Discuss the NoSQL data stores used addressing the following questions & Discuss how partitioning and replication is implemented in the architecture of Twitter

Twitter uses Redis as it NoSQL store. Redis has many features, there are session caches which allows a long time session to be open which means other servers are able to access the data fairly quickly because the session is still open. Page loadings are consistent because of full-page being cached this allows data to be backed up making it quite safe. Pub/Sub are able to broadcast messages to every listening server it provides a consistent way of talking to devices that have a link to the broadcast. The queues in Redis is good for performance as it is in memory storage engine is optimized for messages (Engel, 2017). Analysis of real-time events for Redis is easy for finding specific data example finding the person who tweets most the system can just pull that information. This makes data miners life easier because it allows collecting certain data they need. These are generally the good points of using Redis as a NoSQL data store, the problems of using Redis is that the memory hard to manage, the flexibility of the system is limited, the security of Redis is basic because of the instance level and requires a lot of Redis instance to be deployed which holds back the scalability.

The reason Redis was chosen was that of its real-time system, this allows data to be processed without any delays. Krikorian (2012) mentioned that it was more of a historical thinking of why they choose Redis, and it was a native list which allows them to keep similar to the architecture overall he thought it was a safer choice by choosing Redis.

Image from: (Krikorian, 2012)

Twitter uses a logical partition to sort and store data. The types of data it stores are users, tweets and timelines. Every type of data store has been changed to accept specific data type and the data store can access between 10 to 50 million queries per second and can run many instances (Hashemi, 2017). Twitter also implements a segment partitioner that splits data into smaller parts from the already acquired tweets to make storing the information simple and faster. Another implementation of partitioning is two-dimensional sharding it sends segments on to other Earlybirds, that expands Twitter clusters and the Earlybird machines can be extended to have a greater serving capacity (Zhuang and Burstein, 2014).

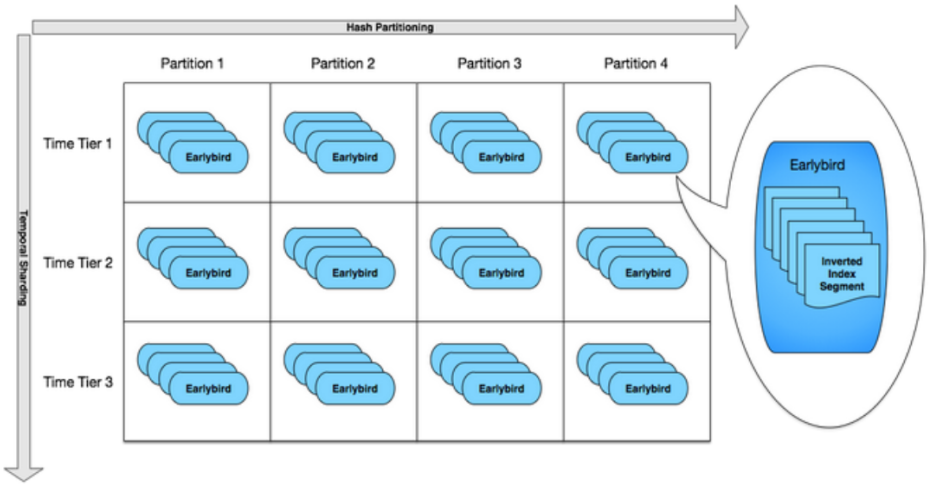

image from: (Zhuang and Burstein, 2014)

Twitter uses prioritization as one if its replication. The process of this is to delay low-priority storage with long service level agreement. This helps the network by using the full capacity and uses the correct resources (Hashemi, 2017). Hashemi (2017) said that they also implement dynamic bandwidth management to help create more LSPs to handle the amount of traffic by removing packets when not needed. Again Hashemi mentioned improving the network they brought in route reflection to make the clients into more routes reflectors (Hashemi, 2017). Twitter also implemented a high performance replicated log service, it replicates data to different dataset independently and this would make all replicas have the same data (Stewart, 2015).

Bibliography:

Hashemi, M. (2017). The Infrastructure Behind Twitter: Scale. [online] Blog.twitter.com. Available at: https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale.html [Accessed 12 Nov. 2018].

Zhuang, Y. and Burstein, P. (2014). Building a complete Tweet index. [online] Blog.twitter.com. Available at: https://blog.twitter.com/engineering/en_us/a/2014/building-a-complete-tweet-index.html [Accessed 12 Nov. 2018].

Stewart, L. (2015). Building DistributedLog: High-performance replicated log service. [online] Blog.twitter.com. Available at: https://blog.twitter.com/engineering/en_us/topics/infrastructure/2015/building-distributedlog-twitter-s-high-performance-replicated-log-servic.html [Accessed 12 Nov. 2018].

Engel, J. (2017). Top 5 Redis use cases | ObjectRocket. [online] ObjectRocket. Available at: https://www.objectrocket.com/blog/how-to/top-5-redis-use-cases/ [Accessed 12 Nov. 2018].

Raffi Krikorian. (2012). Real-Time Delivery Architecture at Twitter. [Online Video]. 23 October 2012. Available from: https://www.infoq.com/presentations/Real-Time-Delivery-Twitter. [Accessed: 12 November 2018].