Toward Scalable RDMA Through Resource Prefetching

Zhenlong Ma, Ning Kang, Fan Yang, Chongyang Hong, Jing Xu, Guojun Yuan, Peiheng Zhang, Zhan Wang, and Ninghui Sun

***Abstract*—RDMA network is being widely deployed in data centers, high-performance computing, and AI clusters. By offloading the network processing protocol stack to hardware, RDMA bypasses the operating system kernel, thereby enabling high performance and low CPU overhead. However, the protocol processing demands substantial communication resources, and due to the limited hardware resources, commercial NICs (Network Interface Cards) experience a significant number of cache misses in large-scale connection scenarios. This results in performance degradation, indicating that RDMA lacks scalability. In this paper, we first analyze the characteristics of resource access in RDMA. Based on these characteristics, we propose a resource access prediction and prefetching mechanism in the hardware, which preemptively fetches the resources required by the protocol processing pipeline to the on-chip cache. This mechanism increases the NIC's cache hit ratio. Evaluation results demonstrate that our approach improves throughput by 125% and reduces latency by 17.9% under large-scale communication scenarios.**

***Index Terms*—Remote direct memory access, scalability problem, high performance network, network interface card.**

I. Introduction

RDMA (Remote Direct Memory Access) is being deployed in various computing systems as high performance network infrastructure to support applications like AI [1], storage [2], RPC [3], etc. However, RDMA faces scalability issues. RDMA offloads the entire protocol stack into the RDMA Network Interface Card (RNIC), which needs a lot of communication resources including connection contexts, communication descriptors and memory translation metadata. These resources are stored on the RNIC's SRAM as a cache. When the communication scales up, cache misses frequently happen, leading to data exchanges through PCIe with the host memory [4], and therefore the throughput of the RNIC drops [4], [5], [6].

We propose a cache management machanism based on resource prefetching on the RNIC hardware. In our work, The RNIC predicts the resources that will be utilized in the subsequent period of time, either based on the information provided by the scheduling module or the user program. Then the RNIC prefetches them into the cache in advance. This strategy improves the cache hit rate in the RDMA protocol processing pipeline and thereby optimize the performance of the RNIC under large scale communication scenarios.

We make the following contributions in this work:

- We raise a resource prefetching framework on the RNIC hardware to achieve high cache hit rate.
- We implement and evaluate the prefetching mechanism in a RNIC simulator based on Gem-5 to trace its performance gains. The evaluation shows that with prefetching, the RNIC can maintain high throughput and low latency when communication scales.

II. RNIC Workflow and Scalability Problem

Traditional network built on TCP/IP moves data between the user and kernel space of the operating system, which takes at least tens of microseconds [7]. In contrast, RDMA eliminates these cumbersome data copies and context switches by offloading the entire network stack into the NIC hardware.

In RDMA, a connection is instantiated as a Queue Pair (QP), which contains Work Queue Elements (WQEs) recording the location and access permission of the message data to be transferred. An RDMA network interface card (RNIC) works as Fig. 1 shows. The user writes WQEs into the QP in the host memory and trigger the doorbell register on the RNIC to launch the transmission. Then the RNIC fetches the QP context (QPC) to get the location of the WQEs, followed by fetching WQEs to get the memory address of the message data. Sequentially, the RNIC fetches the Memory Protection Table (MPT) and Memory Translation Table (MTT) items to check the memory access permission and translate the virtual address given by the user process into physical memory address. And finally, the RNIC transmits the message data from the host memory into the network. In the whole procedure, the RNIC accesses the QPC, WQEs, MPT and MTT frequently. However, the RNIC is not capable to store all the resources on the chip. Therfore, the RNIC stores only part of these resources as caches.

However, when the communication scales, i.e. the RNIC needs to handle a large number of QPs or memory regions, the communication resources are missed in the cache frequently, and the RNIC fetches these resources from the host memory through PCIe, which introduces evident time delay on the processing pipeline and decreases the throughput of the communication, which is shown in Fig. 3. Consequently, when multiple QPs are transmitting data simultaneously, the throughput of RDMA is even lower than the TCP/IP network [4]. In consideration of the fact that nodes in modern data centers often establish more

Received 11 October 2024; revised 11 January 2025; accepted 21 January 2025. Date of publication 27 January 2025; date of current version 26 March 2025. This work was supported by the Beijing Municipal Science and Technology Program under Grant Z241100004224026, in part by the Jiangsu Science and Technology Project under Grant BE2022051-2, and in part by the NSFC under Grant 62401541. *(Corresponding author: Fan Yang.)*

Ning Kang, Fan Yang, Chongyang Hong, Guojun Yuan, Zhan Wang, and Ninghui Sun are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China (e-mail: yangfan2020@ict.ac.cn).

Zhenlong Ma and Jing Xu are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China, and also with University of Chinese Academy of Sciences, Beijing 101408, China.

Peiheng Zhang is with the Institute of Intelligent Computing Technology, CAS, Suzhou 215163, China.

Digital Object Identifier 10.1109/LCA.2025.3534188

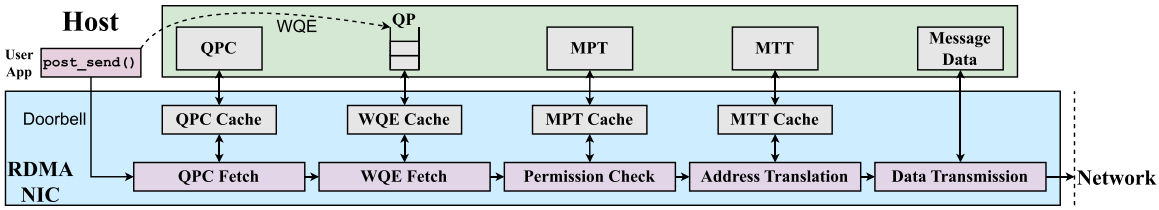

Fig. 1. RDMA NIC workflow.

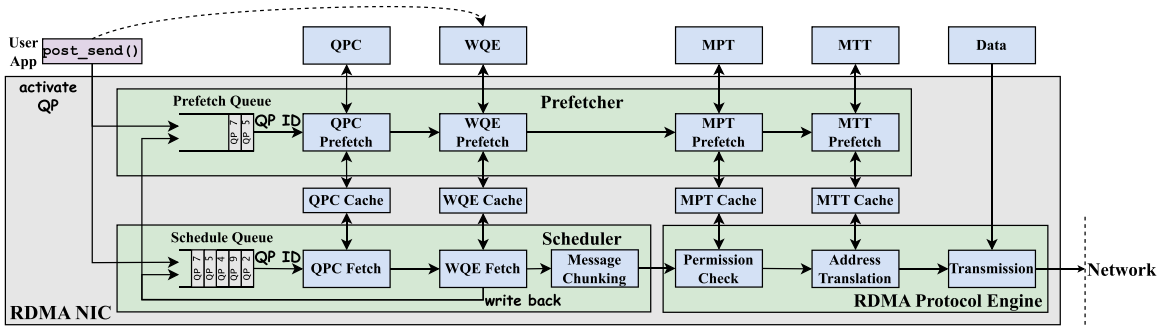

Fig. 2. Scheduling triggered prefetching architecture.

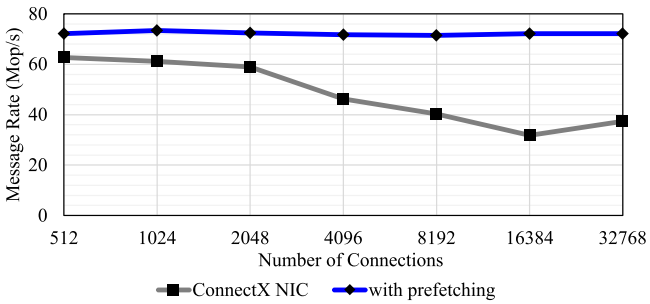

Fig. 3. Connection scalability on message rate of ConnectX-5 NIC and NIC with prefetching.

than 10 K connections [8], it is essential to solve the scalability problem of RDMA.

III. Methodology

Although communication resource caches on the RNIC undertake a similar job with the cache inside CPU, they are accessed and organized in different patterns. Specifically, the resource access of RNIC exhibits the following characteristics:

- The RNIC accesses the communication resource in a stream manner. When a QP is scheduled to process communication tasks, its corresponding resources are fetched and utilized. Once utilized, these resources are not accessed again until this QP is rescheduled next time. This periodic access pattern contrasts with the more continuous patterns with strong locality typically seen in CPU caches.
- Each network flow accesses multiple kind of resources. These resources are highly relevant with each other but resources of different QPs are independent. For example, the access of MPT relies on the information provided by WQEs while MTT resources rely on segments in MPT, but the MPT of QP A has no relationship with QP B.

Based on the characteristics above, we demonstrate that an effective management framework for RNIC caches should not rely on the temporal or spatial locality. In addition, different types of communication resources should not be managed as seperate components. Therefore, we propose to prefetch resource to cache in advance in a chain manner. This strategy increases the cache hit rate and in consequence solve the scalability problem effectively.

Our main challenges can be summarized as follows:

- How to predict the communication resource to be utilized in the following time period?
- When should the NIC prefetch the resource ahead of the main processing pipeline?

To address these challenges, we propose two prefetching policies based on scenarios of heavy-loaded or light-loaded. Our design contains a scheduling module, a cache management module and a prefetching module. In the heavy-loaded scenario, the prefetching is triggered by the scheduling module, while in the light-loaded scenario, it is triggered by the software.

A. Scheduling Triggered Prefetching

In situations involving a large amount of communication connections wating to be processed, the RNIC schedules these QPs by picking one of them and delivering it into the processing pipleline. In addition, the scheduler also specify the amount of data to be transmitted in each processing procedure. The execution sequence is maintained in the scheduler. Thus we use the scheduler to guide the prefetcher. The architecture of the scheduler and the prefetcher is shown in Fig. 2.

We design a simple scheduler by employing a scheduling queue to store the identifiers of all active QPs. When the software triggers the doorbell register, the QP number is placed into the scheduling queue. The fetch module then pops the QP number out and retrieves the QP context and communication descriptors

from the WQE buffer. The chunk module generates descriptors of a specific size and sends them to the RDMA protocol processing pipeline. After the chunk module completes one QP, if this QP has remaining tasks, the QP ID is written back into the sheduling queue.

The processing rate or the RNIC is typically limited by the network line rate or the control logic of the protocol pipeline, therefore the RNIC has enough time to prefetch the resource before the resource is utilized.

In the prefetching module, we have established an additional queue that stores the QP IDs prepared for prefetching. Each element in this prefetching queue is a copy of an element from the scheduling queue. When the software triggers the doorbell or the scheduling module completes a batch of WQE fetch and writes the QP ID back to the scheduling queue, the same QP ID is simultaneously written into the prefetching queue.

In the scheduler triggered prefetching mechanism, the prefetching queue is designed to initiate resource prefetch requests ahead of the scheduling queue by a certain quantity of QPs. This mechanism ensures that resources are preloaded and ready for use when the scheduling queue requires them. The prefetch module operates by popping a QP ID from the prefetching queue once the capacity difference between the prefetch and scheduling queues falls below a specific threshold, which is referred to as the prefetching window.

To achieve the resource prefetching of the ith QP in the scheduling queue in time, assume that each QP is scheduled to transmit data length of $chunk\_size$, and n kinds of resources need to be utilized by the processing pipeline, and the delay of fetching each resource is $delay\_fetch$. The size of the prefetching window wnd should satisfy the following condition:

$$\frac{chunk\_size}{BW} \times wnd \geq n \times delay\_fetch$$

From the formula above, it is evident that the the latency to retrieve resources, and chunk size can significantly influence the required prefetching window size. Specifically, a higher delay and smaller chunk sizes need a larger window. In addition, if the prefetching window is set too large, the resources may be replaced with high probability before they are utilized, resulting in an ineffective prefetching. If it is set too small, there may not be enough time to prefetch the resources, leading to potential pipeline bubbles. Therefore, the size of the prefethcing window should be as small as possible under the premise of meeting the condition above.

Due to the interdependent nature of communication resources on the RNIC, the prefetcher first prefetches the QPC and WQEs, followed by MPT and MTT resources based on the content provided by the WQEs.

B. Software Triggered Prefetching

In the light-load scenario, characterized by a small number of concurrent communications transmitting small messages, RNIC cache misses result in increased latency. In this case, the queuing delay of the scheduling queue is too short so that it is impractical to prefetch the communication resources based on the scheduler. However, before writing the doorbell into the RNIC, the user application needs to generate the WQEs and copy them into the QP area in the memory in advance, which takes at least several hundreds of nanoseconds according to our experiment. Consequently, we utilize this preparation time in the software to hide the time delay of resource fetching.

We modified the Base Address Register (BAR) space on the network interface hardware by allocating a prefetch register within the BAR space. This register's address is mapped to the user address space via mmap. Additionally, we modified the `ibv_post_send` function at the libibverbs layer of the RDMA communication API. When the user invokes the `ibv_post_send` function, the software writes the corresponding QP ID into the prefetch register through Programmed Input/Output (PIO) before generating and copying the WQEs and triggering the doorbell operation. This mechanism notifies the hardware to ensure that the resource related with a specific QP is stored in the hardware storage unit before the RNIC receives the doorbell. If the number of active connections are below the prefetching window size, the hardware launches the prefetching task of this QP by pushing the QP ID into the prefetching queue.

IV. Evaluation

We evaluate our prefetching mechanism using a Gem-5-based cycle-accurate RNIC simulator. We set network link bandwidth to 100 Gbps, same as Nvidia ConnectX-5 RNIC, set PCIe bandwidth to 128 Gbps, same as PCIe Gen3 x16, and set PCIe Round-Trip Time to 500 ns. Since our work focuses on the architecture inside the RNIC, our test is based on two nodes to avoid the interference from the network influencing our test experiment.

We compare the performance of our prefetching mechanism with that of the commodity RNIC and existing scalability optimization for RDMA. The result shows that resource prefetching enhances the RNIC's scalability, offering significant improvements over existing approaches.

A. Connection Scalability

In this experiment, we examined the variation in message throughput with different numbers of concurrently QPs. Each QP was configured to continuously send 50 messages of 64 bytes in size. We compared our results to those obtained from running perftest on a Nvidia ConnectX-5 RNIC.

As shown in the Fig. 3, our platform's peak message rate is close to that of the ConnectX RNIC. However, when the number of QPs exceeds 2048, ConnectX's throughput declines as the number of QPs increases. With 16384 concurrent QPs, its throughput is only half of the peak value. In contrast, our platform maintains a stable throughput at the peak value under large-scale QPs. This is attributed to our scheduling and prefetch modules, which efficiently fetch resources into the on-chip SRAM, thereby mitigating the impact of cache misses on communication throughput.

We also in depth evaluate the influence of the prefetching window size on throughput. In Fig. 4, in the absence of prefetching, i.e. the window size is set to zero, the message rate of the

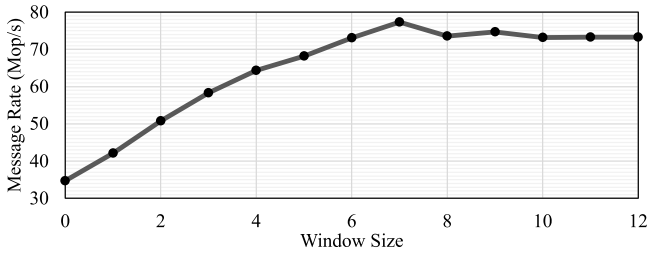

Fig. 4. The relationship between the message rate and prefetching window.

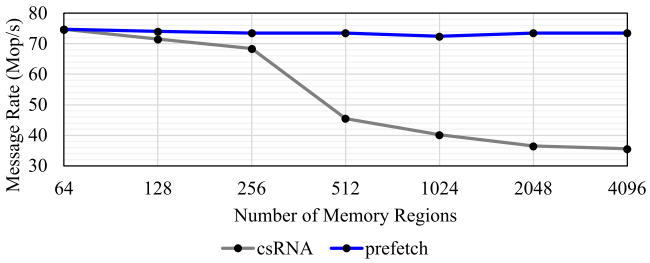

Fig. 5. Memory scalability.

RNIC is about 35 Mop/s, which is only half of the peak value. With the window size expands from zero to 7, the message rate increases continuously. When the windows size exceeds 7, all QP's resources can be fully prefetched in time and the message rate reaches its peak value.

B. Memory Scalability

We evaluated the memory scalability of the RNIC by creating a large number of memory regions. In this set of experiments, we set the capacity of MPT and MTT cache as 256 items. We compare our work with csRNA [5], another work aiming at solving the RNIC scalability targeting at reducing the cache miss penalty of QPC. However, csRNA did not take MPT and MTT into consideration.

As shown in Fig. 5, when the number of memory regions exceeds 256, csRNA's throughput drops significantly from 70 Mop/s to about 40 Mop/s. In contrast, our prefetching-based approach is unaffected by the number of memory regions, demonstrating superior scalability in managing MPT and MTT resources.

C. Software-Triggered Prefetch

Finally, we also conducted the experiment testing the communication latency of a large number of QPs under light-load communication scenarios. We employed software prefetching for the QP contexts.

We established up to 256 QPs, circularly polling these QPs to perform WRITE operations. Once a QP received a completion event, the communication proceeded with the next QP. We set the capacity of the QPC cache as 100. Fig. 6 presents our test results.

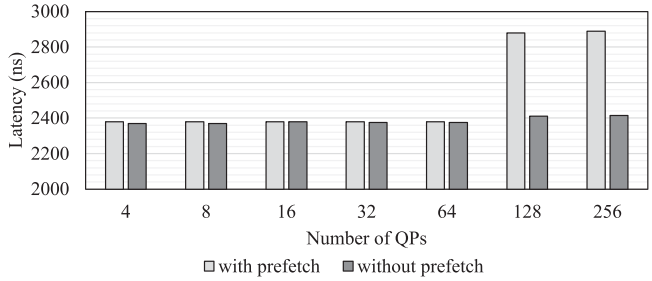

Fig. 6. Latency of light-loaded communication.

When the number of QPs ranges from 4 to 64, no cache miss occurred, resulting in a latency of approximately 2.4 microseconds, regardless of whether prefetching was used. However, when the number of QPs increased to 128 or more, cache misses caused the latency to rise to around 2.9 microseconds. Compared to scenarios without prefetching, software prefetching significantly reduces this increase in latency.

V. Conclusion

This paper proposes a cache management method which enhances RDMA scalability by addressing on-chip cache miss through prefetching resources into the device SRAM from the host memory before they are required in the protocol processing pipeline. Our approach integrates a hardware scheduler and software modification to predict the resources to be utilized. The results demonstrate that cache prefetching significantly improves RDMA performance in scenarios involving a large number of connections and registered memory regions.

References

[1] A. Gangidi et al., "RDMA over ethernet for distributed training at meta scale," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2024, pp. 57–70.

[2] S. Ma, T. Ma, K. Chen, and Y. Wu, "A survey of storage systems in the RDMA era," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4395–4409, Dec. 2022.

[3] A. Kalia et al., "Datacenter RPCs can be general and fast," in *Proc. 16th USENIX Symp. Netw. Syst. Des. Implementation*, Boston, MA, USENIX Association, 2019, pp. 1–16.

[4] Z. Wang et al., "SRNIC: A scalable architecture for RDMA NICs," in *Proc. 20th USENIX Symp. Netw. Syst. Des. Implementation*. Boston, MA, USENIX Association, 2023, pp. 1–14.

[5] N. Kang et al., "csRNA: Connection-scalable RDMA NIC architecture in datacenter environment," in *Proc. IEEE 40th Int. Conf. Comput. Des.*, 2022, pp. 398–406.

[6] A. Kalia et al., "Design guidelines for high performance RDMA systems," in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, 2016, pp. 437–450.

[7] J. Ousterhout, "A linux kernel implementation of the homa transport protocol," in *Proc. USENIX Annu. Tech. Conf.*, USENIX Association, 2021, pp. 99–115.

[8] M. Yu et al., "Profiling network performance for multi-tier data center applications," in *Proc. 8th USENIX Symp. Netw. Syst. Des. Implementation*, Boston, MA, USENIX Association, 2011, pp. 57–70.