CRITIQUE - 7 [Lecture - 8 Persistent memory.]

Summary:

In this lecture, we discussed persistent memory. We talked about memory and storage in general and the history of storage systems. How the memories are fabricated back when the computers were emerging, how it progressed to the current developments in the memory. The storage stack gave an overall view of how the memory is used / accessed by different applications. The programmer's view of how memory is accessed was also briefly touch based - buffer based and memory mapped files. Next topic was super exciting, we talked about NVDIMM-N - a persistent memory. “3D XPoint” is the NVDIMM-N from intel vendor. It’s not out on the market yet, but it’s 1000x faster, durable and 10x denser than a NAND flash. Later topic included types of persistent memory, programming models and examining some of the challenges faced during design the programming model for persistent memory.

The lecture started off by providing a bit of history of how the persistent memory was fabricated back in the late 80’s, and how it truly is a persistent memory (could recover data even from an explosion). It was fascinating to learn about the olden day's memory chips. To increase the performance of the memory we have to tune two parameters - decrease latency and increase bandwidth. The random fun facts and URL's thrown around the slides and in the lecture was really good and interesting. The explanation of the storage stack and programmers’ view of accessing the storage was helpful. I understood what goes on behind the scene when I perform a buffer based IO or a memory mapped IO. Though I understood buffer based and memory mapped IO concepts, I’m still unclear as to when should I use buffer based and when to use memory mapped IO while programming. All Operating systems perform paging which is transparent to all the applications. It lets the operating system swap out and swaps in the required parts of the memory from the disk - virtual memory. However enterprise applications like oracle DB etc don’t like the OS paging since they put these applications to sleep momentarily. Instead, these applications manage their own data on the disk.

In next section, we discussed NVDIMM-N, a persistent memory (pmem). NVDIMM stands for Non-Volatile Dual In-line Memory Module. It's like a DRAM but retains data even when the power is removed either from an unexpected power loss, system crash or from a normal system shutdown. The memory timeline gave an idea of all the developments that happened in the chronological order. The bulk of the storage latency is due to the accessing existing media devices (NVM tread), however, 3D XPoint reduces this latency drastically due to in-place persistent memory. With pmem programmers can access the memory is byte-addressable, can perform load / store operation without the need for demand-paging. What blew my mind was these NVDIMM-N devices are nanosecond devices and we can stall the CPU to read or write to the memory. I found on the internet and also on slides - Transparent persistent memory. I would wish to know more about this in the upcoming lectures. I learnt about different types of persistent memory - battery-backed DRAM, DRAM saved on power failure, NVM with caching, and next generation NVM’s like magnetic tunnel junction, electrochemical cells etc. when the

system resets DRAM can get the cache contents from NAND flash - “warm cache”. No time spent in loading up memory.

In the next section, we briefly went over the programming models for the persistent memory. SNIA technical working group defined 4 programming models required by developers. Interfaces that are pmem aware file systems accessing kernel pmem support, interfaces for an application accessing a PM-aware file system, Kernel support for block NVM extensions and Interfaces for legacy applications to access block NVM extensions. We also talked about the use cases for NVM.PM.Volume and NVM.PM.File system. I learnt that the NVDIMM flushes the data on power failure or only when an application's call sync explicitly. There are different ways of flushing the data from caches to the pmem - synchronous flushing and pipelined flushing.

We briefly discussed some of the challenges faced during designing the pmem programming model. Some of the problems like what happens when the store operation is more than 8 bytes, the “C-clamp” problem - since the NVM doesn’t have a file system layer, how to implement RAID and positional independence. Later, we saw where in the software stack is pmem leveraged. I got to know that higher up the s/w stack we go it add more leverage however it also increases the barrier to adopting pmem. Once these challenges and programming models are defined software vendors had a different set of concerns like they need a persistent memory allocator, support for transactions, replication and support developers favorite programming language. This resulted in the development of a suite of libraries for supporting pmem programming models.

In a nutshell, I found this lecture to be very interesting and lively at the same time. The organization of the contents and presentation was excellent and very informative. Andy gave good information about persistent memory, ample examples, and code snippets. I really liked all the URL and fun facts that were present in the slides. Thank you Andy Rudoff for sharing the information and personal experience working on persistent memory storage.

References:

- Persistent memory lecture slides by Andy Rudoff
- https://en.wikipedia.org/wiki/NVDIMM
- https://en.wikipedia.org/wiki/3D_XPoint