CRITIQUE - 10 [Lecture - 12 Remote replication and Deduplication]

The main topic of today's lecture was remote replication and deduplication. We discussed the need for remote replication with an example and how it works. I learned different ways to replicate the data - synchronous and asynchronous replication. We also looked at some of the challenges with remote replication. In the second half of the lecture, the concept of deduplication was brushed up and dove deeper into how ZFS implements deduplication and the challenges in deduplication.

In the first half of the lecture, we briefly discussed remote replication. Remote replication is a process of copying the data to a device at the remote location. The need for a remote replica was clearly explained in detail. Replicas are used for disaster recovery, data distribution, backup and data migration. In the event of a disaster where an entire site is down, the recovery is done by making the replica node as primary and all client requests are now forwarded to the remote site. To increase the quality of service (QoS) and data availability the data may be distributed at each client site. Replicas serve as a backup of the data on the tape. Replication is also desired to migrate data from one location to another, to do load balancing and improve the quality of service. All the concepts are well presented with ample examples and illustrations.

I learned about the different mechanisms of replication - synchronous and asynchronous. Synchronous replication is a process of copying data over SAN, LAN or WAN so that there are multiple up-to-date copies of the data on remote devices. There is no data loss in case of a disaster recovery. From [2] I got to know that synchronous replication is desired in a high-end transactional application where instant failover is required. Synchronous replication is expensive and introduces latency in I/O. Asynchronous replication is a store and forward later approach to replication. This approached is designed to work well over long distances. In this technique, the data is written to the primary device first and then pushed to the remote devices later or asynchronously in the background. A full copy of the data is sent once and only delta changes are forwarded from there on. An asynchronous replica is cheap when compared to synchronous.

Although replication offers a number of advantages, there are some challenges with remote replication. For the replica to become a primary node, the replica has to be identical in every aspect with that of the primary node. Meaning it has to be identical in terms of hardware, software (firmware) etc. otherwise replica might not be compatible to serve the clients. Since these copies are at a remote location, there is going to be poor performance and high latency due to imperfect networks. In the case of a high

transactional application where there are large updates or dozens of simultaneous updates - keeping up the backup up-to-date can be challenging and can lead to scalability issues. In cascaded replication, if one of the nodes fail all the nodes behind it won't receive any updates and lead to inconsistent data or data loss. In a case of fan-out replication or multicast, it will overload the primary node since it has to send the same updates to multiple devices. The discussion on how to set up remote replication on the appliance was interesting and gave us a practical overview.

The second half of the lecture was about data deduplication. Deduplication is a form of data compression that eliminates the copies of identical data. Only the unique instances of the data blocks are retained in the storage device and redundant data is replaced with a reference to a unique data on disk. Deduplication can be done inline, post processing or during backup. It can be also done on file level, block level or byte level. In order to identify the duplicates, every block should hash to a unique id. If the generated id is already seen then it's a duplicate copy otherwise, it's unique copy. If two unique data blocks hash the same id, then there's going to be a data loss. So the hash function or computation of id has to be elegant without any collisions.

Later we discussed the deduplication in ZFS. I learned that ZFS implements block level deduplication, though it seems simple it poses some challenges such as scalability, performance, space accounting, and administration. Deduplication is managed at dataset level and all dedup enabled datasets will be deduped against each other to achieve high compression. ZFS dedup tables are kept in core for faster lookups, however, this limits the amount of dedupable data to the size of the core. Dedup tables are checksummed for data authentication and validation. Corruption of single dedup data block will cause all the referencing blocks to be corrupted as well. To safeguard against the corruption, the dedup blocks are replicated - called Ditto blocks. I learned the relationship between deduplication ratio and block size. Dedup over the wire and in-memory dedup concepts are discussed in detail and straightforward.

Overall the lecture was good and supplemented with examples. I understood the concept of remote replication, pros and cons and how ZFS deduplication works and how we can fine tune dedup for efficient storage and performance.

Reference:

1. Remote replication & deduplication - Lecture slides by Dr Mark Maybee.
2. http://searchdisasterrecovery.techtarget.com/definition/synchronous-replication