HyperLoop: Group-Based NIC-Offloading to Accelerate Replicated Transactions in Multi-Tenant Storage Systems

Originally published in In Proceedings of ACM SIGCOMM Conference

Daehyeok Kim[1], Amirsaman Memaripour[2], Anirudh Badam[3], Yibo Zhu[4], Hongqiang Harry Liu[5], Jitu Padhye[3], Shachar Raindel[3], Steven Swanson[2], Vyas Sekar[1], Srinivasan Seshan[1]

[1]Carnegie Mellon University, [2]UC San Diego, [3]Microsoft, [4]Bytedance, [5]Alibaba

1 MOTIVATION

Distributed storage systems are an important building block for modern online services. To guarantee data availability and integrity, these systems keep multiple replicas of each data object on different servers and rely on *replicated transactional operations* to ensure that updates are consistently and atomically performed on all replicas.

Such replicated transactions can incur large and unpredictable latencies, and thus impact the overall performance of storage-intensive applications. Recognizing this problem, both networking and storage communities have proposed a number of solutions to reduce average and tail latencies of such systems.

Networking proposals include kernel bypass techniques, such as RDMA (Remote Direct Memory Access) [8], and userspace networking technologies [3]. Similarly, there have been efforts to integrate non-volatile main memory (NVM) [1], and userspace solid state disks (SSDs) [2] to bypass the OS storage stack to reduce latency.

While optimizations such as kernel bypass do improve the performance for standalone storage services and appliance-like systems where there is only a single service running in the entire cluster [7], they are unable to provide low and predictable latency for multi-tenant storage systems.

The problem is two fold. First, many of the proposed techniques for reducing average and tail latencies rely on using CPU for I/O polling [2, 3]. In a multi-tenant cloud setting, however, providers cannot afford to burn cores in this manner to be economically viable. Second, even without polling, the CPU is involved in too many steps in a replicated storage transaction. Take a `write` operation as an example: i) It needs CPU for logging, processing of log records, and truncation of the log to ensure that all the modifications listed in a transaction happen ***a**tomically*; ii) CPU also runs a ***c**onsistency* protocol to ensure all the replicas reach identical states before sending an ACK to the client; iii) During transactions, CPU must be involved to lock all replicas for the ***i**solation* between different transactions for correctness; iv) Finally, CPU ensures that the data from the network stack reaches a ***d**urable* storage medium before sending an ACK.

To guarantee these ACID properties, the whole transaction has to stop and wait for a CPU to finish its tasks at each of the four steps. Unfortunately, in multi-tenant storage systems, which co-locate 100s of database instances on a single server to improve utilization, the CPU is likely to incur frequent context switches and other scheduling issues.

2 HYPERLOOP DESIGN

In this paper, we explore an alternative approach for predictable replicated transaction performance in a multi-tenant environment by *completely removing* the CPU from the critical path. We present HyperLoop, a design that achieves this goal. Tasks that previously needed to run on CPU are entirely offloaded to commodity RDMA NICs, with Non-volatile Memory (NVM) as the storage medium, without the need for CPU polling. Thus, HyperLoop achieves predictable performance (up to 800× reduction of 99th percentile latency in microbenchmarks) with nearly 0% CPU usage. Our insight is driven by the observation that replicated transactional operations are essentially a set of memory operations and thus are viable candidates for offloading to RDMA NICs, which can directly access or modify contents in NVM.

Group-based RDMA primitives: In designing HyperLoop, we introduce new and general *group-based* NIC offloading primitives for NVM access in contrast to conventional RDMA operations that only offload point-to-point communications via volatile memory. HyperLoop primitives provide a necessary mechanism for accelerating replicated transactions, which helps perform logically identical and semantically powerful memory operations on a group of storage servers' durable NVM regions without remote CPUs' involvement. These include group-write (gWRITE), memcpy (gMEMCPY), compare-and-swap (gCAS), and flush (gFLUSH).

These group-based primitives are sufficient to offload operations that are conventionally performed by CPUs in state-of-the-art NVM and RDMA based replication systems. Such operations include consistent and atomic processing and truncating of log updates across all replicas, acquiring the same logical lock across all replicas, and durably flushing the volatile data across all replicas. HyperLoop can help offload these to the NIC.

Realizing these primitives, however, is not straightforward with existing systems. Our design entails two key technical innovations to this end. First, we repurpose a less-studied yet widely supported RDMA operation that lets a NIC wait for certain events before executing RDMA operations in a special queue. This enables us to pre-post RDMA operations which are triggered only when the transactional requirements are met. Second, we develop a remote RDMA operation posting scheme that allows a NIC to enqueue RDMA operations on other NICs in the network. This is realized by modifying the NIC driver and registering the driver metadata region itself to

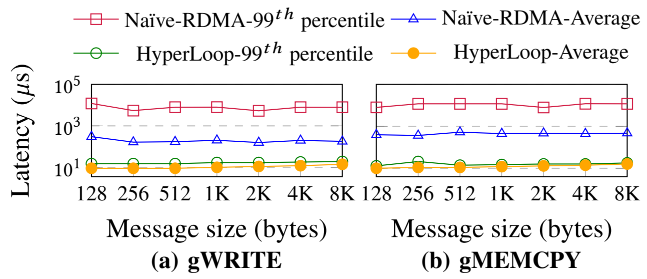

Figure 1: Latency of gWRITE and gMEMCPY compared to Naïve-RDMA.

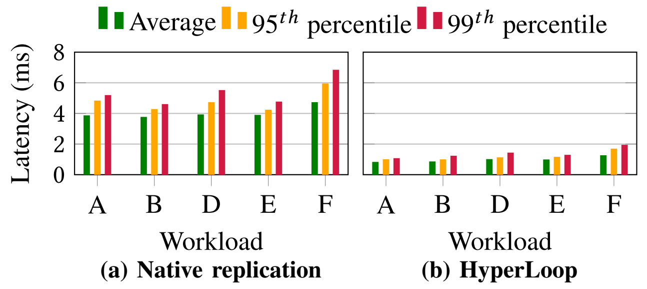

Figure 2: Latency distribution of MongoDB with native and HyperLoop-enabled replication.

be RDMA-accessible (with safety checks) from other NICs. By combining these two techniques, an RDMA-capable NIC can precisely program a group of other NICs to perform replicated transactions.

Our approach is quite general as it only uses commodity RDMA NICs and as such applications can adopt our primitives with ease. In particular, HyperLoop provides APIs built based on our group-based RDMA primitives for transactional storage applications. These include log replication, log processing, and locking for isolation. Using the APIs, we modified RocksDB [5] (an open source alternative to Google LevelDB) and MongoDB [4] (an open source alternative to Azure CosmosDB and Amazon DynamoDB) to use HyperLoop with under 1000 lines of code.

Note that our key idea of HyperLoop, *i.e.,* offloading replicated transactions to NICs in addition to point-to-point read/write operations, can be extended to other hardware combinations as long as the NIC can directly access the storage medium, such as FPGA based-Ethernet NICs that can access SSDs [6].

Implementation: We implement the primitives with 3,417 lines of C library. We also modify 58 lines of code in `libmlx4` and `libibverbs`[1] to implement proposed group-based RDMA primitives for NVM.

3 RESULT HIGHLIGHTS

We evaluate the performance of these systems using microbenchmarks as well as the Yahoo Cloud Storage Benchmark (YCSB) workload. In our evaluation, we assume storage nodes are equipped with battery-backed DRAM [7] which is a form of NVM available today. We emulate battery-backed DRAM by mounting a `tmpfs` file system to DRAM and run applications on it. We expect to observe similar benefits, *i.e.,* no CPU overhead on replicas while exhausting the bandwidth of the network and/or NVM, when replacing battery-backed DRAM with emerging technologies such as 3D XPoint [1].

Figure 1 shows the average and tail latency of gWRITE and gMEMCPY primitives with different message sizes, fixing replication group size (number of member nodes) to 3. For both of gWRITE and gMEMCPY, Naïve-RDMA shows much higher 99^{th} percentile latency than HyperLoop. Particularly, for gWRITE, we can see that with HyperLoop, 99^{th} percentile latency can be reduced by up to 801.8× with HyperLoop. gMEMCPY shows a similar result; HyperLoop reduces the 99^{th} percentile latency by up to 848× compared to Naïve-RDMA which performs the same set of operations as HyperLoop, but involves backup CPUs to handle receiving, parsing, and forwarding RDMA messages.

Further, using YCSB workload, we show that running MongoDB with HyperLoop decreases average latency of insert/update operations by 79% and reduces the gap between average and 99^{th} percentile by 81%, while CPU usage on backup nodes goes down from nearly 100% to almost 0% (Figure 2).

4 LOOKING FORWARD

In this paper, we designed HyperLoop, a new framework that completely eliminates CPUs from the critical path of replicated transactions in multi-tenant storage systems to achieve predictable low latency. HyperLoop entirely offloads replicated transactions to a group of commodity RDMA NICs, with NVM as a storage medium. Looking forward, even though our specific focus in this paper was on storage systems, we believe that the design and insights underlying HyperLoop can be more broadly applicable to other data center applications, especially which leverage NVM.

REFERENCES

[1] 2015. Intel/Micron 3D-Xpoint Non-Volatile Main Memory. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-micron-3d-xpoint-webcast.html. Accessed on 2018-01-25.
[2] 2016. Intel Storage Performance Development Kit. https://software.intel.com/en-us/articles/introduction-to-the-storage-performance-development-kit-spdk. Accessed on 2018-01-25.
[3] 2018. Intel Data Plane Development Kit. http://dpdk.org/. Accessed on 2018-01-25.
[4] 2018. MongoDB. https://www.mongodb.com/. Accessed on 2018-01-25.
[5] 2018. RocksDB. http://rocksdb.org/. Accessed on 2018-01-25.
[6] Adrian M. Caulfield and Steven Swanson. 2013. QuickSAN: A Storage Area Network for Fast, Distributed, Solid State Disks. In *ACM/IEEE ISCA* (2013).
[7] Aleksandar Dragojević, Dushyanth Narayanan, Edmund B. Nightingale, Matthew Renzelmann, Alex Shamis, Anirudh Badam, and Miguel Castro. 2015. No Compromises: Distributed Transactions with Consistency, Availability, and Performance. In *ACM SOSP* (2015).
[8] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. 2016. RDMA over Commodity Ethernet at Scale. In *ACM SIGCOMM* (2016).

[1] `libibverbs` and `libmlx4` are userspace libraries/drivers that allow userspace processes to use InfiniBand/RDMA Verbs on Mellanox hardware.