DESIGN QUORA

The following document is detailed: Purpose, Scope, Architecture and Diagrams, and Original Sources

1. Purpose:

The document, a high-level architectural view of Quora, is meant as an aid to prepare for system design interviews. It is compiled from the information on Quora's technology blog and its presentations. Its focus is to detail Quora's specific choices and hence it leaves out generic topics like consistent hashing, CAP theorem etc. All sources are cited in the "Original Sources" section at this document's end.

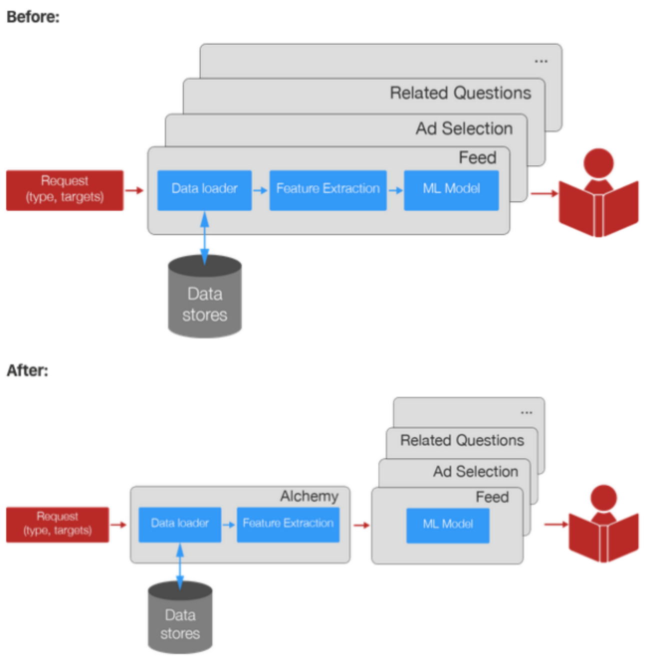
Quora is a question-and-answer website where users can ask questions, post answers, and comment on said questions. It allows users to also see engagement statistics such as views, upvotes, and shares. The website has nearly 300 mil unique visitors every month (2018).

2. Scope (Details):

The scope of this section: Quora's home feed, email digest, disaster recovery/maintainability, full-text search, and their unique counting system.

Not included: Quora's recent efforts with Kubernetes, CI/CD, answer ranking, semantic question matching, neural nets for recommendation, question correction, data pipeline for ML in A2A. Still, I've linked the URLs for those interested in these topics at the bottom of this "Scope" section.

Home Feed – Quora built Alchemy, a high-performance stateless service for machine learning features to help generate their feeds. Alchemy's compute cluster provides real-time responses. What is a compute cluster? A compute cluster is nothing more than a group of connected computers that are independently solving the same problem. The cluster will appear as "one" computer doing "one" task. (Note, the 2019 article on Feature Engineering seems to supersede the 2016 RecSys lecture article).



The key idea of a feed is relevancy to the user. To generate the home feed of a user, machine learning relies on numerous features like upvotes, views etc. that a given post might have. These features could be hundreds per candidate, with thousands of candidates.

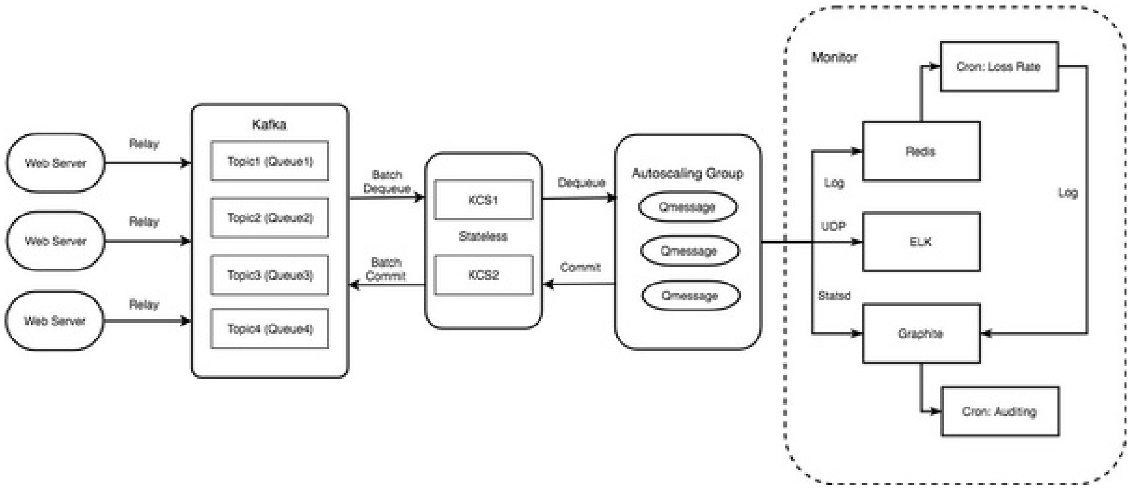
In addition to the home feed, other such as ad selection or related questions would also have their own extractions that need to be calculated. Instead of having feature extraction happening in all three separately. Feature extraction was made into a service. That could feed into ML models for home feed, ad selection, or related questions.

The requests to Alchemy can be either real time or asynchronous. Alchemy uses HBase as a storage layer and Memcached on top of that. To help prevent stale data in Memcached, Quora uses Kafka. This works as the following: The application queues a request. Alchemy picks up that request from the queue. It then knows what it needs to take from the database.

Daily Digest Emails – Quora built its own async task queue called QMessage. QMessage is used for tasks such as push notifications and digest emails. Qmessage processes 15K tasks/second for 500 types of tasks with a latency of 2 minutes. <u>Why did they decide to build QMessage and why did they choose what they did?</u> Their initial queue tool required manual engineering anytime there was a failure. Redis was in-memory so it too had recovery issues in case of a failure. The backbone of Qmessage is Apache Kakfa. <u>What is Kafka?</u> Kafka is simply a distributed streaming platform, with a pub/sub model at its heart. Although Kafka can be used in a couple of ways, Quora uses the Kafka relay, and Kafka consumer service in conjunction with their QMessage.

The web servers relay the information they need to the topics inside the Kafka topics. This is the Kakfa relay step. The topics are dequeued in a batch by the Kafka consumer service. The consumer service stores the request in memory until a worker completes the tasks.

The QMessage dequeue the requests from the consumer service. After the task is executed, the consumer service commits the response to the response back to Kafka. The consumer service is stateless. The QMessage is auto scaled on top of EC2 instances - a cron job helps with this.



(Web Server -> Kafka Topic / Kafka Relay ->Kafka Consumer Service -> Qmessage -> Execute task)

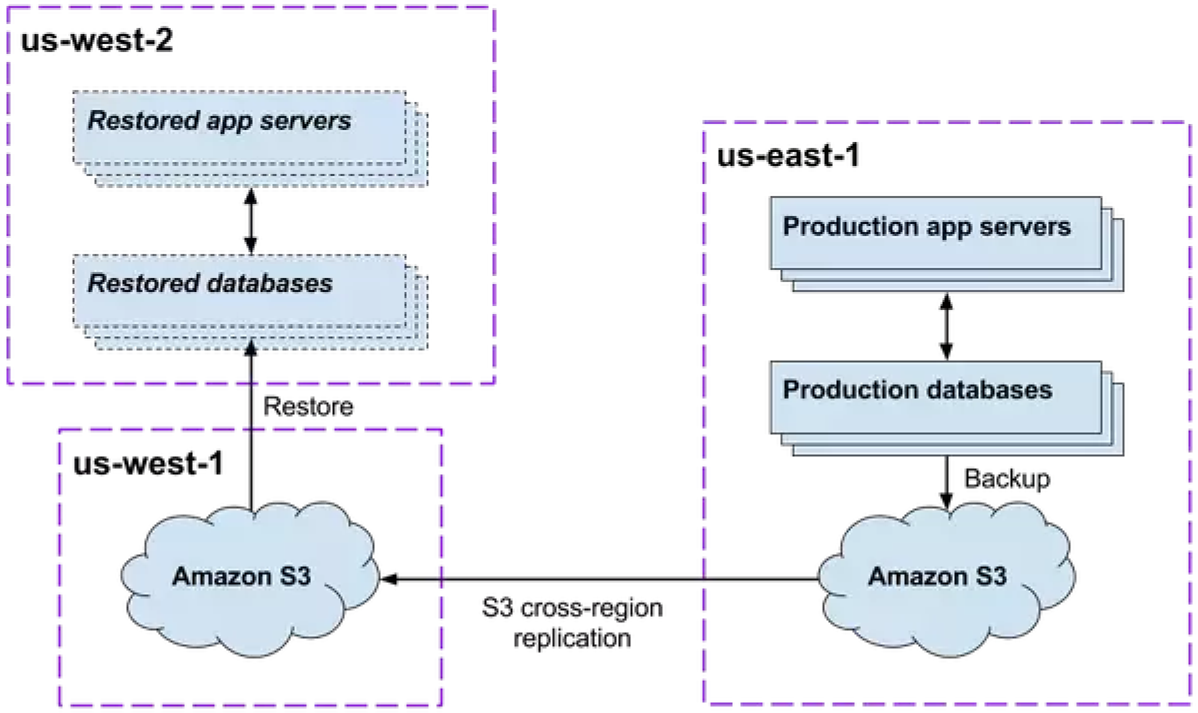
Disaster Recovery/Maintenance - Let's look at how Quora simulated their disaster recovery game plan and how they would restore their service from their replicated backup data stores (MySQL, HBase, Redis) - if there were a catastrophic meltdown

The questions the engineers asked themselves were which parts of the system are critical and which are not, can everything be restored from the backups, and how can they improve the time it takes to restore everything.

To prepare they made sure that one AWS region was working in order to replicate, ensure restore scripts would not interfere with current live systems and which instances were critical in each reason.

It took about 5 hours and the main recovery steps are as follows: start a virtual private cloud, start all data stores from their last available states, web package generator from the CI/CD side, ELB to accept incoming traffic, spin up a PyCache in S3 bucket. This allowed them to finally start web servers, task workers, dependency trackers and the remaining.

This diagram summarizes the backup and replication path that we used for Game Day:



Full Text Search - Initially in 2011, user search was restricted to the titles of the questions. However, since 2013 Quora allows for full-text search. Though I haven't found the exact search engine they currently use, Elastic Search seems to be what they use as per their latest updates.

<u>What?</u> Elastic Search is distributed search and analytics engineer for all types of data including textual, numerical, geospatial, structured, and unstructured. It is built on top of Lucene. Lucene is nothing more than just an indexing and searching library- a text search library written in Java. By itself Lucene is not a database.

When? Use cases include application search, website search, logging analytics

Why? Can be used for real-time queries and it distributed across shards allowing for horizontal scalability.

How? It can be used with simple REST APIs. Data can go into Elastic Search from logs, web apps. It is ingested and indexed. The indexed are just a collection of documents that it has grouped together in the format of JSON docs. The key point is that Elastic Search used inverted index - like the word to page number in a textbook index.

Counting System – Quora built Quanta as their counting system to provide real-time capabilities for things such as views and dashboard information. They had 100K writes per second, more than 1M reads per second.

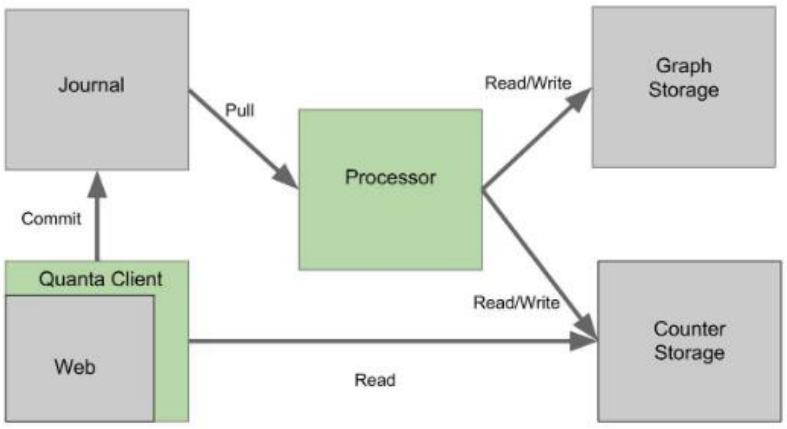
What were they trying to solve?

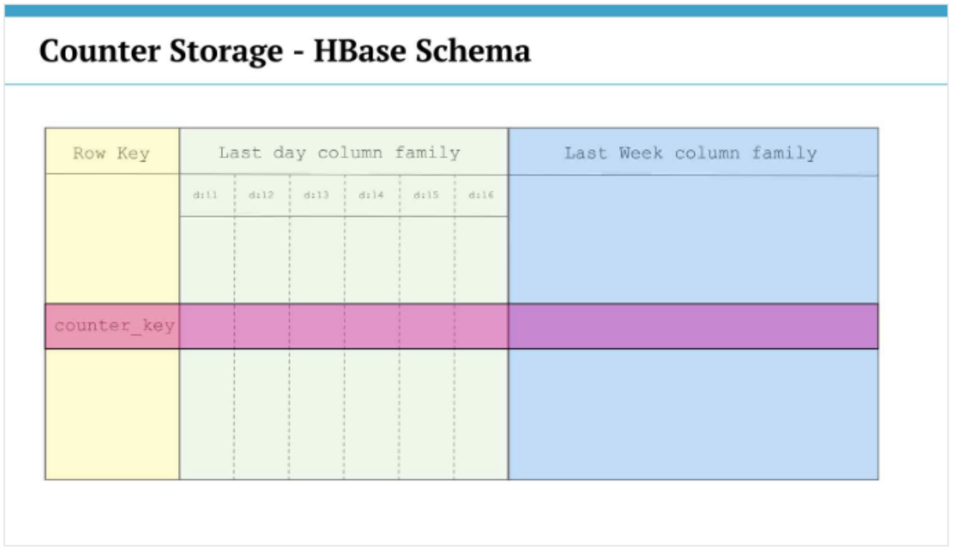
- Once data is committed, it should be permanent and replicated. Have billions of counters not bounded by a single machine.

- Of course, they want low latency for counter requests. Writes can delay a few minutes, not reads

- Support queries like, "how many times did X happen in the last Y days?"

Quora also wanted dynamic counter aggregation in real time. If for example, I am a user who posted two questions. Each of my questions might get 5 views. My profile might get 10 views. My total views should read 20 views. This can be likened to a DAG. The two questions are the nodes that point to a node that represents me. Every node gets eagerly updated. There is a graph update algorithm which powers Quanta. Below is the HLD of quant and the Counter Storage Schema





The Quanta Client sits on a web machine that processes web requests. Client is thin and can be written in any language. Directly reads from counter storage.

Journal is a persistent commit log. Stores data temporarily until processor pulls it to process. It uses Kafka. Kafka enables scalability.

Processor pulls data from queue. Does processing in memory. It does batch asynchronous updates, so it can remain stateless. Stateless means it doesn't have to store info and so they don't worry about availability. This is made possible by their distributed cron architecture. Processor is the only part of Quanta that isn't readily scalable. So how did they scale it? How did they achieve horizontal scalability? They shared the processor and kept watch for race conditions. They sharded it into 2 parts -one for graph and one for counter. Synchronize all processors using a wide shared lock.

Counter Storage is the one storing the counting data. It's a large hash map with a lot of map data. Built on HBase, it has a high write throughput. It allows range scan for related count scans. Variable number of columns allow for time-series and rolling windows. And automatic TTL per column family performing a garbage collection.

Graph storage stores the graph, nodes etc. It is also built on HBase. It is stored as a sparse adjacency matrix. Schema has 2 column family, one for outgoing edges and one for ingoing edges. HBase can serve negative queries through Bloom Filter - great for graph traversals. Bloom filters can help detect which nodes are on the outer parts of a graph. Think of convex hull algorithm. Interestingly, Quanta has something called multi-level graphs – which amazed me.

Kubernetes - https://www.quora.com/q/quoraengineering/Adopting-Kubernetes-at-Quora

Continuous Deployment - https://www.quora.com/q/quoraengineering/Continuous-Deployment-at-Quora

Answer Ranking - https://www.quora.com/q/quoraengineering/A-Machine-Learning-Approach-to-Ranking-Answers-on-Quora

Semantic Question Matching - https://www.quora.com/q/quoraengineering/Semantic-Question-Matching-with-Deep-Learning

Neural Nets - https://www.quora.com/q/quoraengineering/Unifying-dense-and-sparse-features-for-neural-networks

Question Correction - https://www.quora.com/q/quoraengineering/Using-natural-language-models-for-question-correction

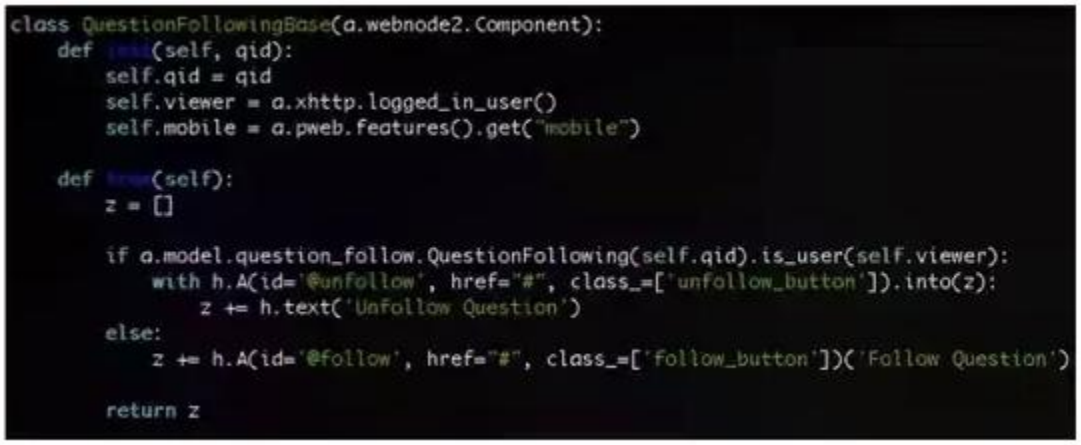
Data Pipeline for A2A - https://www.quora.com/q/quoraengineering/Seeking-Information-Using-Search-and-Question-Askingbbbbbbbb

3. Architecture and Diagrams:

Webnode2, Livenode, Comet, App Server

Webnode2 and LiveNode were two of Quora's original internal systems to manage content shown to users. What is Webnode2 and LiveNode?

Webnode2 is a python library. It acts as an abstraction for the components on a webpage. WebNode2 handles the AJAX request and renders HTML. Here is an example of how it renders HTML.



```
class QuestionFollowingBase(a.webnode2.Component):
    def init(self, qid):
        self.qid = qid
        self.viewer = a.xhttp.logged_in_user()
        self.mobile = a.pweb.features().get("mobile")

    def html(self):
        z = []

        if a.model.question_follow.QuestionFollowing(self.qid).is_user(self.viewer):
            with h.A(id='@unfollow', href="#", class_=['unfollow_button']).into(z):
                z += h.text('Unfollow Question')
        else:
            z += h.A(id='@follow', href="#", class_=['follow_button'])('Follow Question')

        return z
```

LiveNode is a framework that manages WebNode2. It supports real-time updating of pages that have been already loaded. For example, a new answer is pushed to a viewer looking at a question. When there is a change, you do not need to update the whole window. Simply the few components.

The basic gist of these 2 working together is that components on a page are updated and reloaded only on the basis of user interaction.

Quora used the web design paradigm Comet. What is Comet? Unlike a typical request/response relationship that a standard web client might have with a server, the idea behind Comet paradigm is that persistent connections are started by the client and kept open until data is available. This allows for near real-time communication. Out of the two methods available: namely streaming and long polling, Quora depends on long polling (at least in 2011). What is the difference between streaming and long polling? Streaming is a comet connection that is kept open indefinitely. Long polling is a comet connection that is kept open until data is available to the client.

Each browser tab to the Quora website can have several connections. To deal with this many open connections, Quora used Nginx. What is Nginx? It is an open source HTTP web server and reverse proxy server. It can also act as a load balance and HTTP cache server. Quora uses it for proxying requests. To proxy a request means to control client requests without exposing the server components directly.

Now, to properly support Comet programming, Quora required event driven web servers to handle the large number of long polling requests. Quora used the Tornado web framework. What is Tornado? Tornado is a Python web framework which is an application level server. It meets the demands that are placed on a server in Comet setup.

Databases, Caching

Initially Quora started with SQL. Since most of the user generated content on Quora is text and structured, they primarily used SQL databases (2013). A lot of documentation is available on SQL. Let's go over what Quora uniquely does with their data storage. For their cache they use Memcached. What is memcached? It is simply an in-memory key-value store. It holds strings or objects which can be used for DB or API calls and page rendering to save database query time. It stores the data in what are called slabs. A slab contains KV info and metadata. Let's talk about the specific advantages and disadvantages of it as described by the engineers at Quora.

Advantages- Preventing DB overload when for example, querying the list of people who upvoted a question. It is also multithreaded.

Disadvatanges- Networking is expensive in a memcached get. That's why they built Async. What is Async? It is an abstraction framework on top of Memcached that allows developers to write batched cache requests instead of firing individual ones. Async can run either synchronously or asynchronously. Under the hood you can think of these to be a DAG dependency that resolves the dependencies of your current query before allowing other queries to continue accessing the same resource. Code for getting the names of people who upvoted something.

```
from asynq import async

@async()
def get_upvoter_names(aid):
    upvoter_uids = yield upvoters_of_answer.async(aid)
    names = yield [name_of_user.async(uid) for uid in
upvoter_uids]
    return names
```

Another disadvantage of memcached is that it does not allow you to easily monitor performance and efficiency. To optimize their cache performance, they had to keep track of how many resources each function used. To do that they built a custom software called MCInspector. What is MCInspector? It is simply a tool to get the summary of objects and keys of a running memcached process without interrupting it.

MCInspector does this in 3 steps. First, it copies the data in a running memcached process from a virtual filesystem. Second, it parses the copied data to look for specific headers. Third, it aggregates the data and groups it by metadata.

So what? What happens then? MCInspector shows the inefficient hotspots. To improve performance, they shortened keyvalues and improved serialization methods. Memcached uses lazy expiration eviction policy. But by uniquely identifying resources with MCInspector, engineers can purge it memory immediately. A whopping 30% less queries to the DB!

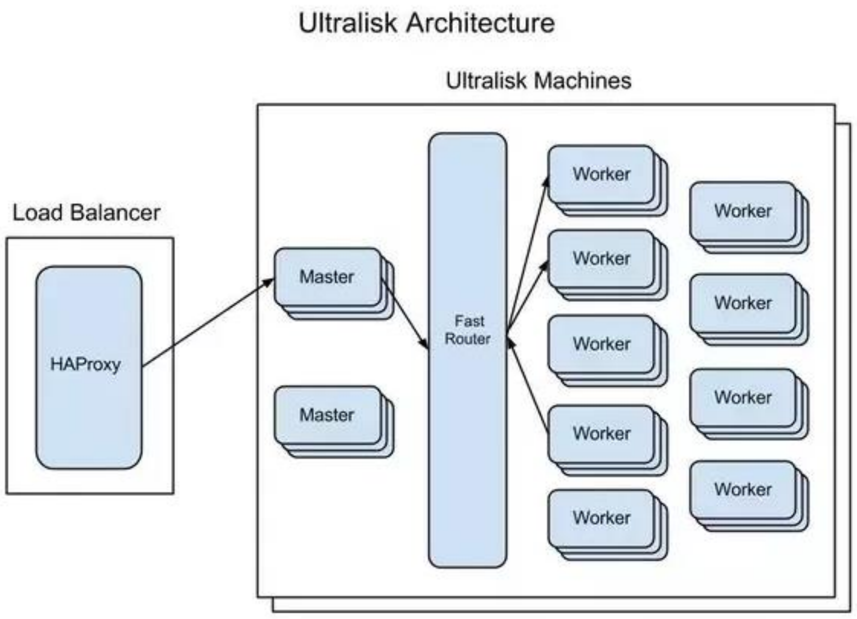
Quora has 4 main levels of data storage:

1. Data warehouse systems. Hive and Redshift. Not used for serving web requests but rather internal analytics. These are generally used for creating reports through ETL.
2. Main DBs. They can be used for serving web requests. SQL and HBase. HBase is a wide-column database.
3. Remote in-memory caching. Used for web requests. Redis and Memcached.
4. Local in-memory caching. They implemented a shared-memory version of Pycache

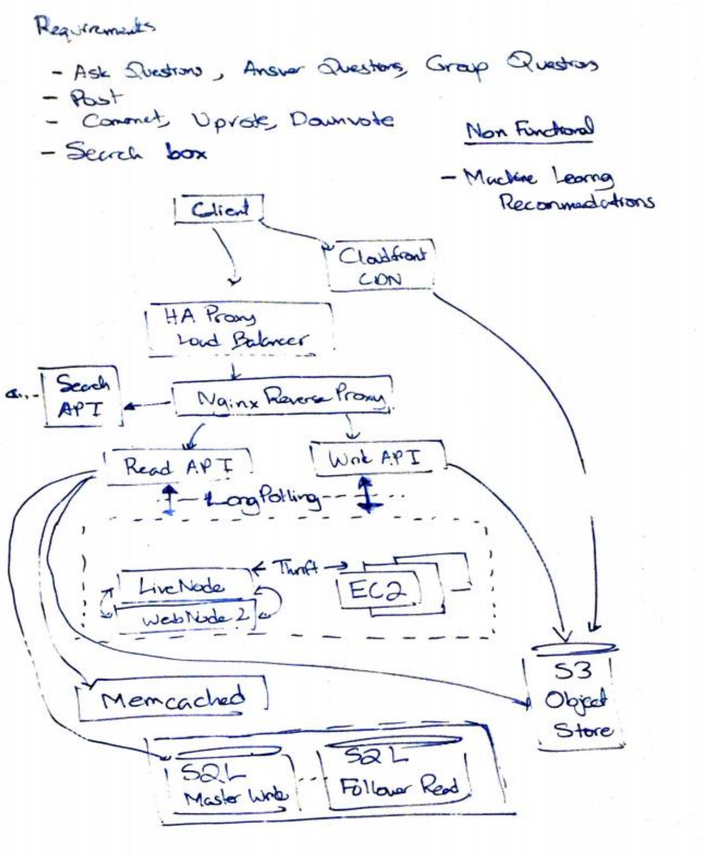
Web Server Architecture, Ultralisk (2013)

Quora had always followed master-worker pattern for their parallel architecture to render webpages. What is master-worker pattern? This is NOT to be confused with master-slave database replication. This master-worker pattern refers to how their web servers are designated and utilized in conjunction to serve requests. Think of the master server as running the main Python processes. The worker servers have smaller Python processes. There is another server which is called fast router writer in C++ which delineates the tasks coming in from the master to the worker. The load balancer send a request to the master. Master asks the workers through the fast router to work on the same task independently. The workers do not talk to each other. The master then receives the combined answer and decides how it will respond to the request.

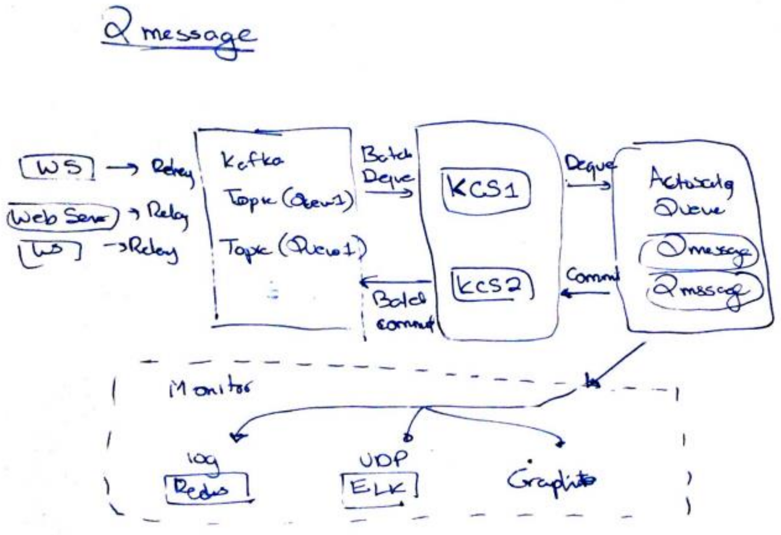
The value of Ultralisk machines from their previous architecture is the placement of these three servers: master, fastrouter, and worker. Initially they were scattered on different machines. However, now they are all on a 32 core EC2 instance. Because master, fastrouter, and worker are on one instance there is a whopping 20-35% increase in speed.



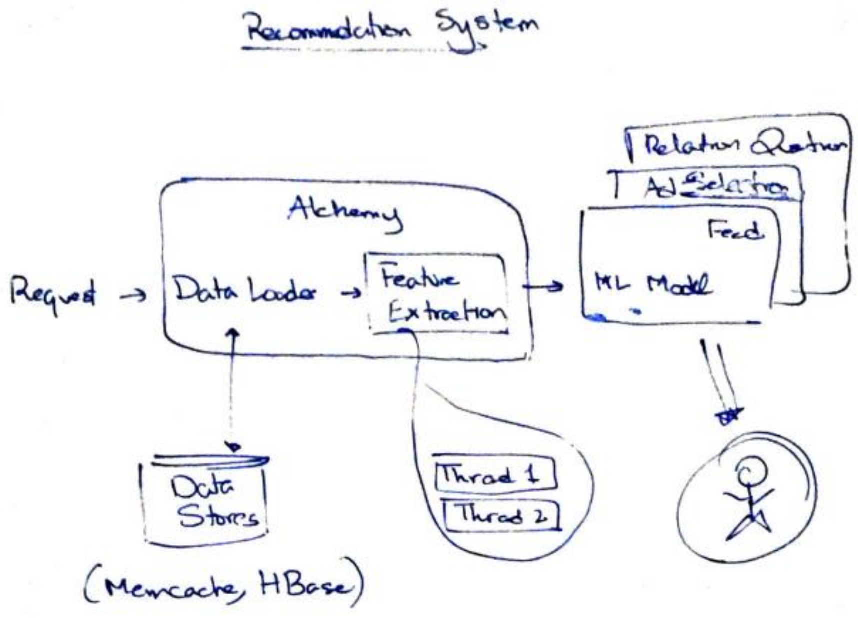
My hand drawn diagrams…



Basic HLD Design (Old Design 2011)



Quora’s QMessage



Alchemy Application

4. Original Sources

http://www.bigfastblog.com/quoras-technology-examined#static-content

https://www.quora.com/Whysending%20d%20AAlc-does-Quora-use-MySQL-as-the-data-store-instead-of-NoSQLs-such-as-Cassandra-MongoDB-or-CouchDB-Are-they-doing-any-JOINs-over-MySQL-Are-there-plans-to-switch-to-another-DB

https://www.quora.com/Quoras-Technical-Infrastructure-What-is-Webnode2
https://www.quora.com/q/hrpvzqrskfvulueb/Tech-Talk-webnode2-http-www-ryin-main-quora-com-topic-webnode2-1-and-LiveNode-http-www-ryin-main-quora-com-
thttps://www.quora.com/q/quoraengineering/Seeking-Information-Using-Search-and-Question-Asking
https://github.com/facebookarchive/scribe

https://www.quora.com/q/quoraengineering/Feature-Engineering-at-Quora-with-Alchemy

https://www.quora.com/q/quoraengineering

https://www.quora.com/q/quoraengineering/Web-Server-Architecture-at-Quora

https://www.quora.com/q/quoraengineering/Seeking-Information-Using-Search-and-Question-Asking

https://github.com/binhnguyennus/awesome-scalability

https://www.quora.com/q/quoraengineering/Adopting-Kubernetes-at-Quora
https://www.quora.com/q/quoraengineering/Pycache-lightning-fast-in-process-caching
https://www.quora.com/q/quoraengineering/Quoras-Distributed-Cron-Architecture
https://www.quora.com/q/quoraengineering/Qmessage-Handling-Billions-of-Tasks-Per-Day
https://www.quora.com/q/quoraengineering/Logging-and-Aggregation-at-Quora
https://www.quora.com/q/quoraengineering/Optimizing-Memcached-Efficiencyhttps://www.quora.com/q/quoraengineering/Ensuring-Quoras-Resilience-to-Disaster
https://www.quora.com/q/quoraengineering/A-Machine-Learning-Approach-to-Ranking-Answers-on-Quora
https://www.infoq.com/presentations/quora-analytics/
https://www.pubnub.com/learn/glossary/what-is-comet-programming/
https://www.webopedia.com/TERM/N/nginx.html
https://www.igvita.com/2009/10/21/nginx-comet-low-latency-server-push/
http://memcached.org/
https://github.com/quora/mcinspector
https://www.elastic.co/what-is/elasticsearch