RMQ: An RDMA-capable Message Queue

Minghua Deng*, Yesheng Ma*, Xueyuan Zhao*
Carnegie Mellon University
{minghuad,yeshengm,xueyuanz}@andrew.cmu.edu

ABSTRACT

Messages are the most broadly used approach for communication in distributed systems. Programming message-based distributed systems was painful for a long time due to the complex nature of network environments. The recent advent of message queues helps alleviate this problem by providing a generic interface for different advanced messaging patterns. However, existing message queues are built upon kernel-based network protocols and have not yet leveraged modern RDMA hardware. This paper presents the design and implementation of RMQ, an RDMA-capable message queue built on top of existing message queue libraries, which achieves 1.5 to 5x throughput improvement compared with existing TCP-based message queue implementation, and supports all the advanced patterns and optimization existing message queue libraries has to offer.

1 INTRODUCTION

(Broker-less) Message queue is a very desirable network programming abstract that are highly optimized for high-performance and has built-in support for many different advanced network programming patterns, such as request-reply, publisher-subscriber, and Push-Pull. As a result, many distributed systems are using message queues as building blocks to build there high-performance network components on. However, through existing message queue implementations supports many different underlying protocols, such as TCP/UDP/IPC, none of them provides native support for another appealing network technology, Remote Direct Memory Access(RDMA).

The idea of remote direct memory access is not new, many research works has been done on this topic starting from 90s. which provides better latency and throughput for the system by bypassing kernel and CPU. As the devices(NICs/Switchs) that natively supports RDMA become more and more available for commercial systems nowadays, many companies that wish to have extremly low-latency and high-performance are replacing their network stacks with RDMA support ones.

It is obvious that if we can combine message queue with RDMA, and do it wisely, we can get both higher performance and desirable network programming abstracts. Moreover, we believe that it provides an alternative solution for building RDMA-based distributed systems, which people now often build from scratch or use RDMA-based TCP libraries. RDMA-capable Message Queue can be used as an building block that hides complicated RDMA settings but provides expressive network primitives in building these kind of systems.

In this paper, we first introduces some related design of both RDMA and messaging systems in section 2 and section 3, and then we dive into the message queue socket abstraction where we base our work on in section 4. After that we will explain some of the design choices we made in and the architecture of our systems RMQ in section 5. Then we will have evaluation in section 6 and related work in section 7. Finally, we will conclude in section 8.

*Authors contributed equally to this research.

2 BACKGROUND

In this section, we will discuss modern hardware and software that are essential for building high performance network application. We will also discuss

2.1 Remote direct memory access

Remote direct memory access (RDMA) is a network technology that has been proposed long ago[14] and has becoming more and more appealing in recent system research[1] [8] [11].

RDMA are now supported through three different ways in commercial clusters, including using RDMA native-supported hardware Infiniband[12], using technology developed by Mellanox called RDMA over Converged Ethernet(RoCE)[2], and Internet Wide Area RDMA Protocol. While technology like RoCE can provide support for RDMA in commercial Ethernet, the performance advantage of RDMA can only be fully utilized using Infiniband which can provide extremely high bandwidth (up to 100Gbps), low latency and link layer guarantees. The wide equipment of infiniband in network performance sensitive cluster is also the reason that brings RDMA back to alive.

RDMA gains its performance advantages over traditional Ethernet based TCP/IP network by bypassing the kernel stack when transmitting packets, and directly reads and writes data from and to main memory through PCIe. It support several different primitives including reliable/unreliable data transmission, and CPU are only involved in posting the action (verbs in RDMA) into queues, and NICs will process them by accessing bound memory without invoking CPU again. As a result, it may help achieve much higher throughput in CPU-bound senario.

RDMA supports two different data transmission patterns, one is two-sided RDMA, and another is one-sided RDMA, the differences between these two are mainly on whether both sides' CPUs need to post actions into the queue before sending or receiving messages. For one sided RDMA, only the side that actively takes action need to post action into NIC, and for two sided RDMA, the other side need also to post an action to prepare for the message transmission. The performance of these two transmission pattern can be varied for different scenario and need to be considered for achieving the optimal performance.[7] [9]

2.2 Messaging Systems

There are two kinds of messaging systems, broker and brokerless. Broker messaging systems persist messages for failure recovery whereas brokerless messaging systems have lightweight messaging