Initial Design of Quora

Transform the Quora requirements into a high-level design.

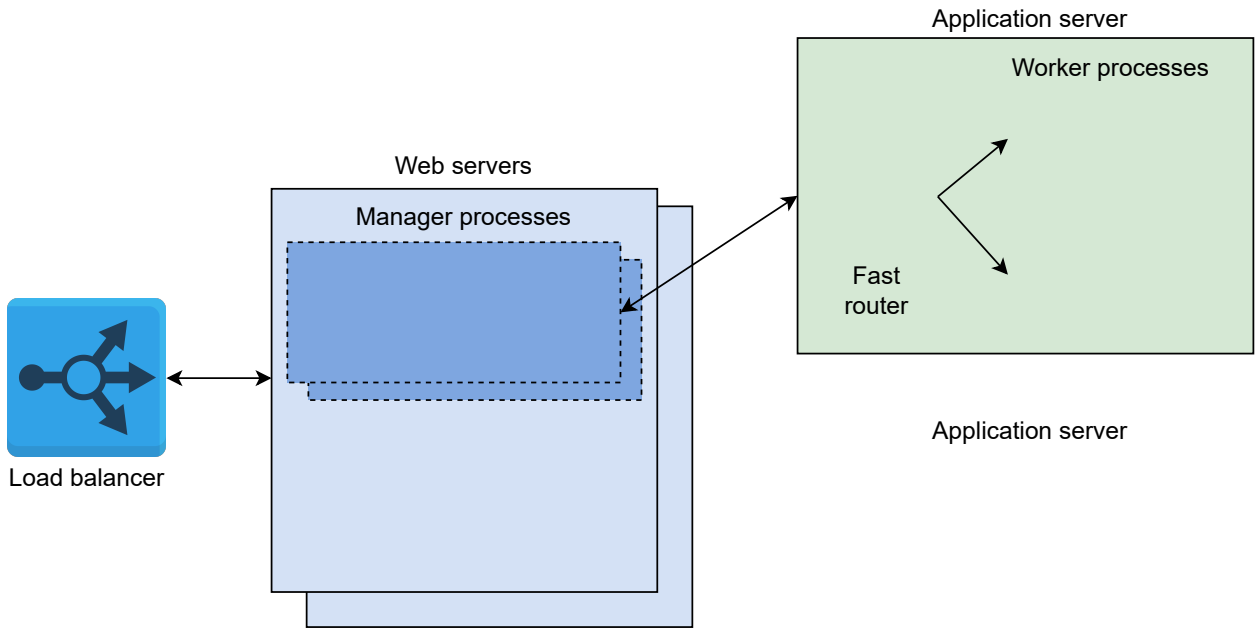
We'll cover the following

Initial design

The initial design of Quora will be composed of the following building blocks and components:

- **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:



Web and application servers at Quora

- **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.

Why HBase?

- **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.

Of course, other basic building blocks like load balancers, monitoring services, and rate limiters will also be part of the design. A high-level design is provided below:

The high-level design of Quora

Workflow

The design of Quora is complex because we have a large number of functional and non-functional requirements. Therefore, we'll explain the

workflow on the basis of each feature:

- **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.

Points to Ponder

Question 1

When would a service like Quora require a notifications feature?

Show Answer

1 of 2

- **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).

Fun Facts

API design

We'll design the API calls for Quora in this section. We'll define APIs for the following features only:

- Post a question
- Post an answer
- Upvote or downvote a question or answer
- Comment on an answer
- Search

> **Note:** We don't consider APIs for a recommendation system or ranking because they are not placed as an explicit request by the user. Instead, the web server coordinates with other components to ensure the service.

Post a question

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

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

Let's understand each parameter of the API call:

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.

The video and image parameters can be `NULL` if no image or video is embedded within the question. Otherwise, it is uploaded as part of the 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.

The rest of the parameters are self-explanatory.

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`.

> **Note:** The downvote API is the same as the upvote API because both are similar functionalities.

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.

> We use a sequencer to generate the different IDs mentioned in the API calls.

Point to Ponder

Question

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

Show Answer

Back

Requirements of Quora's Design

Next

Final Design of Quora

Mark as Completed