CRITIQUE - 4 [Lecture - 5 File systems & ZFS]

Summary:
In this lecture, we discussed in brief about the fundamental concepts in file systems. Different operating subsystems like Process management, virtual memory, scheduler, IO. Some common operations that can be performed on files, directories and file systems. On a File system operations like mount, unmount, syncfs etc were discussed in depth. We looked at few special files created by the UNIX-like operating systems like /dev, /dev/null, IPC files, /proc files etc. We discussed about the Virtual file system and different operations on Vnode and VFS, followed by a short overview of the disk structure. In the later half of the lecture we dived into ZFS (Zettabyte File System), volume interfaces, Copy on write transactions and snapshots in ZFS.

The first half of the lecture was about the brief overview of the file systems. File systems is a data structure where the process keep their volatile data and also provides a way to organize data on the disk (files and directories). I was aware that in unix-like systems everything is represented as a file, but I never knew what it really meant. A live-demo of some of the special files created by the OS like “stack”, “PID”, “cpuInfo”, “/dev/null” etc and showing their contents was very informative and helpful. Every file system part of the kernel has to provide all the file system operations and manage the actual physical media per se. At the same time it also has to be hide the concrete file systems, provide efficient and reliable access to the storage media.

In order abstract the concrete file system to the applications, most file systems implements “Virtual file system” (VFS). VFS provides a standard set of well-defined interfaces that are file system independent. The implementation details are buried in the concrete FS types. I learnt what these VFS node and VFS object interfaces look like and how they get invoked seamlessly from the applications using “Virtual table” (vtable).

The example of a open() system call stack trace provided me with the full information of how an application call like open() gets transformed into a concrete file system open call. I really enjoyed how he kept switching between the presentation and live demo part. The discussion on VFS ended with a side note of how file system really follows object oriented programming (OOP) in C/C++ even though most of the people think of C as procedural oriented programming. Then we discussed about the geometry of the disk, ways to organize data(files) on the media like contiguous allocation of bytes, linked list, linked list via indexing - FAT (File allocation table) and linked via i-nodes.

The second half of the presentation was about ZFS (Zettabyte file system), the main theme of today's lecture. ZFS is a free and open source logical volume manager built by sun microsystems around 2000. I learnt some of the features of ZFS like endless scalability, eliminates the notion of traditional volume interfaces, supports transactional object system, provides end to end data integrity and simple administration. One of the biggest advantage of ZFS is that it's aware of the physical layout of the disk and the existing file system can automatically grow or shrink the storage dynamically when additional disks are added. If a storage is newly added to the storage ZFS makes it available to all of the available file systems that were created. I learnt that ZFS also supports creating a number of different file system each with same or different properties (compressed for space, raw for improved IO etc) instead of one giant monolithic file system.

Comparison of traditional file systems with ZFS was interesting. I got to know how ZFS addressed some of the problem that were existing in the earlier file systems like no defense against silent data corruption, size limitations (number files, directories, capacity of file system etc), painful RAID rebuilds and growing backup times.

I learnt about the different layers in the ZFS. Storage pool allocator (SPA) is the lowest layer that manages the physical drives. The layer above SPA is Data management unit (DMU) which provides for transaction group commit. ZFS POSIX layer (ZPL) residing on top of DMU interfaces with the applications and supports object-based transactions. ZFS ensures the disk consistency through a number of techniques like copy-on-write (COW). What this means is the block is never overwritten when data changes - It's always written to a new block, check-sumed before the parent block pointers are updated. ZFS supports snapshot capability of the configured file system. From http://www.oracle.com/technetwork/articles/servers-storage-dev/autosnapshots-397145.html#introduction I understood that these snapshots contain only deltas, which makes snapshots small, faster and efficient. Oracle also provides a graphical application called "Time slider" for navigating through the snapshots.

Although the first half of the lecture on overview of the file system was informative and interesting, I felt it was repetitive and spent too much time on it since it was already covered in the lecture 1 & 2. I wish he could have spent more time talking about the ZFS, which was supposed to be main theme of this lecture. In a nutshell I felt Chrisil did a great job with the presentation and live hands on demo with special file system.

References:

- Lecture slides on "File system overview" and "ZFS" by Chrisil Arackaprambil.
- The Z File System - Tom Rhodes, Allan Jude, Benedict Reuschling and Warren Block.
- http://www.oracle.com/technetwork/articles/servers-storage-dev/autosnapshots-397145.html#introduction