Buffer sizing with HPCC

Rui Miao, Bo Li, Hongqiang Harry Liu, Ming Zhang
Alibaba Group

1 INTRODUCTION

We conduct comprehensive experiments to understand the performance and buffer size requirement of HPCC [2], a High Precision Congestion Control algorithm designed for large-scale, high-speed networks. HPCC claims to maintain a near-zero queue for ultra-low latency by leveraging the precise link load information from INT(In-band Network Telemetry) to compute accurate flow rate updates. However, we are facing practical challenges in running HPCC in production data centers and therefore whether a near-zero queue is achievable and its impact on the transport performance is still questionable.

In this paper, we present our first experience in running HPCC in production. We deploy HPCC in two state-of-the-art high-performance transports, user-space TCP and RDMA RoCEv2. We show extensive experimental results to demonstrate the performance and challenges in running HPCC in software and hardware transports respectively. We provide evidence that buffer sizing is directly related to the performance of data center applications, where the congestion control (CC) algorithm is an essential scheme for this purpose. Compared with baseline congestion control algorithms, we show that HPCC keeps only a small buffer size and achieves high performance.

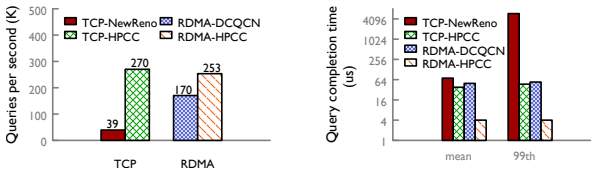
2 PERFORMANCE WITH BUFFER SIZING

2.1 Experiment setup

Network topology The testbed topology mimics a small-scale cluster in our production. The testbed includes two PoDs (point-of-delivery) connected by two core switches. Each PoD contains two ToRs and two Agg switches connected via four 100Gbps links. The testbed includes 8 servers in total and each server has two 25Gbps NICs.

Tool and parameters We use a Remote Procedure Call (RPC) tool for performance evaluation which employs both RDMA and user-space TCP as its transports. We compare HPCC with NewReno [1] in user-space TCP and DCQCN [4] in RDMA respectively, which are the default CC schemes used in production. For RDMA, we use the same parameter settings as in [2] for HPCC and DCQCN. For TCP, we use delayed ACKs and set minRTO to 2ms. Unlike in RDMA, processing INT data in TCP for every single packet incurs a high CPU overhead. Instead, the TCP sender sends a particular probing packet in every RTT, which collects INT data along the path and then is reflected by the receiver.

BS'19, December 2-3, Stanford, CA



(a) QPS. (b) QCT.

Figure 1: QPS and QCT for victim client.

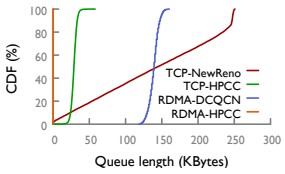


Figure 2: Queue length distribution.

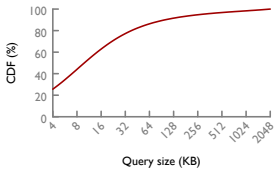
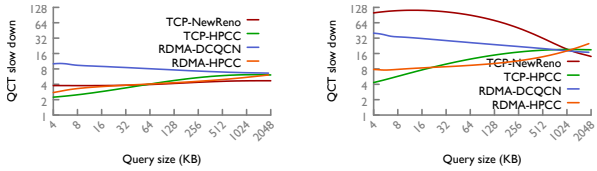


Figure 3: Query size distribution.

Switch buffer. We use dynamic buffering in switches. For TCP, we set the egress admission to allow a single congested queue to use up to 256KB buffer. For RDMA, we set the ingress admission so that the PFC is triggered when an ingress queue consumes more than 11% of the free buffer.

2.2 Handling Flooding Traffic.

When the network under heavy flooding traffic, we evaluate the performance of other traffic sharing the same bottleneck link. Three servers, each with 8 connections, send RPC queries to the same receiver continuously. Two servers acting as the flooding clients send 64 queries per connection at a time, while the other one acting as the victim client sends 2 queries per connection at a time. The query size is fixed at 4KB and the response size is small. We run the experiment for both TCP and RDMA over 2 minutes and measure the query performance of the victim client.



(a) Median. (b) 99th percentile.

Figure 4: QCT slowdown.

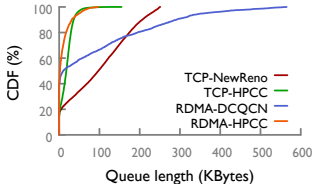


Figure 5: Queue length distribution.

HPCC improves QPS and QCT. Figure 1a shows the average queries per second (QPS) for the victim client. Using HPCC improves QPS by 7X in TCP and 1.5X in RDMA respectively. This improvement shows the HPCC's better fairness and less performance interference under the flooding traffic. Figure 1b shows query completion time (QCT) statistics for the victim client. HPCC achieves a consistently low QCT, while QCT in TCP NewReno experiences large inflation with up to 5.9ms in the 99th percentile.

Figure 2 shows the distribution of queue length at the bottleneck link. HPCC keeps a small queue length, even in the tail of the distribution. For example, HPCC maintains a consistently close-to-zero queue in RDMA and has a median of 29KB and 99th-percentile of 38KB in TCP. This provides direct evidence for its better fairness and lower latency.

Towards near-zero buffering. The essential obstacle in hindering from the near-zero queue is traffic bursts. In RDMA, those traffic bursts stem from newly-instantiated flows that start at line-rate. HPCC leaves a bandwidth headroom (5% by default) to absorb bursts and keeps almost zero-queue. In TCP, however, achieving zero-queue is more challenging due to the bursts from the software stack. To reduce the load and host CPU usage, using delayed ACKs is important in TCP over high-speed networks(>25GE). The receiver sends one cumulative ACK for a few consecutively received packets. On receiving the delayed ACK, the sender sends multiple packets in a burst, typically 5-10 packets. Given the typical bandwidth-delay product (BDP) within data centers is around 30-50 MTU packets, those traffic bursts make it hard for the CC algorithm to accurately estimate network utilization and thus control the queue size. One way to mitigate those bursts is to use hardware rate-limiting primitives to pace packets for sending [3].

2.3 Benchmark Traffic

We now evaluate performance and buffer sizing using the traffic trace collected from a production distributed storage system (Figure 3). The traffic pattern represents the prevalence of short RPC messages in data center networks. For example, 22.9% of queries have a size of 4KB and 90.5% of queries are less than 32KB. We vary the query generation rates to set the average link loads to 80%.

HPCC reduces QCT for short queries Figure 4 shows the median and 99th percentile of QCT slowdown across different query sizes, where QCT slowdown means a query's measured QCT normalized by its ideal QCT when there is no congestion.

Using HPCC in both TCP and RDMA achieves similar slowdowns with TCP NewReno in the median(Figure 4a). However, the median QCT of short queries in RDMA with DCQCN has large slowdowns. For example, DCQCN has a slowdown of 11.9 for 4KB queries in the median.

HPCC achieves much better slowdowns for short queries in the 99th percentile(Figure 4b). For example, HPCC reduces 99th-percentile QCT slowdown from DCQCN's 41.6 down to 7.8 in RDMA and NewReno's 100.0 down to 4.3 in TCP respectively. As the majority of queries are short(90.5% are less than 32KB), this slowdown reduction is very important.

HPCC has small queues Figure 5 shows the distribution of queue lengths at switches, which explains the reason. HPCC achieves a median queue length of 18KB in TCP and 2.2KB in RDMA respectively. This explains precisely why HPCC reduces QCT slowdown compared to NewReno and DCQCN. Besides, HPCC maintains a low queue length even at the tail of the distribution. For example, HPCC has the 99th-percentile queue length of 65KB in TCP and 74KB in RDMA, which incurs only 21.3μs and 23.7μs queueing delay in a 25GE link. As the comparison, TCP NewReno's 99th-percentile queue size is 247KB and DCQCN's size is 519KB.

REFERENCES

[1] Tom Henderson, Sally Floyd, Andrei Gurtov, and Yoshifumi Nishida. The NewReno Modification to TCP's Fast Recovery Algorithm (RFC6582). (2012).

[2] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. HPCC: high precision congestion control. In *SIGCOMM, 2019.*

[3] Sivasankar Radhakrishnan, Yilong Geng, Vimalkumar Jeyakumar, Abdul Kabbani, George Porter, and Amin Vahdat. SENIC: Scalable NIC for End-Host Rate Limiting. In *NSDI, 2014.*

[4] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion Control for Large-Scale RDMA Deployments. In *SIGCOMM, 2015.*