.

We just saw how adding change logs increases the time needed to look up a number. Now imagine a new phone company enters the market and offers a bargain phone plan. Suddenly, lots of people at once change their phone numbers. So many numbers are changing that the printing process needs to speed up. Delivery services are inundated. Trash cans are filled with old phone books. All in all, speeding things up takes an already complex situation and makes it downright chaotic.

Your ecommerce website isn't immune

All these problems afflict ecommerce websites, too.

It's easy enough to build a website with a few items. But when you have thousands of products, you need advanced search functions and meaningful categories to help customers find what they need. The more products, the more effort this takes from your application. You run into the search problem.

Most open-source ecommerce platforms are built around inventory truthfulness, prioritizing consistency over concurrency. As we've just seen with the phone book analogy, you can't have both at the same time.

And when popular sales like Black Friday drive website traffic through the roof, the speed problem rears its ugly head, explaining why mega-retailers like Amazon, H&M, and Walmart have seen their websites crash.

As Paul King writes,

> "The basic goal of a database is to maintain the **illusion** that there is only one copy, only one person changes it at a time, everyone always sees the most current copy, and it is instantly fast."

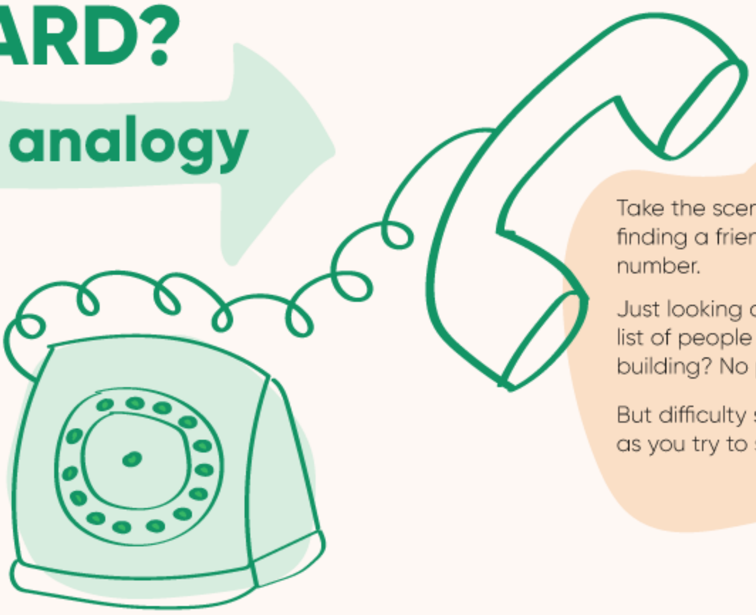
Systems engineers use hundreds of interlocking algorithmic tricks to try to maintain this illusion. But when your website needs to truly—and worse, quickly—scale to growing demand, the fabric holding together this illusion stretches and tears. Your visitors experience this in real time, as your website slows and eventually crashes.

3. Infographic showing the difficulty of server scaling

Here's an infographic we've put together outlining the problems you face when trying to scale your web application.

WHY IS SCALING A WEBSITE SO HARD?

an analogy



Take the scenario of finding a friend's phone number.

Just looking at a contact list of people in your building? No problem.

But difficulty skyrockets as you try to scale...

SEARCH PROBLEM

How do I find the phone number I need in a really big list?

Now you expand to 1 million city residents.

So the list turns into a book.

Finding one name is now harder & slower than scanning the original list.

CONCURRENCY PROBLEM

How do multiple people look up phone numbers at once?

Now multiple people want to look up a number..

CONCURRENCY PROBLEM

How do multiple people look up phone numbers at once?

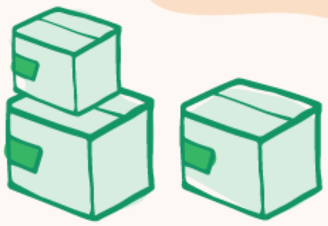
Now multiple people want to look up a number..

But there is only 1 phone book for the whole city.

CONSISTENCY PROBLEM

How do the phone books stay up-to-date?

Over time people move & change their numbers...



meaning all those phone books get out of date.

SPEED PROBLEM

What happens when the rate at which people change numbers increases dramatically?

A phone company offers a great plan & tons of people change numbers.

Speeding things up takes an already complex situation & makes it downright chaotic.

Speeding things up takes an already complex situation & makes it downright chaotic.
Everything has limits, so at scale every system breaks

4. Why does server scaling become so costly?

We know that scaling up your whole web application is incredibly complex. But the more website visitors increase, the complexity builds exponentially—and so can the costs.

Vertical scaling costs

To understand why vertical scaling is so expensive, let's compare the engines of the Ferrari 812 Superfast with the Ford Focus.

The Ferrari's motor produces a maximum of 789 horsepower. That's a lot of power! But, you pay a premium for such a high-performing car. Ferrari takes on a lot of costs designing, engineering, and producing a high-quality engine capable of such power. Top-notch materials. Incredibly efficient design. You get the picture. The $315,000 price tag reflects these costs.

The Ford Focus engine pumps out a respectable 160 horsepower. It's no Ferrari 812 Superfast. But at a price of $19,000, it's also far more economical. You'd need 5 Ford Focus engines to reach the output of the Ferrari, but this would cost $95,000, or over 3 times less than the Ferrari engine. If you could connect the 5 Ford engines, you'd realize the same power at a fraction of the cost.

When vertical scaling requires increasingly higher-end servers, costs increase exponentially.

789 hp
800 hp
$315,000
$95,000

What's more, if you do need to swap out one server for a higher-end one, you'll need downtime to make the switch. That's why vertical scaling doesn't mesh well with autoscaling. Every time you need to scale up or down, you'd incur downtime.

Horizontal scaling costs

Horizontal scaling is the cheaper of the two when it comes to hardware costs. Data center costs can be higher because of increased space, cooling, and power usage. And licensing fees can increase as you have more nodes, or groupings of servers, to license. Depending on your licensing structure, you might end up needing annual licenses for servers even if you only use them for one or two days in the year.

However, the **hidden costs come with the dedicated expertise needed to set up and maintain the autoscaling and application scaling**. Many IT teams consist of devoted DevOps engineers handling the technical details. If Amazon’s 2018 Prime Day failure tells us anything, it’s that it’s incredibly challenging to autoscale a website under heavy load.

Early improvements are easier to find and give better performance. But the more you try to squeeze out of your application, the less benefit any changes will have. More bugs appear as the code becomes more complex. Every time you find one bug, there will be a new bottleneck.

Many bugs aren’t “bugs” until there’s high load on your system. For example, your IT team might shard data to improve performance, so that all product whose SKU code start with A-M is in server 1 and N-Z is in server 2. If you have one really hot item, all the users will be hitting that one database server, causing it to fail earlier than expected.

So even though horizontal scaling gives more flexibility for autoscaling and has lower administrative costs than vertical scaling, resource costs can skyrocket.

5. Server scaling's pitfalls are especially bad in ecommerce

Let’s assume you have autoscaling set up perfectly. There’s still a glaring problem most people gloss over.

You can't just take any off-the shelf software product and think you can scale it. There will be crippling bottlenecks in the application that you have just not thought about.

Websites are made up of cached and dynamic content. Cached content includes images, text, files, etc. that are similar across visitors. Think of your Home or Contact Us pages. These can be pre-loaded using a content delivery network (CDN) so your servers don’t have to deliver the content to each new visitor.

Dynamic content, however, is unique to the visitor and depends on actions he or she takes. The best example is when a visitor adds a product to a checkout cart. Behind the scenes there is a function that checks inventory, updates the database, and changes the cart icon to show the number of products it contains.

Dynamic content exists at critical parts of the customer journey, and delivering dynamic content has limits. The worst bottlenecks are the inventory system and the payment gateway.

Related: How Your Technology Stack Can Fail on Major Ecommerce Days

Inventory systems strive to preserve the database integrity for order and stock volumes. As the number of customers adding products to shopping carts explodes, inventory systems often can’t keep up. The complicated juggling act overheats the systems, harming website performance.

Payment gateways might be even worse. They’re third parties with throughput limits out of your control. All the server or autoscaling you do doesn’t change that. Plus, during big online sales payment processors are inundated with requests from other websites too. As a result, they’ll be slower than normal. All this helps explain why payment gateways are the point of failure for about 75% of companies we support.

The complexity of scaling explains why IT business expert Eric David Benari has said,

> “Performance has very little to do with database tuning and a lot more to do with reducing database requests.”

Learn how to control website traffic and control database requests

6. How over-reliance on scaling can let you down

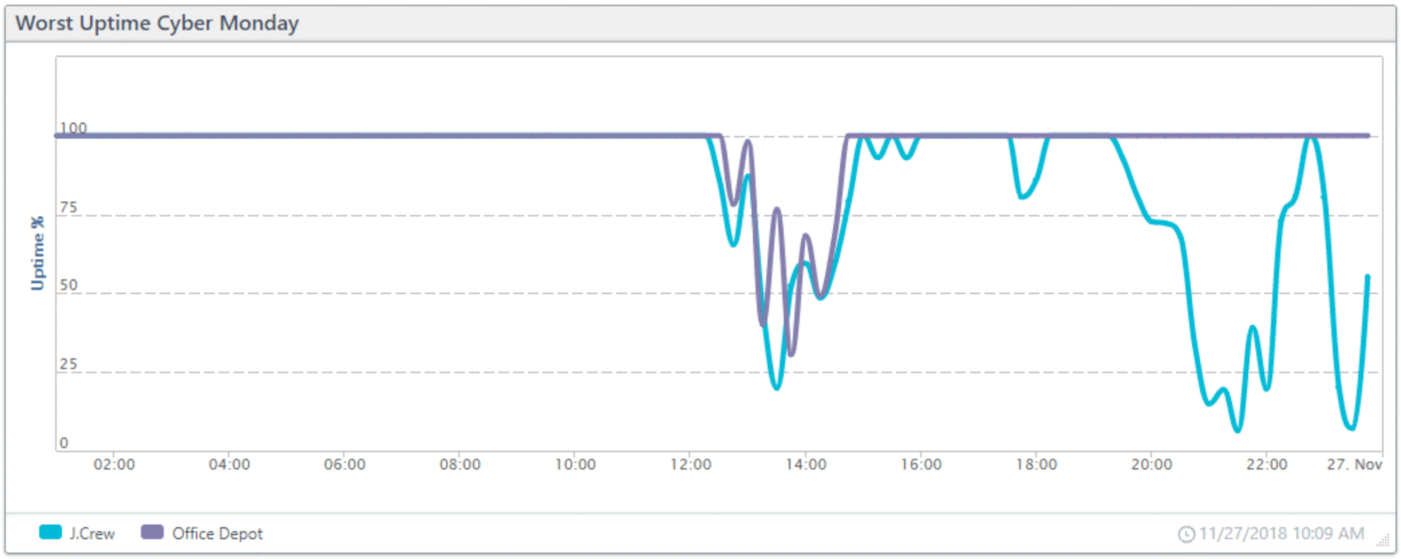
Picture this:

It's Cyber Monday 2018 at Office Depot and J. Crew.

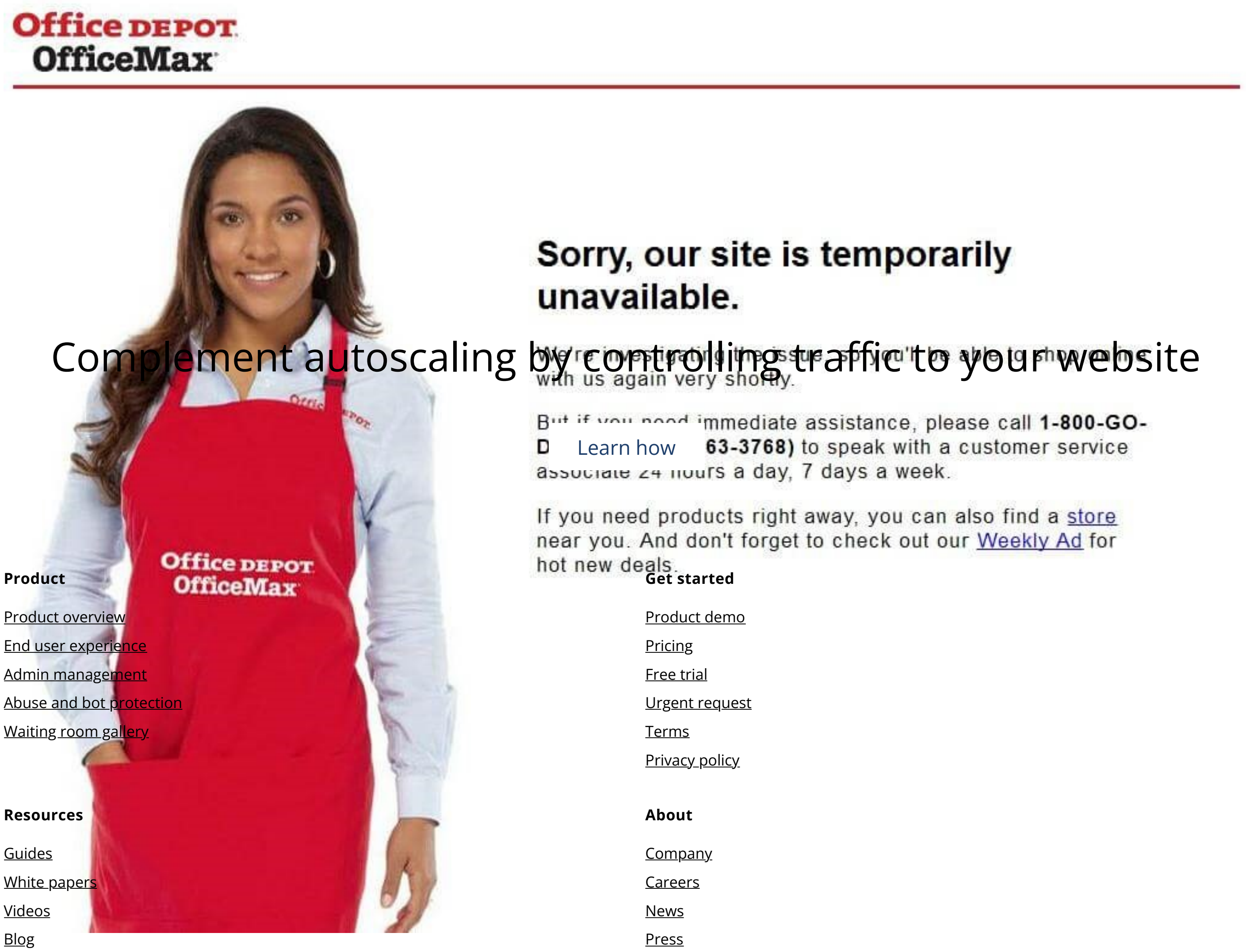
All hands are on deck for what will be the largest single day of online shopping in U.S. history—$7.9 billion in spending.

War rooms are filled with technicians monitoring dashboards. Teams are checking competitors' pricing. Marketing has several campaign tricks up their sleeve.

But when traffic comes streaming to the websites, they falter. Both show temporary error pages like this one:



Both websites suffered outages over several hours during the busy shopping day, leading shoppers to move on to competitors' websites. Experts estimate that J. Crew's failure during Black Friday, just a few days prior, cost the company over $700,000 in sales alone.


Office DEPOT OfficeMax

Sorry, our site is temporarily unavailable.

We're investigating the issue, so you'll be able to shop online with us again very shortly.

B[illegible] if [illegible] immediate assistance, please call **1-800-GO-** D[illegible] **63-3768)** to speak with a customer service [illegible] 24 hours a day, 7 days a week.

If you need products right away, you can also find a store near you. And don't forget to check out our Weekly Ad for hot new deals.

Complement autoscaling by controlling traffic to your website

Learn how

Product

Product overview
End user experience
Admin management
Abuse and bot protection
Waiting room gallery

Get started

Product demo
Pricing
Free trial
Urgent request
Terms
Privacy policy

Resources

Guides
White papers
Videos
Blog
Developers

About

Company
Careers
News
Press
Partners
Contact us

Use cases

Cases
Ecommerce
Ticketing
Public sector
Education

Follow us on social media

What went wrong?

At least with J. Crew, it wasn't that the website was too bulky. Its Google Pagespeed was a better-than-average 85/100. Rather, as Uptrends writes, "The small page size and higher performance score tell us that most likely J. Crew is a victim of its own marketing and does not have the infrastructure to support the influx of visitors [illegible]'s such as these."

In other words, despite attempts to scale up to prepare for the sale, performance bottlenecks took down two major online retailers during their most business-critical day of the year.

7. Summary

- Scaling up servers is one step you should take to prepare your website for increased demand.
- Vertical scaling (upgrading to higher-end servers) costs increase drastically once you get to higher levels.
- Horizontal scaling (connecting many smaller servers) is cheaper in terms of hardware costs but brings immense complexity in combining all the servers into a unified system that requires specialized—and expensive—personnel to manage.
- Scaling your website involves much more than just increasing server capacity, but when you scale your website you run into problems of search, concurrency, consistency, and speed.
- Server scaling does not address website bottlenecks like payment gateways. It takes a holistic strategy to build performance into a web application.
- Disregarding how difficult it is to scale servers and websites will put you at risk for failure during your busiest and most business-critical sales.

Scaling up your website is a natural first step in handling growing demand. But understand there are limits. Hopefully, this blog has helped explain in layman's terms why server scaling, autoscaling, and application scaling are not as easy as they might otherwise seem.