What it takes to run Stack Overflow

Nov 22, 2013

I like to think of Stack Overflow as running *with scale* but not *at scale*. By that I meant we run very efficiently, but I still don't think of us as "big", not yet. Let's throw out some numbers so you can get an idea of what scale we are at currently. Here are some quick numbers from **a 24 hour window** few days ago - November 12th, 2013 to be exact. These numbers are from a typical weekday and only include our active data center - what *we* host. Things like hits/bandwidth to our CDN are not included, they don't hit our network.

- 148,084,883 HTTP requests to our load balancer
- 36,095,312 of those were page loads
- 833,992,982,627 bytes (776 GB) of HTTP traffic sent
- 286,574,644,032 bytes (267 GB) total received
- 1,125,992,557,312 bytes (1,048 GB) total sent
- 334,572,103 SQL Queries (from HTTP requests alone)
- 412,865,051 Redis hits
- 3,603,418 Tag Engine requests
- 558,224,585 ms (155 hours) spent running SQL queries
- 99,346,916 ms (27 hours) spent on redis hits
- 132,384,059 ms (36 hours) spent on Tag Engine requests
- 2,728,177,045 ms (757 hours) spent processing in ASP.Net

(I should do a post on how we get those numbers quickly, and how just *having* those numbers is worth the effort)

Keep in mind these are for the entire Stack Exchange network but still don't include everything. With the exception of the 2 totals, these numbers are only from HTTP requests we log to look at performance. Also, whoa that's a lot of hours in a day, how do you do that? We like to call it magic, other people call it "multiple servers with multi-core processors" - but we'll stick with magic. Here's what runs the Stack Exchange network in that data center:

- 4 MS SQL Servers
- 11 IIS Web Servers
- 2 Redis Servers
- 3 Tag Engine servers (anything searching by tag hits this, e.g. /questions/tagged/c++)

- 3 elasticsearch servers
- 2 Load balancers (HAProxy)
- 2 Networks (each a Nexus 5596 + Fabric Extenders)
- 2 Cisco 5525-X ASAs (think Firewall)
- 2 Cisco 3945 Routers

Here's what that looks like:

200-240V
24A
4.8-5.8KVA
200-240V
24A
4.8-5.8KVA
CIRCUIT 1
CIRCUIT 2

We don't *only* run the sites, The rest of those servers in the nearest rack are VMs and other infrastructure for auxiliary things not involved in serving the sites directly, like deployments, domain controllers, monitoring, ops database for sysadmin goodies, etc. Of that list above, 2 SQL servers were backups only until *very* recently - they are now used for read-only loads so we can keep on scaling without thinking about it for even longer (this mainly consists of the Stack Exchange API). Two of those web servers are for dev and meta, running very little traffic.

Core Hardware

When you remove redundancy here's what Stack Exchange *needs* to run (while maintaining our current level of performance):

- 2 SQL servers (SO is on one, everything else on another…they could run on a single machine still having headroom though)
- 2 Web Servers (*maybe* 3, but I have faith in just 2)
- 1 Redis Server
- 1 Tag Engine server
- 1 elasticsearch server
- 1 Load balancer
- 1 Network
- 1 ASA
- 1 Router

(we really should test this one day by turning off equipment and seeing what the breaking point is)

Now there are a few VMs and such in the background to take care of other jobs, domain controllers, etc., but those are *extremely* lightweight and we're focusing on Stack Overflow itself and what it takes to render all the pages at full speed. If you want a full apples to apples, throw a single VMware server in for all of those stragglers. So that's not a large number of machines, but the specs on those machines typically aren't available in the cloud, not at reasonable prices. Here are some quick "scale up" server notes:

- SQL servers have 384 GB of memory with 1.8TB of SSD storage
- Redis servers have 96 GB of RAM
- elastic search servers 196 GB of RAM
- Tag engine servers have the fastest raw processors we can buy
- Network cores have 10 Gb of bandwidth *on each port*
- Web servers aren't that special at 32 GB and 2x quad core and 300 GB of SSD storage.
- Servers that don't have 2x 10Gb (e.g. SQL) have 4x 1 Gb of network bandwidth

Is 20 Gb massive overkill? You bet your ass it is, the active SQL servers average around 100-200 Mb out of that 20 Gb pipe. However, things like backups, rebuilds, etc. can completely saturate it due to how much memory and SSD storage is present, so it does serve a purpose.

Storage

We currently have about 2 TB of SQL data (1.06 TB / 1.63 TB across 18 SSDs on the first cluster, 889 GB / 1.45 TB across 4 SSDs on the second cluster), so that's what we'd need on the cloud (hmmm, there's that word again). Keep in mind that's all SSD. The average write time on any of our databases is **0 milliseconds**, it's not even at the unit we can measure because the storage handles it that well. With the database in memory and 2 levels of cache in front of it, Stack Overflow actually has a 40:60 read-write ratio. Yeah, you read that right, 60% of our database disk access is writes (you should know your read/write workload too). There's also storage for each web server - 2x 320GB SSDs in a RAID 1. The elastic boxes need about 300 GB a piece and do perform much better on SSDs (we write/re-index very frequently).

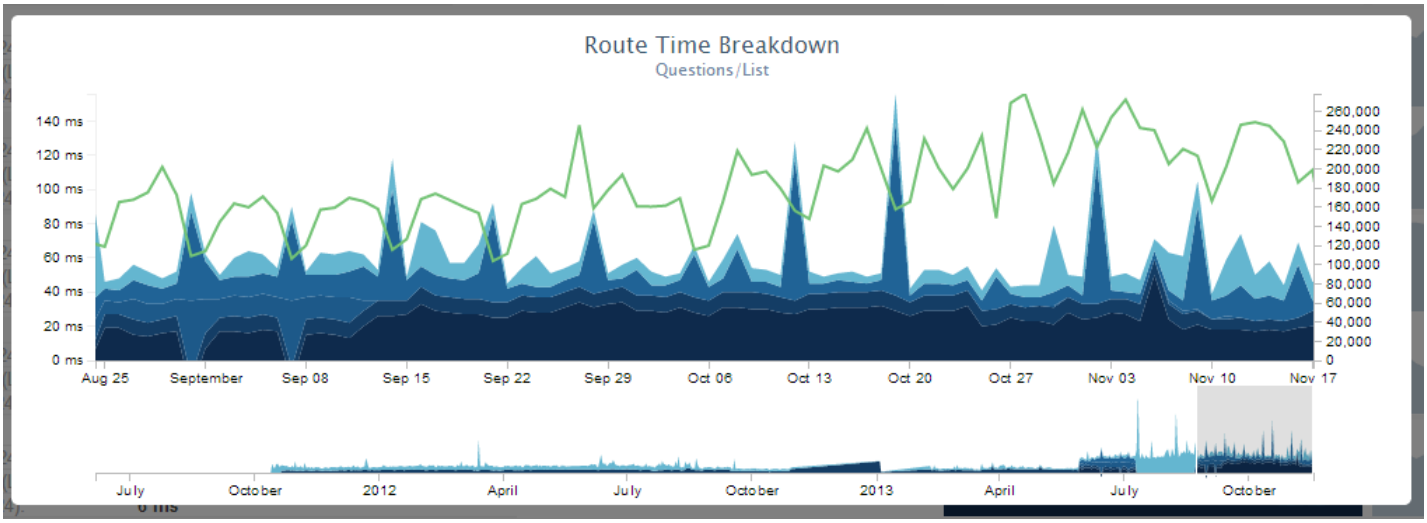
It's worth noting we do have a SAN, an Equal Logic PS6110X that's 24x900GB 10K SAS drives on a 2x 10Gb link (active/standby) to our core network. It's used exclusively for the VM servers as shared storage for high availability but does not really support hosting our websites. To put it another way, if the SAN died the sites would not even notice for a while (only the VM domain controllers are a factor).

Put it all together

Now, what does all that do? We want performance. We *need* performance. Performance is a feature, a very important feature to us. The main page loaded on all of our sites is the question page, affectionately known as Question/Show (its route name) internally. On November 12th, that page rendered in an average of **28 milliseconds**. While we strive to maintain 50ms, we *really* try and shave every possible millisecond off your pageload experience. All of our developers are certifiably anal curmudgeons when it comes to performance, so that helps keep times low as well. Here are the other top hit pages on SO, average render time on the same 24 hour period as above:

- Question/Show: 28 ms (29.7 million hits)
- User Profiles: 39 ms (1.7 million hits)
- Question List: 78 ms (1.1 million hits)
- Home page: 65 ms (1 million hits) *(that's very slow for us - Kevin Montrose will be fixing this perf soon: here's the main cause)*

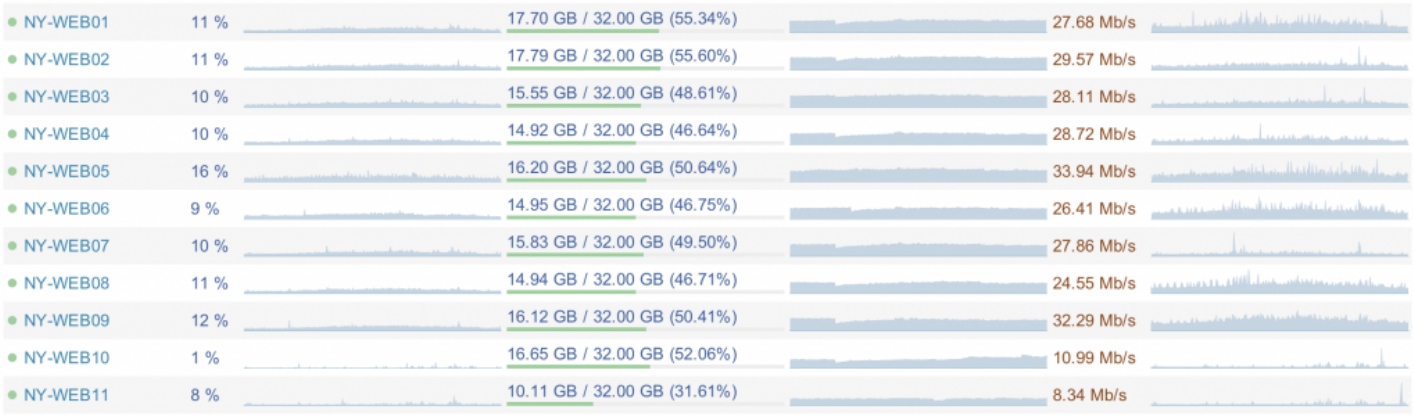
We have high visibility of what goes into our page loads by recording timings for *every single request* to our network. You need some sort of metrics like this, otherwise **what are you basing your decisions on?** With those metrics handy, we can make easy to access, easy to read views like this:



After that the percentage of hits drops off *dramatically*, but if you're curious about a specific page I'm happy to post those numbers too. I'm focusing on render time here because that's how long it takes our server to produce a webpage, the speed of transmission is an entirely different (though admittedly, very related) topic I'll cover in the future.

Room to grow

It's definitely worth noting that these servers run at *very* low utilization. Those web servers average between **5-15% CPU**, 15.5 GB of RAM used and 20-40 Mb/s network traffic. The SQL servers average around **5-10% CPU**, 365 GB of RAM used, and 100-200 Mb/s of network traffic. This affords us a few major things: general room to grow before we upgrade, headroom to stay online for when things go crazy (bad query, bad code, attacks, whatever it may be), and the ability to clock back on power if needed. Here's a view from Opserver of our web tier taken just now:



The primary reason the utilization is so low is efficient code. That's not the topic of this post, but efficient code is critical to stretching your hardware further. Anything you're doing that doesn't need doing costs more than *not doing it*, that continues to apply if it's a subset of your code that could be more efficient. That cost comes in the form of: power consumption, hardware cost (since you need more/bigger servers), developers understanding something more complicated (to be fair, this can go both ways, efficient isn't necessarily simple) and likely a slower page render - meaning less users

sticking around for another page load…or being less likely to come back. **The cost of inefficient code can be higher than you think.**

Now that we know how Stack Overflow performs on its current hardware, next time we can see why we don't run in the cloud.

Core Dev (Q&A site) ~6-7 people
Core Dev Mobile - 6 people
12 | • Reply • Share ›

Kasra Rahjerdi ➦ Wesley Workman
• 9 years ago
Good point.

Our main dev teams: SRE (System Reliability Engineer), Core Q&A (StackOverflow + SE sites), Core Mobile (Android/iOS devs), and Careers 2.0 are all around 6-7 people. However there's a lot of movement between them, Nick is pretty much in every single internal chatroom we have 24/7.
3 | • Reply • Share ›

Jimmy Chandra ➦ Kasra Rahjerdi
• 9 years ago
!!! Cactuar member :)
| • Reply • Share ›

Rubens Mariuzzo ➦ Wesley Workman • 9 years ago
I would love to know that too!
| • Reply • Share ›

NighthawkFoo • 9 years ago
Why do you run on a Windows stack instead of Linux + Apache?
13 | • Reply • Share ›

Jan Corazza ➦ NighthawkFoo • 9 years ago • edited
Because SE is built on .NET technologies and switching would make no sense. Also Microsoft does its best to make it unportable (when Mono got better than their own implementation, they simply stopped standardizing it).

And I though I had mentioned this but apparently I hadn't: most of their team are experienced with Microsoft technologies (especially Joel).
16 | • Reply • Share ›

Nick Craver **Mod** ➦ Jan Corazza
• 9 years ago
Portability has little to do with it, we are very happy with where we are - why spend a large amount of time and money to change? Would we change it from the start if

the team expertise was different? Sure, we would have used whatever that team knew best.

8 | • Reply • Share ›

Jan Corazza ➚ Nick Craver
• 9 years ago

Absolutely - switching to a totally different platform would make zero sense. I should probably edit the comment to make the separation between my answer and my little rant clearer...

6 | • Reply • Share ›

Steve Sims ➚ Jan Corazza
• 8 years ago • edited

double-post

| • Reply • Share ›

Steve Sims ➚ Nick Craver
• 8 years ago • edited

Not switching over when you're already on Windows is a separate question. Why didn't SO hire a team that could do Linux? Did Joel Spolsky personally do much of the early development? That would be my guess, I don't really know the history of SO beyond the Wikipedia article.

| • Reply • Share ›

Nick Craver **Mod** ➚ Steve Sims
• 8 years ago

Jeff Atwood was most comfortable in the .Net world at the time, as were Jarrod Dixon and Geoff Dalgas - so that's what they built. Also under BizSpark it's free to get started the first 3 years.

Go with what you know; smart people can make that scale and it saves a lot of time later. These days it's just fun to show people just how well .Net scales with efficient code :)

11 | • Reply • Share ›

dingo ➚ Jan Corazza • 8 years ago

@**Jan Corazza**: can you please give some data on "when Mono got better than their own implementation" part?

1 • Reply • Share ›

Jan Corazza ➔ dingo
• 8 years ago • edited
Here you go: http://stackoverflow.com/qu....
1 • Reply • Share ›

dingo ➔ Jan Corazza • 8 years ago
Thanks -- is there something more comprehensive than this? The article you pointed at only tests Math.Pow function, which hardly proves mono is better.
I'm sure there are some functions or microtests that do better in Mono, but I'm looking for something that more comprehensive and representative of the general usage.
2 • Reply • Share ›

Sam Wise ➔ NighthawkFoo • 8 years ago
Why not?
• Reply • Share ›

Gary • 9 years ago
How do you deal with votes? Are they all stored in a massive table or in multiple smaller tables? Do you update the cached version of a page every time there is a new vote on a comment or answer? I would imagine that could be quite often.

Thanks for writing this, it's really interesting to see how a busy site handles the load!
11 • Reply • Share ›

Nick Craver **Mod** ➔ Gary • 9 years ago
Votes are in 1 table per item, for example 1 table for post votes, 1 table for comment votes.
Most pages we render real-time, caching only for anonymous users. Given that, there's no cache to update - we just re-query.
5 • Reply • Share ›

Gary ➔ Nick Craver • 9 years ago
But the tables must be huge! It's amazing that the query returns in under 10 seconds. How big are the vote tables and what sort of info is in them? I assume the minimum is commentID, userID and voteValue?
4 • Reply • Share ›

Nick Craver Mod ➦ Gary
• 9 years ago
Well we also have the score denormalized, so we don't need to query often. It's all IDs and dates, the post votes table just has 56,454,478 rows currently. Most queries are just a few milliseconds due to indexing of what we're after.
10 ^ | v • Reply • Share ›

Gary ➦ Nick Craver • 9 years ago
Thanks for explaining - really interesting
4 ^ | v • Reply • Share ›

Alpesh Gediya ➦ Nick Craver
• 9 years ago
Nick , can you please elucidate further on Vote table part.
2 ^ | v • Reply • Share ›

tsilb • 9 years ago
"Hardware is Cheap, Programmers are Expensive" ~ Jeff Atwood
http://www.codinghorror.com...
10 ^ | v • Reply • Share ›

Yuri Fedoseev • 9 years ago • edited
Thank you, very interesting. SO is my main argument against ".Net stack is not for high load".
Can you write more about "efficient code". Thank you
6 ^ | v • Reply • Share ›

Charlie Barker ➦ Yuri Fedoseev • 8 years ago
For high load you need good engineers who know how to tune and optimise the code. The alternative is to get out the big cheque book and buy more/faster equipment. These days the stack is not the limiting factor.
2 ^ | v • Reply • Share ›

syntekz • 9 years ago
I use Stack Overflow from time to time to help with questions that arise at work, but I by no means use it on a consistent basis.

With that said, it is really interesting to see this type of data and information.

Would like to see this type of post made by other websites too.

Thanks for sharing!

4 | • Reply • Share ›

Joel Howard • 9 years ago

If you don't mind me asking... How are you guys handling SSL termination with HAproxy?

We also run HAproxy as our frontend load balancers and I've been running the development branch since it supports SSL termination.

3 | • Reply • Share ›

Kyle Brandt Joel Howard • 9 years ago

We were using Nginx for a while, but now we have completely transitioned to using HAProxy to terminate SSL.

1 | • Reply • Share ›

Eduard Luca Joel Howard • 9 years ago

The alternative to HAProxy 1.5 (which I believe is still beta -- was when I last checked), is to put an stunnel in front of HAProxy that handles requests coming in through HTTPS and sends them to HAProxy via HTTP. That's what I did and never had any issues with it.

| • Reply • Share ›

Nick Craver Mod Eduard Luca

• 9 years ago

Yes, we used to do this via nginx, but have moved to just HAProxy as it allows us much easier config management and more flexibility. I'm not saying everyone should, but in our setup the shift made a lot of sense.

2 | • Reply • Share ›

Eduard Luca Nick Craver

• 9 years ago

I believe it does make sense to only have 1 service handling everything in all cases, but aren't you worried that HAProxy 1.5 is still a dev build? I am, that's why we still didn't switch.

| • Reply • Share ›

Nick Craver Mod Eduard Luca

• 9 years ago

Not at all, it's been rock solid and Willy is awesome at fixing anything that comes up or

helping us diagnose a performance issue. There have been 2 minor ones since the 1.5 upgrade, and we very quickly solved both (neither affected running production).

If no one tests dev builds then they either don't get released or get released with more bugs. We have the tech staff to quickly troubleshoot anything that arises, so by testing things like this under pretty decent load it's just another thing we can do to help other developers and sysadmins. If there's a feature we want in a dev/beta version, we'll go with it as soon as it's stable enough. For example, we are also running SQL 2014 CTP2.

7 ^ | v • Reply • Share ›

Guest ➚ Joel Howard • 9 years ago

You need haproxy 1.5 to support multiple SSL backends. And needs to be compiled as such:

- USE_PCRE=1 to use libpcre, in whatever form is available on your system (shared or static)

- USE_STATIC_PCRE=1 to use a static version of libpcre even if the dynamic one is available. This will enhance portability.

- USE_OPENSSL=1

- USE_ZLIB=1

^ | v • Reply • Share ›

Andy • 9 years ago

What software do you use for your Tag Engine?

2 ^ | v • Reply • Share ›

Nick Craver **Mod** ➚ Andy • 9 years ago

It's called "tag engine" :) It's code written just for our specific use cases and highly optimized for them...it wouldn't be useful to anyone else. We do some performance tricks with the code which Marc Gravell and Sam Saffron have blogged about a bit, that's more useful to other devs than the raw code would be IMO.

1 ^ | v • Reply • Share ›

Andy ➚ Nick Craver • 9 years ago

Lee. Can you point me to the blog post you

I see. Can you point me to the blog post you mentioned? Thanks.

1 • Reply • Share ›

Stuart Blackler Andy • 9 years ago

These posts talk about the initial re-write of the tag engine of doom. From there you should be able to find more info. Hth :)

http://marcgravell.blogspot...