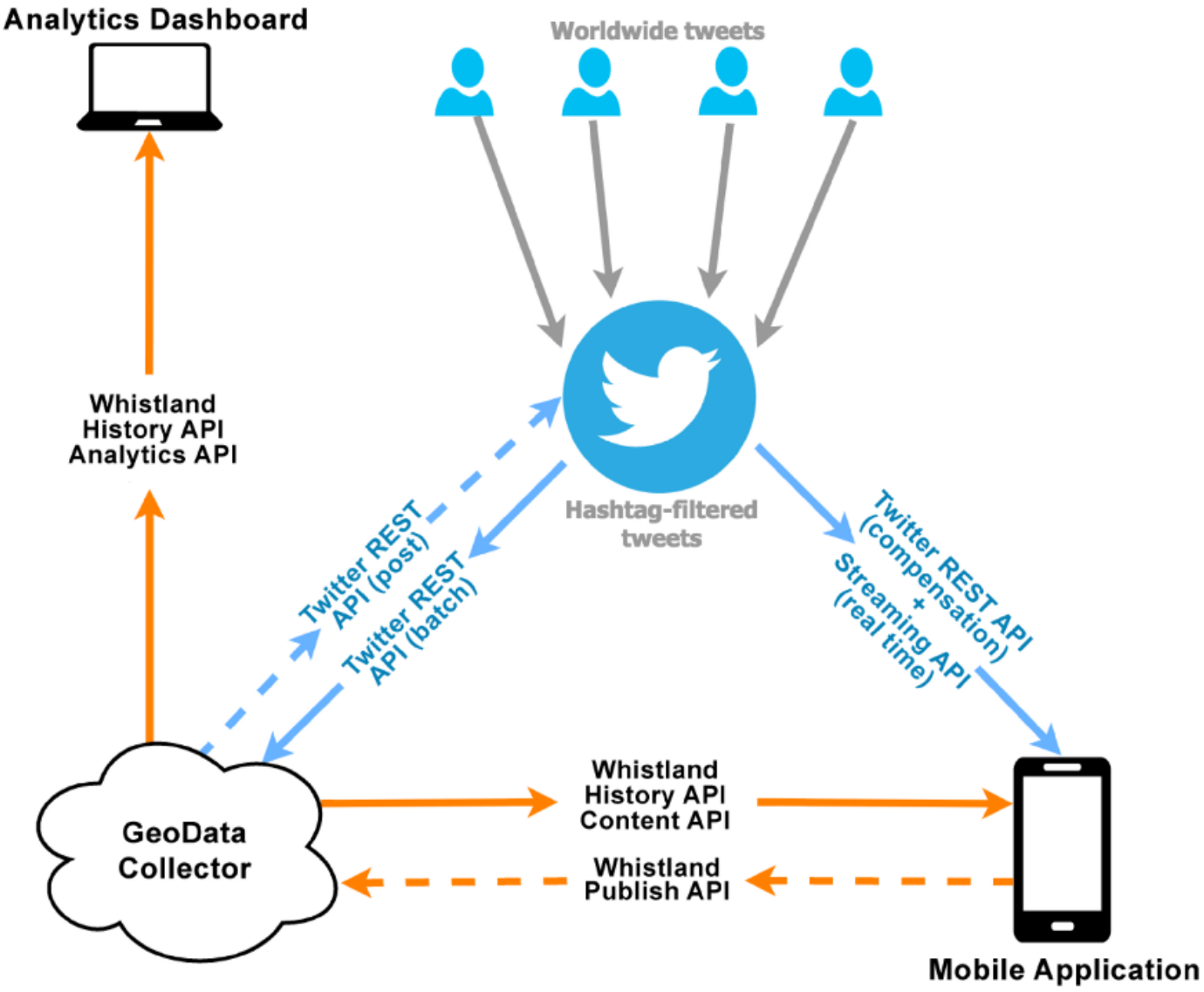
Twitter System Architecture

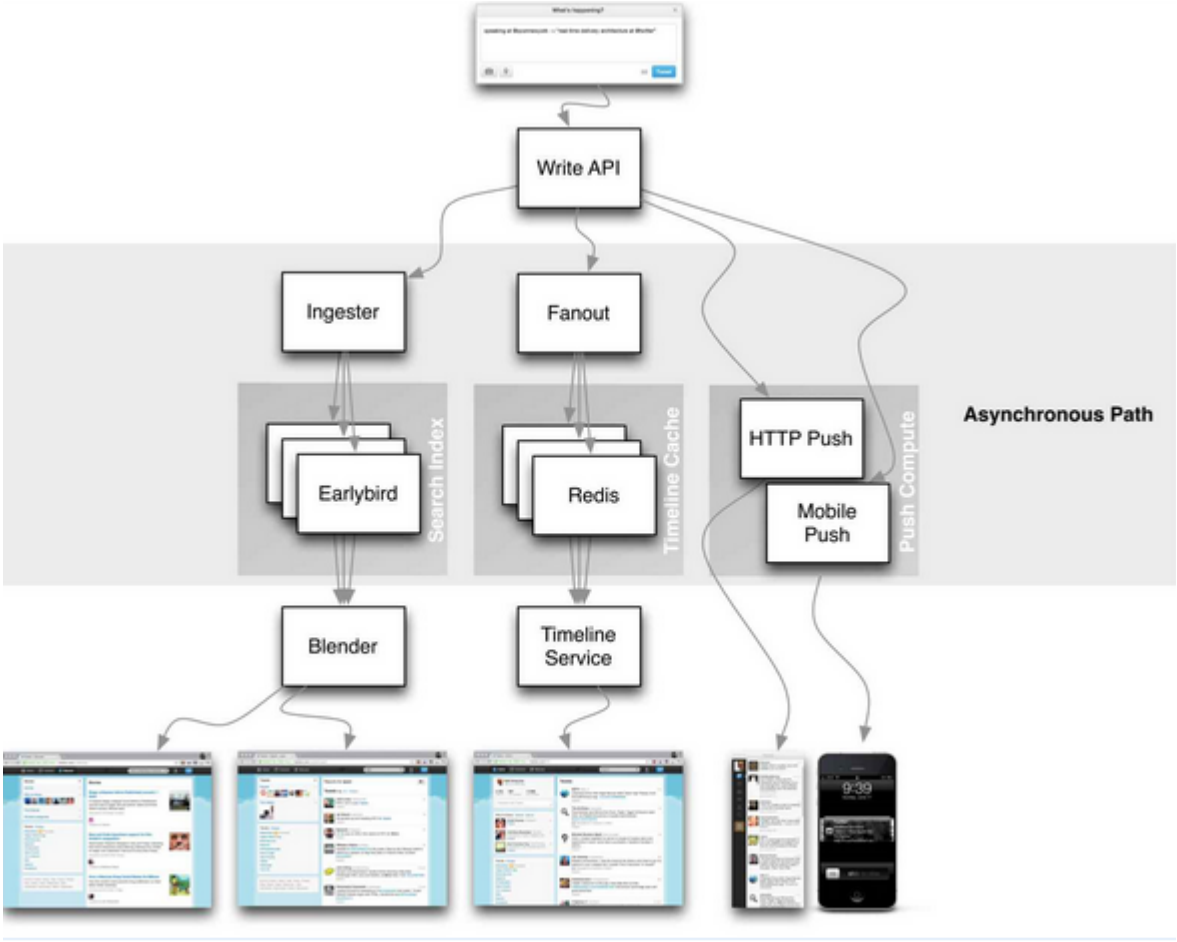
Please clap and share if you like.



Twitter is an online social networking service where users can post and read short messages called “tweets”. Most candidates will design twitter as a monolithic service in system design interviews. However, designing a gigantic service like Twitter as a monolithic service shows that the candidate lacks experience in designing distributed systems. Nowadays, it is normal to design distributed systems in terms of microservices and even lamdas (or functions). The current trend is that no one is writing a new service as a monolithic service and companies are gradually converting their big monolithic services to a set of microservices. Hence, candidates should design Twitter with a microservice.

Functional requirements

1. The users can post or share new tweets
2. the size of a tweet is 140 characters at most
3. The user can delete his tweets but not update/edit his posted tweets (this is a write operation)
4. The users can mark favorite tweets (write operation)
5. The users can follow or unfollow another user (write operation) — following a user means that the users can see the other user's tweets on his timeline
6. 2 types of timelines can be generated (read operation) — a user timeline comprising of the last N number of his tweets. A home timeline that is generated from the popular tweets of the users he is following in the descending order of time
7. The users can **search** the tweets based on keywords (read operation)
8. The users need an account in the service to post or read tweets (we will be using an external identity provider for the time being.
9. The users can register and delete the account
10. Twitter supports tweets comprising text and pictures/videos. However, we will only support text tweets.
11. Analytics/Monitoring of the service to determine its load, health, and functionality
12. Analytics also provides the user's suggestions or recommendations for who to follow, tweet notifications, trending topics, push notifications, and sharing a tweet.

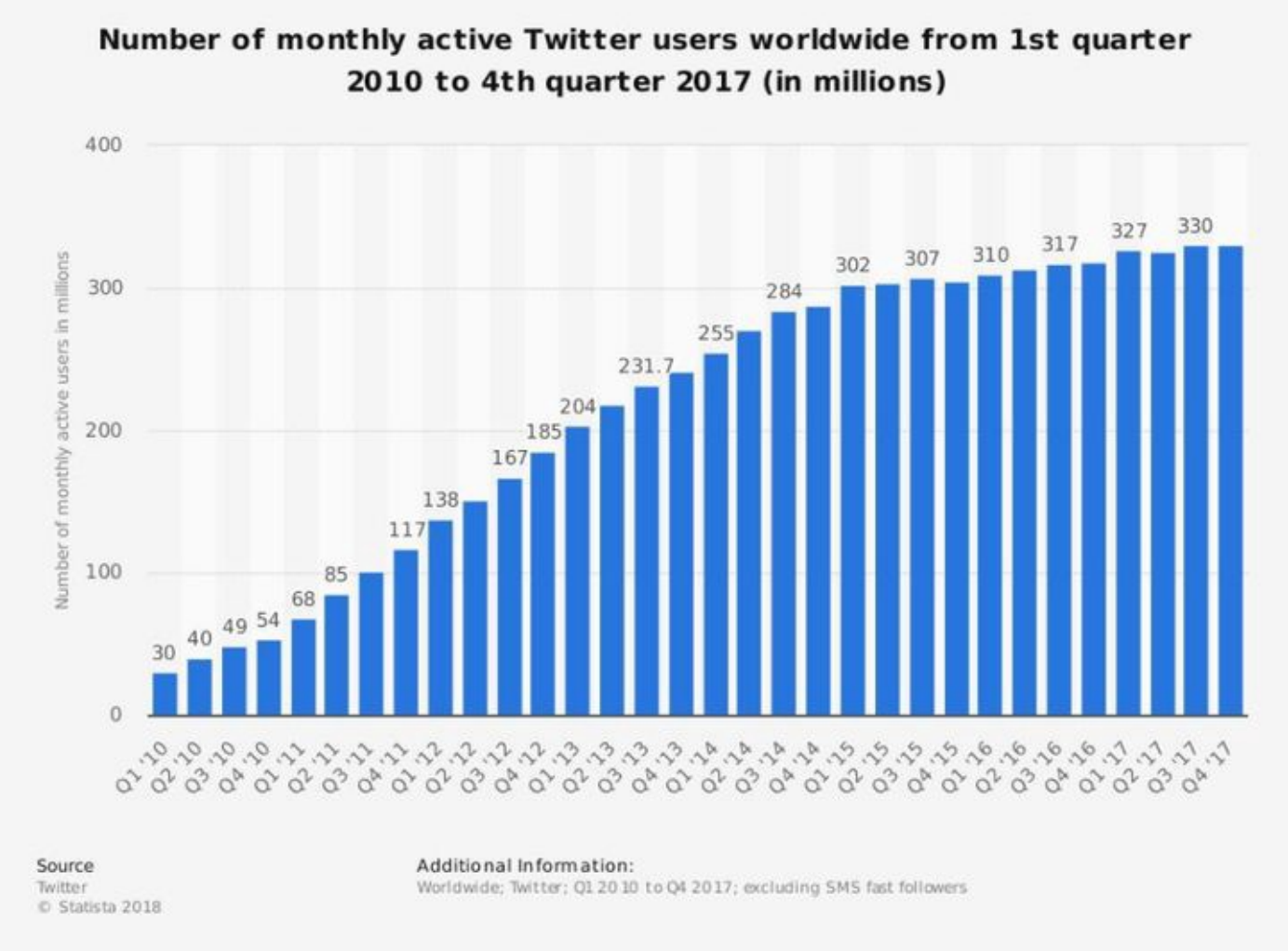


Non-functional requirements

1. The important requirement is that the service should be highly available. It means that the users can read tweets on their home timeline without any downtime
2. Generate the timeline must be within half a second at most
3. The system doesn't need very strong consistency — eventual consistency is applicable. Keyword database is for searching tweets based on keywords.
4. The system should be scalable with the increasing load for increasing users and increasing tweets
5. User data should be durable

Now, we do some mathematics

- the number Daily active user average request/ day = 150M*60/86400 = 100k per second
- Peak users = average concurrent user* 3 = 300k
- Max peak users in 3 months = Peak users *2 = 600k
- Read QPS (queries per second) = 300k
- write QPS (queries per second) = 5k

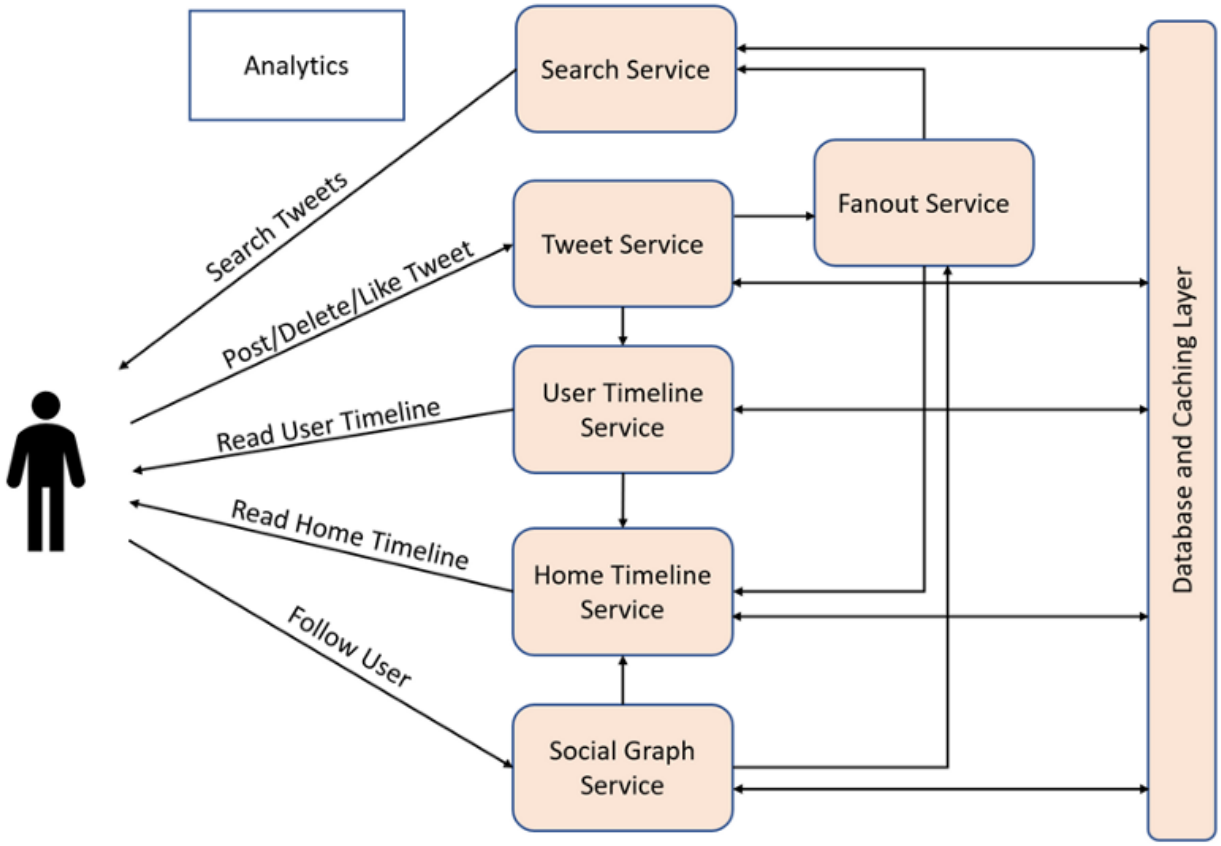


The high-level design of the Twitter service

For such complexity of the system design, we can divide and conquer into several services which comprises several micro-services

1. Tweet service
2. User timeline service
3. Fanout Service
4. Home timeline service
5. Social graph service
6. search service

The following is the architecture of different logical components or micro-services in the Twitter service.

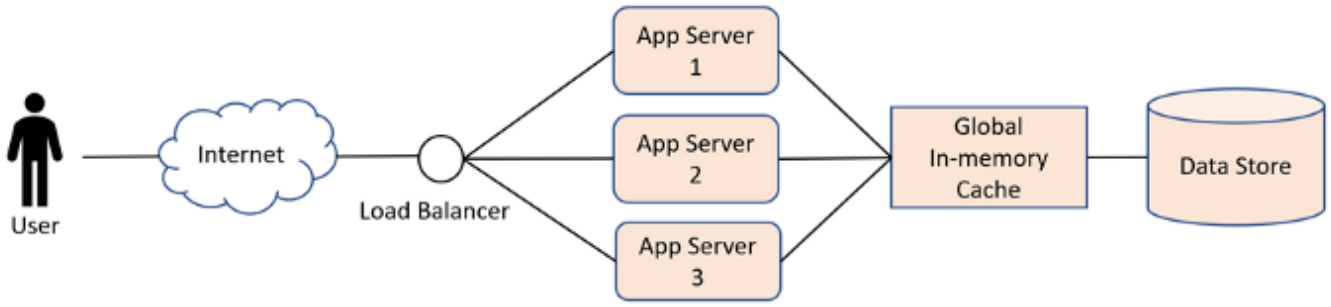


The Detailed Design of the Twitter service

All these micro-services can be called modules.

1. **Tweet service**

- receiving user tweets, forwarding user tweets to the follower's timeline and search service
- store the user's information, tweet information, including the number of tweets from a user, user likes
- It comprises application servers along with a distributed data store that is behind a distributed in-memory cache or an in-memory cache backed by a data store (e.g., Redis)



Now, we show the database schema of our tweet service.

	Users
PK	UserId: varchar(16)
	Name: Varchar (100) Email: Varchar (100) CreationTime: datetime LastLoginTime: datetime IsHotUser: boolean

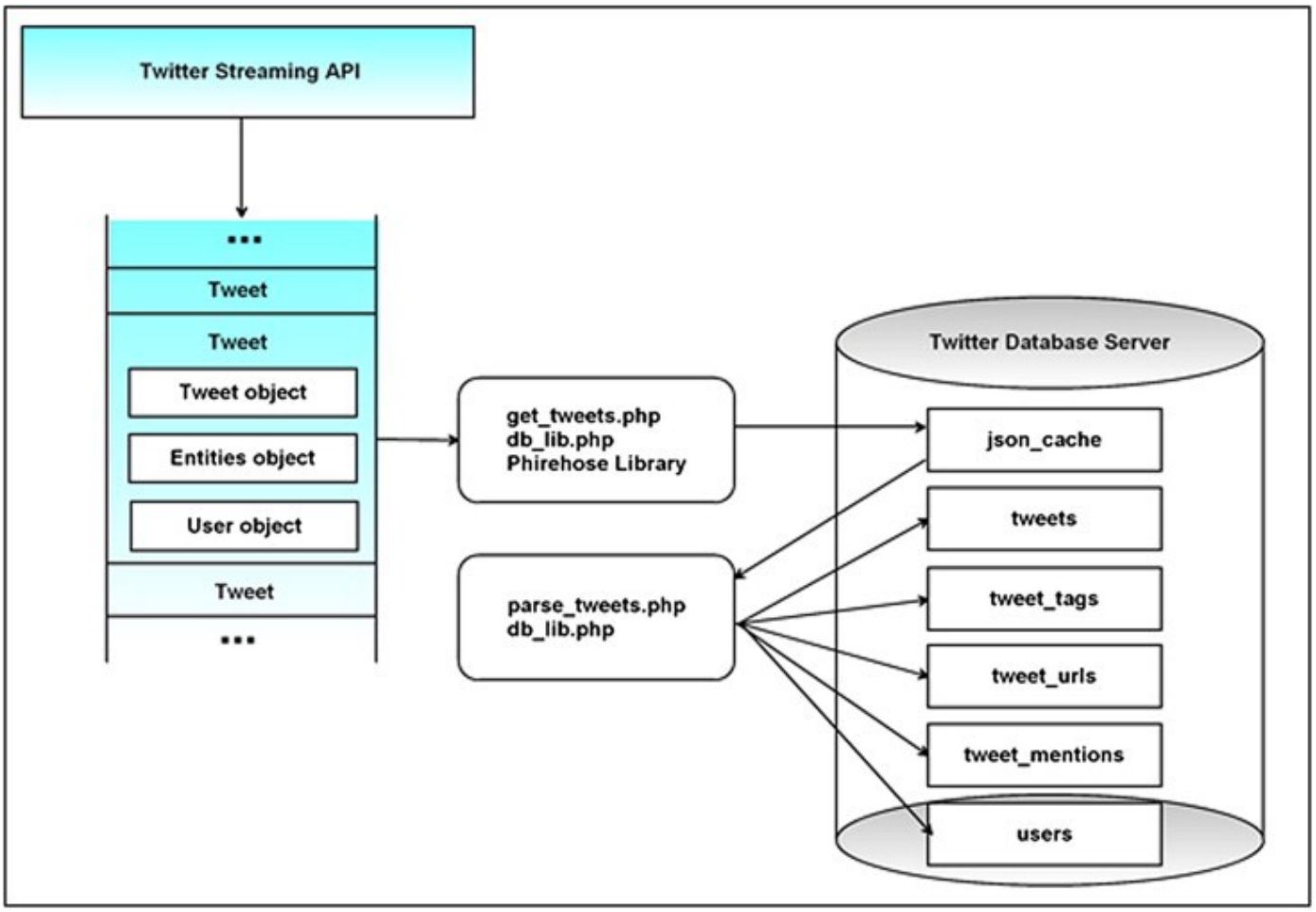
	Tweet
PK	TweetId: varchar (16)
SK	UserId: varchar(16) CreationTime: datetime Content: varchar(140)

	Favourite_Tweet
PK	TweetId: varchar (16) UserId: varchar(16)
	CreationTime: datetime

PK: Primary Key
SK: Secondary Index Key

The user's table contains all the information about different users, and the tweet table stores all the tweets. The favourite_tweet table stores favorite tweet records, i.e., whenever a user likes a tweet, then a record is inserted in the favourite_tweet table.

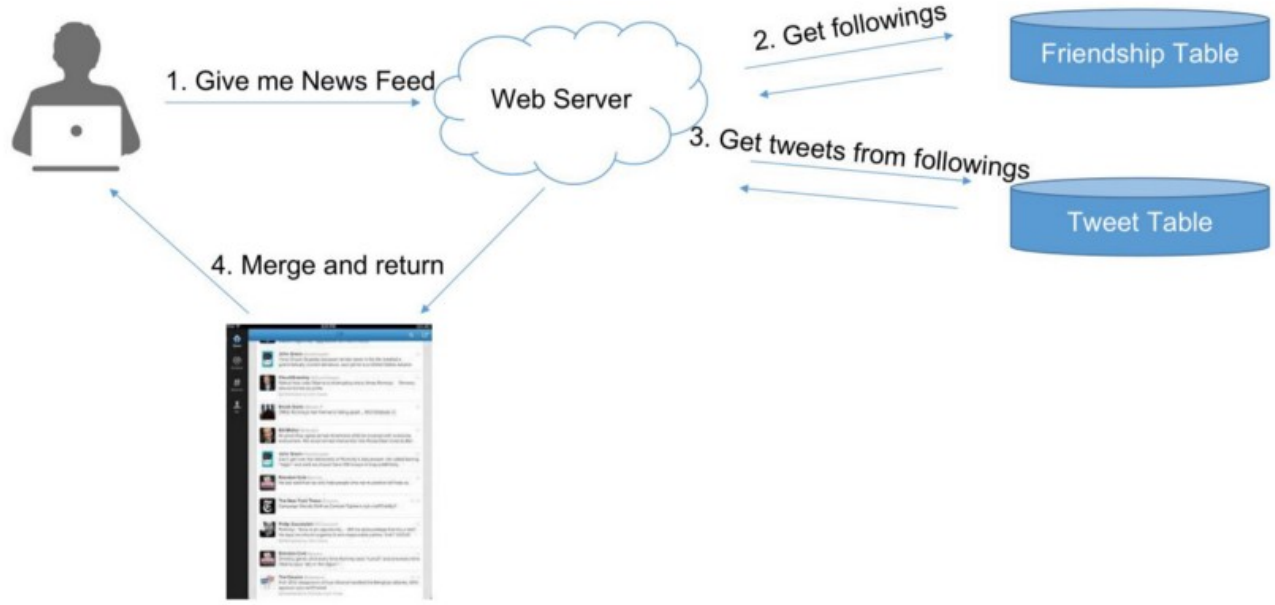
2. Generating a unique tweet Id

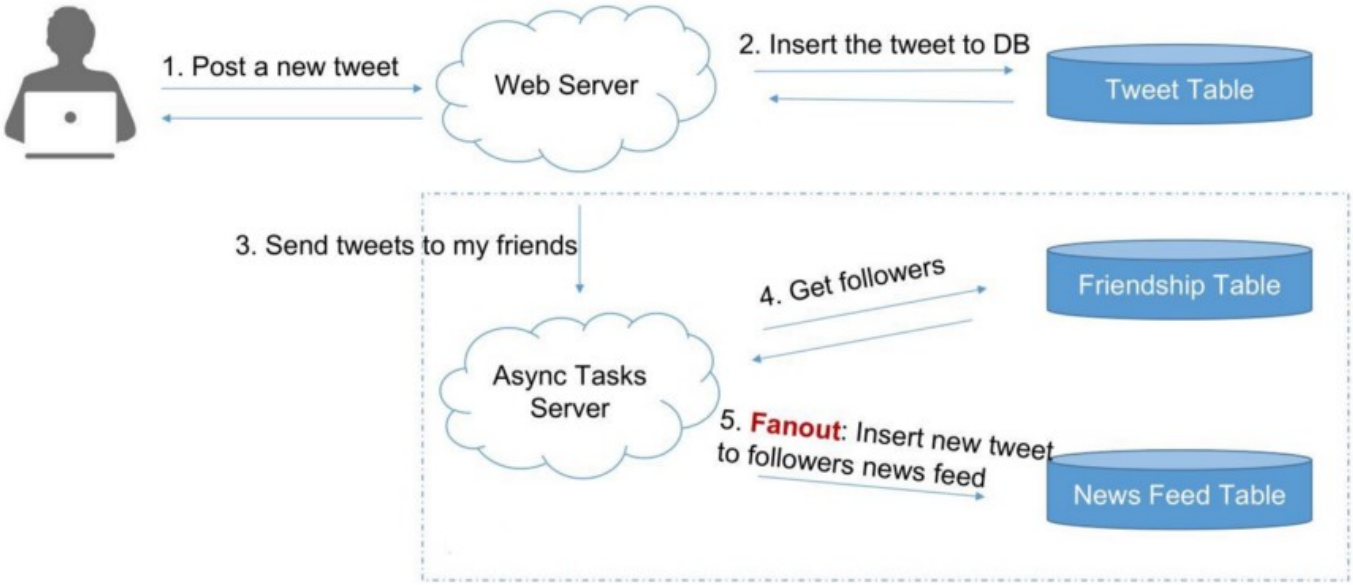


- the users call postTweet(), the call comes to one of the app servers. The app server generates a unique id for the tweet. We can use the same mechanism to generate short URLs, for the tweet. Or another method is a Universally unique identifier (UUID) on the app server. After the Twitter ID is generated, the app server inserts the tweet into the distributed cache and the tweet table in the datastore. The write-through cache mechanism is used because it can write all the create/update/delete operations to be performed to both the cache and the datastore simultaneously.

3. Scaling the tweets

- We can partition both the distributed cache and the datastore into several partitions and replicas.
  * shard by user ID
  *Shard by tweet ID
  * Two-layer/ level shard by user ID and tweet ID

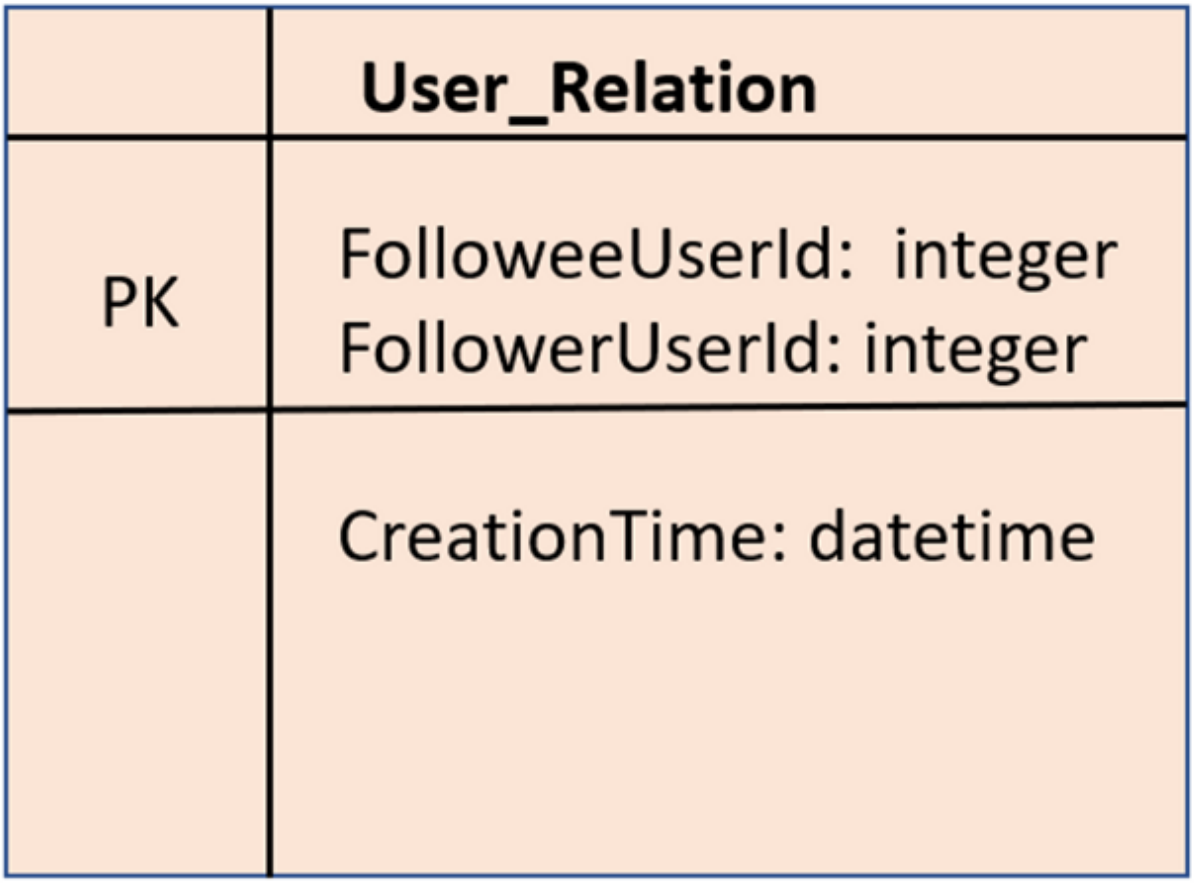




4. Social Graph Service

- Keep track of which users are following which users, we implement the following APIs

- Also have application servers, a write-through distributed cache, and a datastore
- The database schema that is used to store user relationships

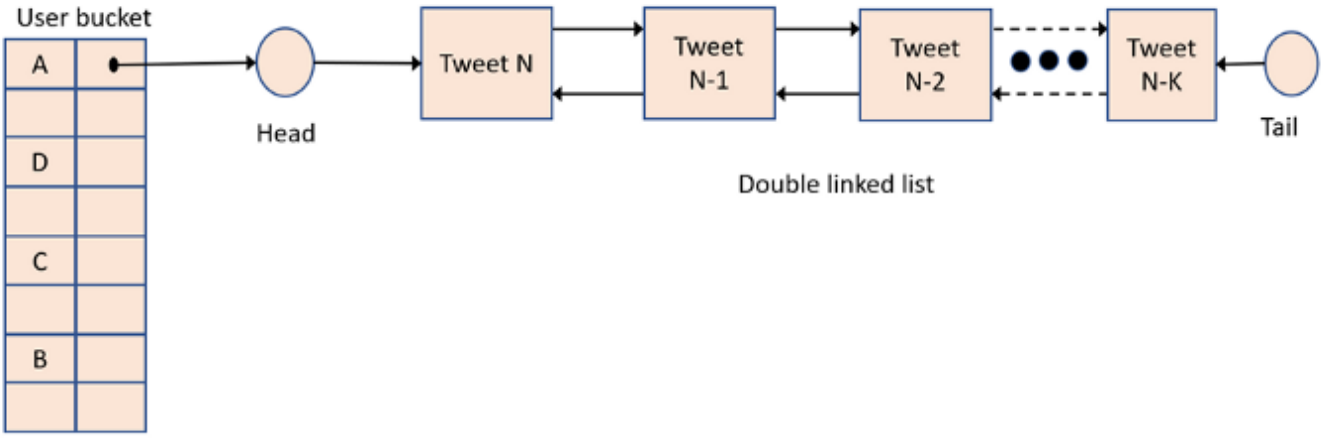


- **The following API**
- merge his timeline into your news feed asynchronously
- After unfollowing a user, asynchronously remove his tweets from your newsfeed
- pick out tweets from the news feed asynchronously
- The asynchronous async is used because this process is not fast at all, and users get feedback quickly while they follow and unfollow the other users immediately

- The disadvantage of asynchrony: refresh the news feed after unfollowing and find that this information is still there but it will be deleted eventually

5. User Timeline service

- This can return the user his timeline which contains all the user tweets in descending order of time. this service is for the home timeline or other users' timeline.
- This service comprises the application servers and the distributed in-memory cache, but has no datastore involved in this service.
- The user timeline is designed using a data structure comprising a linked list of user tweets



- When a user posts a tweet, the tweet service calls the user timeline service to insert the tweet at the head of the user timeline tweet list. This is a 0/1 operation.
- Also, the analytics dashboard can configure K to a suitable value. Initially, the K is equal to 1000 tweets. The last K tweets in the user timeline can be kept because K is configurable.
- In the user timeline list, the tweets are stored in descending order of the creationTime. When the user timeline list reaches its maximum size of K tweets, we will remove the oldest entry.

6. Fanout Service

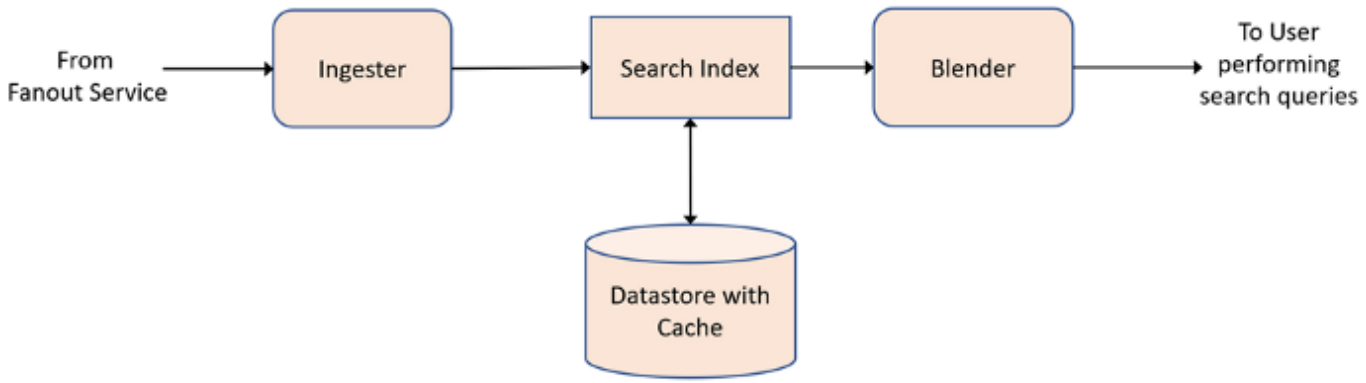
- to forward new tweets to search and home timeline services. In case there are other components/ micro-services as well like trending or notification services
- comprises multiple distributed queues
- When a user sends a tweet message, this service will enter the message in the queue for the tweets because the social graph service has to get the list of followers of the user and insert as many messages in the second set of queues. For celebrity users, they have a very large number of followers greater than the threshold. So, how can this be handled?
- It is a first-in-first-out task queue list. It will process doing tasks share the same list and feedback the queue server after completion. Queue server is an essential part of asynchronous tasks. So, it might not have an immediate effect, but it is eventual consistency.

7. Home timeline service

- To display the user's home timeline
- This comprises all the tweets from the other users, which the user is following. The tweets are displayed in descending order of creation time of the tweet.
- The design is similar to that of the user timeline service.
- But it is a bit more complex than the user timeline service because the user will insert the latest tweet and remove the oldest tweet when the user exceeds the K value and also the user is following many other users (followee), the service needs some mechanism to rank the tweets from the different followee users.

8. Search service

- to serve users' search queries
- the fanout service passes the tweet to the search service



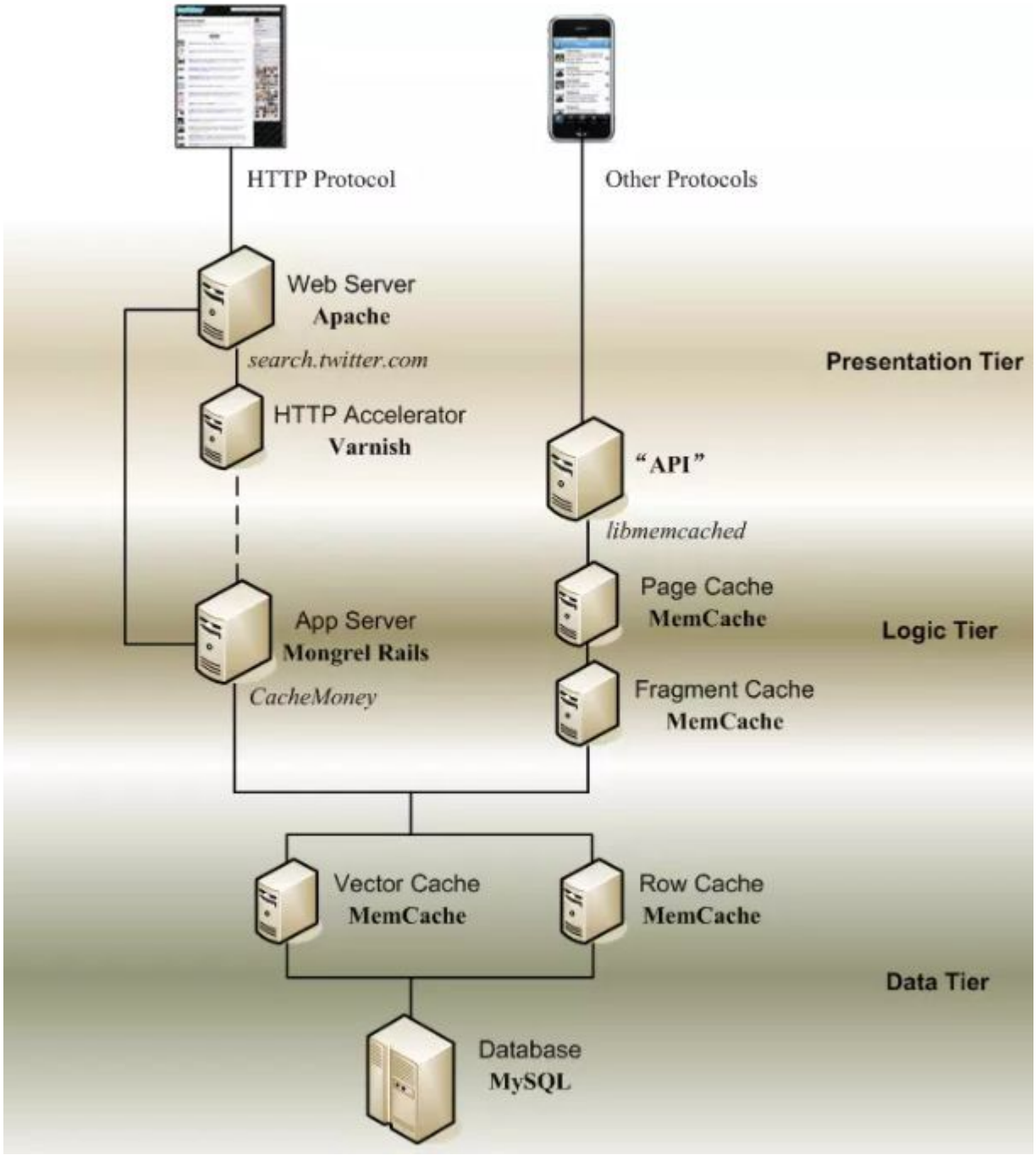
- The ingester (or ingestion engine): tokenize the tweets into a number of tokens or terms or keywords. For example, this tweet: “I want to become a very rich man like Amazon Jeff Bezos”… will then filter out those words which are not useful to be used in a search. All the words will be discarded except for Jeff Bezos and Amazon. The ingester can have this list of words either from configuration or from some database to which it is connected.
- A process called “ stemming” on the remaining words to determine their root words. Stemming is the process of reducing inflected (or derived) words to their word stem, base, or root form. So, there is a lookup table is needed in the database. The advantages of this approach are simple, fast, and easy handling of exceptions. The disadvantages are new or unfamiliar words are not handled, even if they are perfectly regular.
- Pass to the search index
- Search index micro-service will create an inverted index and store a mapping index of terms from the content such as words to their location in a document or set of documents, which in our case, is a tweet or set of tweets.
- Blender service: serve the search queries by the users on the Twitter platform. It first determines the search terms when a search query is requested. IT also does a process of steaming and then uses the root words to run search queries on the inverted index of terms.

9. Photos and Videos

- NoSQL Database is used
- Media Files (file system is used)
- Data schema

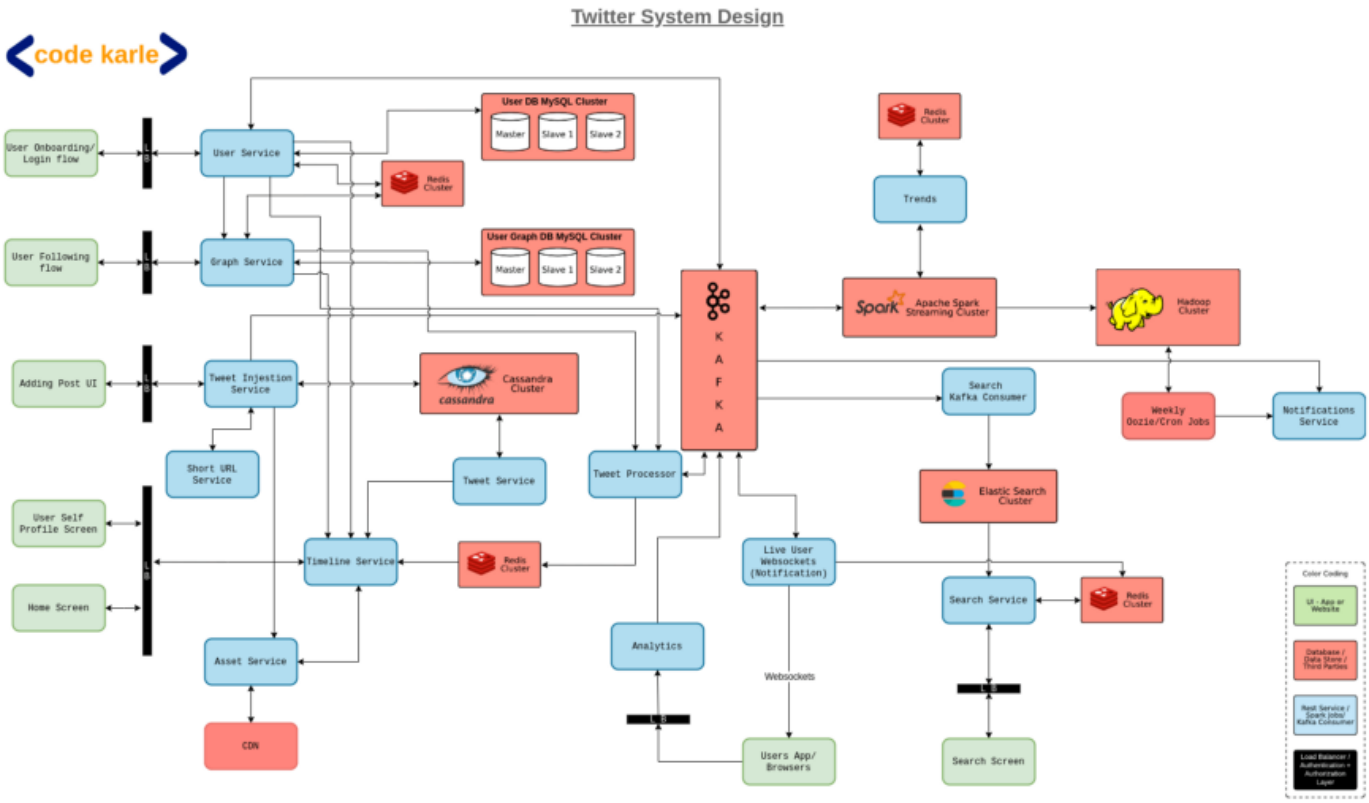
id	status	file	has_deleted	deleted_at	created_at	user_id
1	0	/img/1.jpg	0	null	2021-04-06 12:00	1
2	2	/img/2.jpg	0	null	2021-04-06 12:00	1

Network for Twitter

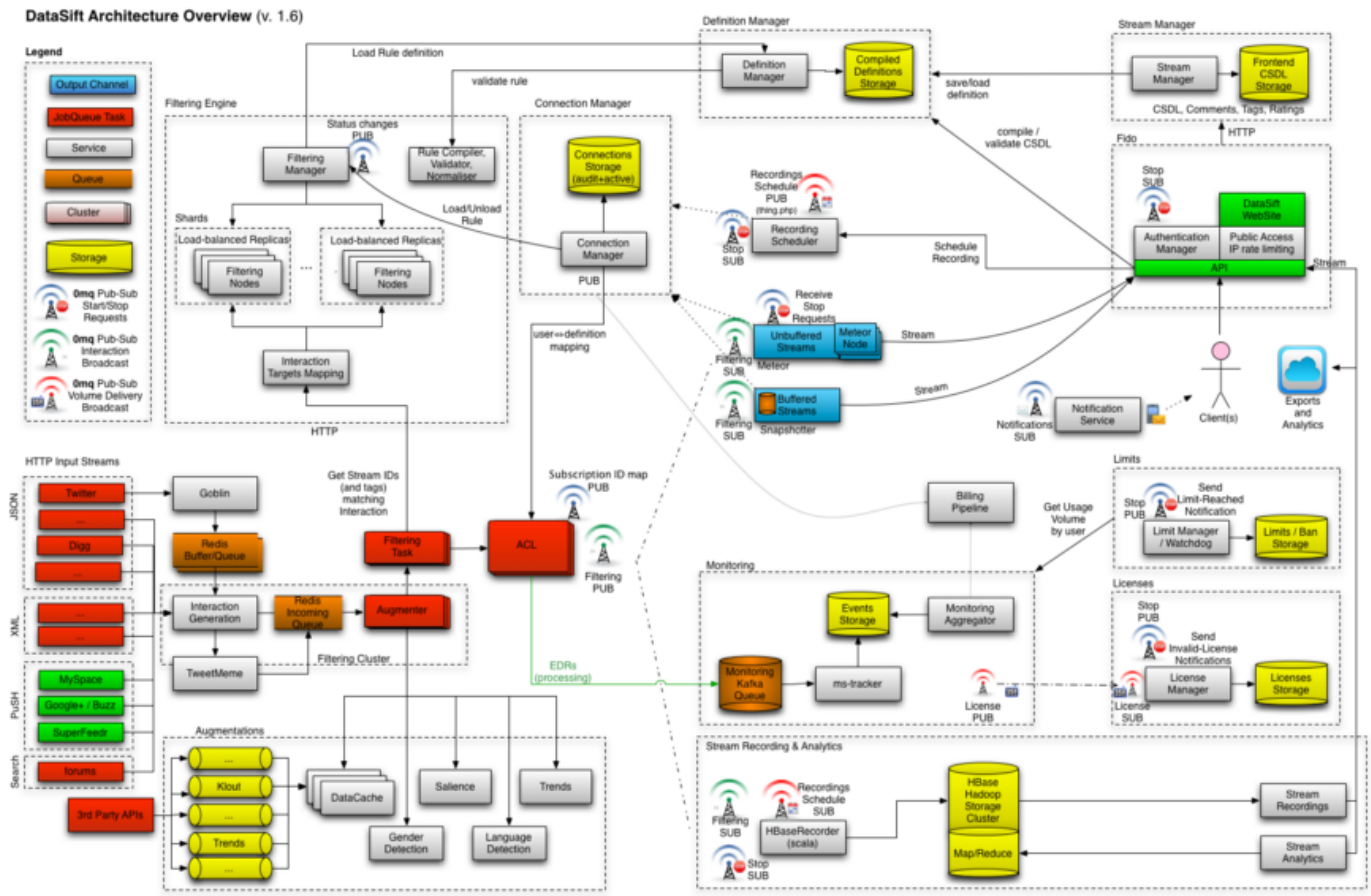


The final detailed design for the twitter

System design



Data Architecture



References

1. http://highscalability.com/blog/2011/11/29/datasift-architecture-realtime-datamining-at-120000-tweets-p.html
2. https://www.codekarle.com/system-design/Twitter-system-design.html
3. https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale
4. http://highscalability.com/blog/2013/7/8/the-architecture-twitter-uses-to-deal-with-150m-active-users.html