Dynamo and Cassandra

Overview of Dynamo and Cassandra

Cassandra was designed to solve the inbox search problem. The Inbox search is an attribute that allows users to search through their Facebook inbox. The Inbox Search problem was a challenge that has eluded researchers at Facebook since its inception. However, with the development of Cassandra, the storage requirements of the Inbox Search Problem was attained [1]. It is also worth mentioning that since the deployment of Cassandra in 2008, the Inbox Search has catered for the needs of well over 250 million users without any major issues. Several key attributes that made the Cassandra stand out among other storage systems include; the ability to scale incrementally, the use of replication to attain high durability and availability, and finally, the reliance of the local file system for data persistence. Overall, the success of Facebook is highly dependent on a storage system that is reliable, highly available, scalable and has high performance, all of which the Cassandra does magnificently.

Managing an e-commerce website like Amazon from a business standpoint requires that the site is always available. Using relational databases to manage Amazon services such as items in the shopping carts, providing best seller lists and product catalog would reduce availability and scale. However, the Dynamo meets the requirements of most of Amazon's services by using a simple primary-key access to a data store [2]. Dynamo provides the intended levels of availability and performance and it is exceptional in handling of data centre failures, network partitions and server failures. The advantage Dynamo has over other databases is that it allows service owners to modify their storage system to meet their intended durability, availability and performance.

Dynamo

- Used virtual nodes
    - If a node fails, then the data it was responsible for was evenly dispersed to other nodes
    - If a node was added to a system then it accepts a roughly equivalent amount of load from all other nodes
- Provides a simple primary-key only interface when searching for things like best seller lists, shopping carts, sales rank etc.
    - A typical relational database would limit scale and availability
- Dynamo is built for a trusted environment
- Dynamo doesn't require BigTable, GFS, Chubby. it's a stand-alone service
    - Why didn't Google make a stand-alone service?
        - Amazon is making a platform, Google is making a product.
        - Google's "2 Pizza rule": teams should not be bigger than what can be fed by 2 pizzas.
            - It follows that each project will be smaller which means teams can work autonomously from other teams but ultimately the entire product is highly coupled
    - Amazon takes the service-based architecture very seriously

        - When you're working on a service, it's exposed to the public
- Dynamo's problem: shopping carts and web sessions
    - Consistency is not a big deal, they care about availability
        - When you are shopping on Amazon, they want it to be available and when you add something to your cart they want to remember that
    - On Christmas the load on the system increases a lot

Cassandra

- Wanted to solve the "Facebook inbox search" problem.
    - Had to scale to very large inboxes
- Model based on some parts of GFS, BigTable and Dynamo.
    - They liked everything about BigTable except that it needed to have a distributed file system which they saw as a flaw
- Uses Zookeeper instead of PAXOS
- Replication system used a Gossip-based mechanism so every node knew the route to the other node it was looking for
- Optimized write throughput without sacrificing reads
    - Facebook inbox system handles billions of writes per day from millions
    - Most of the time we'll be writing to this system, and occasionally we'll be reading from it
- Cassandra's problem: "You have a bunch of messages. You should be able to search through it."
- This isn't a big-data problem compared to Google's "Searching a keyword across the entire internet".

Similarities

- Gossip protocol
    - Harder to get a stricter guarantees from a gossip protocol, but it's much less centralized
    - Uses a ring protocol where a node can talk to other nodes close to it (extends further than just immediate neighbours)
    - Different than PAXOS where a small number of nodes maintain state and you have a consistent view of that where everything else is viewable through a hierarchy

Anecdotes

- The past few systems have been quite similar, but the big difference is the problem they're aiming to solve.
- Feels like the systems we're reading about are trying to do all things (Jack of all trades) instead of focusing on one thing

    - Similar to Multics and we know a big reason Multics lost to UNIX is that it was packaged with a lot of bloatware where a typical user would never use all the provided features.
- Related work should be considered for these papers
- Consistent hashing is where we don't have to re-hash everything when changing the size of hash buckets

References

[1] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kaku lapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In ACM SIGOPS operating systems review, volume 41, pages 205–220. ACM, 2007.

[2] Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 44(2):35– 40, 2010.