CRITIQUE - 5 [Lecture - 6 Caching]

Summary:

In this lecture, we discussed “caching” - one of the important topics in storage systems that increases the performance of read operation dramatically. We talked about the basics of caching, different types of caching - hardware and software, processor cache and different levels of caching in the processor. The bulk of this lecture was mainly composed of ZFS cache called Adaptive replacement cache (ARC). We were introduced to the theoretical and practical concepts of ARC. We briefly went over the ARC code, how a typical read or write operations happens in ARC, ARC buffer eviction, shrinking the ARC buffer and adapting the size per IO requests. In the last part of the lecture, L2ARC was discussed. An additional layer of caching (secondary cache) between ARC (primary cache) and disk. Overall the lecture was well organized and presented in a way easy to grasp.

First and foremost, the basics of caching were discussed. This refreshed my memory about the basic concepts of caching. Caching is a way of storing data in the secondary memory for high-speed retrieval. I learnt about the processor cache and the different levels of cache organization. Each core has its own high-speed L1 cache, next to it is the L2 cache slower than L1 but still faster than memory, L3 is a shared cache among all the cores in a CPU. Page cache accelerates multiple access to the files on the disk. When the file is first read or written from the disk, it is stored also in the unused area of memory, which acts as a cache. If this data is read again later, it can be quickly read from the cache in main memory. Page cache uses the technique of memory mapped file for improved IO performance on large files. However, on small files, a lot of space is wasted as pages are fixed size 4KiB or 8KiB. http://duartes.org/gustavo/blog/post/page-cache-the-affair-between-memory-and-files/ provides a lot of information about the page caches and memory mapped files in depth.

After a brief but concise introduction to basics of caching, we talked about the caching in ZFS. This was a new totally new topic to me and I learnt a bunch of new concepts. ZFS apart from offering a vast pool of storage for different file system it also includes ARC and L2ARC to boost the performance IO operations. ARC stands for “Adaptive Replacement Cache”, a super speed cache located in the server’s memory. ARC sits between transactional object layer and the storage pool thereby intercepting all the IO requests. If the data is cached it sends the ARC buffer reference to the application otherwise loads the data from the disk and then sends the reference. ARC is inspired by a paper that suggests adapting the size of MRU and MFU for better performance. ARC supports variable sized data block caching typically 512B - 1MB.

Next, we dove into how the ARC algorithm works. ARC composes of two types of cache - MRU cache and MFU cache. MRU caches the most recently requested blocks of the file system whereas most frequently requested blocks of the file system are cached in the MFU cache. The evicted pages from the MRU or MFU cache are stored on the disk to save space in the MRU or MFU. These evicted pages are still traced by the “Ghost MRU” and “Ghost MFU” on the disk. The ghost list continuously readjusts the size of MRU (p bytes) and MFU(c-p bytes) to adjust to the load of the system. https://pthree.org/2012/12/07/zfs-administration-part-iv-the-adjustable-replacement-cache/ gives a good explanation of the working of the ARU cache with visual diagrams. Though you did a great job explaining the working of ARU algorithm, It would have been easier to comprehend if visual diagrams were included.

In the second half of the lecture, we looked into the building blocks of the ARC and pseudo code representation of ARC data structures. The consumer can access the ARC data using refs. The ARC data blocks are represented as ARC buffers. We discussed how ARC states are represented, state eviction list, and look up data in ARC. During ARC read if the block to be read is found in the hash table then a new ref to the existing cached block is returned. Otherwise, the data is read into the cache first, ARC buffer is created and then reference to this buffer is returned.

The last section of the lecture was about L2ARC. I learnt that L2ARC is an additional caching sandwiched between ARC and the disk. It's slower than ARC but still much faster than retrieving the data from the disk. L2ARC are usually SSDs. when there's an ARC miss the IO request is intercepted by the L2ARC. If the data is cached in L2ARC then it's loaded into ARC other IO request is passed down to the memory controller to load the data from disk to the ARC. The advantage of using L2ARC cache other than the boost in performance is that L2ARC is non-volatile. So the data cached in L2ARC cache can be used to warm up the ARC on restart. To sum it up, I found this lecture interesting and informative. I learnt a lot of new concepts in Caching, ZFS cache, ARC, and L2ARC. Showing a peek of the data structures of ARC gave me an idea of how ARC is represented.

References:

1. Lecture slides on Caching by Mark Maybee
2. Brandon's blog: http://blogs.oracle.com/brendan/entry/test
3. https://pthree.org/2012/12/07/zfs-administration-part-iv-the-adjustable-replacement-cache/
4. http://duartes.org/gustavo/blog/post/page-cache-the-affair-between-memory-and-files/