ISBN: 978-3-95450-249-3 ISSN: 2673-5350 doi: 10.18429/JACoW-IBIC2024-WEP65

DESIGN OF DATA TRANSMISSION SCHEME BASED ON RDMA

Y. Q. Zhang, K. Zhou, M. Li, R. S. Mao, Institute of Modern Physics, Lanzhou, China

Abstract

With the development of precise radiotherapy, high-throughput data transmission has become a critical component of beam diagnostics. As the volume of generated measurement data rapidly increase, the data transmission mode that utilizes traditional Ethernet protocol, remote CPU and operating system to control memory read and write can not meet the transmission performance requirements. To break through these bottlenecks and achieve more real-time and efficient data transmission and processing, this paper designs a prototype data transmission system based on RDMA technology. By directly transferring memory data between hosts, the system bypasses the operating system kernel and CPU intervention, thereby minimizing transmission latency and enhancing data throughput. The system utilizes the RoCE v2 network protocol and is implemented through the libibverbs dynamic link library to establish stable RDMA sessions and develop corresponding network programs. Performance evaluations in terms of transmission latency, throughput, and CPU utilization indicate that the network transmission scheme proposed in this paper offers lower latency, higher throughput, and reduced CPU usage compared to schemes using the TCP protocol during large-scale data transfers.

INTRODUCTION

RDMA is a network protocol that enables the rapid transfer of data from one system to the memory of a remote system over various network technologies (such as InfiniBand or TCP/IP) without impacting the operating system. It eliminates context switching and data copy operations, thereby freeing up bus bandwidth and CPU cycles to enhance the performance of application systems. This paper designs a data transmission scheme based on RDMA technology by utilizing the OFED (OpenFabrics Enterprise Distribution) open-source project and the rdma-core user-space library. After a comprehensive consideration of compatibility, cost, and performance, the RoCE protocol, which offers a good balance of lower cost and satisfactory performance, was selected. Based on considerations of data integrity and transmission reliability, the Reliable Connection (RC) service type was selected.

DESIGN OF THE SYSTEM

The data transmission system is a server/client application designed to establish a duplex data channel between the communicating parties, allowing them to exchange data. As illustrated in Figure 1, the network adapters are physically connected to the transmission terminals via the PCIe interface, enabling direct read/write access to terminal memory through their DMA controllers. The network adapters between terminals are directly connected by fiber optics, serving as the physical data channel during the transmission process. Resources like the work queue pairs and completion queues created in the "registered memory" of the terminals can be abstracted as the data interface with the network adapters, forming the logical data channel for transmission.

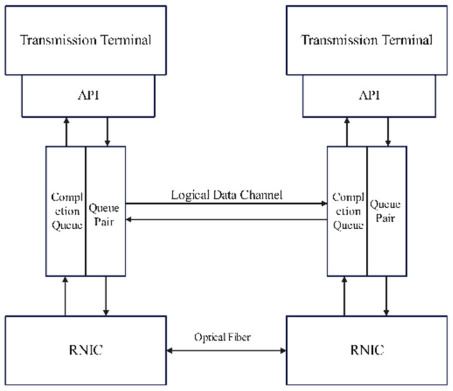

Figure 1: Transmission channel of the transmission system.

As shown in Figure 2, the client of the data transmission system is divided into five modules: resource management, communication establishment, RDMA operations, queue pair management, and resource cleanup. The server introduces a thread invocation module, which is used for concurrently handling POSIX threads from different clients for subsequent communication. Although the server and client implement different functionalities, both are integrated within the same system architecture.

The resource management module is responsible for initializing, creating, and managing all necessary system resources while controlling their lifecycle. This includes handling device context structures, protection domains (PD), completion queues (CQ), queue pairs (QP), memory regions, and data buffers, among others. The system dynamically allocates RDMA-specific memory regions (MR) within the associated PD to prevent resource leakage. At the same time, it generates unique Local Key (L_KEY) and Remote Key (R_KEY) fields for access permission verification.

In the IB (InfiniBand) specification, the QP is a core component of communication and can be abstracted as a virtual interface between software and hardware. It sequentially stores the Work Queue Elements (WQE) that the hardware needs to execute, where WQE corresponds to the Work Requests (WR) mentioned earlier from the software perspective. From a data perspective, the interface of a QP to the application layer is divided into “send (post_send)” and “receive (post_recv)” operations. This means that application layer software fills a WR into the QP, requesting the underlying hardware to perform a “send” or “receive” operation. This also helps distinguish whether the QP acts as the “sender” or “receiver” in a particular communication session. The queue pair management module triggers state

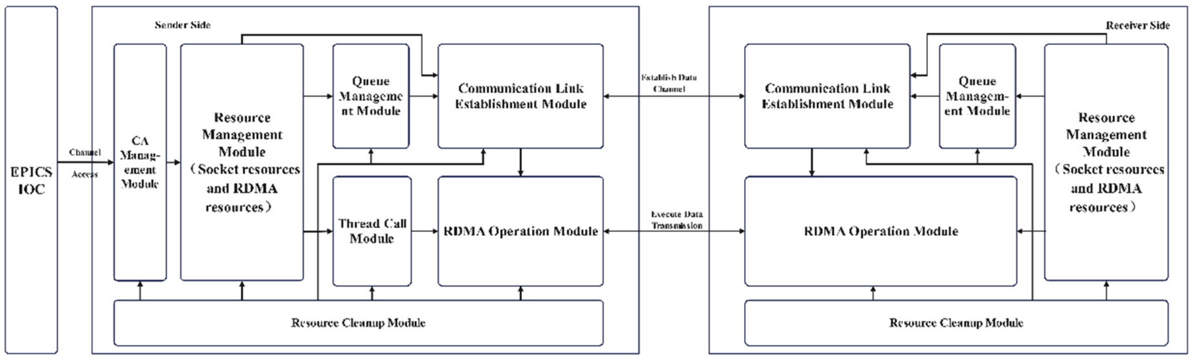

Figure 2: The basic structure of the transmission system.

transitions of the QP to make it available for invocation by other modules. Once a queue pair (QP) is created, it automatically enters the RST (Reset) state. To transition the QP into the INIT (Initialized) state, the program sets a series of QP attributes and then calls an interface to complete the state transition. The RTR (Ready To Receive) state indicates that the QP is ready to receive. The RTS (Ready To Send) state signifies that the SQ can operate normally.

Communication establishment is a necessary preparatory step for the local and remote QPs before exchanging data. This paper utilizes the Socket API for establishing the connection, allowing for more flexible

configuration of related parameters, which control the communication pathway. The main control information includes the registered memory address, R_KEY, QP Number (QPN), and Global Identifier (GID), among others. Once the Socket communication is established, the QP state can be adjusted to create the data path.

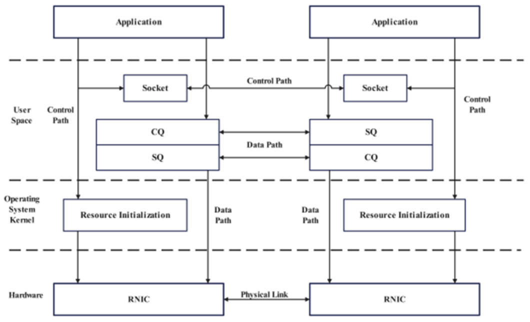

Figure 3: Pathway of the RDMA transmission system.

As shown in Figure 3, the control pathway is used for the creation, synchronization, and initialization of RDMA basic elements, establishing the connection between the server and client, and exchanging control information necessary for transmission. The data pathway is established after the connection between the QPs on both ends is complete and is primarily used for data transmission. Low-frequency operations such as initialization and configuration are executed in the operating system kernel mode and exchanged between the control pathway and the peer to avoid impacting transmission efficiency. Data transmission occurs directly through the data pathway, bypassing the kernel to achieve kernel bypass.

The RDMA operations module requires asynchronous operation of the send queue (SQ), receive queue (RQ), and completion queue (CQ) during RDMA send and receive operations. To ensure multithreaded concurrency, dedicated worker threads are created for each of the three queues. The basic workflow is illustrated in Figure 4.

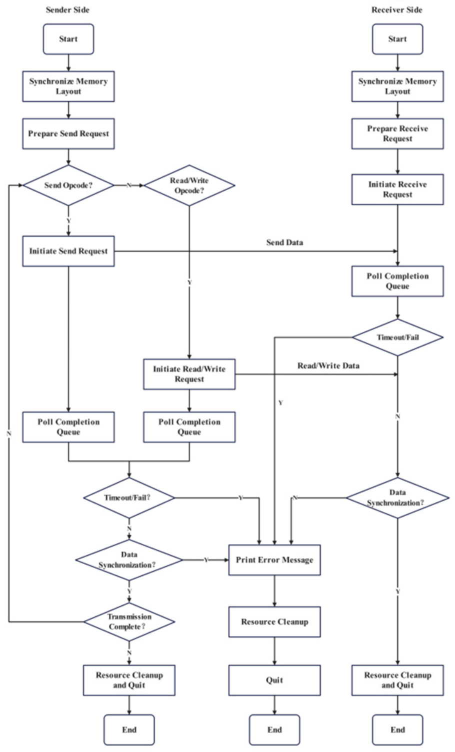

Figure 4: Basic process of RDMA operations.

The send queue thread receives all resources necessary for network communication and sets the operation code. The Send Request (SR) is a structure that includes multiple attribute fields, such as scatter/gather element pointer addresses to specify the data memory layout, the length of the data to be sent, and the L_KEY for memory access. The SR is then submitted to the hardware; if the submission fails, an error handling pointer is returned, pointing to the problematic work request. Finally, the resource cleanup module is invoked to destroy the thread. The receive queue thread needs to define a Receive Request (RR) structure for initialization before submitting the RR to the underlying hardware.

The completion queue (CQ) stores Completion Queue Elements (CQE), which can be abstractly viewed as the interface between the underlying hardware and the upper-level software, used to notify the software of the completion of an RDMA operation. Since the hardware asynchronously places CQEs into the CQ, the software needs to actively retrieve the CQE through the data plane interface. This paper employs a non-blocking polling method to handle CQEs.

After the data transmission is complete, the system employs an event-driven communication framework for data synchronization. This approach allows for the simultaneous handling of read and write events, which ensures the reliability of data transmission.

TEST AND RESULTS

The system testing aims to compare the performance of the RDMA-based data transmission system designed in this paper with that of traditional TCP-based data transmission, focusing on three performance metrics: transmission latency, throughput, and CPU load. The experimental environment is set up based on the slow control network of the CEE (CSR External Target Experiment) project at the Institute of Modern Physics, Chinese Academyof Sciences. Both the server and client run on PCs equipped with the Ubuntu 20.04 operating system, and RDMA-related drivers (MLNX_OFED_LINUX-5.8-3.0.7.0) are loaded on both ends. The RDMA test network cards used on both sides are high-speed cards based on the Mellanox MT28908A0, specifically the M2890-2QSFP 200G model, with optical fiber serving as the transmission medium. The RDMA-based data transmission system using the RoCE protocol serves as the experimental group, while the traditional data transmission system utilizing the TCP protocol serves as the control group, both groups perform 10,000 data transmission trials.

Transmission latency is defined as the time interval between the start of data transmission and the completion of data transmission. Data packets ranging from 64 bytes to 128 KB are transmitted, ensuring that the RoCE send and receive queues are sufficiently long and that the traffic is adequate. The average transmission latency test results for both groups are shown in Figure 5.

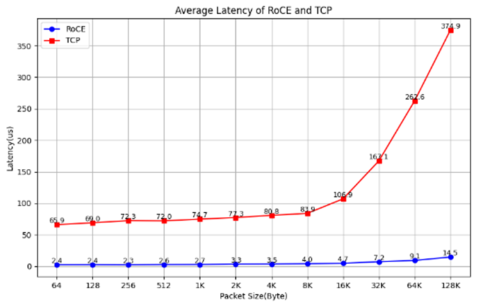

Figure.5: Comparison of average transmission latency between RoCE and TCP.

The test results indicate that for packet sizes smaller than 16 KB, the average transmission latencies for both systems are relatively stable, with RoCE latency being significantly lower than that of TCP, averaging a difference of 74.99 microseconds. When the packet size exceeds 16 KB, the TCP latency begins to increase significantly, while the RoCE latency also increases but with a less pronounced rise, resulting in an average difference of 257.95 microseconds between the two. The calculations show that the transmission latency of the RDMA-based data transmission system is reduced by an average of 96.3% compared to the TCP transmission method.

Under conditions ensuring that the RoCE send and receive queues are sufficiently long and that there is adequate traffic, throughput tests are conducted for both the experimental group and the control group. The test results are shown in Figure 6.

The test results indicate that as the size of the transmitted data packets increases, both the experimental and control groups exhibit a linear growth in throughput. Moreover, the throughput of RoCE significantly surpasses that of TCP. Calculations reveal that the transmission throughput based on the RDMA data transfer system shows an average improvement of 95.1% compared to the TCP transfer method.

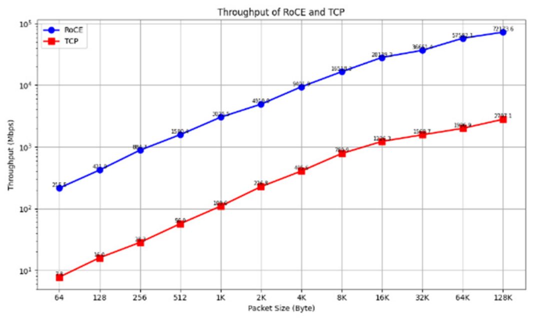

Figure.6: Comparison of throughput between RoCE and TCP.

CPU load tests were conducted on both the experimental and control groups, and the test results are presented in Figure 7.

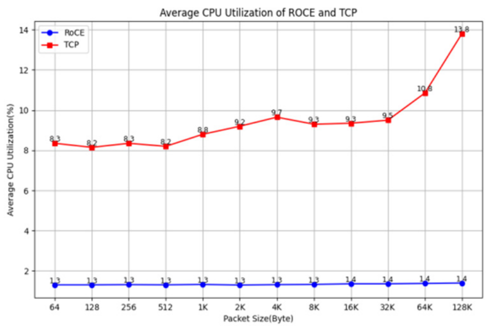

Figure.7: Comparison of CPU Load between RoCE and TCP.

The test results show that across all data packet length ranges, the CPU load for RoCE remains extremely low, significantly lower than that of the TCP mode. The transmission CPU load based on the RDMA data transfer system demonstrates an average reduction of 85.6% compared to the TCP transmission method.

CONCLUSIONS

This paper designs and implements a data transmission system based on RDMA technology. A testing environment was constructed to conduct long-term stability transmission tests using this system. The test results indicate that the proposed system significantly outperforms the TCP transmission mode in transmission latency, throughput, and CPU load. These findings demonstrate the superior performance and reliability of the system.