Summary: FaSST: Fast, Scalable and Simple Distributed Transactions with Two-sided (RDMA) Datagram RPCs

COL862 : Advanced Topics in Operating Systems, Sem I, 2017-18

September 19, 2017

Keywords

Networking, RDMA, RPC, Data Center, Transaction Processing System, One sided RDMA, Two sided RDMA

Brief Summary

One-sided RDMA *verbs* are popular because it bypasses the remote CPU and directly access main memory of the remote machine to fetch the data. This reduces the latency of the whole operation. Because of this, till now the main focus of development was on operations using one-sided RDMA verbs. However, in certain systems like a Transaction system or when traversing a pointer based data structure, it requires multiple round trips to finish the operation. This leads to low throughput, high latency and reduces the net CPU savings from remote CPU bypass.

Authors implements FaSST RPCs using two-sided RDMA vers(where the remote cpu is involved) over unreliable datagrams , which provides much more flexibility in terms of capability of the remote machine. Authors build a distributed transaction processing system using these FaSST RPCs, which is simple, fast and scalable when compared with RPCs implemented using one-sided RDMA verbs (FaRM , DrTM).

Related Work

Recent works, where RPC is implemented using one-sided verbs tries to reduce the number of READ operations that needs to be carried out with some optimizations.

- **FaRM:** Here the main idea is to store the data with the index, which allows data to be READ with the index. However, this amplifies the size of read by a factor of 6x-8x.
- **DrTM:** It caches the index of its Hash table in all of the machine, eliminating the need to READ an index from different machine. However, this is not practical as the size of the indexes can be large.

Network Primitves

Connection Oriented

- NIC need to use connected transport connection for one-sided verbs such as READ and WRITE.
  - As READ and WRITE are higher level abstractions and hence the NIC needs to maintain state during its operations which requires a connection oriented transport.

Scalability:
In large cluster, using one sided verbs over connected transport, number of QPs required is equal to the number of possible communicating entities. For example if there are N machine with t threads each and every thread wants to communicate with every other threads, then total QP required is $N * t$.

Because of this, it does not scale very much because, as the number of machine increases, the number of QP (for concurrent communication among different machines/threads) also increases. Eventually the NIC runs out of memory and it starts thrashing which affects the throughput.

QP Sharing:
There are some methods to control the number of QPs, such as **QP sharing**. This limits the number of QPs in the system, but threads contend over the QP which reduces the performance.

Connectionless

- Two-sided verbs such as SEND and RECV are relative low level abstractions and do not require any states to be maintained and hence can be made to work with connectionless, datagram approach.

Scalability:
Using the two-sided datagram approach limits the number of QP to the same as the number of cores in the machine (each core has exclusive access to a QP) as a single QP can be used to communicate with more than one entity.

FaSST RPC

RPC implemented using two-sided datagram verbs over un reliable channel are *simple, fast and scalable* when compared to RPC implemented using one sided verbs (FaRM, DrTM). Although a single READ is faster in case of one-sided RDMA verbs, RPCs outperforms them for transaction like queries where the one-sided verbs needs to issue multiple requests.

FaSST RPC uses some optimizations to make them more efficient.

- Coroutines are used to hide the network latency of RDMA networks which is around $20\mu s$. It is claimed that 20 co routines will be enough to hide the latency. Coroutines are used instead of threads because of low cost of scheduling and concurrency.

- RPC Interface and Optimizations
  Some of the advantages of using a Unreliable data gram transport are:
  - **Request Batching**: One of the optimizations to send multiple request is to batch them which reduces the number of doorbell to NIC to 1. It also allows message to the same machine to coalesced together.
  - **Response Batching**: The server also tries to batch the response. However, it does not wait for a particular number of message before sending them as this can increase the latency for some of the operations.
  - **Cheap RECVs**: CPU needs to get a descriptor from the CPU each time it gets a packet over the network. This is costly. To minimize this, the NIC asks for a set of descriptors during the initialization and then reuses them over the course of its operation.

FaSST Transaction

Transactions

FaSST Transactions involves

- **Read and Lock:** Read the header and value of keys from their primaries. Also request the primaries to lock their header. FaSST RPC allows both operation to be done in a single round trip. One-sided verbs needs two round trips.
- **Validate:** It might be possible that some other operation changed the value of the first primary during the lock operation was going on as the primary 1 is not locked. After the locking it validates the read set again to check if any of them has changes or is locked. In this case transaction is aborted. If no such case then it proceeds.

  It is possible that the value of primary 1 may change after the validate phase, which does not affect this transaction as the operations can be serialized with this transactions and then can be argued upon.
- **Log:** After the validation it commits the log record. The logs are committed at f+1 place to survive f failures. Log contains key-value items and their fetched versions.
- **Commit backup:**
  If logging succeeds it sends update RPCs to backup of W. Then it waits from ACK from each of them before it updates the primaries.
- **Commit Primary:**
  After receiving ACK from each of the back up coordinator sends update RPCs to primaries. Primary updates the key's value, increment its version and then release the lock.

Packet Misses

Using an unreliable mechanism generally includes providing a mechanism for fault tolerance. Authors claim that as the RDMA network is highly reliable, packet loss will be rare. However, they capture a packet loss by using a timeout of 1 second at requester side and treat packet loss as hardware failure by killing the FaSST process on that machine (receiver).

Future Trend

- Increase the NIC cache so that they can support more QPs.
  This will not work as the number of core in the CPU is also increasing. It takes a small cluster to saturate the cache of the current NIC and hence it is not feasible to use this technique for moderately large or large clusters.
- Dynamically Connected Transport(DCT)
  This provides software the illusion of using one QP to communicate with multiple remote machines, but at a prohibitively large performance cost for our workloads: DCT requires additional network messages.
- Portals: Scalable one-sided RDMA using a connection less design.

Test You Understanding

1. In which situations or for what kind of operations the two-sided RPC will outperform the one-sided RPC operations?
2. What is read size amplification and how does the approach by FaRM is affected from this.
3. The latency of RDMA network is around $10\mu s$, much higher than the time required by applications to finish the local computations. How this is handled in the *FaSST* ?
4. Why Request/Response batching is not possible in one-sided RDMA verbs such as READ and WRITE.
5. Why is there a need to validate the read of Primary 1 after locking the Primary 2? Why is it sufficient to validate only once?