Optimizing Redis with RDMA

Liyan Zheng, Kezhao Huang, Tianhui Shi, Zixuan Ma, Yutian Wang, Jidong Zhai

Tsinghua University

2018/10/9

Overview

Redis is an open-source in-memory key-value database, using TCP socket to set up communication between clients and servers as well as within clusters.

Standalone Mode

- Event handling mechanism: handling both file event and time event
- I/O event notification mechanism: using epoll to get informed when there are messages sent or received(file events happen)

Cluster Mode

- High availability: Redis Cluster is able to survive partitions where the majority of the master nodes are reachable and there is at least one reachable slave for every master node that is no longer reachable
- Acceptable degree of write safety: The system tries (in a best-effort way) to retain all the writes originating from clients connected with the majority of the master nodes. However usually there are small windows where acknowledged writes can be lost.
- High performance and linear scalability

Cluster Mode

There are two kinds of messages across clusters:

- Status message: Ping pong, ACK with timestamps and so on. Usually sent in low frequency. Its magnitude can be ignored compared with the massive messages sent between cluster and clients.
- Data message: The master will backup all commands that update data in the database on slaves. It is of the same magnitude with the client-server communication.

The choice of RDMA verbs

Instead of using RDMA Receive and Send to send messages to server from client which are more similar with socket, we choose to utilize RDMA write. The client can write to server without the server knowing it, and we designed poll function for RDMA write to realize I/O event notification. Write can help us achieve better performance than Send-Recv.

To decrease clients' CPU use, we choose RDMA send to pass server's reply back to clients. This helps us increase every client's throughput.

The choice of RDMA verbs

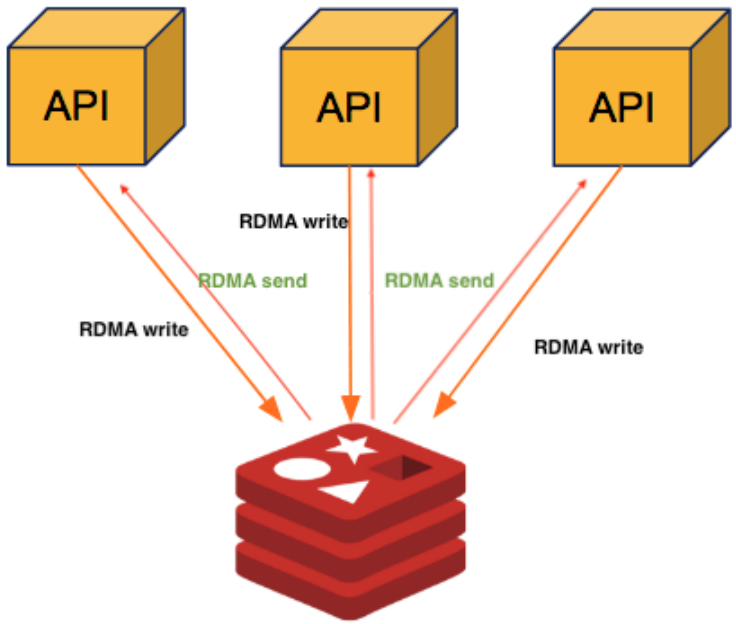


Figure 1: RDMA between clients and server

Pipeline

To maximize the advantage of RDMA write, a server have multiple slots. Each slot is able to contain a message from the client, so there can be more than one outstanding messages, making it possible for server to handle more operations within the same time.

Dividing the buffer for inbound RDMA Write of server in multiple parts can help with the slots, the configuration of the slot number and size will be synchronized between server and clients. So that the process of passing message from client to server can fully bypass the server's CPU

Optimization

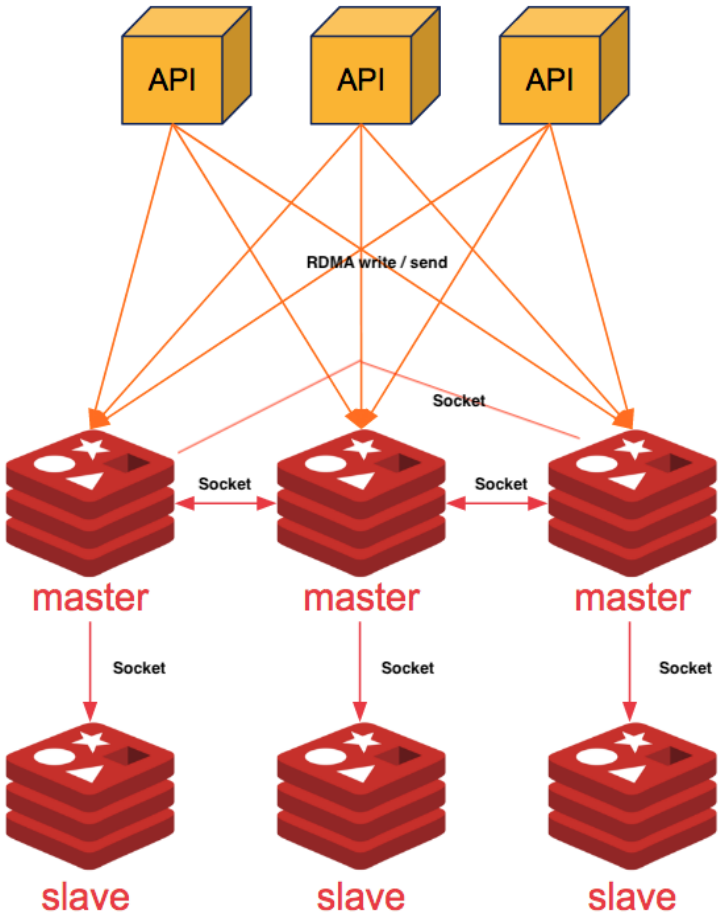
- Selectively signaled and inline SEND
  Small message is a common use case for Redis, such as implementing expires on items, counting and queues.
  Overhead of RDMA events themselves (not transferring) significantly influence performance in such case.
- Send response immediately
  Latency of IB link is noticeably lower than it of Ethernet.
  Separating Send operations can achieve both low latency and high throughput.

Cluster Mode

By default, Redis Cluster uses asynchronous replication between nodes to achieve high performance and scalability while preserving weak but reasonable forms of data safety and availability. And the original Redis can achieve linear scalability.

So when using asynchronous replication between nodes, the communication between nodes is not the bottle neck of the performance. However, Redis sacrifice better data safety for high performance because it is unaffordable to use synchronous replication with TCP socket. But with the low latency brought by RDMA, it is possible for us to achieve stronger data safety. We get RDMA Write in charge of the message passing within clusters, which can achieve least latency.

Cluster Mode



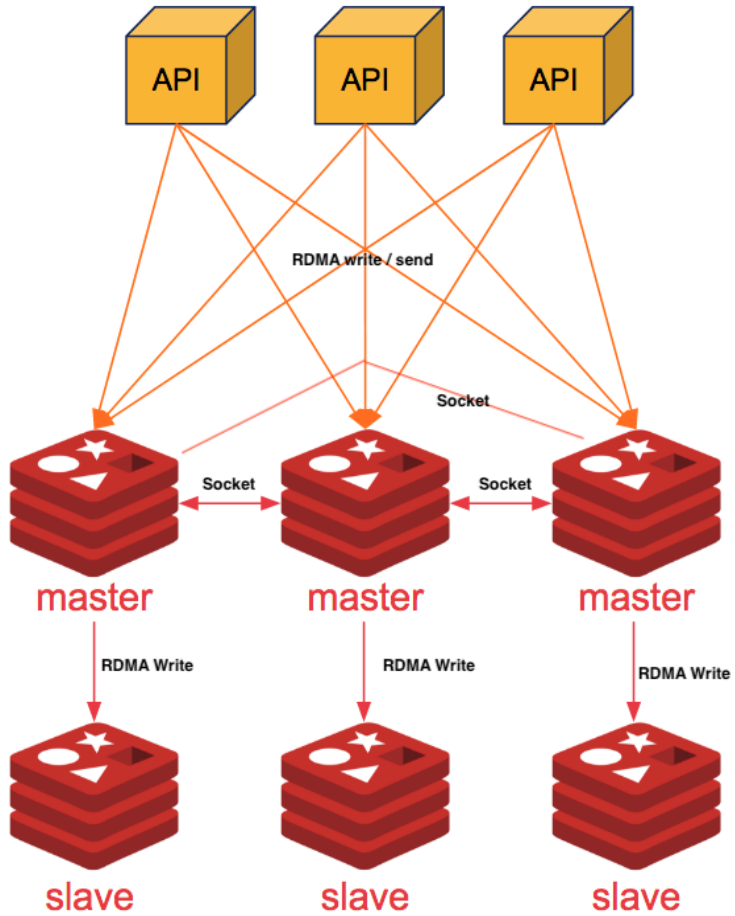


Figure 2: Using socket to send data Figure 3: Using RDMA to write data

RDMA between clients and server

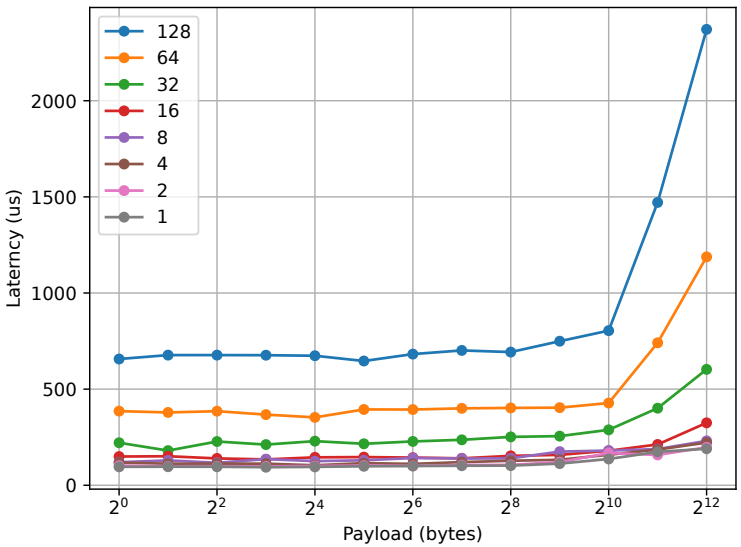


Figure 4: latency of socket Redis

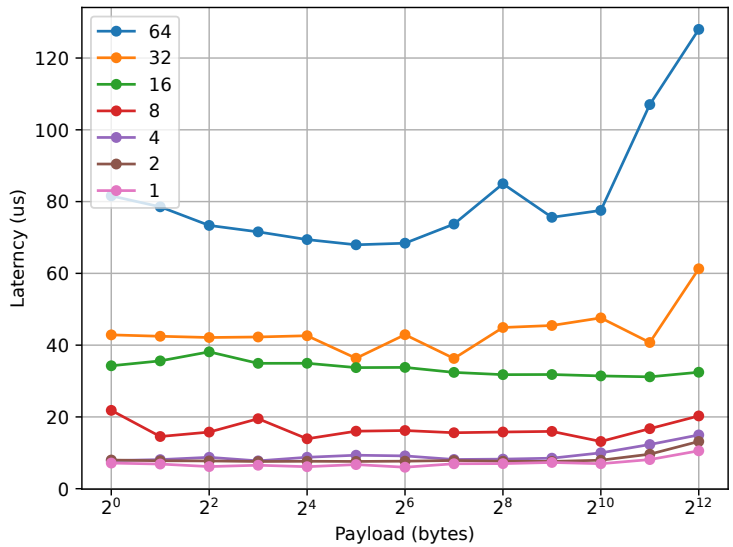


Figure 5: latency of RDMA Redis

RDMA between clients and server

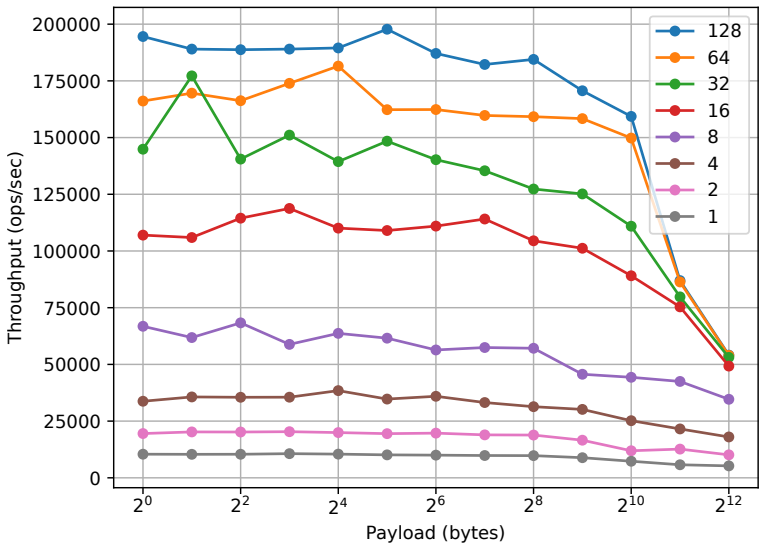


Figure 6: throughput of socket Redis

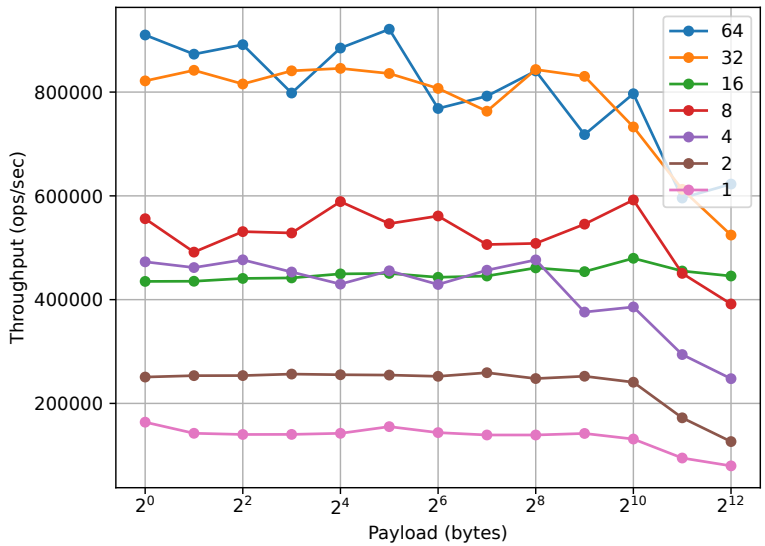


Figure 7: throughput of RDMA Redis

RDMA between clients and server

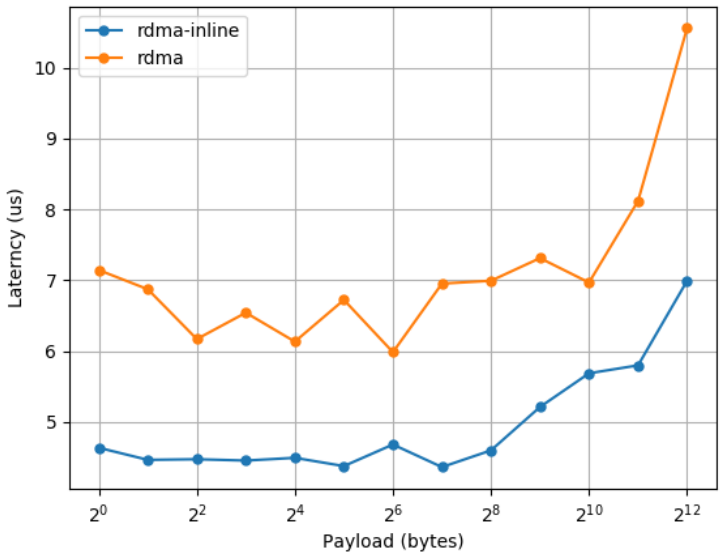


Figure 8: latency of inline and normal

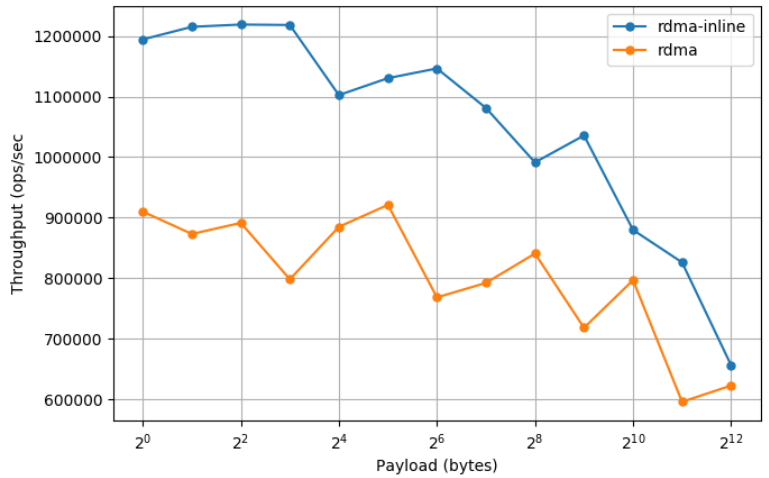


Figure 9: throughput of inline and normal

To saturate IB bandwidth

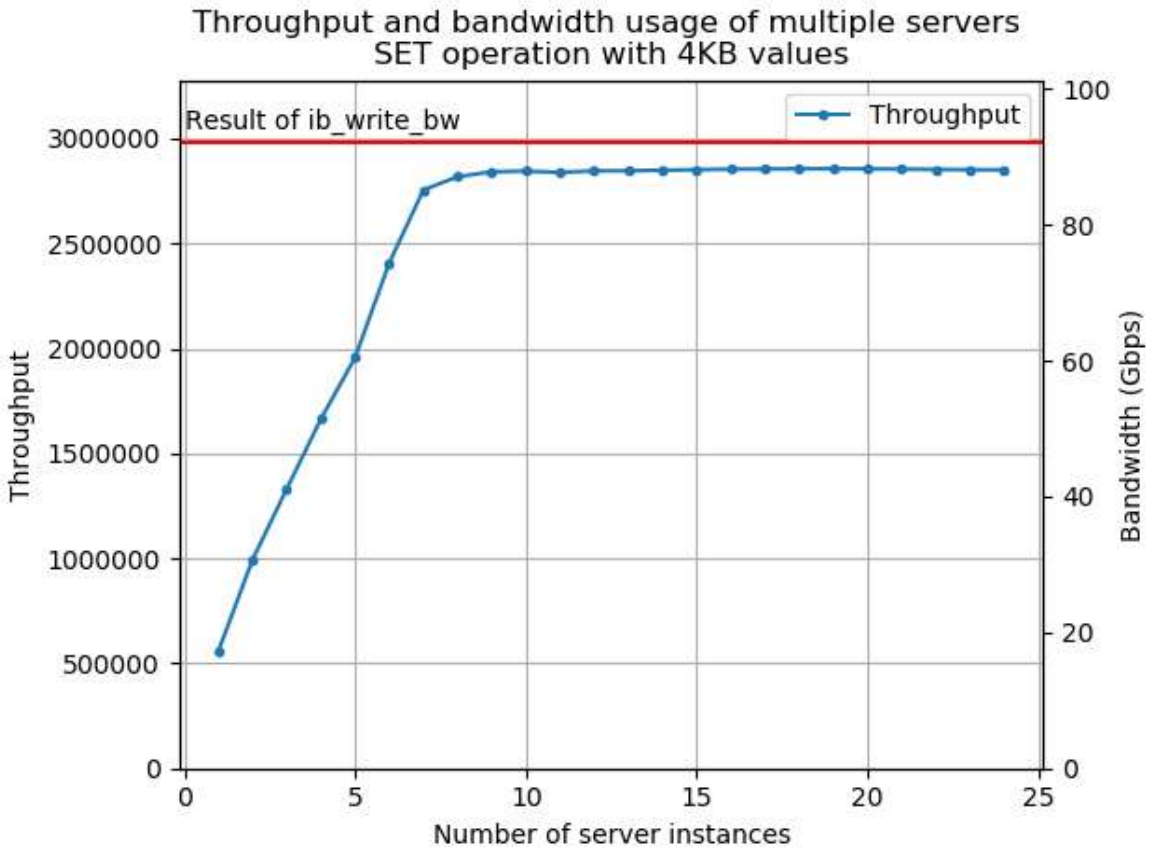


Figure 10: Total throughput and bandwidth usage grows with RDMA server instances increase

RDMA Cluster

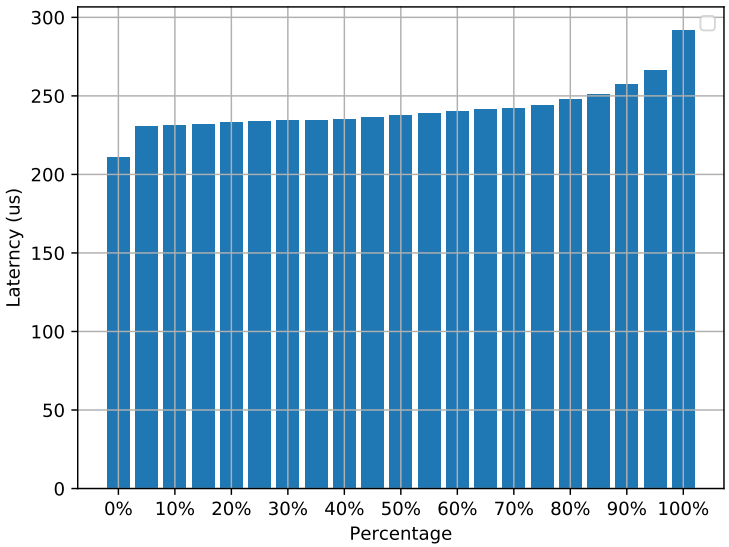


Figure 11: latency distribution of socket cluster

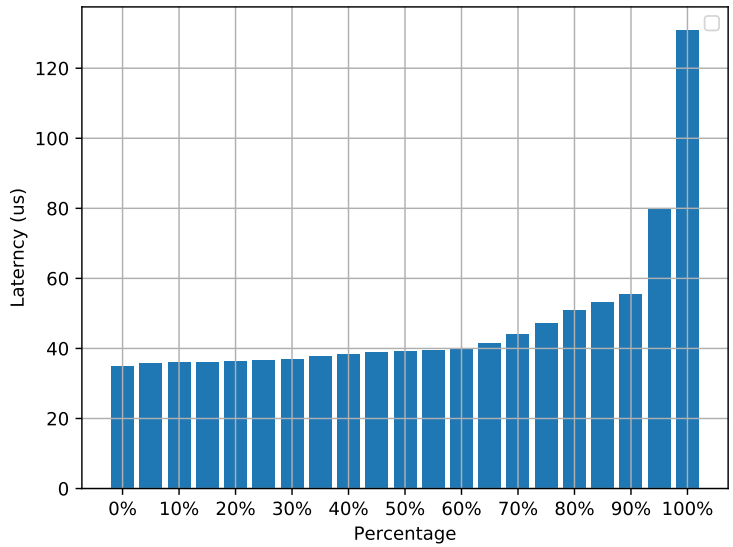


Figure 12: latency distribution of RDMA cluster

References

References

- Jian Huang, Xiangyong Ouyang, Jithin Jose, Md. Wasi-ur Rahman, Hao Wang, Miao Luo, Hari Subramoni, Chet Murthy, and Dhabaleswar K. Panda. High-performance design of HBase with RDMA over Infini- Band. pages 774–785. IEEE.
- Anuj Kalia, Michael Kaminsky, and David G Andersen. Design guidelines for high performance RDMA systems. page 15.
- Anuj Kalia, Michael Kaminsky, and David G Andersen. Using RDMA efficiently for key-value services. page 22.
- Wenhui Tang, Yutong Lu, Nong Xiao, Fang Liu, and Zhiguang Chen. Accelerating redis with RDMA over InfiniBand. In Ying Tan, Hideyuki Takagi, and Yuhui Shi, editors, Data Mining and Big Data, volume 10387, pages 472–483. Springer International Publishing.

Q&A

Thank you for listening!