Reddit

[Reddit: Lessons Learned From Mistakes Made Scaling To 1 Billion Pageviews A Month]

[7 Lessons Learned While Building Reddit To 270 Million Page Views A Month]

As a decade-long user of Reddit since its inception in 2005, it has grown into one of the largest websites across the internet. Now it has become a worldwide discussion website that allows its members to submit content either through text posts or direct links and share. It allows categorization of these posts into “subreddits”, which allow for more specialized content and gathering of users for the same purpose. Though these two articles are rather dated it shows that even in 2013, Reddit was able to support 1 billion page views a month, on top of that, there are massive amounts of data that must be stored, managed, and backed up in the form of text files, images, audio files, and even videos depending if users utilize Reddit’s own multimedia manager.

In general, they serve a broad market of users that simply want to get together, discuss, argue, and all in all, learn. At the start, the website was situated in a datacenter but as users and content grew, the costs of maintaining and monitoring these servers also grew as well. Over the next few years, Reddit slowly moved functionality towards AWS’s EC2 for storage of logos and thumbnails and soon enough they had 240 servers supporting 2+ billion pageviews a month and more than 2TB of data stored in their database. In due time, Reddit fully moved to AWS’ EC2 service and decommissioned their physical servers and datacenter.

Though the article itself does not mention the operating system, looking through the other products and stack technologies that it does utilize, Reddit’s information architecture appears to run off a variant of an open source operating system, though it is not clear exactly what but because of the products like Apache, Python, PostgreSQL and more. The fact that Reddit does have its own GitHub account that pushes code into its own production is surprising as well. Through further analysis of their GitHub account, it appears that they may run off of Ubuntu Linux 14.04 based off of their install script to spin up a private instance of Reddit with even its own Vagrantfile to help speed up the installation. Because of the small start that Reddit had with only a handful of employees, a Linux operating system simply appeared to be the most simple and cost-effective operating system at the time and by the time Reddit became much larger like it is today, the company probably saw no need to migrate to a Windows OS at all.

Using Siftery as well, I found that the site is built through Python, with the use of Django/Pylons, which are python web frameworks, and supplemented by JavaScript frameworks, Backbone.js. This is run on nginx as the web server as well to terminate SSL and serve static content while HAProxy is for aiding in load balancing. . It has users connect to a web interface which communications with the application layer to utilize: memcache, Cassandra, and Postgres. As it sounds, memcache is utilized for caching to allow users to speed up dynamic web applications and load up content, and in fact Daniel Ellis, a Senior

Software Engineer, actually posted on the RedditBlog an article all about “Caching at Reddit” on January 17th, 2017. Apache Cassandra is utilized for its voting system, whereas Postgres is utilized for much of data storage.

Reddit is a private company, so it does not have any stock offerings and what not. There have been many obstacles with Reddit despite being more than a decade old to hammer out the kinks. Some of which was a javascript vulnerability that allowed users to inject malicious javascript code, but due to the open source nature of Reddit, was quickly patched. Other obstacles came especially with the transition to being fully hosted on the cloud, and even today during peak hours, users can encounter latency, inability to load, searching functions crashing and much more that caused client-side problems of course. In fact, in the first article, it mentions that the key to scaling is finding the bottlenecks before users do, however, that also requires key monitoring, but at the start there were issues with monitoring because their system was not virtualization friendly which led to much downtime and latency. As these issues were fixed, the architecture was modified to allow for more easily scaling in times of peak traffic. As load balancers were introduced into the stack, separation of services also became a thing to help group like processes and data on different boxes to ensure that it remains clean and the “boxes” can scale as needed without the worry traffic bottlenecks due to data storage. Other issues came with using bleeding edge products and pushing its usage into production instead of allowing for further testing especially when these products were still early in its development cycle. Soon enough, everything was automated and everything was memcached which now allows for fast speeds when searching, loading, rendering etc.

All in all, these two articles provided good insight that needs to be kept in mind when developing a web application, especially with the ideas of caching, data retrieval, and scalability. Additionally, these provided good insight on the different tools that were used.

Stubhub

[StubHub Architecture: The Surprising Complexity Behind The World’s Largest Ticket Marketplace]

As someone who often uses StubHub to purchase tickets to go to music concerts and events, I was glad to learn that StubHub has been praised for its complexity and ability to thrive in the cutthroat entertainment e-commerce sector. Though this article was written in 2012, its numbers were still significant and I can only imagine the growth it has seen in the past 5 years. At the time, StubHub was growing at 20% a year, with hundreds of thousands of page requests, and making millions of transactions per year.

Because of the nature of entertainment, traffic is unpredictable and can be centered around game outcomes, events, schedules, announcements etc and as a result there are a lot of complex factors that must be accounted for in its real-time ticket market component. Though its “marketplace” environment may turn many off, it’s also been a great way to find some cheap tickets last minute as well and for added security it does implement an escrow model to provide trusty and safety as sort of a “middleman” in terms of transactions. StubHub has three sources of transaction mechanisms, through the web, point of sale systems, and bulk upload, all of which provide the information architecture of the organization.

Once again, the article does not have a clear operating system but looking through the platform and technology stack from Siftery, it does appear to utilize another open source operating system but no specific model and version. Since it is primarily an e-commerce organization, it does have a lot of services and products that focus on marketing through search engine optimization, consumer data analytics and various social media dashboards and plugins. However, it does appear to use Java but from looking through the platform and the stack, it appears that StubHub is more of an amalgamation of 3rd party vendor products that are integrated into this organization and there is very few in-house programming and design and despite researching throughout the internet, I cannot seem to find many specifics regarding the technologies being used.

It seems that most of the technologies do utilize the database, which is Oracle, because it relies so heavily on smoothing out and balancing the load for scalability as well as provide a backbone for a listing catalog service. It uses LCS, Listing Catalog Service to help keep a table up to date with changes as they occur in the database as well as helping to form an interactive graphic for selecting tickets and other shopping experience developments. Like Reddit, StubHub heavily uses memcached to help with the distribution and transactions of tickets to the sellers. Lastly, StubHub uses a technology called Solr which is used for searching with flexible output formats as needed. StubHub itself is a private company, however, it is owned by eBay, which is a publicly traded company.

Because of the conditions and obstacles that an e-commerce company can face in unpredictable situations, an obstacle is clearly providing consistent service and ability to transition quickly and scale as needed with bursty traffic all the while providing a safe buying experience and not escalating into a digital scalper as it has been called on many occasions. On the note of a safe buying experience, StubHub has to ensure it is able to provide a safe experience through integration between systems and ensure there is a failsafe in case of failure points in the transaction, which brings it back to consistent service to prevent situations like that.