CRITIQUE - 6 [Lecture - 7 ZFS Cont.]

Summary:

This lecture is a continuation of the lecture 4 on ZFS by Chrisil. In this lecture, we discussed in depth about the ZFS. The concepts like the latest trends in storage integrity, End to end data integrity - Self-validating Merkle tree, data correction in ZFS, RAID-Z, ZFS resilvering, scrubbing, and ditto blocks. In the second half of the lecture we discussed the data management unit (DMU) of ZFS, DMU object hierarchy, transaction sequence and engine, live demo of the some of the concepts in ZFS like the creation of file systems in ZFS, DTrace etc.

The lecture started with brushing up some of the ZFS concepts covered in the last lecture like general introduction to ZFS and copy on write (COW). From the live demo, I learnt how to create multiple ZFS pools or file systems, how to enable different file system settings like mirroring, compression, limit storage and ZFS snapshots for individual file system that are carved out of ZPOOL. We discussed the trends in the storage integrity. The error rate of the disk has remained same over the years however the storage pool capacity have been increasing, which increases the chances of disk error rate. I learnt that noisy error is good when compared to silent errors. Disk errors can be detected and recovered using checksum and mirroring (any of the RAID techniques).

In traditional file systems, the checksum is stored in the same block as data which can cause unable to detect the error when a phantom read or write happens. ZFS avoids this problem by storing checksum and data in separate blocks. The blocks of the ZFS pool forms a “Merkle tree” which validates all of its children (data and data block pointer) - called “Self-Validating Merkle tree”. ZFS uses end to end checksum to detect and correct the data corruption. If mirroring is enabled in ZFS or RAID-Z then ZFS will both detect and correct the corrupted data on the other disk - called “Self-healing data”. Self healing data and copy on write techniques ensures that the data on ZFS pools are always consistent. I also got to know that the blocks pointers or the disk virtual address (DVA) are replicated as well - called “Ditto blocks”. This prevents the failure of millions or probably more data blocks due to a single node failure at the top of the root file system. The motivations for having Ditto blocks was not clearly explained and convincing.

In ZFS the error detection and correction is done through data scrubbing the disks. It uses the same algorithms as Resilvering. We discussed the ZFS performance aspects. ZFS random writes are sequential whereas the sequential reads are scattered. Thus sequential reads can be time-consuming. But it is not going to be a problem due to ZFS ARC caching and “prefetching” the data. The example given for intelligent prefetch was a bit hard to follow. The goal of any set of storage pools is to spread the IO uniformly across all available disks, in ZFS this is done dynamically on the fly - “Dynamic striping”.
Next, we talked about the motivation for the ZFS Intent log (ZIL). ZFS includes logging feature (journal) to recover from the system crashes or power failure. All the file system related events

are logged as a transaction record in the ZIL, these records contain the information to recover in case if there is a crash or a failure.

In the second half of the lecture, we talked about the important component of ZFS - Data management unit (DMU). DMU sits on top of SPA (Storage pool allocator) layer and uses SPA to get the blocks for writing objects. DMU also provides transaction engine for disk consistency. I learnt about the sequence of operations involved when a ZFS object (new file) is created. First ZFS creates a new dnode, inserts a entry in parent's ZAP (ZFS attribute processor), modify space maps and indirect blocks. I got to know about the DMU object hierarchy and DMU transaction sequence. The transaction in ZFS are not processed individually but rather they are aggregated and processed as a transaction group for performance. These transactions can be in any of one of these states - open, Quiescing and syncing.

I learnt about how to instrument a system for diagnosing the problems. Print statement is the most primitive instrumentation one can use. I wasn't aware of DTrace, a dynamic instrumentation framework created by sun microsystems for debugging and analyzing the errors in kernel and application in real time. Overall I learnt a lot about ZFS file systems, what it has to offer and new features introduced into ZFS to solve existing traditional problems. This lecture was very informative and useful. Live demo and code walk through provided me with some practical knowledge of to use ZFS.

References:

- Video lecture by Chrisil Aracaparambil.
- https://blogs.oracle.com/bill/entry/ditto_blocks_the_amazing_tape
- https://blogs.oracle.com/bonwick/entry/zfs_end_to_end_data
- https://blogs.oracle.com/realneel/entry/the_zfs_intent_log