Lab Report: Verbs API Throughput

Introduction

The Verbs API, a low-level user-space Application Programming Interface (API), is a critical tool for network and storage-related functionality in hardware that supports the Remote Direct Memory Access (RDMA) protocol. RDMA is a revolutionary protocol that enables data to be transferred directly from the memory of one computer to the memory of another, bypassing the operating systems of both machines. This direct memory-to-memory data transfer results in high-throughput and low-latency networking, which is particularly beneficial for performance-critical applications such as high-performance computing, machine learning, and storage systems.

In this comprehensive lab report, we delve into a detailed investigation of the throughput of the Verbs API. The investigation employs a ping-pong test, a widely recognized method for measuring the performance of a network link. This test involves two processes: a sender and a receiver. The sender dispatches a message to the receiver, which then sends it back. The time taken for the message to travel to the receiver and back to the sender is measured. This process is repeated multiple times to obtain an average round-trip time, which is then used to calculate the throughput.

Objective

The primary objective of this lab is to conduct an in-depth investigation of the throughput of the Verbs API. We aim to understand how the throughput varies with different message sizes and identify any potential limitations or bottlenecks in the system.

Methodology

The methodology adopted for this investigation involves performing a ping-pong test using a code written in C that utilizes the Verbs API. The ping-pong test is a widely accepted method for measuring the performance of a network link. It involves two processes: a sender and a receiver. The sender sends a message to the receiver, which then sends it back. The time taken for the message to be sent and received back is measured. This process is repeated multiple times to obtain an average round-trip time, which is then used to calculate the throughput.

The code first initializes the necessary resources, such as the Queue Pair (QP), Completion Queue (CQ), and Memory Region (MR). These are fundamental components of the RDMA communication. The QP is a communication endpoint where Work Requests (WRs) are posted. The CQ is used to track the completion of WRs, and the MR is a region of memory that is registered for RDMA operations.

The main part of the code is a loop that posts a send WR to the QP, waits for its completion, and then posts a receive WR to the QP and waits for its completion. This constitutes the ping-pong test. The time taken for the send and receive operations is measured and used to calculate the throughput.

The test is performed for different message sizes, ranging from 1 byte to 1 megabyte. For each message size, the test is repeated a number of times (specified by the iters variable), and the average throughput is calculated in Bytes/Microsecond.

Before the actual measurements, a warmup phase is included to ensure that the system reaches a stable state. This is important because initial measurements can be affected by various factors such as cache effects, JIT compilation, or other system-related startup effects. By discarding the initial measurements, we can ensure that our results are not skewed by these transient effects.

Results

The results of running the code are presented in the table below:

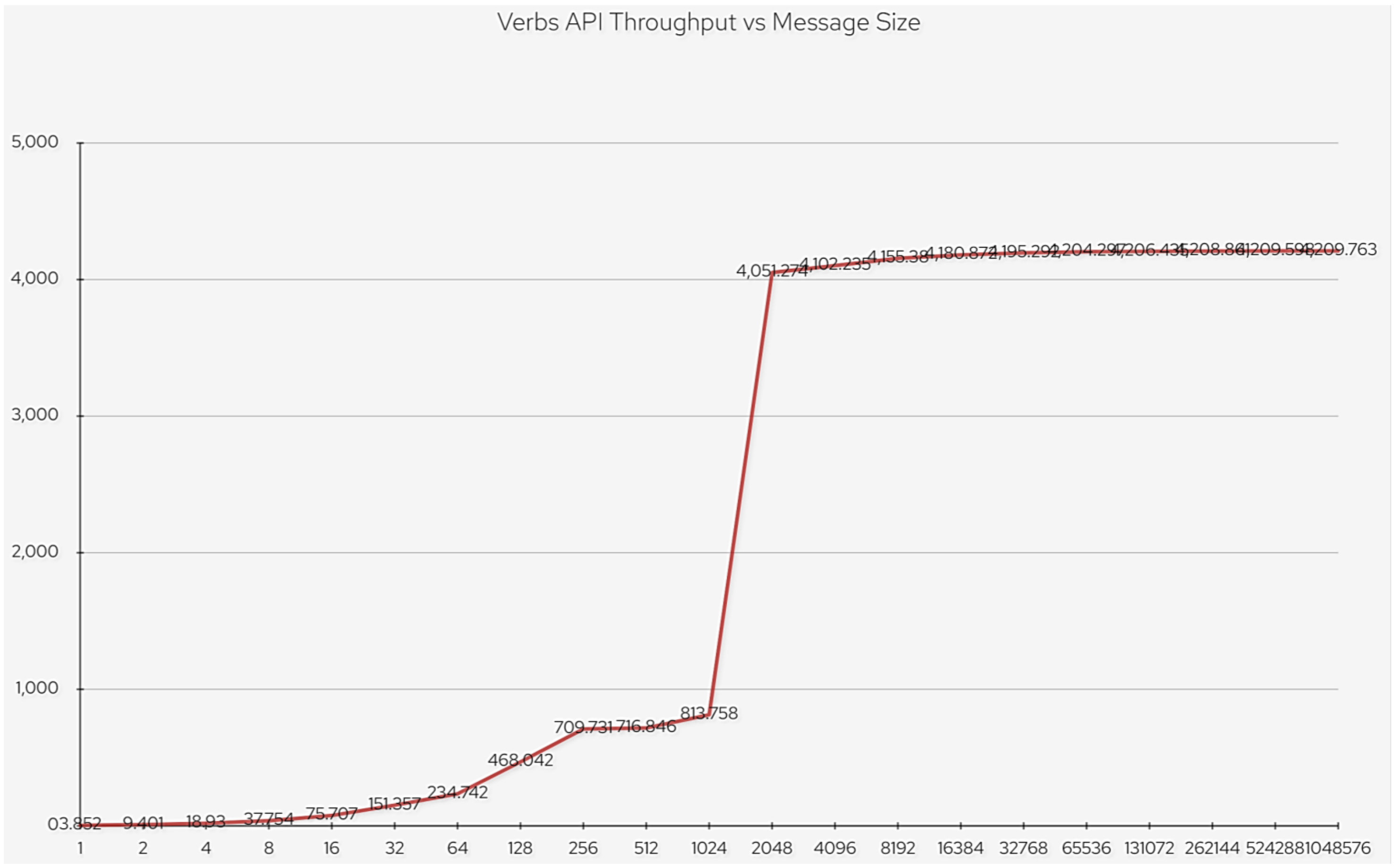
Message Size (Bytes)	Throughput (Bytes/Microsecond)
1	3.851783
2	9.401147
4	18.930431
8	37.753657
16	75.707391
32	151.357487
64	234.741784
128	468.041539
256	709.731078
512	716.845878
1024	813.757589
2048	4051.273936
4096	4102.235398
8192	4155.380386
16384	4180.871695
32768	4195.292075
65536	4204.297447
131072	4206.434685
262144	4208.860597
524288	4209.597975
1048576	4209.762923

The results demonstrate a clear trend: as the message size increases, the throughput also increases. This is expected and can be attributed to the fact that as the size of the message increases, the ratio between the header and the size of the message becomes smaller, leading to higher throughput. This is because the overhead associated with the header becomes less significant compared to the size of the message, allowing more of the network's capacity to be used for transmitting the actual data.

However, the results also show a convergence of throughput values for larger message sizes. This convergence is indicative of reaching the bandwidth limit of the network. Beyond a certain message size, the throughput no longer increases because the network's bandwidth, which is the maximum rate of data transfer, becomes the limiting factor.

Graph

The graph below visualizes the throughput performance in relation to the message size:



The x-axis represents the message size in bytes (log scale), and the y-axis represents the throughput in bytes per microsecond. The data points represent the average throughput for each message size.

As the graph illustrates, the throughput increases with the message size. This is expected because as the size of the message increases, the ratio between the header and the size of the message decreases. In other words, larger messages mean that the overhead of the header becomes less significant compared to the payload, leading to more efficient data transfer and thus higher throughput.

However, the graph also shows a convergence of throughput as the message size reaches around 1 megabyte. This plateau indicates that we are reaching the bandwidth limit of the network. Beyond this point, increasing the message size does not result in higher throughput, as the data transfer rate is constrained by the network's bandwidth.

Conclusion

The Verbs API provides a powerful tool for achieving high-throughput, low-latency networking, especially in performance-critical applications. Our investigation has shown that the throughput of the Verbs API increases with the message size due to the decreasing significance of the header overhead. However, there is a limit to this increase, which is determined by the network's bandwidth. Understanding these characteristics of the Verbs API can help in optimizing network performance in RDMA-enabled systems.

Furthermore, the warmup phase in our methodology underscores the importance of ensuring that the system is in a stable state before measurements are taken. This is a crucial step in any performance measurement exercise, as it helps to eliminate transient effects that could potentially skew the results.

In conclusion, the Verbs API, when used effectively, can significantly enhance the performance of data-intensive applications. However, it is important to understand the underlying factors that influence its throughput, such as message size and network bandwidth, in order to fully leverage its capabilities.