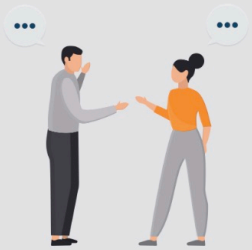
Designing Quora ?

Search engine	People
	
Fast	Slow
Shallow	Deep
Read-only	Conversational

Quora is a social question-and-answer service that allows users to ask questions to other users. Quora was created because of the issue that asking questions from search engines results in fast answers but shallow information. Instead, we can ask the general public, which feels more conversational and can result in deeper understanding, even if it's slower. More than 300 million monthly active users post thousands of questions daily related to more than 400,000 topics on Quora.

Requirements

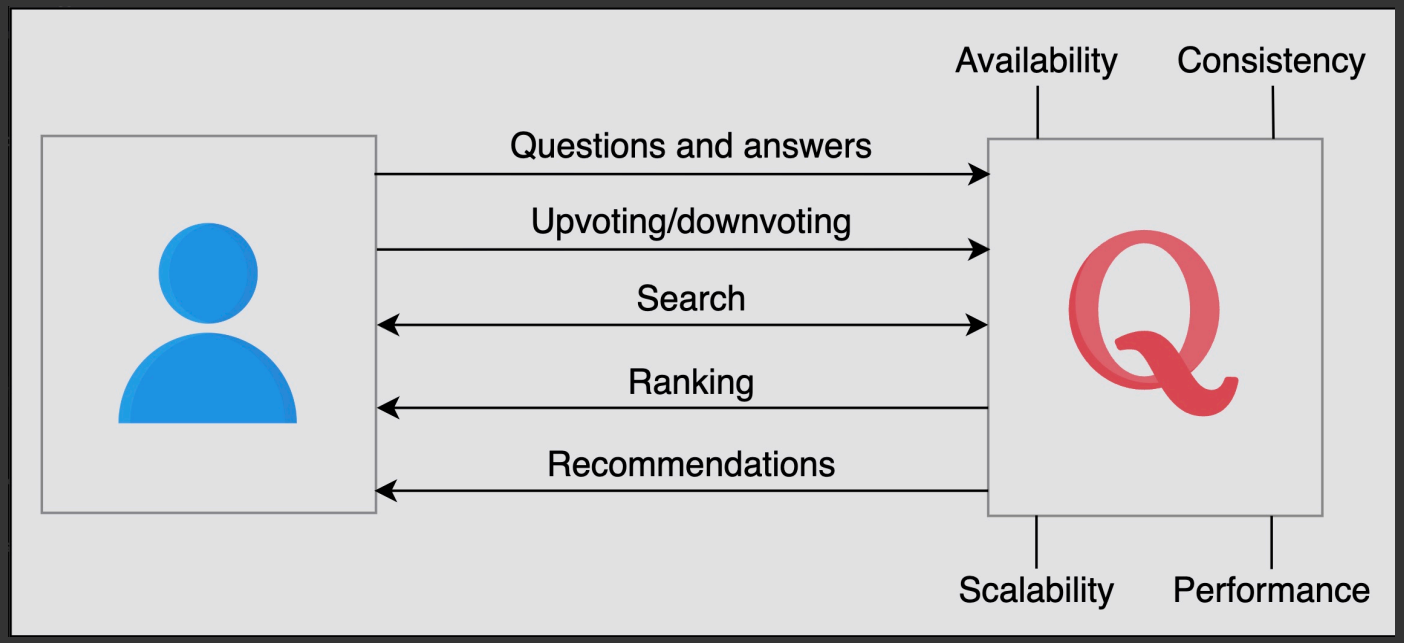
Functional requirements

A user should be able to perform the following functionalities:

- **Questions and answers**: Users can ask questions and give answers. Questions and answers can include images and videos.
- **Upvote/downvote and comment**: It is possible for users to upvote, downvote, and comment on answers.
- **Search**: Users should have a search feature to find questions already asked on the platform by other users.
- **Recommendation system**: A user can view their feed, which includes topics they're interested in. The feed can also include questions that need answers or answers that interest the reader. The system should facilitate user discovery with a recommender system.
- **Ranking answers**: We enhance user experience by ranking answers according to their usefulness. The most helpful answer will be ranked highest and listed at the top.

Non-functional requirements

- **Scalability**: The system should scale well as the number of features and users grow with time. It means that the performance and usability should not be impacted by an increasing number of users.
- **Consistency**: The design should ensure that different users' views of the same content should be consistent. In particular, critical content like questions and answers should be the same for any collection of viewers. However, it is not necessary that all users of Quora see a newly posted question, answer, or comment right away.
- **Availability**: The system should have high availability. This applies to cases where servers receive a large number of concurrent requests.
- **Performance**: The system should provide a smooth experience to the user without a noticeable delay.



Resource Estimation

Assumptions

1. There are a total 1 billion users, out of which 300 million of DAU.
2. Assume 15% of questions have an image, & 5% of questions have a video embedded in them. A question cannot have both at the same time
3. We will assume 1 image estimated to be 250 KB & video is considered 5MB.

Number of servers estimation

Let's estimate our requests per second (RPS) for our design. If there are an average of 300 million daily active users and each user can generate 20 requests per day, then the total number of requests in a day will be:

$300 \times 10^6 \times 20 = 6 \times 10^9$

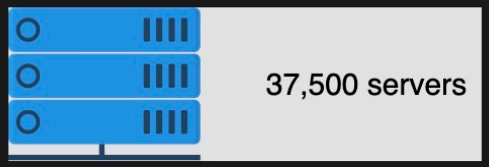
Therefore, the RPS = $\frac{6 \times 10^9}{86400} \approx 69500$ approximately requests per second.

Estimating RPS

Daily active users	300	million
Requests per day per user	20	
Requests Per Second (RPS)	f 69444	

We already established in the back-of-the-envelope calculations chapter that we'll use the following formula to estimate a pragmatic number of servers:

$$\frac{Number\ of\ daily\ active\ users}{RPS\ of\ a\ server} = \frac{300 \times 10^6}{8000} = 37500$$



The estimated number of servers required for Quora

> Therefore, the total number of servers required to facilitate 300 million users generating an average of 69,500 requests per second will be 37,500.

Storage Estimation

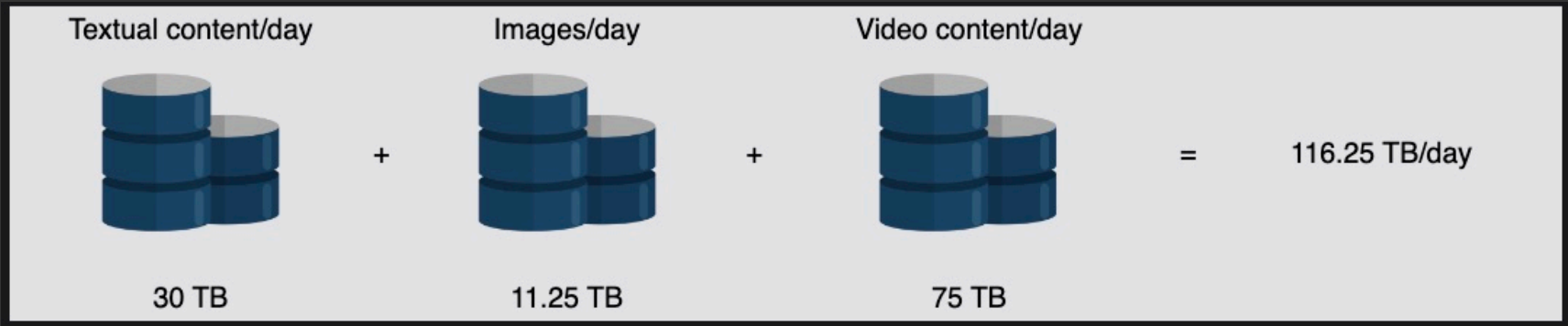
- Each of the 300 million active users posts 1 question in a day, and each question has 2 responses on average, 10 upvotes, and 5 comments in total.
- The collective storage required for the textual content (including the question, answer(s), and comment(s) text) of one question equals 100 KB.

Storage Requirements Estimation Calculator

Questions per user	1	per day
Total questions per day	*f* 300	millions
Size of textual content per question	100	KB
Image size	250	KB
Video size	5	MB
Questions containing images	15	percent
Questions containing videos	5	percent
Storage for textual content	*f* 30	TB
Storage for image content	*f* 11.25	TB
Storage for video content	*f* 75	TB

The following are the default calculations:

- Total questions: $300M \times 1 = 300 \times 10^6$ questions per day.
- Storage required for textual content of all questions in one day:
  $300 \times M \times 1 \times 100 \times 10^3 B = 30TB$
- Storage required for images for one day: $\frac{300 \times 10^6 \times 15}{100} \times 250 \times 10^3 B = 11.25TB$
- Storage required for video content for one day: $\frac{300 \times 10^6 \times 5}{100} \times 5 \times 10^6 B = 75TB$



> Total storage required for one day = $30\ TB + 11.25\ TB + 75\ TB\ = 116.25\ TB$ per day

The daily storage requirements of Quora seem very high. But for service with 300 million DAU, a yearly requirement of $116.25\ TB \times 365 = 42.43\ PB$ is feasible. The practical requirement will be much higher because we have disregarded the storage required for a number of things. For example, non-active (out of $1\ B$) users' data will require storage.

Bandwidth Estimation

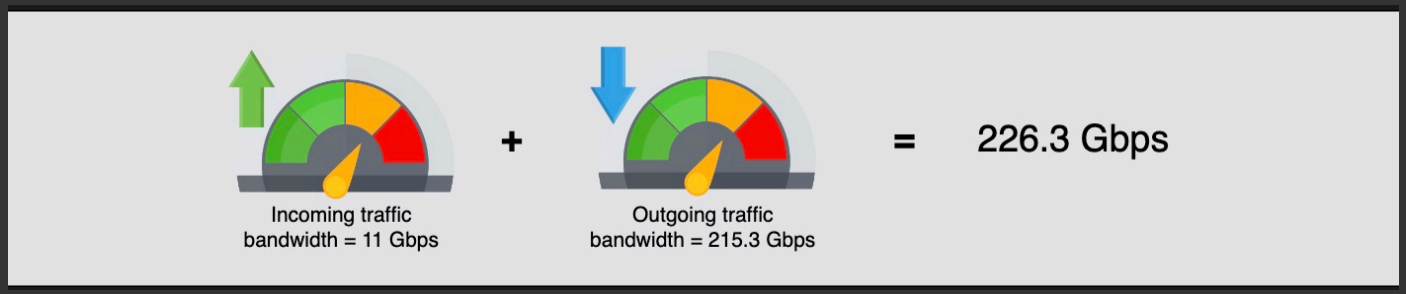
- **Incoming traffic**: The incoming traffic bandwidth required per day will be equal to $\frac{116.25\ TB}{86400} \times 8 = 10.9\ Gbps \approx 11\ Gbps$
- **Outgoing traffic**: We have assumed that 300 million active users views 20 questions per day, so the total bandwidth requirements can be found in the below calculator:

Bandwidth Requirements Estimation Calculator

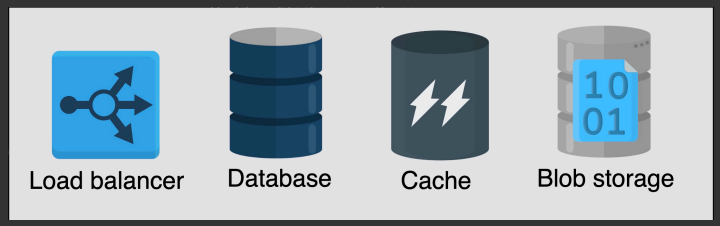
Total storage required per day	116.25	TB
Incoming traffic bandwidth	11	Gbps
Questions viewed per user	20	per day
Total questions viewed	69444	per second
Bandwidth for text of all questions	55.56	Gbps
Bandwidth for 15% of image content	20.83	Gbps
Bandwidth for 5% of video content	138.89	Gbps
Outgoing traffic bandwidth	215.3	Gbps

- $300M \times 20\ questions = 6$ billion questions are viewed per day.
- Questions viewed per second: $\frac{6 \times 10^9}{86400} = 69.4 \times 10^3$ questions are viewed per second.
- Bandwidth for the textual content of all questions and their answers:
  $69.4 \times 10^3 \times 100 \times 10^3 \times 8\ bits = 55.56\ Gbps$
- Bandwidth of the 15% of content which contain images per second:
  $69.4 \times 10^3 \times \frac{15}{100} \times 250 \times 10^3 \times 8\ bits = 20.83\ Gbps$
- Bandwidth for the 5% of content that contains video per second:
  $69.4 \times 10^3 \times \frac{5}{100} \times 5 \times 10^6 \times 8\ bits = 138.9\ Gbps$
- Total outgoing traffic bandwidth: $55.56\ Gbps + 20.82\ Gbps + 138.9\ Gbps = 215.3\ Gbps$

We use rounding at each step in this explanation. The answers in the calculator above are slightly different due to rounding.



Basic building blocks we will use



Load balancers will be used to divide the traffic load among the service hosts.

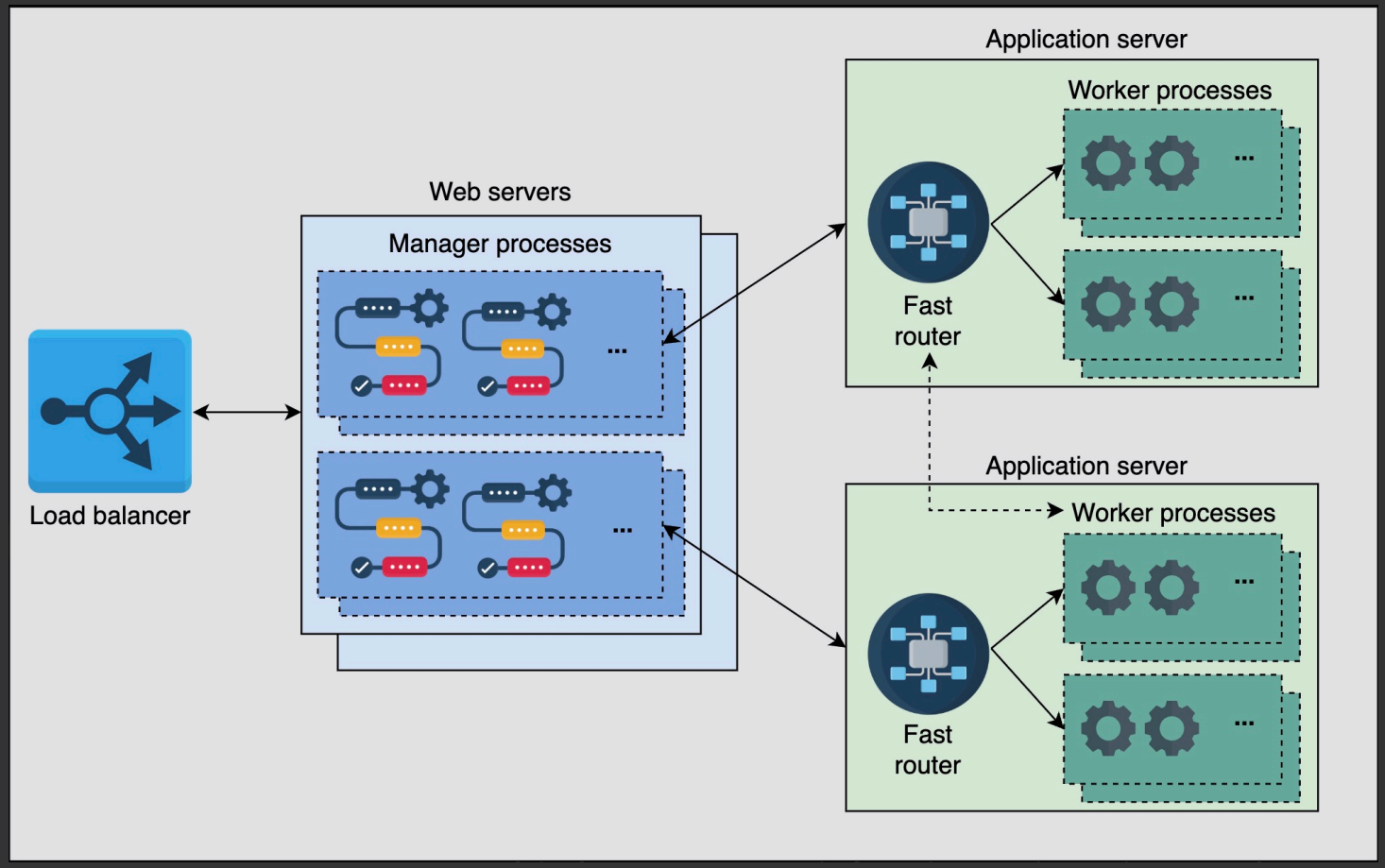
Databases are essential for storing all sorts of data, such as user questions and answers, comments, and likes and dislikes. Also, user data will be stored in the databases. We may use different types of databases to store different data.

A distributed caching system will be used to store frequently accessed data. We can also use caching to store our view counters for different questions.

The blob store will keep images and video files.

Initial design

- **Web and application servers**: A typical Quora page is generated by various services. The web and application servers maintain various processes to generate a webpage. The web servers have manager processes and the application servers have worker processes for handling various requests. The manager processes distribute work among the worker processes using a router library. The router library is enqueued with tasks by the manager processes and dequeued by worker processes. Each application server maintains several in-memory queues to handle different user requests. The following illustration provides an abstract view of web and application servers:

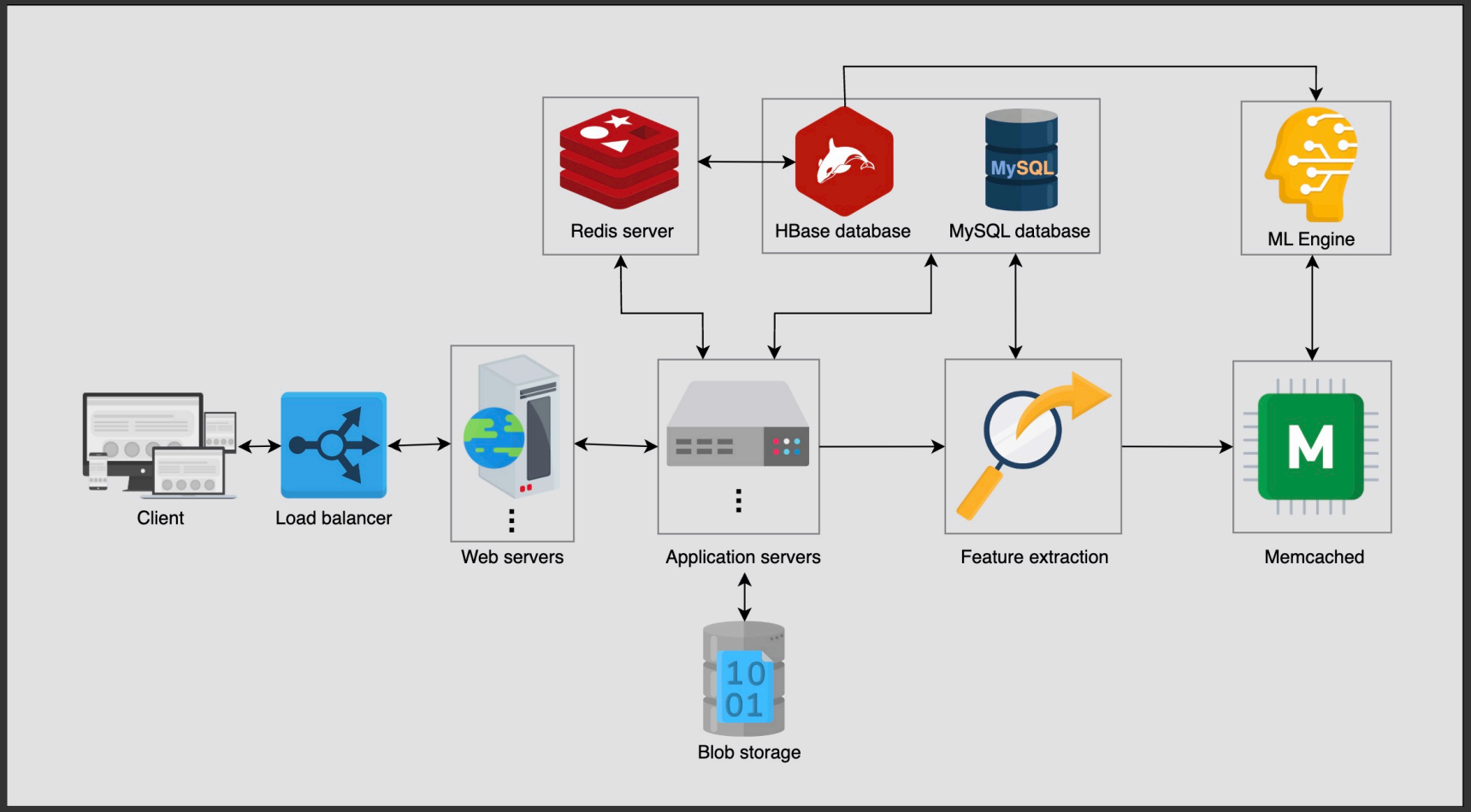


Data stores: Different types of data require storage in different data stores. We can use critical data like questions, answers, comments, and upvotes/downvotes in a relational database like MySQL because it offers a higher degree of consistency. NoSQL databases like HBase can be used to store the number of views of a page, scores used to rank answers, and the extracted features from data to be used for recommendations later on. Because recomputing features is an expensive operation, HBase can be a good option to store and retrieve data at high bandwidth. We require high read/write throughput because big data processing systems use high parallelism to efficiently get the required statistics. Also, blob storage is required to store videos and images posted in questions and answers.

Distributed cache: For performance improvement, two distributed cache systems are used: Memcached and Redis. Memcached is primarily used to store frequently accessed critical data that is otherwise stored in MySQL. On the other hand, Redis is mainly used to store an online view counter of answers because it allows in-store increments. Therefore, two cache systems are employed according to their use case. Apart from these two, CDNs serve frequently accessed videos and images.

Compute servers: A set of compute servers are required to facilitate features like recommendations and ranking based on a set of attributes. These features can be computed in online or offline mode. The compute servers use machine learning (ML) technology to provide effective recommendations. Naturally, these compute servers have a substantially high amount of RAM and processing power.

Compute servers: A set of compute servers are required to facilitate features like recommendations and ranking based on a set of attributes. These features can be computed in online or offline mode. The compute servers use machine learning (ML) technology to provide effective recommendations. Naturally, these compute servers have a substantially high amount of RAM and processing power.



Posting question, answers, comments: The web servers receive user requests through the load balancer and direct them to the application servers. Meanwhile, the web servers generate part of the web page and let the worker process in the application servers do the rest of the page generation. The questions and answers data is stored in a MySQL database, whereas any videos and images are stored in the blob storage. A similar approach is used to post comments and upvote or downvote answers. Task prioritization is performed by employing different queues for different tasks. We perform prioritization because certain tasks require immediate attention—for example, fetching data from the database for a user request—while others are not so urgent—for example, sending a weekly email digest. The worker processes will perform tasks by fetching from these queues.

When would a service like Quora require a notifications feature?

Hide Answer

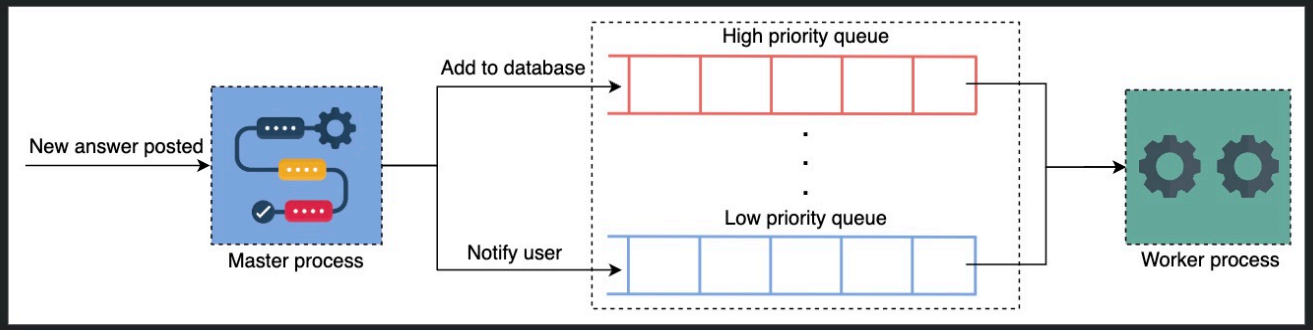
Quora will need a notification service in the following scenarios:

- A user posts a question on a topic subscribed by a would-be respondent.
- A user posts an answer to a question asked by another user.
- A post a user is interested in or wrote received new comments or upvotes/downvotes, and so on.

How is it possible for Quora to send notifications to users through the above-described mechanism?

Hide Answer

Since tasks are added to different priority queues, it is possible to maintain a queue for notifications. While higher priority tasks are enqueued to high-priority queues, a notification task will be added to a medium or lower priority queue. The illustration below visualizes the described concept:



- **Answer ranking system**: Answers to questions can be sorted based on date. Although it is convenient to develop a ranking system on the basis of date (using time stamps), users prefer to see the most appropriate answer at the top. Therefore, Quora uses ML to rank answers. Different features are extracted over time and stored in the HBase for each type of question. These features are forwarded to the ML engine to rank the most useful answer at the top. We cannot use the number of upvotes as the only metric for ranking answers because a good number of answers can be jokes—and such answers also get a lot of upvotes. It is good to implement the ranking system offline because good answers get upvotes and views over time. Also, the offline mode poses a lesser burden on the infrastructure. Implementing the ranking system offline and the need for special ML hardware makes it suitable to use some public cloud elastic services.

- **Recommendation system**: The recommendation system is responsible for several features. For example, we might need to develop a user feed, find related questions and ads, recommend questions to potential respondents, and even highlight duplicate content and content in violation of the service's terms of use. Unlike the answer ranking system, the recommendation system must provide both online and offline services. This system receives requests from the application server and forwards selected features to the ML engine.

- **Search feature**: Over time, as questions and answers are fed to the Quora system, it is possible to build an index in the HBase. User search queries are matched against the index, and related content is suggested to the user. Frequently accessed indexes can be served from cache for low latency. The index can be constructed from questions, answers, topics labels, and usernames. Tokenization of the search index returns the same results for reordered words also (see Scaling Search and Indexing in **Distributed Search** chapter for more details).

API Design

Post a Question

Post a question#

The POST method of HTTP is used to call the `/postQuestion` API:

```
postQuestion(user_id, question, description, topic_label, video, image)
```

Parameter	Description
`user_id`	This is the unique identification of the user that posts the question.
`question`	This is the text of the question posed by the user.
`description`	This is the description of a question. This is an optional field.
`topic_label`	This represents a list of domains to which the user's question is related.
`video`	This is a video file embedded in a user question.
`image`	This is an image that is a part of a user question.

Post an answer

For posting an answer, the POST method is a suitable choice for `/postAnswer` API:

```
postAnswer(user_id, question_id, answer_text, video, image)
```

Parameter	Description
`question_id`	This refers to the question the answer is posted against.
`answer_text`	This is the textual answer posted by the responder.

Upvote an answer

The `/upvote` API is below:

```
upvote(user_id, question_id, answer_id)
```

Parameter	Description
`user_id`	This represents the user upvoting the answer.
`answer_id`	This represents the identity of the answer that is upvoted for a particular question, which is identified by the `question_id`.

Comment on an answer

The `/comment` API has the following structure:

```
comment(user_id, answer_id, comment_text)
```

Parameter	Description
`user_id`	It represents the user commenting on the answer.
`comment_text`	It represents the text a user posts against an answer identified by the `answer_id`.

Search

The `/search` API has the following details:

```
search(user_id, search_text)
```

Parameter	Description
`user_id`	This is the `user_id` performing the search query. It is optional in this case because a non-registered user can also search for questions.
`search_text`	This is the search query entered by a user.

Why is there a custom routing layer between the web and application servers instead of a load-balancing layer?

Hide Answer

The primary reason is performance. A generic load-balancing layer will have little application understandability and higher latency.

As stated, the manager processes generate the skeleton of the web page and leave the majority of the work to worker processes. In that sense, one manager process can generate multiple worker tasks. The custom router library is a queue between manager and worker, where any worker process can consume a task generated by the manager process.

This design improves the overall performance and supports scalability.

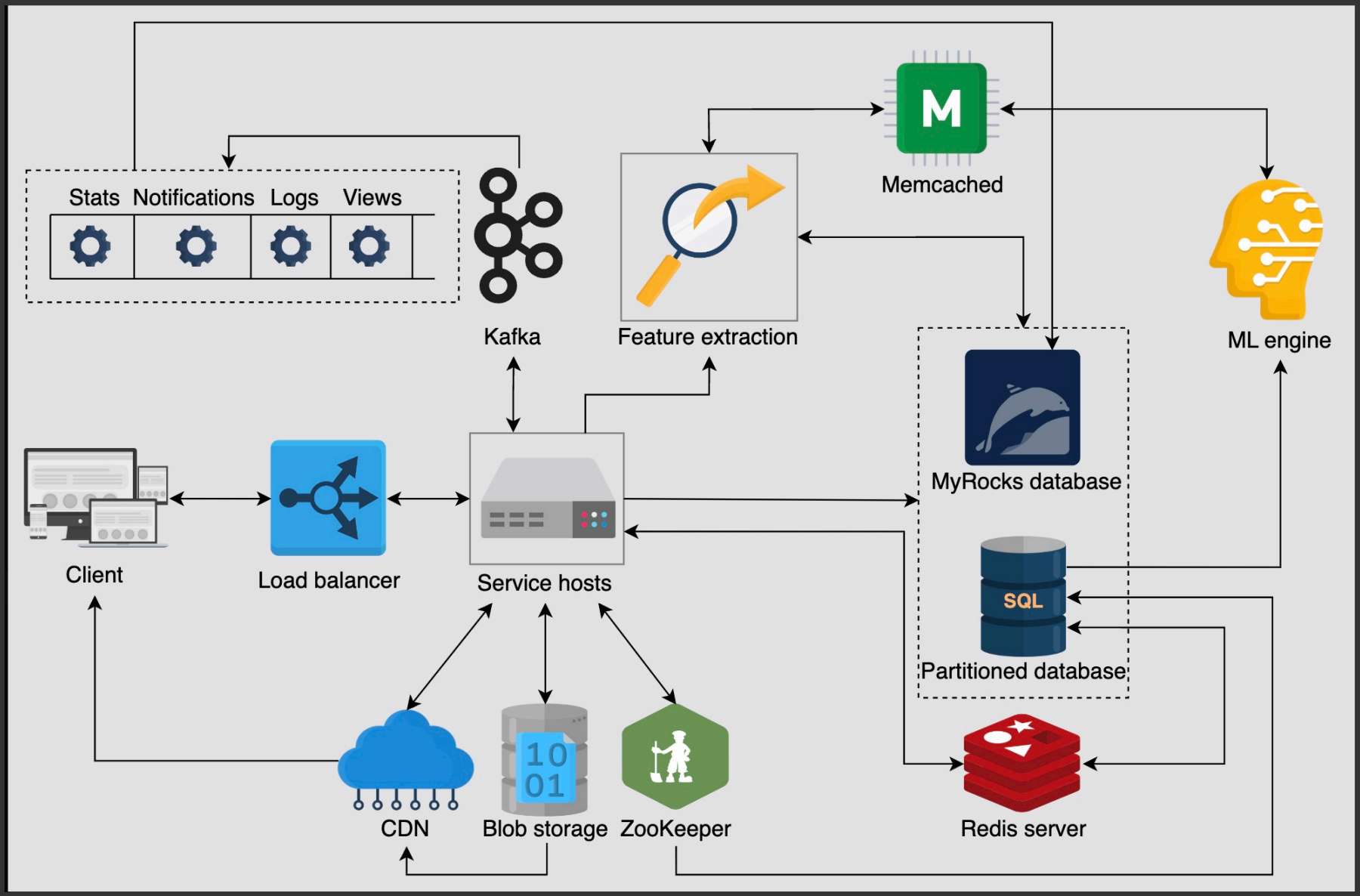
Final design of Quora

- **Limitations of web and application servers**: To entertain the user's request, payloads are transferred between web and application servers, which increases latency because of network I/O between these two types of servers. Even if we achieve parallel computation by separating the web from application servers (that is, the manager and worker processes), the added latency due to an additional network link erodes a user's experience. Apart from data transfer, control communication between the router library with manager and worker processes also imposes additional performance penalties.

- **In-memory queue failure**: The internal architecture of application servers log tasks and forward them to the in-memory queues, which serve them to the workers. These in-memory queues of different priorities can be subject to failures. For instance, if a queue gets lost, all the tasks in that queue are lost as well, and manual engineering is required to recover those tasks. This greatly reduces the performance of the system. On the other hand, replicating these queues requires increasing RAM size. Also, with the number of features (functional requirements) that our system offers, many tasks can get assembled, which results in insufficient memory. At the same time, it is not desirable to choke application servers with not-so-urgent tasks. For example, application servers should not be burdened with tasks like storing view counts for answers, adding statistics to the database for later analysis, and so on.

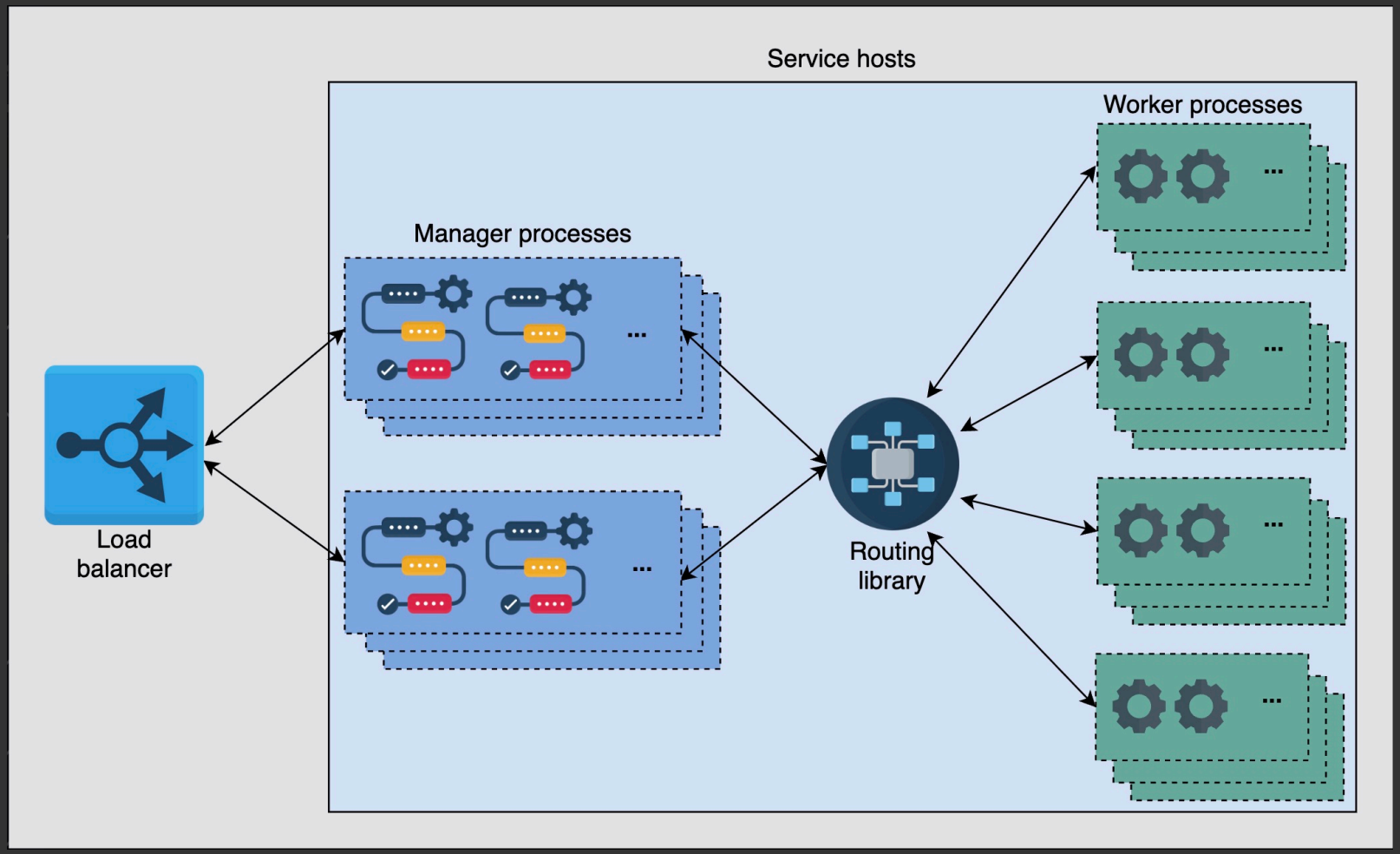
- **Increasing QPS on MySQL**: Because we have a higher number of features offered by our system, few MySQL tables receive a lot of user queries. This results in a higher number of QPS on certain MySQL servers, which can result in higher latency. Furthermore, there is no scheme defined for disaster recovery management in our design.

- **Latency of HBase:** Even though HBase allows high real-time throughput, its P99 latency is not among the best. A number of Quora features require the ML engine that has a latency of its own. Due to the addition of the higher latency of HBase, the overall performance of the system degrades over time.


Stats
Notifications
Logs
Views
Kafka
Memcached
Feature extraction
ML engine
MyRocks database
Client
Load balancer
Service hosts
SQL
Partitioned database
CDN
Blob storage
ZooKeeper
Redis server

Detailed design

Service hosts
We combine the web and application servers within a single powerful machine that can handle all the processes at once. This technique eliminates the network I/O and the latency introduced due to the network hops required between the manager, worker, and routing library processes. The illustration below provides an abstract view of the updated web server architecture:

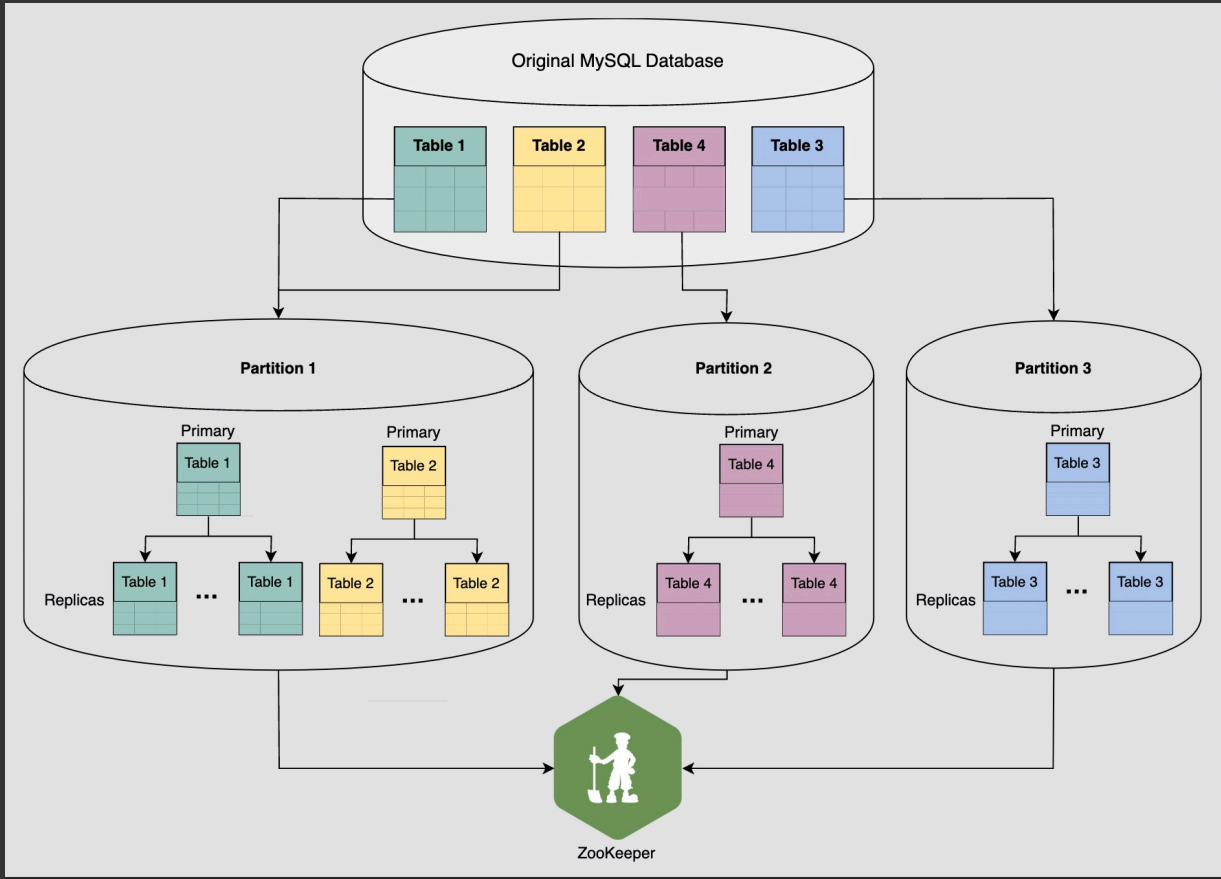


Vertical sharding of MySQL

Tables in the MySQL server are converted to separate shards that we refer to as **partitions**. A partition has a single primary server and multiple replica servers.

The goal is to improve performance and reduce the load due to an increasing number of queries on a single database table. To achieve that, we do vertical sharding in two ways:

1. We split tables of a single database into multiple partitions. The concept is depicted in Partitions 2 and 3, which embed Tables 4 and 3, respectively.
2. We combine multiple tables into a single partition, where `join` operations are anticipated. The concept is depicted in Partition 1, which embeds Tables 1 and 2.




```
Which partitions contain which tables and columns?
Which hosts are primary and replicas of a particular
partition?
Both of these mappings are maintained by a service
like ZooKeeper.
```

MyRocks

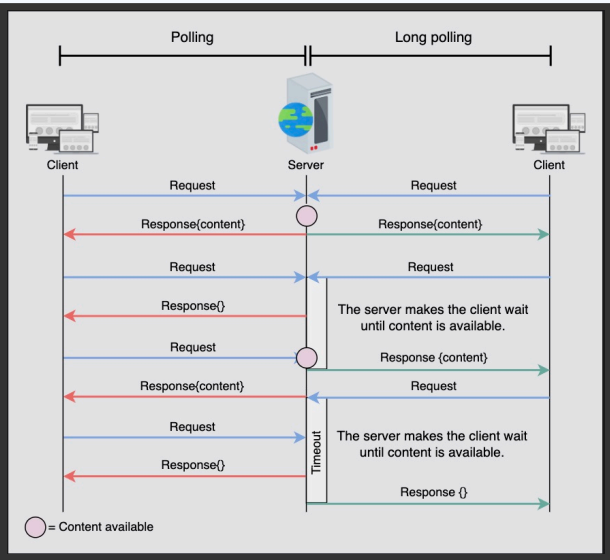
The new design embeds MyRocks as the key-value store instead of HBase. We use the MyRocks version of RocksDB for two main reasons:

1. MyRocks has a lower p99 latency instead of HBase. Quora claims to have reduced P99 latency from 80 ms to 4 ms using MyRocks.
2. There are operational tools that can transfer data between MyRocks and MySQL.

Kafka

Our updated design reduces the request load on service hosts by separating not-so-urgent tasks from the regular API calls. For this purpose, we use Kafka, which can disseminate jobs among various queues for tasks such as the view counter (see Sharded Counters), notification system, analytics, and highlight topics to the user. Each of these jobs is executed through cron jobs.

```
Polling is a technique where the client (browser) frequently requests the server for new updates. The server may or may not have any updates but still responds to the client. Therefore, the server may get uselessly overburdened. To resolve this issue, Quora uses a technique called long polling, where if a client requests for an update, the server may not respond for as long as 60 seconds if there are no updates.
```



What would be considered a good approach for communication between different manager and worker processes within the service hosts?

Hide Answer

Two approaches are feasible for communication:

1. UNIX sockets
2. TCP connections

Sockets (Unix or TCP) allow data streaming between sender and receiver with appropriate flow control (and congestion control in the case of TCP). That means the sender and receiver can send variable-size data in a decoupled fashion.

Other interprocess communication techniques like shared memory may not be feasible because they require estimating the size of the required memory segment, which makes the participants more coupled. Also, it will not work across physical servers.

So, we prefer sockets due to their high decoupling, flow control, and ability to work for both single servers or over the network.

In our detailed design, we have employed Kafka to handle our view counter of answers. Why do you think we made this decision?

Hide Answer

Depending on the topic label, many users can view the response to a question at once. This can choke the servers. Instantaneous update of the view counter is also not an important product feature. Therefore, Kafka is suitable for handling these tasks.

Quora handles these tasks in two minutes or less.

Nonetheless, we can use sharded counters as an effective solution to the view counter problem.

What is the main advantage of using long polling instead of polling?

Hide Answer

Long polling transfers the control to the server side instead of the client, which has no information about the updates in content. If the server is in control, it can reply as soon as there is fresh content. As a result, it reduces the request load on itself. However, long polling is a resource-intensive solution because it keeps the connection persistent or alive for a longer period of time.

WebSockets are another low-latency solution with low overhead. However, WebSockets might be an overkill for the features offered by Quora.

Evaluation of Quora design

Scalability: Our system is highly scalable for several reasons. The updated design uses powerful and homogeneous service hosts. Quora uses powerful machines because service hosts use an in-memory cache, some level of queueing, maintain manager, worker, and routing library. The horizontal scaling of these service hosts is convenient because they are homogeneous.

On the database end, our design shards the MySQL databases vertically, which avoids issues in scalability because of overloaded MySQL servers. To reduce complex `join` queries, tables anticipating `join` operations are placed in the same shard or partition.

Consistency: Due to the variety of functionalities offered by Quora, different consistency schemes may be selected for different types of data. For example, certain critical data like questions and answers should be stored synchronously. In this case, performance can take a hit because users don't expect instantaneous responses to their questions. It means that a user may get a reply in five minutes, one hour, one day, or no response at all, depending on the user's question and the availability of would-be respondents.

Availability: Some of the main ideas to improve availability include isolation between different components, keeping redundant instances, using CDN, using configuration services like ZooKeeper, and load balancers to hide failures from users.

Performance: This design has a strong performance because we have employed the right technology for the right feature. For example, we have used several datastores for different reasons. On top of that, we used different distributed caches depending upon the use case and access frequency. Also, we employed Kafka to queue similar tasks and assign them to cron jobs that otherwise take a long time if executed via API calls.

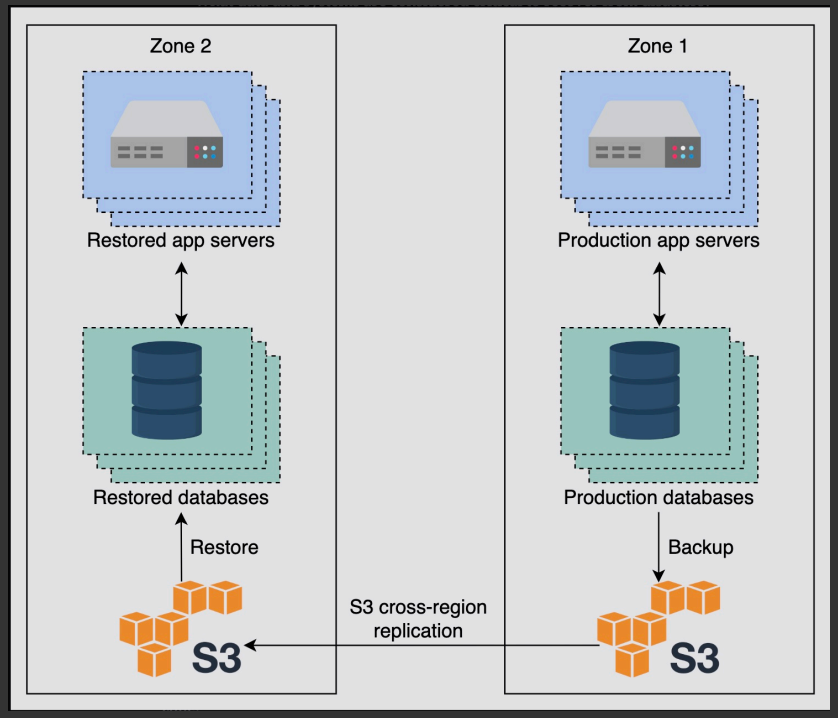
Requirements	Techniques
Scalability	• Based on AWS, which supports automatic scaling. • Uses the same servers to reduce complexity in horizontal scalability • sharding of MySQL database. • Employs various data stores for different purposes. • Asynq can enable developers to code quickly by batching cache requests, • separate compute, and feature extraction modules. Therefore, a generic feature extraction facility allows scalability of different recommendation systems.
Consistency	• Uses MySQL and synchronous replication within a data center for critical data. • Offers eventual consistency for non-critical data like the view counter.
Availability	• Use of different data stores prevents failure of multiple services at once by • using database sharding and replicas. • Uses CDN as a backup to serve static/dynamic data in case of failures. • ZooKeeper enables service hosts to get updates about MySQL shards. • Load balancers hide server failures from end users. • AWS supports an availability above 99 percent. • Thrift isolates services and therefore failures.
Performance	• MyRocks has a much lower P99 latency. • Uses the right programming language to deliver tasks quickly, such as C++ for the routing library. • Uses `Multiget()` to retrieve multiple entries from Memcached at once. • Eliminates network round trip time with Asynq. • Kafka improves the performance of service hosts. • Sharding improves QPS of MySQL. • Custom, in-memory caching system reduces the latency of frequently accessed data.

Disaster Recovery

The first and foremost approach of handling a disaster is frequent backups. The frequency of backups depends on the size of the data. Daily backups are suitable for our design because we can backup individual data stores and shards without any hassle. Of course, backups will be stored at remote destinations because natural disasters can wipe out the entire facility at a location.

The following are important questions for designing a disaster recovery plan:

- What data and systems are considered critical to recover from disasters?
- How fast is the restoration from the backup facility?
- Can all systems be recovered through backups?
- How can we deal with potential loss of data that we couldn't replicate before the disaster hit?



The approach is fairly straightforward. The data, application servers, and configurations are backed up in the Amazon S3 storage service in the same zone. Zonal replication between S3 storage facilitates transfer to another zone. Later, the application and database servers can be restored from the S3 storage in another zone.

Conclusion

Throughout this design, we learned how Quora is able to scale its services as the number of users increases. One interesting aspect of the design includes the vertical sharding of the MySQL database. Apart from that, the Quora design discusses a variety of techniques to meet the functional and non-functional requirements. However, our scope did not include the usage of techniques like natural language processing (NLP) to remove spelling mistakes in user's questions or typeahead services during the search.

How can using different data stores for different types of data be beneficial in disaster recovery?

Hide Answer

When we use different data stores, it means that we can recover various forms of data simultaneously instead of one large blob of data. Since we recover data stores with variable recovery times, the final recovery time equals the maximum recovery time among all data stores.

However, using many databases also means that administrators need to know the details of checkpointing or restoring them.