RDMA Load Balancing via Data Partition

Yi Wang
School of Electronic Information and Communications
Huazhong University of Science and Technology
Wuhan, China, 430074
Email: ywang@hust.edu.cn
Phone: +86-027-87543236

Ya-nan Jiang
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China, 210023
Email: mf1733026@smail.nju.edu.cn
Phone: +86-25-89681372

Qiufang Ma
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China, 210023
Email: mg1633053@smail.nju.edu.cn
Phone: +86-25-89681372

Chen Tian
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China, 210023
Email: tianchen@nju.edu.cn
Phone: +86-25-89681372

Bo Bai
Future Network Theory Lab
Huawei
Hong Kong, China, 999077
Email: baibo8@huawei.com
Phone: +852-3547-2722

Gong Zhang
Future Network Theory Lab
Huawei
Hong Kong, China, 999077
Email: nicholas.zhang@huawei.com
Phone: +852-3547-2722

***Abstract*—With the development of data center networks, traditional TCP/IP cannot support the demand in data centers. Remote Direct Memory Access (RDMA) technology could improve the performance of DCN significantly because of high throughput and low latency. However, load balancing is a key issue in RDMA which has not been solved distribute. This paper will propose an algorithm to solve the load balance problem in RDMA on application layer without hardware changing. The main idea is to divide data to chunks and data chunks on multiple reachable paths for transmission. However, it is no trivial to find the optimal chunk size and the path number, some empirical values are found by varieties of experiments and tests. Moreover, the chunk allocation scheme also needs to consider the traffic condition in DCNs to find more free paths to transmit. We evaluate the algorithm in application layer with ns3 simulator. The experiment results show that with our algorithm the completion time can decrease 81.03% at most.**

***Index Terms*—Load Balance, RDMA, Data Center Networks.**

I. Introduction

With the rapid growth of online services and cloud computing, large-scale data centers (DCs) are being built around the world. High speed, scalable data center networks (DCNs) are needed to connect the server in one DC. However, traditional TCP/IP cannot meet the requirement of DC workloads because of high CPU overhead and latency. Remote Direct Memory Access (RDMA) supports the operations such as read and write directly to the computer memory without involving the kernel. With the characteristic of zero copy, RDMA greatly reduces the load and overhead of CPU which decreases the latency. Thus, it attracts much attention to deploy RDMA in data centers[1], [2], [3], [4], [5], [6], [7], [8].

The traditional design of RDMA did not consider the load balance issue in DCN. Load balancing is important to reduce congestion [9], [10], [11], [12], [13], [14], [15], [16]. Specifically, when two hosts transmit data by RDMA, it needs three-way handshake to establish connection at first [17]. The connection is built between the work queues of the host. When only one connection exists, all packets will carry exactly the same five-tuple between two hosts. When multiple connections exist, however, there are no change of the five-tuples in different connections. Therefore, no matter single connection or multiple connections, all the data in transmission will follow the same path. For a large amount of data transmission [18], this phenomenon can easily lead to imbalance in the network, which may cause congestion, and low utilization of other reachable network resources (as Figure 1).

In this paper, we consider the load balancing issue of RDMA. The objective is to combine RDMA with multi-path transmission [19]. This combination can further reduce the transmission time, and the possibility of congestion in the network, so as to improve the utilization of network resources [20] [10] [21] [22]. MP-RDMA [19] has implemented it by changing the RDMA NICs. Our target is to solve the problem in application layer, without IP and lower level involved. The proposed algorithm mainly contains two parts. First, we establish multiple connections in multiple work queues between the hosts and change the message of the five-tuple in different connections so that the data can be transmitted by multi-path. Second, we divide the data to be transmitted to chunks with proper size [23], [24], [25], which will be allocated to proper connections.

There are some challenges we need to solve. First, the driver of RDMA cards is encapsulated, it's not easy to change the five-tuple which generated by the driver. Second, how to choose the optimal chunk size of data and the number of paths is also non-trivial. Fine-grained partition will cause many other workloads such as request elements and complete elements. Coarse-grained partition may not make any sense of load balancing in different paths [26], [27]. Meanwhile, the number of paths also need a compromise in a similar way. Finally, how to detect the traffic and the extent of congestion

978-1-7281-1856-7/19/$31.00 ©2019 IEEE

also needs to be considered, as the traffic in the paths may impact the choice of data chunks [28], [29].

We realized the algorithm in ns3 simulator. We do some test to observe the impact on transmission complete time by increasing the data chunk size and the number of paths. Due to the mechanism of completion elements, the driver could detect if the request in the work queue complete or not. By estimating the rate of processing request in different queues, we could find the optimal parameters for data chunk. With the simulation, we find that using multiple paths to transmit data can decrease the transmission time by 81.03% at most. The decrease by chunk sizes could range from 13.74% to 69.71%.

In the rest of this paper, we introduce the related works in Section II. The process of proposing and optimizing in detail is presented in Section III. In Section IV, we describe the experiments and compare the performance. Section V concludes the paper.

II. Background

RoCEv2: RoCEv2 [30] [31] encapsulates the RDMA packet within an Ethernet/IPv4/UDP packet. Thus, RoCEv2 is compatible with most existing data center networking infrastructure. The UDP header in the packet is needed for ECMP-based multi-path routing (as shown below). The source UDP port can be randomly chosen for each queue pair (QP) where the destination port is always set to the specific value 4791. The intermediate switches always use standard five-tuple hashing. Thus, packets belonging to the same QPs will follow the same physical path for transmission, while different QPs (even between the same pair of two terminals) can follow different paths.

OFED: The OpenFabrics Enterprise Distribution (OFED) [32] / OpenFabrics Software is an open-source software for RDMA. OFS supports highly efficient networks, storage connectivity and parallel computing. Thus, it is widely used in business, research and scientific environments. The computing evolves applications that require extreme speeds, massive scalability and utility-class reliability. OFS includes kernel-level drivers, supports channel-oriented RDMA and send/receive operations and kernel bypasses. OFS also provides both kernel and user-level application programming interface and services for parallel message passing, sockets data exchange and so on.

ECMP: Equal-cost multi-path routing (ECMP) is a routing strategy which finds multiple ”best paths” for packets with the same destination. Since it is a per-hop decision in a single router, multi-path routing can be used in conjunction with most routing protocols. It can increase bandwidth substantially by load-balancing on multiple paths.

MPTCP: Multipath TCP[10] can open additional subflows to transport across multiple network paths. An additional subflow can use the same pair of src/dst IP addresses as the first flow with different ports, or use any IP addresses that the client or server may have. Based on ECMP routing, the subflows can be hashed to different paths. Each subflow also maintains a congestion window to control the packet sending and adapt to the conditions along the path.

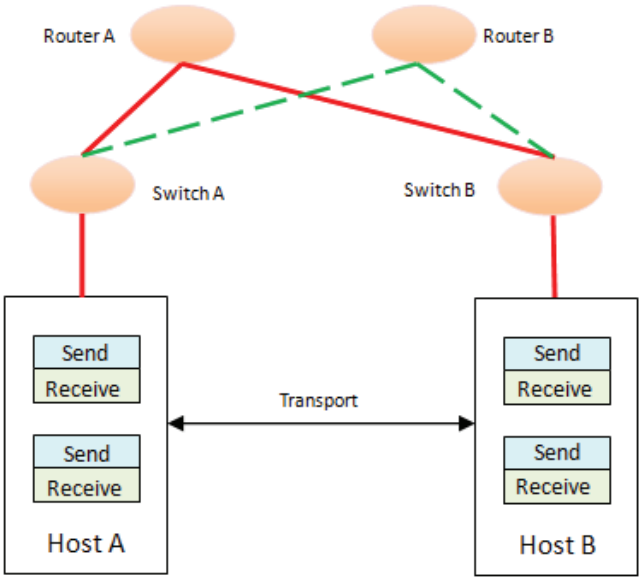

Fig. 1. The utilization of single transmission path.

MP-RDMA: Multi-path RDMA [19] is based on NIC hardware. It requires additional 66 Bytes on-chip memory to each connection state compared to single-path RDMA. It uses multi-path ACK-clocking mechanism and out-of-order aware path selection mechanism to choose network paths and distribute packets in a congestion-aware manner.

III. Design

A. Problem/Statement

The transmission tasks of RDMA will establish connections through the QPs in the terminal at first. It means that the data will be sent in turn from the QPs, while different QPs may be out-of-order. Ideally, the data transmission tasks between different QPs is parallel.

If only one connection between two terminals exists, all the data packets follow the only connection path to transmit. When multiple connections exist, the packets in different connections transmit through different physical paths. However, in our test, we found that all the data packets between two specific terminals carry the same five-tuple. It means no matter how many connections we built, how many paths are available, there is only one physical path been used.

There are some contents referring to the support for ECMP in RoCEV2[31]: It is stated that the UDP source port must be the same for a series of packets with sequential constraints. The packets without this constraint can carry different UDP source values.

Distributing data packets on multi-path is a way to do load balancing. As the other four items in five-tuple (source/destination IP, UDP destination port and the protocol) are specific, if the UDP source port is the same, all packets

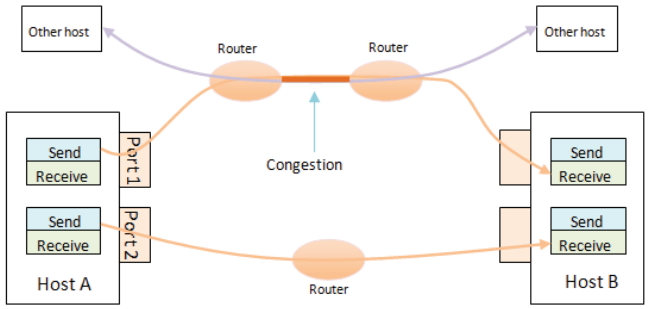

Fig. 2. The congestion caused by evenly allocation .

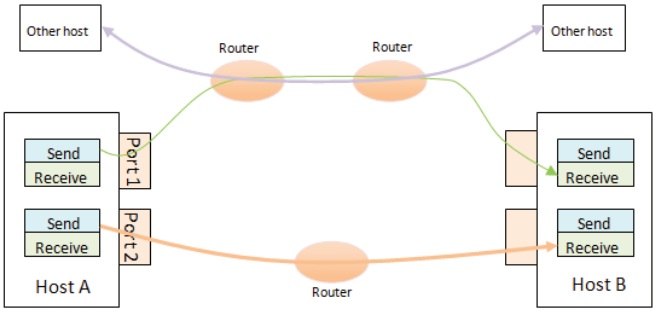

Fig. 3. The condition with dynamic allocation.

carry the same five-tuple. As a result, all the packets will follow the same path during the transmission tasks. If the packets' number is large enough, not only the load and congestion risk will be seriously increased, but also waste other available network resources(as shown in Figure 1). In order to be more balanced, we try to do some change on source port. Then, data packets in different connections can be transmitted independently.

B. Algorithm

We will try to change the UDP source port value as the other four items in five-tuple are specific. Recalling the RDMA transmission characteristics, we can bind different QPs with different UDP source ports. Therefore, the packets sent by the same QPs carry the same port value, while different QPs have different port values. In this way, the number of QPs is the maximum number of the possible physical paths.

While an application establishes multiple pairs of QPs for transmission, how to allocate the data packets is also an important issue. If we allocate the packets evenly to each QP, the network load should be equal in all used paths. However, the network resources is public, we need to take other devices into account. It may lead to congestion at a certain part in a transmission path. For example, one of an application's transmission path includes the part with heavy traffic, the rate on this path is slow while other paths are normal. Then, the path will cause too much delay to cumber the total transmission(as shown in Figure 2). Hence, the static allocation scheme on average is not reasonable.

To avoid the case above, we decided to adopt a dynamic approach. Since Infiniband generates a work completion message when completing a work request, the OFED driver provides an interface to detect the completion information, which allows the upper layer to get the completion status. Then, we assigned equal and small amount of data packets to each pair of QP for sending at the beginning of the transmission. While we detect a complete information, add the next send task to the corresponding QP until all the data sent completed. In this dynamic scheme, once a path delay increases, the number of packets allocated to this path will be reduced accordingly. As a result, the less under loaded paths will be more fully utilized, which can adapt to the dynamic changes in network resources(as shown in Figure 3).

When the data packets are assigned to different QPs, the original order of the data will be broken. If the receiver can not reorder the data correctly, the transmission is meaningless. For the complex IBA network layer, the change will be costly, and may cause a series of problems. Our target is to realize these functions in application layer without lower layer involved. Therefore, we put the data sequence relation into the Payload. i.e., as a normal data content to transmit.

In this context, we add several elements as custom header to the data chunks [**?**] after the split. The custom header mainly contains three elements: data ID, total data size, current chunk offset. Data ID is mainly used to distinguish different data transmission task at the same time to avoid confusion when the receiver reorders the data chunks. Total data size is used to prompt the receiver machine of the memory size should be allocated to process the data receiving and reordering operations. Current chunk offset is used to determine the location of the current chunk in the total data, which is useful for data combination. We then send customized data chunk to the corresponding QPs for transmission. After receiving the disordered data, the receiver collects customized header information, and allocates memory according to the value of the total data size. Then, it combines the data chunks according to the data ID and the current chunk offset.

There is one step before placing the send task of chunks to QPs. When the completion messages of one of the QPs is detected, the total data calls the split function, which will return a data chunk as the next sending task. The algorithm will be executed under the premise that the function returned correctly. It will continue to send the task until all the data chunks are sent completely. At the receiver terminals, all the QPs must keep active to receive data chunks before the data reception completed.

In algorithm design, we need to answer another question: how to appropriately determine the number of QPs and the size of data chunks? If the number of QPs is too small, the available physical paths may not be fully utilized. The algorithm cannot make full use of the network resources between the two machines. If the number of QPs is too large, even beyond the actual number of available physical paths, more than one pair of QP may share the same path. It will greatly increase the system overhead for maintaining QPs

Algorithm 1 The load balancing algorithm in client.

```
Get initial information;
Create QPs and register buffer, connect to server ;
for still have chunks not be transmitted do
    for all the QPs in client do
        Check the state of current QP;
        if last sending task complete in current QP then
            Call the function of data partition and get a chunk
            to send;
            send the new chunk;
        end if
    end for
end for
disconnect with server;
```

Algorithm 2 The load balancing algorithm in server.

```
Get initial information;
Create QPs and register buffer;
keep listenning and get request to accept;
for still have chunks not be accepted do
    for all the QPs in server do
        Check the state of current QP;
        if last receive task complete in current QP then
            ready for next receive task;
            Call the function of data reordering;
        end if
    end for
end for
disconnect with client;
```

status. Similarly, if the size of the data chunk is too small, QPs need to place request elements, process tasks, detect the completion messages and execute other operations frequently, which also cause more system overhead. If the chunk size is too large, the algorithm also cannot achieve a good utilization for the network resources as data packets cannot choose paths freely. Therefore, in the following experiments, we will focus on changing the value of these parameters for performance comparison to find the optimized value.

IV. Experiments

We test our algorithm in ns3 simulator [33] [34]. We designed three experiments [35], respectively, the impact of the topology is considered for choosing the parameters. We take use of the real world network topology [36], i.e., the most common local area network structure. The topology is tested from the small range, then expanded as much as possible within the limit of the server. In the first experiment, we established a topology with 20 terminal nodes (numbered 0 - 19) and 10 switch nodes (numbered 20 - 29). The topology is shown in Figure 4. Each switch in the second layer connected with four terminal nodes, and connected with all switches in the first layer. In this topology, there are several equivalent yet different paths between any two terminals to meet our requirement. The transmission rate we set is 40Gbps, with the delay 0.001ms, the transmission state is ideal, the error rate we set is 0.

We test the impact of the number of QPs. The number of QPs determines the maximun number of paths that packets can be transmitted through by. If it is too large, it will cause the redundancy situation and increase the overhead of the computer system. Thus, we test it from 1 to 16, while 1 means using single path to transmit. The sender is terminal No. 0, which sends 10MB messages to terminal No. 19 with 200 background streams added randomly.

The simulation results are shown in Figure 7. All the curves significantly decrease when the QPs number increases from 1 to 2. When the number of QPs is only one, single path transmission happens. The size of data chunk has little effect to the completion time, all in 20639592ns. When the number of QPs is 2, with the chunk size ranging from 1K to 32K, the completion time is similar, which is from 12530000 ns to 12650000 ns. However, when the chunk size is 64K, the completion time is 15009224ns, with 128K to 17816130 ns. All these results are better then the original test which does not use multi-path for transmission.

In Fig 7, if the data chunk size is large, such as 64K and 128K, with the number of QPs increasing larger than 2 the completion time is almost no oscillation. If the chunk size is slightly smaller, for example, set to 32K, 16K, 8K and 4K, the completion time will be significantly decreased when the number of QPs increased from 2 to 3, but the slope decreases with the increase of the chunk size. The completion time tends to be stable when we continue to increase the number of QPs. When we divide the data chunk into the limit size, such as 2K, 1K for transmission, it is interesting that the completion time oscillates negligibly at the size of 2K and tends stable when the number of QPs is larger than 5. For the size of 1K, the time continuously decreases, but the slope is decreased with the number of QPs increased.

The phenomenon above confirms that the large number of QPs cannot reduce the completion time. There are limited number of physical paths between any two terminals. Besides, some paths may contain many hops, which may increase the delay and congestion. Using the paths mentioned above is unwise. Taking the system overhead into account, choosing the number of QPs from 2 to 5 is more reasonable.

We then test the completion time with different chunk sizes when the number of QPs sets to 2, 3, 4 and 5. The results are shown in Figure 10. It can be seen that, no matter how to change the QPs number, with the chunk sizes increased, the completion time increased monotonically. Clearly, when the data chunk size is large, the difference of load scale between paths will be smaller. If congestion occurs, the transmission delay of the path will increase. As the remaining data packets in the chunk have no choice to transfer through other paths, it will still be waiting in this congestion path. When the chunk size is set to the smallest value, the choice of each packet for paths will be more flexible. When the same congestion occurs, because of its small data volume, although the remaining chunk packets should be followed, it will not cause much

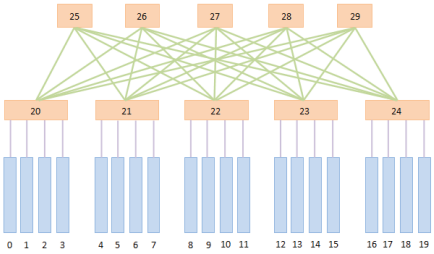

Fig. 4. The topology in the first test.

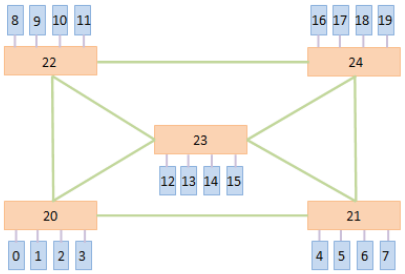

Fig. 5. The topology in the second test.

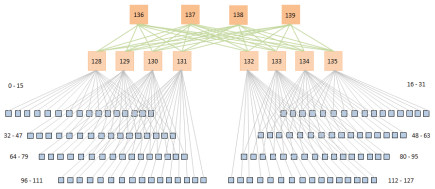

Fig. 6. The topology in the third test.

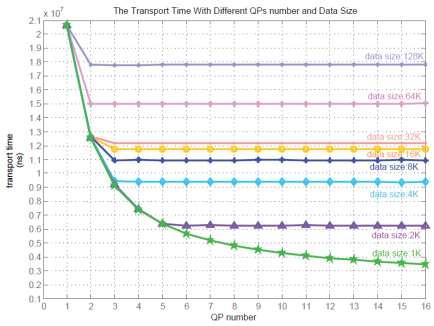

Fig. 7. The relationships between the QPs number and time in topology1.

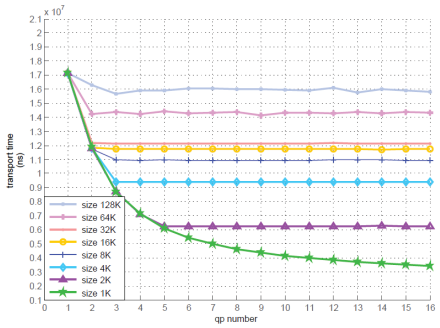

Fig. 8. The relationships between the QPs number and time in topology2.

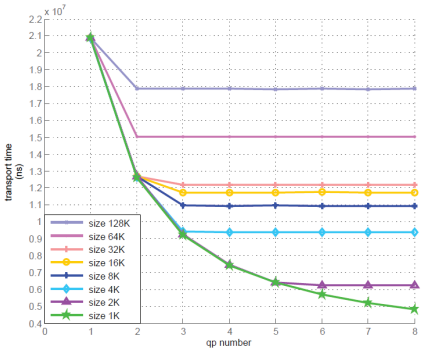

Fig. 9. The relationships between the QPs number and time in topology3.

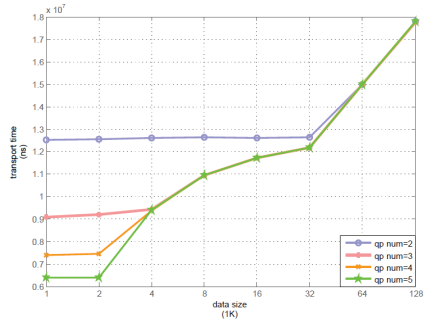

Fig. 10. The relationships between the data block size and time in topology1.

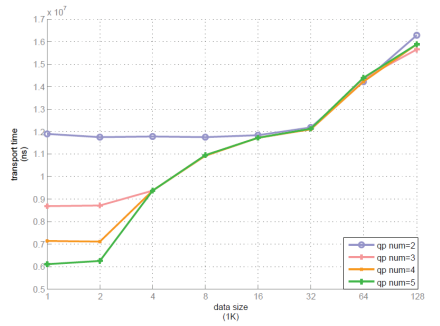

Fig. 11. The relationships between the data block size and time in topology2.

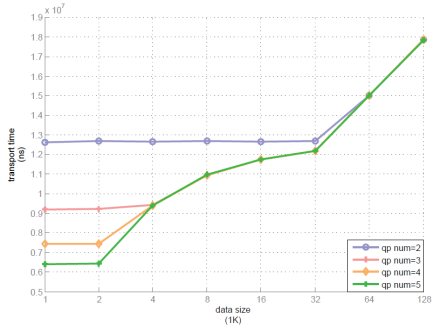

Fig. 12. The relationships between the data block size and time in topology3.

delay. Meanwhile, other data chunks will select the others to avoid the congestion path.

Our objective is to divide the large data to transmit by multi-path to improve the utilization of network resources and reduce the transmission delay. The simulations also show that the best results are achieved by splitting the data into 1K units. Indeed, 1K data packet can be transported flexibly, so that the idle network resources can be fully utilized. However, to support a fine-grained division, it will cause high overhead for the data split and reorganization in application layer. In addition, recalling the transmission mechanism of RDMA, sending packet needs to post a request element in the special QPs at first. Then, the QPs process the task, generate the complete information, and then call the completion notice to inform the upper applications. If the data is divided into 64K, 10M data needs to be carried out these operations 160 times. For the chunk to 1K, it needs 10240 times, which increases the cost of computer system sharply.

In Figure 10, when the number of QPs is 2, the chunk size in the range from 1K to 32K has little impact for the completion time. When the number of QPs is from 3 to 5, the chunk size changes from 2K to 32K, the completion time increases slowly. The common feature of the four curves is that a significant increase of completion time happens when the chunk size larger than 32K. Therefore, 32K is threshold.

In the following, we design the second experiment to test whether the topology will make a difference. As shown in Figure 5, the topology contains 20 terminal machines,

numbered from 0 to 19, and 5 switches, numbered from 20 to 24.

In the first experiment, each terminal is connected to only one switch. The switch node is connected to several upper routing nodes. In this structure, if the terminals' transmission tasks do not coincide, there is no resource preemption. However, in the second topology, each router is connected with the terminal and also directly connected with other routers. This topology may not reduce the length of the transmission paths, but it will cause congestion and resource preemption under the same background.

We still send 10MB data from terminal No. 0 to terminal No. 19. The transmission rate, delay, error rate and 200 background streams are the similar as the first experiment. The effect of the number of QPs and chunk size on the completion time is shown in Figure 8 and Figure 11, respectively.

The trend of the curves in the two figures under the new topology is the same as them in the first experiment. With the absence of multi-path, the total transmission time of 10MB data is 17772320 ns, which is lower about 3000000 ns than the initial time of the first experiment. In Figure 8, when the chunk size is 128K, the curve is stable when the number of QPs is larger than 2 in the first topology. In the second experiment, the value is larger than 3 which achieves the best results into a steady state, but the oscillation is more obvious than the first one. Its range is about 180000 ns, which is nearly 10 times of the first. Similarly, the oscillation of 64K curve is larger than the first 150000 ns. When the chunk size is 16K or 32K, there is a small decrease when the number of QPs increases from 2 to 3 in the first experiment. In the second experiment, however, there is almost no decrease which trends to a steady state ahead.

Although the second experiment shows the similar results as the first, two experiments are carried out in a small topology, which is not much different between the two environments. Thus, the parameters we previously obtained can only be used for similar circumstances. In the second experiment, we made slightly changes on the topology structure, the transition points of the curves showed some differences. We have to test the impact of the number of QPs and the data chunk size when the topology is further expanded with more complicated structures.

We design the third experiment to expand the scale of the topology [36] to determine the range of parameters. As shown in Figure 6, this topology contains 128 terminal machines, numbered from 0 to 127, 12 routers, numbered from 128 to 139. Each of the second level routers connects to 16 terminals and all the first level routers. This topology is identical to the structure of the first topology with much larger scale. The communication still happens between nodes No. 0 and No. 19. 200 background flows settings remain unchanged. However, in such a large topology, 200 background flows may have little impact on the transmission time of the testing data, which should not cause the curves oscillating.

The results of the third experiment are shown in Figure 9 and Figure 12. Although we narrow the range of the number of QPs, the trend of the parametric curves are similar to those of the first half in the first experiment, including the variation of the curves. There is only a little increase in the specific transmission time with about 200000 ns. Thus, the scale of the topology does not affect the range of parameters in a certain extent.

V. Related Work

Recent years, a lot of researches about RDMA performance improving have been done[37], [38], [39], [40], including multi-path transport and packet loss handling. MP-RDMA [19] is based on NIC to deploy traffic to multiple paths. It uses multi-path ACK-clocking mechanism to distribute packets according to congestions, and uses bitmap to track the out-of-order packets. MP-RDMA also actively selects fast paths with similar delay and prune slow paths to improve the overall performance. This work has similar idea with us, the difference is that MP-RDMA mainly focus on improving NIC hardware, and our work only changes applications.

IRN [2] also makes some changes to RoCE NICs. It uses selective retransmission mechanism and BDP-FC mechanism which bounds the number of in flight packets to handle the problem of packet loss. RaaS [1] improves the scalability of RDMA and CPU/memory utilization, whose prototype RDMAvisor can achieve high throughput for thousand of connections with low CPU and memory overhead.

VI. Conclusion

This paper gave an algorithm to solve the load balancing issue in RDMA from application layer. The main idea is to bind different QPs with different UDP source ports so that the packets sent by different QPs carry different five-tuple. Then, we divide the total data into chunks with proper size which is allocated to different QPs according to the congestion conditions of the corresponding paths. In addition to the multi-path routing algorithm, the data chunks can be transmitted by multi-path to improve the utilization of network resources and speeding up the data transmission rate. Through the simulation experiments, we show that the algorithm can decrease the transmission time by 81.03%. Therefore, the combination of multi-path routing algorithm and RDMA solve the load balancing issue effectively.

Acknowledgment

The authors would like to thank anonymous reviewers for their valuable comments. This research is supported by the National Key R&D Program of China 2018YFB1003505, the National Natural Science Foundation of China under Grant Numbers 61602194, 61772265, and 61802172, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.

REFERENCES

[1] Z. Wang, X. Wang, Z. Qian, B. Ye, and S. Lu, "Rdmavisor: Toward deploying scalable and simple RDMA as a service in datacenters," *CoRR*, vol. abs/1802.01870, 2018.

[2] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for RDMA," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*, 2018, pp. 313–326.

[3] J. Yang, J. Izraelevitz, and S. Swanson, "Orion: A distributed file system for non-volatile main memory and rdma-capable networks," in *17th USENIX Conference on File and Storage Technologies, FAST.*, 2019, pp. 221–234.

[4] D. Kim, T. Yu, H. H. Liu, Y. Zhu, J. Padhye, S. Raindel, C. Guo, V. Sekar, and S. Seshan, "Freeflow: Software-based virtual RDMA networking for containerized clouds," in *16th USENIX Symposium on Networked Systems Design and Implementation, NSDI, Boston.*, 2019, pp. 113–126.

[5] H. Li, T. Chen, and W. Xu, "Improving spark performance with zero-copy buffer management and RDMA," in *IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2016, San Francisco, CA, USA, April 10-14, 2016*, pp. 33–38.

[6] A. Kalia, M. Kaminsky, and D. G. Andersen, "Using RDMA efficiently for key-value services," in *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, pp. 295–306.

[7] Q. Cai, W. Guo, H. Zhang, D. Agrawal, G. Chen, B. C. Ooi, K. Tan, Y. M. Teo, and S. Wang, "Efficient distributed memory management with RDMA and caching," *PVLDB*, vol. 11, no. 11, pp. 1604–1617, 2018.

[8] D. Y. Yoon, M. Chowdhury, and B. Mozafari, "Distributed lock management with RDMA: decentralization without starvation," in *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pp. 1571–1586.

[9] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," in *ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 465–478.

[10] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *ACM SIGCOMM 2011 Conference*, 2011, pp. 266–277.

[11] S. Ghorbani, Z. Yang, P. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 225–238.

[12] D. Jiang, P. Zhang, Z. Lv, and H. Song, "Energy-efficient multi-constraint routing algorithm with load balancing for smart city applications," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1437–1447, 2016.

[13] Y. Gao, Y. Yang, T. Chen, J. Zheng, B. Mao, and G. Chen, "DCQCN+: taming large-scale incast congestion in RDMA over ethernet networks," in *2018 IEEE 26th International Conference on Network Protocols, ICNP 2018, Cambridge, UK, September 25-27, 2018*, 2018, pp. 110–120.

[14] Y. Pan, C. Tian, J. Zheng, G. Zhang, H. Susanto, B. Bai, and G. Chen, "Support ECN in multi-queue datacenter networks via per-port marking with selective blindness," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018, pp. 33–42.

[15] J. Zheng, B. Li, C. Tian, K. Foerster, S. Schmid, G. Chen, and J. Wux, "Scheduling congestion-free updates of multiple flows with chronicle in timed sdns," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018, pp. 12–21.

[16] D. Jiang, P. Zhang, Z. Lv, and H. Song, "Energy-efficient multi-constraint routing algorithm with load balancing for smart city applications," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1437–1447, 2016.

[17] I. T. Association, "Infiniband architecture specification volume 1 release 1," *http://www.infinibandta.org/content/*, 2015.

[18] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-datacenter transfers," in *Proceedings of the Tenth European Conference on Computer Systems, EuroSys 2015, Bordeaux, France, April 21-24, 2015*, 2015, pp. 20:1–20:14.

[19] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and T. Moscibroda, "Multi-path transport for RDMA in datacenters," in *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, 2018, pp. 357–371.

[20] C. Guo, H. Wu, G. Soni, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in *Conference on ACM SIGCOMM 2016 Conference*, 2016, pp. 202–215.

[21] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp," in *Usenix Conference on Networked Systems Design and Implementation*, 2011, pp. 99–112.

[22] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath tcp," in *Usenix Conference on Networked Systems Design and Implementation*, 2012, pp. 29–29.

[23] C. Tian, J. Yan, A. X. Liu, Y. Tang, Y. Zhong, and Z. Li, "Macroflow: A fine-grained networking abstraction for job completion time oriented scheduling in datacenters," in *24th IEEE International Conference on Network Protocols, ICNP 2016, Singapore, November 8-11, 2016*, 2016, pp. 1–2.

[24] B. Tian, C. Tian, J. Sun, J. Yan, Y. Tang, W. Wang, H. Dai, N. Xia, G. Chen, and W. Dou, "Using the macroflow abstraction to minimize machine slot-time spent on networking in hadoop," in *Proceedings of the 2nd Asia-Pacific Workshop on Networking, APNet 2018, Beijing, China, August 02-03, 2018*, 2018, pp. 36–42.

[25] B. Tian, C. Tian, H. Dai, and B. Wang, "Scheduling cofiows of multi-stage jobs to minimize the total weighted job completion time," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 864–872.

[26] C. Tian, A. Munir, A. X. Liu, Y. Liu, Y. Li, J. Sun, F. Zhang, and G. Zhang, "Multi-tenant multi-objective bandwidth allocation in datacenters using stacked congestion control," in *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*, 2017, pp. 1–9.

[27] S. Hu, W. Bai, K. Chen, C. Tian, Y. Zhang, and H. Wu, "Providing bandwidth guarantees, work conservation and low latency simultaneously in the cloud," in *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, 2016, pp. 1–9.

[28] W. Bai, K. Chen, H. Wang, L. Chen, D. Han, and C. Tian, "Information-agnostic flow scheduling for commodity data centers," in *12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 15, Oakland, CA, USA, May 4-6, 2015*, 2015, pp. 455–468.

[29] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "PIAS: practical information-agnostic flow scheduling for commodity data centers," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 1954–1967, 2017.

[30] I. T. Association, "Annex a 16: Roce," *http://www.infinibandta.org/content/*, 2010.

[31] ——, "Annex a 17: Rocev2," *http://www.infinibandta.org/content/*, 2014.

[32] Mellanox, "Mellanox ofed for linux user manual rev 4.0 software version 4.0," *http://www.mellanox.com/related-docs/prod_software/Mellanox_OFED_Linux_Release_Notes_4_0-2_0_0_1.pdf*, 2017.

[33] Bobzhuyb, "ns3-rdma," *https://github.com/bobzhuyb/ns3-rdma/*, 2016.

[34] Ns-3, "ns-3 tutorial release ns-3.26," *https://www.nsnam.org/releases/*, 2016.

[35] S. Ma, J. Kim, and S. Moon, "Exploring low-latency interconnect for scaling out software routers," in *IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era*, 2016, pp. 9–15.

[36] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, and P. Germano, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *Communications of the Acm*, vol. 45, no. 4, pp. 183–197, 2015.

[37] H. Qiu, X. Wang, T. Jin, Z. Qian, B. Ye, B. Tang, W. Li, and S. Lu, "Toward effective and fair RDMA resource sharing," in *Proceedings of the 2nd Asia-Pacific Workshop on Networking, APNet 2018, Beijing, China, August 02-03*, 2018, pp. 8–14.

[38] J. Xue, M. U. Chaudhry, B. Vamanan, T. N. Vijaykumar, and M. Thottethodi, "Fast congestion control in rdma-based datacenter networks," in

Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos, Budapest, Hungary, August 20-25, 2018, pp. 24–26.

[39] M. Miao, F. Ren, X. Luo, J. Xie, Q. Meng, and W. Cheng, "Softrdma: Rekindling high performance software RDMA over commodity ethernet," in *Proceedings of the First Asia-Pacific Workshop on Networking, APNet 2017, Hong Kong, China, August 3-4, 2017*, pp. 43–49.

[40] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*, pp. 523–536.