NoSQL Data Stores

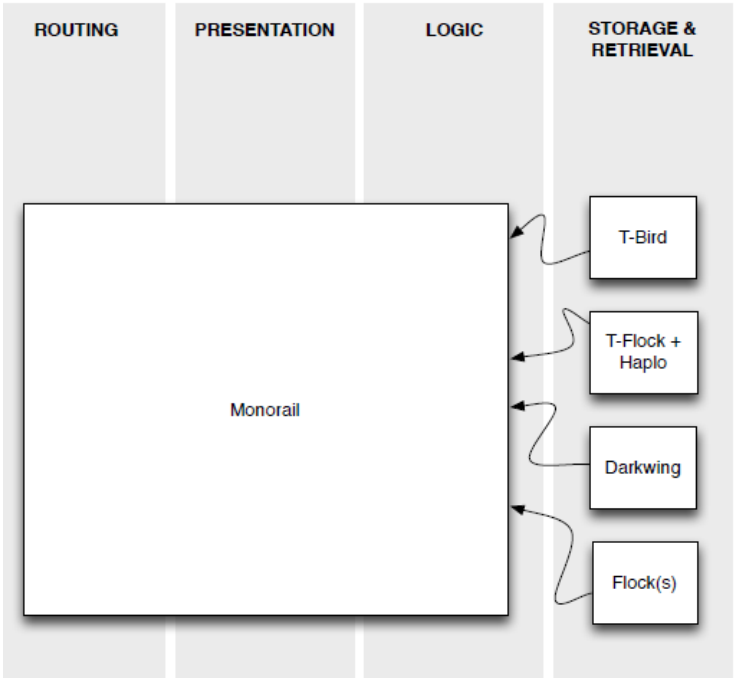
NoSQL is a schema less cluster-oriented database that is being used in Twitter Architecture. It provides platform services in milliseconds across worldwide. Typical Architecture and Framework that Twitter uses is covered in this scope of essay.

Twitter comprises of 4 types of timelines, out of which two are push-based and two are pull-based.

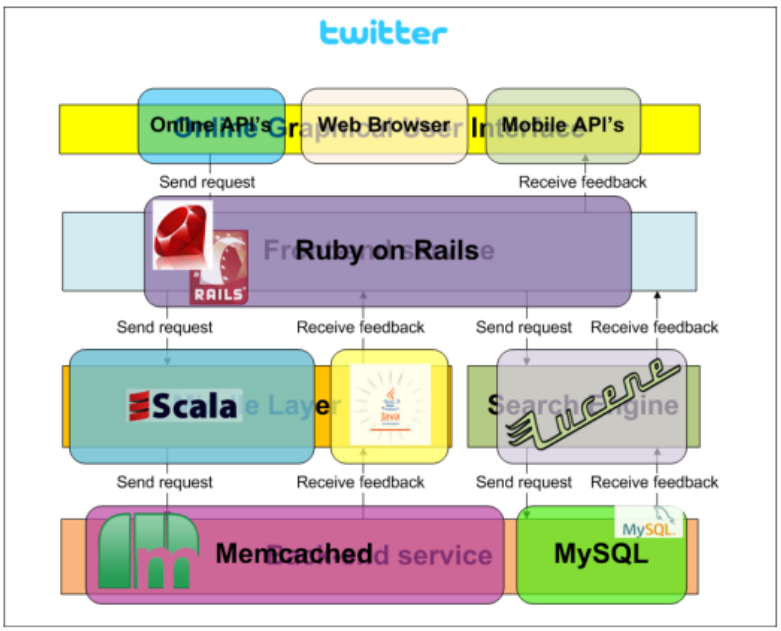
1. Push-based: It has the http push and mobile-push mechanisms to do real-time delivery.
2. Pull-based: It has the user's timeline and explicit search query [1].

Twitter Architecture:

Twitter uses a basic 4-layer architecture where it has the largest Ruby Monorail App which is formed by the Routing, Presentation, Logic layers and has Database on its back-end. Database is built by many storage applications like T-Bird, T-Flock, Darkwing and Flocks storing user tables, tweet tables and social graphs, etc. [1]



Real Time Delivery in Twitter [1]

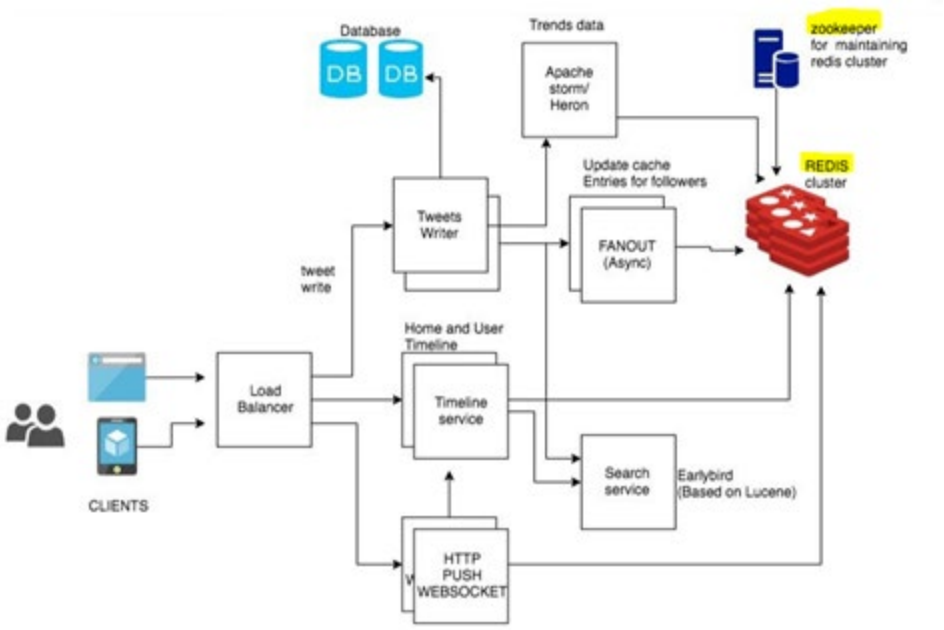


Overwritten Twitter Architecture [2]

In recent times, Twitter uses Memcached database, which is the Redis NoSQL database to reduce the read-heavy in Twitter.

NoSQL Datastores in Twitter:

1. **Key – Value Stores**:
   Redis is Memcached database which provides in-memory cache. It is a key-value database where user id is the key. Redis has its own data-structures like lists, sets, hash tables, etc to save the values. [3]



System Design for Twitter [3]

Role in Twitter:

Redis performs the following functions:

1. When a user tweets, it goes to the Load balancer and moves to Tweet writes. It is persistently stored in database and fan-out asynchronously. It is then written onto the users' timeline which is usually maintained on Redis Cache.
2. The tweet is inserted onto the tweet-user's home timeline Redis cache as well.
3. Thus the write operation is optimized using these pre-cached Redis instances. When one writes a user tweet onto the Redis, it is fan-out to multiple Redis cache to avoid single-point failure.
4. It also provides faster access to tweets as one is using the in-memory cache and not querying from database at every instance.

Zookeeper:

This provides a co-operation service for distributed components. Redis will have many nodes in the cluster and there will be a master node to co-ordinate all the other nodes. Zookeeper ensures that all the nodes are configured properly and check if they are online or offline [3].

2. **Column-Family Datastore:**
   a. **Hadoop:**
      Hadoop is a distributed computing framework. It breaks the file into chunks of blocks and distributes the data so that even if 2 out of 100 nodes or systems goes down there won't be any data loss. [4]

      Hadoop in Twitter:
      It provides **scalability** and **interoperability**. Hadoop helps hosting 300PB of data on ten thousands of servers. Instead of restoring it in a single large java heap space, one can scale the node count. Also, one can have one single namespace or URI to access the cluster instead of individual namespaces which is interoperable.
      Twitter sends emails of interested tweets to users using Hadoop.

   b. **HBase:**
      HBase is column-oriented distributed database built on top of HDFS.

      HBase in Twitter:
      Twitter runs HBase across the Hadoop cluster and "People Search" function in Twitter application rely on HBase for data generation. HBase is also used as a time series database for cluster monitoring. HBase uses Zookeeper which helps co-ordinates the master node in a cluster [6].

3. **Apache Cassandra**:
   Cassandra is a column-store NoSQL Datastore. It is an eventually consistent database. [5]

   Cassandra in Twitter:
   Cassandra avoids long-way delay for Twitter users, while allowing some latency in fetching the tweets in the last fraction of seconds. Then to

ensure increased consistency, Twitter used Cassandra along with Gizzard, a strongly consistent database.

4. **FlockDB**:
   It is a graph datastore which shows the connectivity of users and followers in Twitter [4].

References

[1] InfoQ. (2019). *Real-Time Delivery Architecture at Twitter*. [online] Available at: https://www.infoq.com/presentations/Real-Time-Delivery-Twitter/ [Accessed 17 Nov. 2019].

[2] Lossek, Matthias, Rik Janssen, and Tim de Boer. "Matthijs Neppelenbroek." (2011) [online] Available at: http://www.timdeboer.eu/paper_publishing/Twitter_An_Architectural_Review.pdf [Accessed 18 Nov. 2019]

[3] Medium. (2019). *System design for Twitter*. [online] Available at: https://medium.com/@narengowda/system-design-for-twitter-e737284afc95 [Accessed 17 Nov. 2019].

[4] Anon, (2019). [online] Available at: https://blog.twitter.com/engineering/en_us/a/2010/hadoop-at-twitter.html [Accessed 16 Nov. 2019].

[5] Metz, C., Metz, C., Barber, G., Matsakis, L., Pardes, A., Edelman, G., Finley, K. and Knight, W. (2019). *This Is What You Build to Juggle 6,000 Tweets a Second*. [online] WIRED. Available at: https://www.wired.com/2014/04/twitter-manhattan/ [Accessed 16 Nov. 2019].

[6] Aljawarneh, Shadi. [online] Available at: https://books.google.co.uk/books?id=6K2eBQAAQBAJ&pg=PA308&lpg=PA308&dq=Cloud+Computing+Advancements+in+Design,+Implementation,+and+Technologies+pdf&source=bl&ots=8jgM2OZTvL&sig=ACfU3U07EyicofBNpiKHQ6AjBgzUTERuaw&hl=en&sa=X&ved=2ahUKEwjAm5C7wvPlAhXHTMAKHaNICpwQ6AEwB3oECAoQAQ#v=onepage&q=Cloud%20Computing%20Advancements%20in%20Design%2C%20Implementation%2C%20and%20Technologies%20pdf&f=false [Accessed 18 Nov. 2019]