The document titled "System Design: Quora - Grokking Modern System Design Interview for Engineers & Managers" is a comprehensive guide aimed at helping engineers and managers understand the intricacies of designing a system like Quora. Here is a detailed summary:

### **Introduction**

* **Purpose**: The document outlines the key components and steps involved in designing Quora, a social question-and-answer service.
* **Challenge**: Finding answers online can be daunting; search engines like Google provide fast but often shallow information, whereas asking people can provide more in-depth insights despite taking longer.

### **What is Quora?**

* **Overview**: Quora allows users to ask questions to other users and receive answers, often from domain experts. It aims to provide more conversational and detailed responses compared to search engines.
* **User Base**: Quora has over 300 million monthly active users who post thousands of questions daily across more than 400,000 topics.

### **Design Approach**

The document breaks down the system design of Quora into four key lessons:

1. **Requirements**:
   * **Functional Requirements**: Features and functionalities needed for the platform to operate, such as user authentication, question posting, answer submission, upvoting/downvoting, etc.
   * **Non-Functional Requirements**: Performance, scalability, reliability, and other quality attributes.
   * **Resource Estimation**: Estimating the resources needed to build and maintain the system.
2. **Initial Design**:
   * **Building Blocks**: Identifying core components such as databases, load balancers, distributed caches, etc.
   * **Integration and Workflow**: How these components interact and work together to form the initial system design.
   * **API Design**: Designing the APIs required for interaction between different parts of the system.
3. **Final Design**:
   * **Limitations of Initial Design**: Identifying and addressing the limitations found in the initial design.
   * **Enhanced Design**: Updating the design to meet both functional and non-functional requirements.
   * **Vertical Sharding**: An interesting aspect of the design to improve database performance and scalability.
4. **Evaluation**:
   * **Non-Functional Requirements**: Assessing the design against non-functional criteria such as scalability, performance, and availability.
   * **Trade-offs**: Discussing trade-offs and potential improvements to enhance system availability and reliability.

### **Key System Components**

The document details several key components essential for Quora's design:

* **Domain Name System (DNS)**
* **Load Balancers**
* **Databases**
* **Key-value Stores**
* **Content Delivery Network (CDN)**
* **Sequencers**
* **Distributed Monitoring**: For both server-side and client-side errors.
* **Distributed Cache**
* **Distributed Messaging Queue**
* **Pub-Sub Systems**
* **Rate Limiter**
* **Blob Store**
* **Distributed Search**
* **Distributed Logging**
* **Distributed Task Scheduler**
* **Sharded Counters**

### **Practical Applications**

The guide also includes practical design examples for various systems beyond Quora:

* **Designing YouTube**
* **Designing Google Maps**
* **Designing a Proximity Service (e.g., Yelp)**
* **Designing Uber**
* **Designing Twitter**
* **Designing a Newsfeed System**
* **Designing Instagram**
* **Designing a URL Shortening Service (e.g., TinyURL)**
* **Designing a Web Crawler**
* **Designing WhatsApp**
* **Designing Typeahead Suggestion**
* **Designing a Collaborative Document Editing Service (e.g., Google Docs)**

### **Conclusion**

* **Failures and Lessons**: The document also discusses spectacular failures in system design and the lessons learned from them.
* **Certification**: Completing the course offers a certification to validate the knowledge and skills acquired.

### **References and Resources**

The document includes numerous links to external resources and annotations for further reading and exploration on system design topics.

This guide provides a detailed roadmap for understanding and implementing system design principles, specifically tailored for a complex platform like Quora, and extends these principles to various other applications .

The document titled "Requirements of Quora's Design - Grokking Modern System Design Interview for Engineers & Managers" provides a detailed analysis of the system requirements for designing Quora, a social question-and-answer service. Below is a comprehensive summary of the key points discussed in the document:

### **Overview**

* **Purpose**: The document outlines the functional and non-functional requirements necessary for designing Quora. It includes resource estimation, server estimation, storage needs, bandwidth requirements, and essential building blocks.

### **Functional Requirements**

1. **Questions and Answers**:
   * Users can ask questions and provide answers, including images and videos.
2. **Upvote/Downvote and Comment**:
   * Users can upvote or downvote answers and leave comments.
3. **Search**:
   * A search feature allows users to find questions already asked by others.
4. **Recommendation System**:
   * Users can view a feed with topics of interest, questions needing answers, and interesting answers. A recommender system facilitates user discovery.
5. **Ranking Answers**:
   * Answers are ranked according to their usefulness, with the most helpful answers listed at the top.

### **Non-Functional Requirements**

1. **Scalability**:
   * The system should handle growth in the number of features and users without impacting performance and usability.
2. **Consistency**:
   * The design should ensure consistency in viewing content across different users, particularly for critical content like questions and answers. However, immediate consistency is not necessary for newly posted content.
3. **Availability**:
   * The system should have high availability, especially under heavy concurrent request loads.
4. **Performance**:
   * The user experience should be smooth, without noticeable delays.

### **Resource Estimation**

* **Assumptions**:
  + Total users: 1 billion
  + Daily active users: 300 million
  + 15% of questions have images; 5% have videos.
  + Image size: 250 KB; Video size: 5 MB.

1. **Number of Servers**:
   * **Requests Per Second (RPS)**: With 300 million daily active users generating 20 requests per day, the total requests per day are 6 billion. Dividing this by 86,400 seconds (one day) gives approximately 69,444 RPS.
   * **Server Calculation**: Using an assumed RPS handling capacity of 8,000 per server, the total number of servers required is estimated to be 37,500.
2. **Storage Estimation**:
   * **Daily Storage Requirements**:
     + Textual content: 0.3 TB
     + Image content: 11.25 TB
     + Video content: 75 TB
   * **Annual Storage Requirements**: Approximately 31.6 PB (petabytes).
3. **Bandwidth Estimation**:
   * **Incoming Traffic**: 8 Gbps
   * **Outgoing Traffic**:
     + Text: 0.56 Gbps
     + Images: 20.83 Gbps
     + Videos: 138.89 Gbps
   * **Total Bandwidth**: 168.3 Gbps (incoming + outgoing).

### **Building Blocks**

The initial design of Quora will use the following components:

1. **Load Balancers**:
   * To distribute traffic load among service hosts.
2. **Databases**:
   * For storing user data, questions, answers, comments, and likes/dislikes. Different types of databases may be used for different data.
3. **Distributed Caching System**:
   * To store frequently accessed data and view counters for questions.
4. **Blob Store**:
   * To store images and video files.

### **Conclusion**

* **Design Considerations**: The document highlights the importance of considering both functional and non-functional requirements, estimating necessary resources accurately, and using appropriate building blocks to ensure scalability, consistency, availability, and performance.
* **Future Steps**: The subsequent chapters will focus on the initial design, addressing limitations, finalizing the design, and evaluating it against non-functional requirements.

This detailed analysis provides a foundational understanding of what is required to design a scalable and efficient system like Quora, ensuring it meets both user expectations and technical constraints.

The document titled "Initial Design of Quora - Grokking Modern System Design Interview for Engineers & Managers" provides a detailed approach to transforming Quora's requirements into a high-level design. Here is a comprehensive summary:

### **Overview**

The document aims to create an initial design that fulfills Quora's functional requirements, discussing the system's building blocks, integration, workflow, and API design.

### **Initial Design Components**

1. **Web and Application Servers**:
   * **Manager Processes**: Web servers have manager processes to generate web pages.
   * **Worker Processes**: Application servers handle various requests, distributing work through a router library.
   * **In-memory Queues**: Each application server maintains in-memory queues for different user requests.
2. **Data Stores**:
   * **Relational Database (MySQL)**: Used for critical data like questions, answers, comments, and upvotes/downvotes due to its consistency.
   * **NoSQL Database (HBase)**: Stores page views, answer scores, and features for recommendations. It handles high bandwidth and parallel processing for big data.
   * **Blob Storage**: Stores videos and images posted in questions and answers.
3. **Distributed Cache**:
   * **Memcached**: Stores frequently accessed critical data.
   * **Redis**: Maintains online view counters for answers due to its in-store increment capability.
   * **Content Delivery Networks (CDNs)**: Serve frequently accessed videos and images.
4. **Compute Servers**:
   * Facilitate features like recommendations and answer ranking using machine learning (ML).
   * Possess high RAM and processing power.

### **Workflow**

1. **Posting Questions, Answers, Comments**:
   * User requests are received through load balancers and directed to application servers.
   * Web servers generate part of the web page, and application servers complete the page generation.
   * Data is stored in MySQL, while videos and images go to blob storage.
   * Task prioritization is done using different queues.
2. **Answer Ranking System**:
   * Uses ML to rank answers based on extracted features stored in HBase.
   * Offline implementation reduces infrastructure burden and uses public cloud elastic services.
3. **Recommendation System**:
   * Provides features like user feed, related questions, ads, and content moderation.
   * Operates in both online and offline modes, interacting with the ML engine.
4. **Search Feature**:
   * Builds an index in HBase from questions, answers, topic labels, and usernames.
   * Matches user queries against the index and serves frequently accessed indexes from cache for low latency.

### **API Design**

1. **Post a Question**:
   * postQuestion(user\_id, question, description, topic\_label, video, image): Handles posting questions with optional fields for video and image.
2. **Post an Answer**:
   * postAnswer(user\_id, question\_id, answer\_text, video, image): Manages posting answers to specific questions.
3. **Upvote/Downvote**:
   * upvote(user\_id, question\_id, answer\_id): Used for upvoting answers. The same structure applies to downvoting.
4. **Comment on an Answer**:
   * comment(user\_id, answer\_id, comment\_text): Allows users to comment on answers.
5. **Search**:
   * search(user\_id, search\_text): Facilitates searching questions, with user\_id being optional for non-registered users.

### **Conclusion**

The initial design includes essential building blocks such as load balancers, monitoring services, and rate limiters to ensure a robust and scalable system. The document provides a high-level overview of the workflow, component interactions, and API design necessary to meet Quora's functional requirements. The subsequent steps involve refining this design to address limitations and fulfill all non-functional requirements, ensuring the system's scalability, performance, and reliability.

The document titled "Final Design of Quora - Grokking Modern System Design Interview for Engineers & Managers" outlines the limitations of the initial design and provides detailed improvements to fulfill both functional and non-functional requirements. Here is a comprehensive summary:

### **Limitations of the Initial Design**

1. **Web and Application Servers**:
   * **Network Latency**: Latency is increased due to network I/O between web and application servers.
   * **Performance Penalties**: Additional control communication between the router library, manager, and worker processes imposes further performance penalties.
2. **In-Memory Queue Failure**:
   * **Task Loss**: In-memory queues are prone to failures, leading to the loss of tasks and the need for manual recovery.
   * **Memory Insufficiency**: Replicating queues increases RAM usage, and the accumulation of tasks may result in insufficient memory.
3. **High QPS on MySQL**:
   * **Latency**: High query per second (QPS) on MySQL tables results in increased latency.
   * **Lack of Disaster Recovery**: The design does not include a scheme for disaster recovery.
4. **Latency of HBase**:
   * **High P99 Latency**: HBase has high P99 latency, affecting overall system performance, especially when used with the ML engine.

### **Improvements in the Final Design**

1. **Service Hosts**:
   * **Unified Servers**: Web and application servers are combined into a single powerful machine, eliminating network I/O and reducing latency. This approach improves performance by avoiding network hops and simplifying the architecture.
2. **Vertical Sharding of MySQL**:
   * **Partitioning**: Tables in the MySQL server are split into separate shards (partitions), each with a single primary server and multiple replica servers.
   * **Co-Locating Data**: Related data is co-located in a single partition to reduce traffic on hot data and improve performance.
   * **Scalability**: Vertical sharding enhances scalability by allowing further sharding and increasing the number of read-replicas for hot shards.
3. **MyRocks**:
   * **Key-Value Store**: MyRocks replaces HBase as the key-value store, offering lower P99 latency and operational tools for data transfer between MyRocks and MySQL.
   * **Performance**: Quora reduced P99 latency from 80 ms to 4 ms using MyRocks, significantly improving system performance.
4. **Kafka**:
   * **Task Offloading**: Kafka is used to offload not-so-urgent tasks from regular API calls. Tasks such as view counters, notifications, analytics, and topic highlights are managed through Kafka queues and executed via cron jobs.
5. **Technology Usage**:
   * **Cloud Infrastructure**: Quora employs AWS for various infrastructure elements, including S3 for blob storage and Redshift for storage solutions.
   * **Programming Languages**: Python is used for faster development, while C++ is used for the feature extraction service to reduce latency and support the ML engine.
   * **Interoperability**: Thrift service is used for interoperability between different components programmed in various languages.
6. **Polling vs. Long Polling**:
   * **Long Polling**: To handle frequent page updates for comments, upvotes, and downvotes, Quora uses long polling. This technique reduces server burden by responding to client requests only when there are updates, thus avoiding frequent, unnecessary server responses.
7. **Memcached Multiget**:
   * **Cache Retrieval**: Memcached employs multiget() to retrieve multiple keys from cache shards, reducing the retrieval latency of multiple keys and improving performance.

### **Conclusion**

The final design addresses the limitations of the initial design by optimizing server architecture, improving data storage and retrieval, offloading non-critical tasks, and employing appropriate technologies for better performance and scalability. The improvements ensure that Quora can handle a growing user base while maintaining high performance, reliability, and availability.

### **References**

* Grokking Modern System Design Interview for Engineers & Managers .

The document titled "Evaluation of Quora’s Design - Grokking Modern System Design Interview for Engineers & Managers" provides an in-depth analysis of how Quora's design fulfills its non-functional requirements, covering scalability, consistency, availability, and performance. Here is a detailed summary:

### **Fulfilling Requirements**

The document evaluates the design of Quora against its non-functional requirements, highlighting the following key points:

1. **Scalability**:
   * **Service Hosts**: The design employs powerful and homogeneous service hosts, which use in-memory cache, some level of queuing, and maintain manager, worker, and routing libraries. These homogeneous service hosts allow for convenient horizontal scaling.
   * **Database Sharding**: The MySQL databases are sharded vertically, which helps avoid scalability issues due to overloaded MySQL servers. Tables anticipating join operations are placed in the same shard or partition to reduce complex join queries. However, vertical sharding alone may not be sufficient for very large MySQL tables, and horizontal sharding may be necessary to handle write bottlenecks.
2. **Consistency**:
   * **Different Schemes for Different Data**: The design uses various consistency schemes for different types of data. Critical data like questions and answers are stored synchronously to ensure immediate consistency. However, this can impact performance as users may experience delays in receiving responses.
   * **Eventual Consistency**: Non-critical data like view counts are stored with eventual consistency to improve performance, as it is not essential for all users to see the same view count immediately.
3. **Availability**:
   * **Isolation of Services**: Services are isolated to prevent the failure of one component from affecting others. Redundant instances, CDNs, and configuration services like ZooKeeper help maintain availability.
   * **Load Balancers**: Load balancers are used to hide server failures from users.
   * **Redundant Instances**: The system keeps redundant instances to ensure continuous availability.
4. **Performance**:
   * **Right Technology for Right Feature**: The design employs different datastores for different purposes, uses distributed caches based on access frequency, and employs Kafka for queuing similar tasks. This helps maintain high performance and reduce latency.
   * **Custom Queuing Solution**: Quora's custom queuing solution can handle roughly 15,000 tasks per second, which significantly enhances performance.

### **Disaster Recovery**

The document acknowledges a gap in the design concerning disaster recovery management and suggests mechanisms to address it:

1. **Frequent Backups**:
   * **Backup Frequency**: Daily backups are recommended for critical data stores and shards.
   * **Remote Storage**: Backups should be stored at remote locations to protect against natural disasters.
2. **Restoration Strategy**:
   * **Simple and Effective**: Data, application servers, and configurations are backed up in Amazon S3 storage with zonal replication to facilitate transfer to another zone.
   * **Drawbacks**: Potential loss of data not backed up since the last backup, and long restoration times during which databases may not serve queries.
3. **Critical Questions**:
   * **Data Recovery**: What data and systems are critical to recover?
   * **Restoration Speed**: How fast can data be restored from backups?
   * **Completeness of Backups**: Can all systems be recovered through backups?
   * **Handling Data Loss**: How to deal with potential data loss that couldn't be replicated before the disaster?

### **Conclusion**

The document concludes by summarizing how the design of Quora scales its services as the number of users increases. Key techniques include vertical sharding of MySQL databases, isolation of services, use of multiple datastores, and custom queuing solutions. While the design meets most non-functional requirements, it lacks comprehensive disaster recovery management.

### **References**

* Grokking Modern System Design Interview for Engineers & Managers .

This summary encapsulates the critical aspects of the evaluation of Quora’s design, emphasizing scalability, consistency, availability, performance, and disaster recovery.

ChatGPT can make mistakes. Check important info.

ChatGPT