CRITIQUE - 2 [Lecture -3]

Summary: In this lecture we discussed about the storage area network(SAN), Network attached storage systems(NAS), and some of important storage block protocols like SCSI, iSCSI, iSER and FC. we also discussed about how we can streamline some of these protocols to optimize for the performance. In the first part we discussed about the SAN, NAS and storage virtualization how it abstracts the logical storage from the actual physical storage. Some of the modern SAN applications that can take advantage of block protocols like databases(oracle, IBM DB2), microsoft exchange, hypervisor(desktop/server virtualization like VMware etc). The later discussions included overview of SCSI, some relevant standards for the disk storage, SCSI command flows for read/write operations and COMSTAR architecture. At last we briefly discussed some of the implementation challenges like STMF threading models.

First and foremost I learnt about the differences between Network attached storage(NAS) and Storage Area Network(SAN). NAS is used for file storage/file sharing and runs a striped down version of OS. It's runs only file storage protocols like NFS, SMB etc. Whereas SAN is used for block storage and runs block storage protocols like SCSI etc. SAN gets mounted as a physical drive unlike shared network drive(in NAS).

Block storage protocols was something that I was not aware of. Block protocols store or retrieve data based on blocks(512 Bytes) as opposed to file protocols. I understood that the SCSI protocol provides a provision to read/write blocks but who tracks these blocks, who reassembles these blocks to construct the object ? Say some application wants to read a file spread across multiple blocks. Is it the responsibility of the application to track the blocks, request read from SCSI and re-assemble the blocks to get the file? If so then the application on some level is also doing the file system management which clearly it's not supposed to (separation of concern). I learnt about SCSI(small computer system interface) protocol itself. It consists of a "Initiator" and "Target". The target is a pool of storage media which can be broken down into small addressable pieces called "logical unit number"(LUN). The standards for disk storage are SCSI architecture model, SCSI primary commands and SCSI block commands. I found this useful link http://www.globalknowledge.net/mea-shared-content/documents/645372/645377/WP_CI_FundamentalsofSCSI1.pdf on internet with gave a brief overview of the SCSi protocol and roles. I got to know that the initiator can be implemented either using software or hardware. The software initiator uses the code in OS(kernel resident device driver) whereas the hardware initiator uses dedicated hardware called HBA(host bus adaptor) which runs a firmware that implements SCSI.

I learnt about the hand-shake process between initiator and target. Which is a 3-phase process - Command phase, Data phase and Status phase. The command phase contains the request for transfer(read/write) and life-time of the request. The actual data is transferred during the data phase. And the last phase just indicates the status of the transfer - success or failure. I liked the way you would explained the SCSI read or write operation and let us figure out how we can optimize the protocols for the performance, reduced latency and seek time. Performance boost

can be achieved by bi-directional data transfer, piggybacking the status phase to last data transfer - called “Status collapse”.

I learnt about the different networking protocols like SCSI over FC, iSCSI and iSER. iSCSI runs on top of the TCP/IP protocols and transports SCSI commands over ethernet (NIC). This reduces the total cost of deployment when compared to SCSI over FC. In SCSI over FC most of the transfer is done by HBA. iSER is an extension of iSCSI with RDMA protocol. This offloads the processing from the host. iSCSI command and responses are send through “RDMA send” which bypasses TCP/IP stack.

In second half of the lecture, I learnt about the common SCSI target(COMSTAR) architecture. This framework basically allows any server to become a block-based storage device. This framework allows all SCSI devices like tape, SES etc to be connected to any transport protocols like FC, FCoE, iSER, iSCSI etc with concurrent access to all of the LUNs and single point of management. The block based array built using COMSTAR has 4 pieces namely Port provider, SCSI target mode framework(STMF), LU provider. Port provider provides the connection points(like HBA, NIC etc) for all transports like FC, FCoE, SAS etc. LU provider implements the functionality of SCSI logical unit like tape, disk, ZFS etc. STMF layer is simple and implements thread scaling and “LUN masking”. I also found some useful information about COMSTAR in oracle page http://www.oracle.com/technetwork/articles/javase/index-135035.html. I got to know that it’s the world’s first open-source target framework. I noticed in the slide-1 COMSTAR stands for COMmon SCSI target however on the oracle web page it’s defined as Common multiprotocol SCSI target.

In the last part we went over some of the technical challenges faced during the implementation of COMSTAR. We discussed STMF threading model designs i.e creating as many threads as the number of disks, creating one non-blocking I/O thread per CPU core. I liked discussing the merits and demerits of push vs pull task threading models. In the original STMF threading model they would create as many threads as the number of disks in the pool though there is only limited CPU cores. To overcome the problem of more threads than the cores they created a new model that creates one non-blocking I/O thread per CPU core. In this model the threads pull the works from the queue and execute rather than the task being pushed to the threads in the previous implementation.

Resources:

http://www.globalknowledge.net/mea-shared-content/documents/645372/645377/WP_CI_FundamentalsofSCSI1.pdf

http://www.oracle.com/technetwork/articles/javase/index-135035.html