Presenters:
Ankit
Shailja

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

COL862 : Advanced Topics in Operating Systems
Sem I, 2017-18
Date :2017-09-14

1 Summary:

One-way RDMA READs/WRITEs are costly and doesn't perform well when used for distributed transaction system. There are many reasons behind this, for example:

- One way read can't dereference multi-level indirections in one network round trip and need more number of round trips. There are many way to resolve this: Store full index on each node or flatten key-value store. These solution can't resolve the problem without other implications.
- One-way RDMA operations need connection oriented binding and there is a limit of maximum number of QPs that can be created on NICs. If we create more number of QP then there is a high chance that performance will degrade due to NIC cache miss. QP sharing among cores is also not a good idea because of locking overheads.

These problems can be solved if we use RPC over RDMA instead of one-sided RDMA operations. Remote CPU is involved in case of RPC, however the performance for transaction systems is more when used RPCs as compared to one-way RDMA READs/WRITEs

2 Discussion on Section 1

Q: What do you mean by a distributed transaction system?
A transaction is a small part of code between *BEGIN* and *END*. Most of OLTP systems supports small transactions and guarantees serializability, durability and high availability. Due to the large size of database, the database is partitioned among machines in a cluster. There are many protocols to achieve above mentioned properties for a distributed transaction system.

Q: What is special about transaction systems and why RPCs are better than one-sided RDMA operations for such systems?
The abstraction which are needed to implement an efficient transactional system are higher level abstractions which can't be offloaded to NICs in an optimal way. For example: index dereferencing, locking, logging, replication. One sided RDMA READs/WRITEs support reading and writing on remote memory and not the higher abstractions.

3 Discussion on Section 2

Q: Why should READs be connection oriented and SENDs/RECVs can be connectionless?
Read is a higher level abstraction and for that we need to maintain state at the hardware level. To maintain state at the hardware level the NIC need to use connection oriented transport.

Q: Why QP sharing is a problem?
If QPs are shared then CPU and QPs work in a producer- consumer setting. As we are aware of the problems in producer-consumer setting, like locking and concurrency control. This may lead to a performance drop to a large extent.

Q: What is NIC cache?
NICs have memory for storing buffer and descriptors. NICs have high speed cache memory to store descriptors and slow memory for other storage. Non-efficient use of NIC cache memory might degrade the performance.

Q: Explain zero-copy networking.
For high speed networking, hardware provides a part of memory which can be accessed by NIC directly. When NIC receives the packet it puts the packet directly in the host memory, instead of copying it in kernel space and then user space. The protection of this memory mapping is maintained at the hardware level.

4 Discussion on Section 3

Q: Difference between OLTP and OLAP.
OLTP(Online Transaction Processing) systems are class of programs which perform transaction on a large set of data. These transaction are performed online, means whenever the operations are performed. OLAP(Offline Analytical Processing) system performs analytics over the data in an offline mode. With the help of the analytical techniques OLAP tell about the trends and other interesting insights about the data and database users.

Q: What is MMIO and why are MMIO expensive and why do we need memory barriers for networking primitives?
Memory-mapped I/O is a method of performing I/O between CPU and I/O devices which uses the same physical address space to address both memory and I/O devices. Memory specific instructions such as READ and WRITE are used by the CPU to communicate with the I/O devices instead of IN/OUT instructions. MMIO are expensive because these operations take place via PCIe bus which is a high bandwidth-high latency medium. We need memory barriers for networking primtiives to ensure the ordering in the operations.

Q: What is symmetric model for transactional systems?
As we know that, in a distributed transaction system data is stored in cluster of machines. In symmetric model, all the machine work both as a client and server. User can query any machine in the system and if data is not present on that system and this machine will act as a client and will fetch the data from a remote machine.

Q: Interrupt vs Polling for network IO.
Both the techniques have their own advantages, if the machines are specially assigned for a particular task. In the paper, the machines are part of a distributed transaction system and the goal is to increase the throughput. In such cases, polling is a good idea to reduce the overhead associated by interrupt generation and handing.

Q: Why is partitioning of data-store and indexes needed?
The amount of data current transaction system needs to support is huge, which can't not be stored on one machine. The data is partition(can be done in many ways ex: based on the primary key) and stored on different machines in a cluster.

Q: Why do requests per second decrease with increase in request/response size in Figure 1?
In the graph, the comparison is between size of the read and number of requests served in one second. If we increase the size of a request, the number of request which will go through the network will decrease as the network bandwidth is limited. So, with increase in the read size, the request served per second is decreasing and not the bits per second.

Q: Related to figure 1: Why are 11 machines used for RPCs and 6 machine for one-way READs to show to throughput comparison between the two?
They wanted to compare the peak READ throughput and to get the peak READs throughput it is a necessity to keep the number of QPs small so that the NIC cache can handle those many number of connections. For RPCs they have used 11 number of machines, as they wanted to show the effect of doorbell batching and which can be shown with a reasonably large number of machines in the cluster.

Q: Compare RPC over Ethernet and RPC over RDMA.
It is not discussed in the paper and would be an interesting question to answer.

5 Discussion on Section 4

Q: Why are coroutines used instead of threads?
The switching overhead is more in case of threads and additionally lock and concurrency control is needed to use the threads in an efficient manner. On the other hand, coroutines switching is fast and the program can be written in such as way that we don't need to use additional mechanism for concurrency control.

Q: Why does optimal throughput come with 20 coroutines per thread?
The network RTT is around 10us for the setup used in the paper. Switching overhead and request creation time would be around .5us and that is the reason to use 20 coroutines to hide the network latency.

Q: Why is source-destination thread mapping used?
As mentioned in the paper, the NIC has limited cache to support RECV queue. If the RECV queue size is mode then the performance might decrease due to cache thrashing. To keep the queue size small, they have put the restriction that one peer thread can send requests to other peer thread and not all the threads on the remote machine.

Q: Explain Cheap RECVs posting?
In generic network setting NIC needs to create a new descriptor for each packet receive, which is an overhead for CPU and memory subsystem. Instead of that, CPU can create a many descriptor at once and can reuse those descriptors in a circular fashion for each packet receive.

Q In Figure 4, what is the bottleneck and what hypothesis supports it?
All the optimizations are used to reduce the CPU overheads. Significant increase in the throughput shows that particular optimizations(request/response batching) favour the CPU more and less increase in throughput shows that the particular optimization(cheap recv) is not very useful.

6 Discussion on Section 5

Q: Why is validate phase needed in two phase commit used in the paper?
In the paper, they have not used locking for read operations and some other transaction might change the version for that data. To avoid that they are validating the reads before committing the transaction.

Q: Why don't we need to validate the reads after validate phase, when version can change after validate phase too?
It is not needed, since all the transactions are serialized in reference to the current transaction.

7 Discussion on Section 6

Q: In Figure 10, median and 99th percentile latency are being used and not average latency, why?
In the network and distributed systems, vendors put SLAs on time limit and graphs shows the median and some upper percentile to show how many transactions are with in SLAs and how many transactions have violated the SLAs.

8 Discussion on Section 7

Q: What is a good criteria of CPU onload and device offload in systems? What is the trade off?(in reference to the paper, too broad of a question in general setting)
It depends on the type of abstraction and how the abstractions are used for the system in question. For example, one-sided RDMA read which is a higher level abstraction might be suited for bulk transfer but not for OLTP systems. On the other hand, TCP is also a high level abstraction but its functionality can be offloaded to NIC in an efficient manner.
Trade-off: With one-sided RDMA, remote CPU is being bypassed which is not suited to OLTP workloads. So trade-off is between saving the CPU cycles and throughput(due to increased round trips).