Distributed System for Reddit

case study report submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

On 15-April-2024

For Partial Fulfilment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

In

Information Technology

By

Harshita Gupta - 210911224 – 39
Harsh Vardhan Mishra - 210911268 - 45
Aastha Sinha - 210911320 - 54
Charu Yadav - 210911396 – 69

Distributed Systems (ICT 3254)

Under the guidance of

Ms. Sucheta V Kolekar
Assistant Professor
Department of I&CT
Manipal Institute of Technology

1 Introduction

Reddit, established in 2005, has emerged as a premier social news aggregation, web content rating, and discussion platform that is structured into numerous communities known as "subreddits." Each subreddit is centered around a particular topic or interest, ranging from technology and science to entertainment and lifestyle. Users can submit content such as text posts, links, and images to these subreddits, where other members can vote the submissions up or down, thus determining their visibility on the site based on popularity.

The interactive nature of Reddit allows users not only to vote on content but also to engage in discussions by commenting on posts. This dynamic interaction fosters a rich community environment where debates, advice, and in-depth discussions thrive. Moreover, Reddit's platfom supports various forms of media integration and content presentation, enhancing user engagement and content diversity.

As Reddit caters to millions of users worldwide, the platform must efficiently manage a vast amount of user-generated content and ensure that user interactions are processed in real-time. This requirement demands a robust, scalable, and highly available distributed system architecture. The system must support complex user interactions, content filtering, dynamic content updates, and extensive data analytics—all while maintaining fast response times and high reliability.

The design of Reddit's distributed system is crucial in handling peak traffic loads, storing and retrieving massive amounts of data, and ensuring consistent user experience across different global regions. Additionally, the system must address various challenges such as data consistency, system failure recovery, and security threats, which are inherent to managing a large-scale, distributed online platform. This background sets the stage for discussing the intricate details of the distributed system architecture tailored for Reddit, encompassing aspects of system design, failure management, and security frameworks.

2 Architecture Diagram & Explanation

Architecture Diagram:

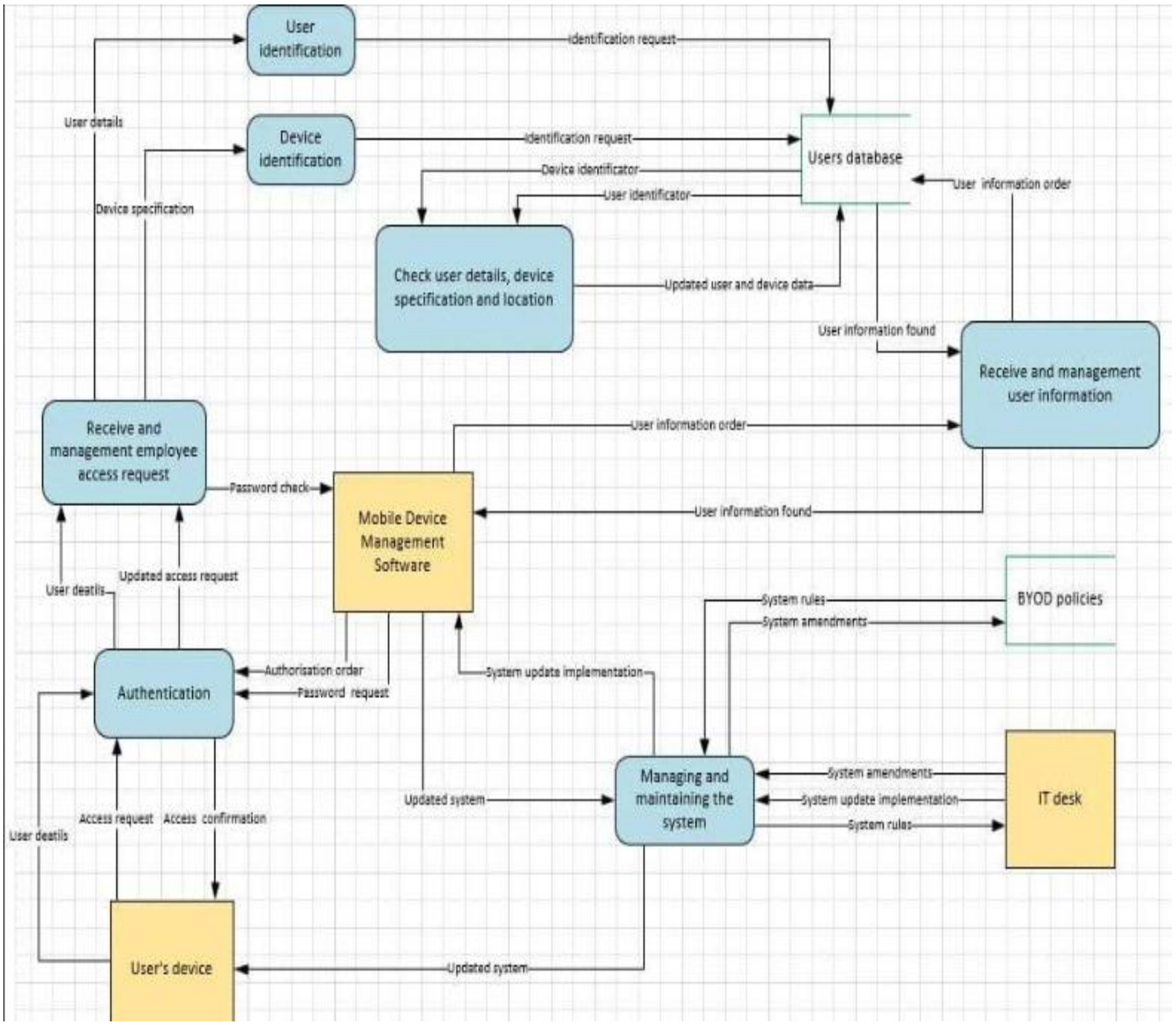

Figure 1: Architecture Diagram of Distributed System for Reddit

Middleware Explanation:

Figure 1 shows the architecture diagram of the entire distributed systems for reddit. The middleware in Reddit's distributed architecture is crucial for enabling efficient, scalable, and reliable service communication. Middleware components manage the interaction between different parts of the system, particularly focusing on message queuing, caching, and session management. Key aspects include:

- **Message Queues (e.g., Kafka, RabbitMQ):** These act as a buffer and a communication layer between different services, which is essential in a distributed system where components need to communicate asynchronously. For example, when a user makes a post, the post data is sent to a queue from which it is processed by different services for tasks like updating feeds, sending notifications, and indexing for search. This decouples the services, allowing them to operate independently and scale as needed.
- **Caching (e.g., Redis, Memcached):** Caching is used to store popular or frequently accessed data, such as the front-page posts or hot topics, reducing the number of direct queries to the databases, thus decreasing latency and load. This is especially important in Reddit's context where milliseconds in response time can significantly impact user experience.
- **Session Management:** Middleware also helps manage user sessions across different servers and processes, ensuring a consistent user experience without requiring re-authentication or data loss as users interact with different parts of the platform that may be served by different underlying servers.

This middleware layer ensures that Reddit can handle large volumes of data and user requests efficiently, maintaining performance and scalability even during peak loads.

3 Failure and Security Models for the Designed Distributed System

Failure Models:

To ensure the resilience and reliability of Reddit's distributed system, the following strategies are employed:

1. **Redundancy:** Critical components, such as load balancers, web servers, and database servers, are duplicated across different physical and geographical locations. This redundancy ensures that the failure of a single component or data center does not bring down the entire service.
2. **Data Backup and Replication:** Data is backed up regularly, and real-time replication is employed. This means data is continuously copied from primary servers to secondary servers, so if the primary server fails, the system can quickly switch to a secondary without data loss.
3. **Fault Detection and Recovery:** The system includes comprehensive monitoring tools that constantly analyze the health of all components. These tools can automatically detect hardware or software failures and initiate recovery processes such as rerouting traffic or restarting services without human intervention.

4. **Graceful Degradation:** In the event of partial system failure, the system is designed to degrade functionality gracefully. Non-critical features may be temporarily disabled to keep the core functionalities running smoothly.

Security Models:

Reddit's distributed system incorporates robust security measures to protect user data and ensure secure operations:

1. **Data Encryption:** All data is encrypted both in transit over the network and at rest on storage devices using strong encryption protocols. This prevents unauthorized access to sensitive user information.
2. **Authentication and Authorization:** Secure authentication mechanisms such as two-factor authentication (2FA) are implemented to verify user identities. Additionally, a comprehensive role-based access control (RBAC) system ensures users have access only to the resources pertinent to their role within the system.
3. **Network Security:** Firewalls and intrusion detection/prevention systems (IDS/IPS) are deployed to protect against external attacks. Regular security assessments and penetration testing are conducted to identify and mitigate vulnerabilities.
4. **Compliance and Privacy:** Adherence to international standards and regulations, such as GDPR and CCPA, ensures that user data is handled securely and with respect for privacy.

4 Features and Inter-process Communication Model

Reddit's distributed system supports several key features, each relying on sophisticated inter-process communication (IPC) mechanisms:

Features:

- **Content Submission:** Users submit content like posts and comments, which are then stored and displayed.
- **Voting System:** Users can upvote or downvote content, influencing its visibility.
- **Real-time Updates:** Live updates for comments, votes, and other interactions are provided to users.

Inter-process Communication Model:

- **RESTful APIs:** These APIs are used for synchronous communication between client applications and servers, facilitating requests and responses for user actions like posting or voting.
- **WebSockets:** For real-time features such as live comments and notifications, WebSockets provide a full-duplex communication channel that allows the server to send messages to the client without the client having to request it repeatedly.

- **Message Queues:** Services like Kafka are used for asynchronous communication among various backend services. This is crucial for decoupling services and ensuring that operations like sending notifications or updating feeds do not interfere with user interactions.
- **Event-Driven Architecture:** This architecture allows services to react to events as they occur. For example, when a new post is submitted, an event is generated which triggers various actions like indexing for search, updating user timelines, and triggering notifications.

5 Distributed File System Requirements and Architecture

Figure 2 gives a clear overview of the distributed file system for reddit.

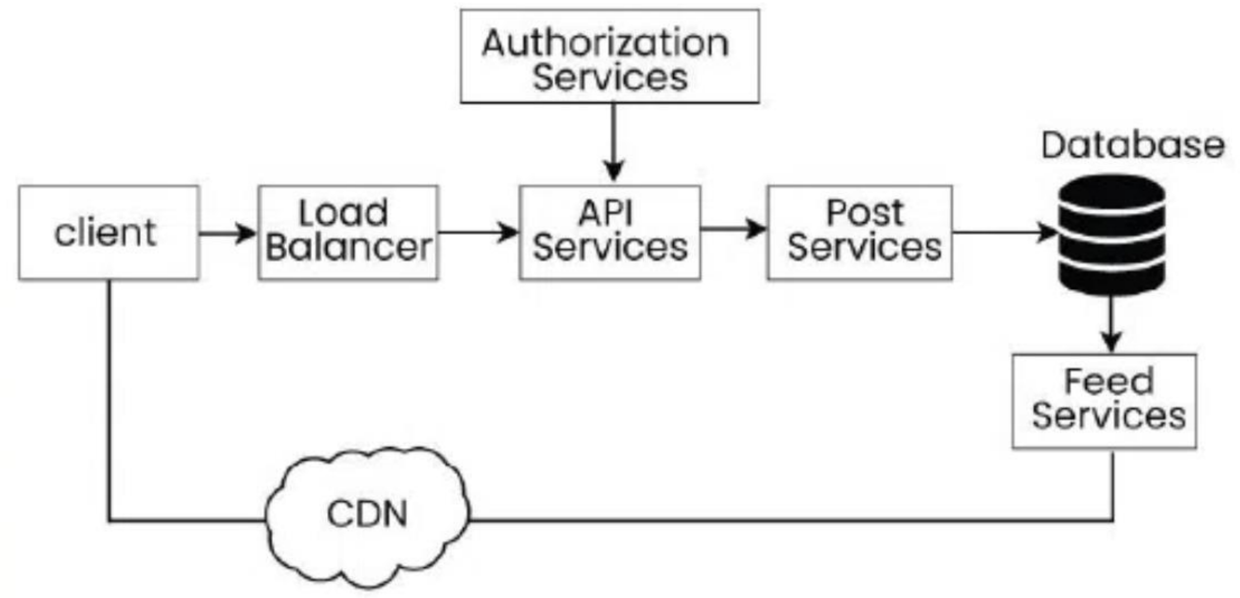

Figure 2: Distributed File Systems

Requirements:

Given the scale of Reddit's operations, the distributed file system must be highly scalable, available, and consistent:

1. **Scalability**: As Reddit receives millions of page views and user interactions, the system must scale horizontally to handle the increasing load without degradation in performance.

2. **Fault Tolerance**: The system must be resilient to failures, ensuring that the service remains available even if individual components fail.
3. **Consistency**: Reddit must maintain data consistency across various nodes to ensure that all users see the latest data regardless of which node serves their requests.
4. **Latency**: Minimizing latency is crucial for a seamless user experience. Distributing servers geographically can help in serving users from their nearest location, thus reducing latency.
5. **Load Balancing**: Proper load balancing mechanisms must be in place to distribute the traffic evenly across servers, preventing any single server from becoming a bottleneck.
6. **Data Storage and Management**: Reddit involves complex data structures, from user profiles to comments and voting. Efficient storage solutions like databases that support horizontal scaling (like NoSQL databases) are vital.

Architecture:

1. **Client Layer (Web and Mobile Applications)**
    - **Clients**: Serve as the front end for user interactions with Reddit. They handle user requests, process client-side logic, and present data, providing the interface through which users browse, post, comment, and interact with the community.
2. **Networking Layer**
    - **Load Balancers:** Distribute incoming requests across multiple servers to optimize resource use, minimize response times, and avoid overload on any single server, enhancing the system's reliability and scalability.
3. **Application Layer**
    - **API Services:** Act as gatekeepers that manage all API requests. They authenticate these requests, route them to appropriate backend services, and ensure smooth communication between the client layer and more granular backend services.
    - **Post Services**: Manage all functionalities related to posts, ensuring that user interactions with posts are processed accurately and stored reliably.
    - **Feed Services:** Tailor the user experience by generating personalized content feeds based on user activity and preferences, using sophisticated algorithms to sort and recommend content.
4. **Authorization Layer**
    - **Authorization Services:** Handle all aspects of user authentication and authorization, ensuring secure access to the system and protecting sensitive user data by verifying identities and enforcing access controls.
5. **Data Storage Layer**
    - **Databases:** Store and manage all structured data, including user profiles, post metadata, comments, and other critical data. They ensure data integrity, support complex queries, and enable efficient data retrieval.
6. **Content Delivery Network (CDN)**

- **CDN**: Enhances content delivery speeds by caching static resources closer to users. This network reduces the load on the main servers and accelerates the delivery of media-rich content like images and videos, crucial for an engaging user experience.

This layered approach delineates the roles of different components within the distributed architecture, demonstrating how they collectively support the functionality, scalability, reliability, and performance of Reddit. Each layer contributes to handling specific types of tasks and data, which are crucial for managing the vast and dynamic environment of a popular social media platform. This structure not only simplifies scaling but also facilitates maintenance and enhances fault tolerance by segregating duties across different components and layers.