Transaction Processing over High-speed Networks

Proposal by:
Erfan Zamanian
erfanz@cs.brown.edu

Research Committee:
Tim Kraska
Maurice Herlihy
Stan Zdonik
{tim_kraska, mph, sbz}@cs.brown.edu

Brown University
April 2015

Abstract

By avoiding high cost of disk I/O, memory-resident OLTP databases reduce the runtime of typical single-sited transactions to a fraction of a millisecond. With disk gone from the picture, network has become the next bottleneck. In fact, the traditional wisdom is that network is the new disk, and distributed transactions must be avoided as much as possible, through techniques such as partitioning. However, recent technological trends toward low-latency, high-bandwidth networks which are capable of minimizing communication overhead through Remote Direct Memory Access (RDMA) are changing our view on the fundamental assumption that network is significantly the bottleneck. For example, the latency in InfiniBand FDR/EDR is less than $1\mu s$, two orders of magnitude faster than the traditional 1Gb Ethernet, and is only one order of magnitude slower than local memory access ($0.1\mu s$).

In this project, we aim to explore the design of a memory-resident OLTP with optimistic concurrency control, in particular snapshot isolation, which leverages RDMA and low latency of fast networks as much as possible. We argue that the new RDMA-enabled architecture is neither shared-memory nor shared-nothing. This new architecture requires us to fundamentally rethink the design of databases. In particular, we plan to address the question that in an RDMA-capable distributed system, what changes have to be made to concurrency control and related components of these databases. We speculate that benefits of using RDMA in OLTP systems are three-fold; first, it decreases the latency of transactions. Second, in this new hybrid architecture, the burden of transaction processing can be potentially shared between servers and clients, resulting in a more balanced design. Finally, separating transaction processing from data management, which are both traditionally handled by the server, could facilitate the scale-out process. Our initial experiments show that exploiting RDMA could result in at least an order of magnitude higher performance compared to the existing design.

I. Introduction

The notion of transaction is one the most fundamental concepts in database management systems. Online Transaction Processing (OLTP) databases rely on concurrency control and other mechanisms to create the illusion for each transaction that it is alone in the system, while actually there might be concurrent transactions being run simultaneously. Besides, transactions are guaranteed to be deterministic, meaning that their effects persist over time, and they are executed in their entirety, transforming the database from one consistent state to another. In a distributed database, transactions may span multiple machines. In order to allow concurrent execution, coordination between the involving nodes over the network is required. For example, in a money transfer between user A and B, whose records are hosted at different machines, whatever amount is deducted from A's account at site S1 must be transferred to B's account at site S2. S1 and S2 must both agree on the verdict of this transaction. Such agreement involves coordination between the two nodes over the network.

Modern distributed main-memory DBMSs are built on the assumption that network communication is slow and network bandwidth is severely limited. Table 1 shows a comparison between different components of a computer system. By getting rid of disk as the main storage, memory-resident OLTP systems have reduced the run-time of transactions, which are typically short-running and do not need complex computation, to less than $100\mu s$ [4, 5], if no network is involved. Distributed transactions, on the other hand, often require multiple network round-trips. For example, the standard version of two phase commit requires 4 end-to-end communications between the initiator and participants, which can take up to $400\mu s$, 4 times the actual work of a local transaction. This added overhead not only increases the transaction latencies, but also increases the chance of contention over locks. As the contention rate rises, it becomes more likely for transactions to abort.

Therefore, it is easy to see why in-memory OLTP systems avoid distributed transactions as much as possible. They achieve this by using complex partitioning schemes to co-locate data to make sure that transactions can access all the data they need in a single site, such that no coordination between sites is needed. While this is a solution, it imposes a new set of challenges for the developer. Besides, some workloads, such as social-graph data, are inherently not perfectly partitionable.

The assumption that network is by far the slowest component of distributed systems and must be avoided at all cost, has started to become false. High-performance Remote Direct Memory Access (RDMA) capable networks, such as InfiniBand FDR/EDR, which were previously only used in High Performance Computing (HPC) environments due to their high cost, have started to gain economic viability and find their ways to datacenters. Table 1 shows the latency and bandwidth of two generations of InfiniBand networks. As can be seen, their latency is only one order of magnitude higher than local main memory random access, as opposed to three orders of magnitude in traditional TCP/IP Ethernet-based message passing. In addition to its low latency, RDMA completely bypasses the traditional network stack and the kernel of sender and receiver, allowing a machine to transfer data to and from another machine's memory with little or no involvement of the CPU on the remote side.

Incorporating RDMA, however, requires a re-design in many components of OLTP systems. This can be best explained by contrasting the existing distributed OLTP architectures, particularly the prevalent design, namely shared-nothing, with the new architecture. Figure 1 depicts a simple representation of these architectures.

In a shared-nothing architecture, server's CPU is the single point of coordination in handling client's requests. It has exclusive access to its own data residing in the main memory. It receives the request, processes it, and returns the result to the client.

An RDMA-enabled architecture, on the other hand, is neither a pure shared-nothing design, where message-passing is the only means of communication, nor a pure shared-memory architecture due to three main reasons. First, the memory access patterns are quite different in RDMA and NUMA architecture. Second, the latency between machines is significantly higher to access a random byte than with today's NUMA systems. Third, in a NUMA architecture, hardware-embedded coherence protocols ensure data consistency, which is not supported with RDMA, since data is always copied.

	End-to-end Latency (μs)	Throughput (GB/s)
Disk (SSD)	10,000	0.3
Main Memory (DDR3)	0.1	12.8
1 Gb Ethernet	100	0.125
InfiniBand 4xFDR	0.7	6.8
InfiniBand 4xEDR	0.5	12.1

Table 1: *Comparison of data transfer in different computer components*

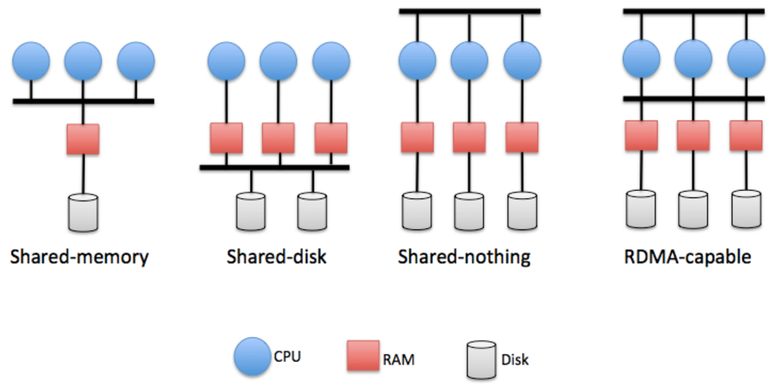

Figure 1: *Different Architectures of databases*

This hybrid architecture, where communication is possible through both message passing and direct memory access, requires a complete re-design of OLTP systems.

II. Background

Among the various RDMA-enabled interconnects, InfiniBand has gained significant acceptance in the HPC community due to its ultra low latency and high bandwidth. While it used to be very expensive in the past, it has recently become cost-competitive with Ethernet, and therefore could be seen as a viable alternative as interconnect in datacenters. InfiniBand implements many layers of the network stack, from transport to physical layer, directly in NICs. In this section, we will give a brief overview of the RDMA technology.

I. RDMA

The application sends or receives request directly to and from its RDMA NIC (RNIC) using the RDMA API, which is based on *verbs*. Invoking this process does not involve any system call and context switch from the user space to the kernel space. As most operating systems impose high overheads for interrupt processing, network protocol stacks, and context switching, avoiding them significantly helps decrease the latency.

There are two communication models: one-sided communication and two-sided communication.

One-sided, or **memory semantic** verbs, are those verbs which are performed without any knowledge of the remote side. RDMA READ, WRITE, and atomic operations, such as Compare and Swap (CS), and Fetch and Add (FA) are one-sided operations. The active side submits the verb, while the passive side is completely unaware of the process. Both active and passive sides must register the memory region to be able to access it via RDMA. The passive side's RNIC directly writes/fetches the desired data using an DMA operation from local memory.

Two-sided, or **channel semantic** verbs, such as SEND and RECEIVE, require both sites to involve in the communication process. The payload of the SEND is written to the memory region specified by a corresponding RECEIVE which must be posted by the receiver before the sender actually sends its request. Therefore, the sender's RNIC does not have to register the remote memory region before performing the operation.

While two-sided verbs are workflow-wise similar to socket programming semantics, namely read() and write(), leveraging the one-sided verbs, such as READ and WRITE, requires a dramatic change in the programming model.

RDMA uses the concept of queue pairs for connections. Application posts verbs to the send queue, which has a corresponding receive queue on the receive side, so the name queue pair. Once RNIC performs the requested memory access at the remote side, it pushes a completion event on a corresponding completion queue, which can notify the sender about the completion of the task. All queues are maintained inside RNIC.

III. Proposal

We argue that there is a need to fundamentally rethink the entire OLTP databases, including data access patterns, replication, log management, etc. to take full advantage of the next generation of network technology.

In this work, we will explore the design of an RDMA-aware OLTP database for the new network, where the transaction logic is pushed as much as possible to clients. Our focus will be primarily on optimistic concurrency control, in particular snapshot isolation. In the most extreme case, where all requests are handled directly by clients, and servers are not involved in transaction processing. As such, servers are only responsible for managing their own data and keeping it "clean", such as garbage collection.

In particular, throughout the course of this project, we would like to address the following questions:

- Given that some workloads are inherently hard or impossible to partition, is it still the case that distributed transaction must be avoided at all cost.
- In what aspects does the design of a pure RDMA OLTP system with snapshot isolation differ from a traditional requesting client/responding server architecture?
- Now that clients can access server's data directly in the new design, how does the server's memory should be organized for efficient access.
- Clients must know the exact location of records and their (possibly) multiple versions in order to access it via RDMA. How should the catalog management be done in this environment? In particular, making the clients aware of new data arrival (e.g. SQL UPDATE and INSERT) and deletion (e.g. SQL DELETE) is of especial interest.
- RDMA operations lack richness. Anything which is more complicated than a simple memory lookup or memory write has to pay for multiple network roundtrips, and could easily offset all the advantages of using RDMA. The research question here is how to use RDMA efficiently to minimize the number of netwrok roundtrips.
- Without any single coordinator for memory accesses, write-write and read-write races can occur. How to avoid them?
- The search space within RDMA itself is rather large. For instance, there are a handful of verb semantics (including channel semantics and memory semantics), transport types (including reliable, unreliable and unconnected) and other important parameters (e.g. signalling, request and response sizes, etc.) that make the design of an RDMA-aware system non-trivial. For OLTP systems, we are

interested to explore the optimal combination of these factors that results in high performance.

The main focus of this project is to explore the design of a pure RDMA-enabled OLTP system. However, we believe that the highest performance is achieved neither in a pure messaging system nor in a pure RDMA one. The optimal design relies somewhere between these two ends of the spectrum, where clients and servers divide the transaction processing logic between themselves. Finding such a design is left as the stretch goal.

IV. Preliminary Results

V. Project Schedule

In the remaining 10 months of the project, we will first complete our literature review, both on related transaction processing papers and RDMA related papers. We plan to divide the remaining time between system design, and system implementation.

In the first half, we are first going to design and run some micro benchmarks on RDMA verbs in InfiniBand. The goal here is to fully understand how various RDMA choices compare to each other, recognize their characteristics, and find out what communication scheme is more suitable for different parts of our system. We will then continue with the actual system design. Here, we aim to address all the questions mentioned in Section III. The second half will be devoted to implementation of our proposed system. Table 2 shows the time table for the project schedule.

VI. Related Work

This work combines the ideas of the well-studied topic of distributed transaction processing with the emerging technology of RDMA-capable fast networks.

I. Distributed Transaction Processing

[4] performed a detailed study of performance overhead of different components in a distributed OLTP database. After breaking down the time that the server's CPU spends on various components of transaction processing, they concluded that buffer management, logging and locking significantly increase the overhead and should be reduced/removed as much as possible. In a related work, [13] argues how the need for exploiting recent technological advances in hardware (main memory in their case) could require a fundamental re-design of the entire system.

II. RDMA-based Data Management Systems

Although RDMA has attracted a lot of attention in the Systems community (e.g. for building distributed lock managers [1, 10], replicated state machines [11], RDMA-based of existing systems such as Hadoop [8]), it is a relatively unexplored topic in the area of databases. FaRM [2] is a general-purpose main memory distributed computing platform which exploits RDMA to improve latency. Similar to our work, it provides ACID guarantees for local and distributed operations. However, it is not an OLTP database and its API is quite limited. FaRM achieves high throughput by using one-sided RDMA WRITE to implement a fast message passing primitive, and RDMA READ for read-only operations. Since access local memory is still much faster than RDMA, FaRM performs locality-aware optimizations.

There has been a number of works that leverage RDMA to build key-value stores [9, 6, 7]. In Pilaf [9], while get() are handled by having clients directly access the server's memory using RDMA READ, put() requests are sent to and serviced by the server via RDMA WRITE. Since clients are not involved in put(), the synchronization process between concurrent memory accesses is much more simplified. In this design, put() is performed in one roundtrip (one RDMA WRITE for sending the request and one for the response), while get()

Task	Time Required (month)	Target Completion Date
Literature study	1	6-1-2015
RDMA micro benchmarks	1	7-1-2015
System design	3	10-1-2015
Implementation	3	1-1-2016
Experiments	1	2-1-2016
Finishing up report	1	3-1-2016

Table 2: *Project schedule*

requires at least two RDMA READ (one for looking up a key in the hash table array to find the memory address of the key-value pair on the server, and one to fetch the actual value associated with the key). As a critique on the aforementioned design, [6] argues that a system which requires multiple roundtrip of one-sided RDMA operations (as in the case of Pilaf, RDMA READ) could be outperformed by a design where RDMA verbs are used merely as a message-passing mechanism, and not memory access. Therefore, in their system, HERD, the client writes its get() or put() request into the server's memory, and the server computes the reply. They showed that the maximum throughput can be achieved by bypassing network stack and CPU interrupts, and not bypassing the CPU entirely.

Due to its high bandwidth, fast networks have gained attention for data analytics. Traditionally, there used to be a large gap between the bandwidth of network and intra-processors, which led to designs of CPU-intensive algorithms for query processing, which would avoid the network traffic as much as possible. The authors in [12] argue that fast networks have started to change this assumption, and proposed a distributed query engine which optimizes the exchange operator for bulk transfer using RDMA. [3] showed how RDMA could be leveraged to boost the performance of join.

References

[1] A. Devulapalli and P. Wyckoff. Distributed queue-based locking using advanced network features. In *Parallel Processing, 2005. ICPP 2005. International Conference on*, pages 408–415. IEEE, 2005.

[2] A. Dragojevic, D. Narayanan, M. Castro, and O. Hodson. Farm: Fast remote memory. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014*, pages 401–414, 2014.

[3] P. W. Frey, R. Goncalves, M. Kersten, and J. Teubner. A spinning join that does not get dizzy. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 283–292. IEEE, 2010.

[4] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker. OLTP through the looking glass, and what we found there. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 981–992, 2008.

[5] E. P. Jones, D. J. Abadi, and S. Madden. Low overhead concurrency control for partitioned main memory databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 603–614, New York, NY, USA, 2010.

[6] A. Kalia, M. Kaminsky, and D. G. Andersen. Using rdma efficiently for key-value services. *SIGCOMM Comput. Commun. Rev.*, 44(4):295–306, Aug. 2014.

[7] H. Lim, D. Han, D. G. Andersen, and M. Kaminsky. Mica: A holistic approach to fast in-memory key-value storage. *management*, 15(32):36, 2014.

[8] X. Lu, N. S. Islam, M. Wasi-ur Rahman, J. Jose, H. Subramoni, H. Wang, and D. K. Panda. High-performance design of hadoop rpc with rdma over infiniband. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 641–650. IEEE, 2013.

[9] C. Mitchell, Y. Geng, and J. Li. Using one-sided rdma reads to build a fast, cpu-efficient key-value store. In *USENIX Annual Technical Conference*, pages 103–114, 2013.

[10] S. Narravula, A. Mamidala, A. Vishnu, K. Vaidyanathan, and D. K. Panda. High performance distributed lock management services using network-based remote atomic operations. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 583–590. IEEE, 2007.

[11] M. Poke and T. Hoefler. DARE: High-Performance State Machine Replication on RDMA Networks. ACM, Jun. 2015. Accepted at ACM HPDC'15.

[12] W. Rödiger, T. Mühlbauer, A. Kemper, and T. Neumann. High-speed query processing over high-speed networks. *arXiv preprint arXiv:1502.07169*, 2015.

[13] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The end of an architectural era (it's time for a complete rewrite). In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 1150–1160, 2007.