# Analyzing Performance Pitfalls of On-Demand Paging of InfiniBand

Takuya Fukuoka[1,a)] Shigeyuki Sato[1,b)] Kenjiro Taura[1,c)]

***Abstract:*** For modern supercomputers, InfiniBand, a high-performance interconnect, is increasingly important because it offers remote direct memory access (RDMA) technology, which enables low-latency communication based on kernel bypassing. Because of the nature of kernel bypassing, conventional RDMA technology, however, requires memory registration to pin down communication buffers to physical memory before issuing communication operations, which is highly problematic both in the efficiency and productivity of memory management. Mellanox addressed this issue with On-Demand Paging (ODP), which is a feature to page-in/-out communication buffers automatically depending on the situation. Although ODP was said to take an acceptable cost, in this work, we reveal that ODP has awful performance pitfalls. We identify that the packet loss and timeout occur in simple conditions and incurs a large penalty on latency. We provide a detailed analysis of this phenomenon and explore ways of improving it.

***Keywords:*** InfiniBand, On-Demand Paging, Packet loss, Reliable Connection

## 1. Introduction

Utilizing multiple nodes is critical with the growing size of the computation, especially in High-Performance Computing (HPC) systems. TOP500[*1] lists the most powerful computing systems in the world and those ranked are mostly composed of multiple nodes. In these systems, an interconnect which connects each nodes plays a important role because the performance depends on more and more communication rather than computation [3]. If the interconnection network becomes to perform better, it should be more possible to improve the overall system performance.

Remote direct memory access (RDMA) is a popular technology employed in interconnects for its ultra-low latency and its high bandwidth. RDMA achieves the benefits by enabling communication without involving remote CPUs and directly transferring the memory between user-level buffer and NICs. These characteristics are called kernel-bypassing and zero-copy transfer, which make it attractive for many distributed systems [2, 4, 11–13, 21, 25, 33, 34, 38]. RDMA is currently supported over a high-performance network such as InfiniBand as well as commodity Ethernet via RoCE and iWARP [7, 26, 40].

While RDMA decreases latency and network bandwidth, there is one big challenge that should be addressed. Because of the nature of kernel bypassing, conventional RDMA technology requires memory registration to pin down communication buffers to physical memory to avoid swapping it out and make it accessible from remote CPUs. It is reported that, however, memory registration is a quite expensive operation with non-negligible latency and CPU load, and registering memory every time communication occurs degrades performance [5, 24]. Thus, effective management of communication buffers is needed to get the most of the performance of RDMA. A lot of research have been conducted about how to manage registration buffers so far [1, 5, 18, 28, 29, 32, 35, 37, 39], and one common approach is employing pin-down cache to reuse the pin-down area to decrease the invocation of registration primitives.

However, the pin-down cache scheme is highly problematic in the efficiency and productivity of memory management. First, it still requires to pin down some communication buffers even with some kinds of optimizations and limits the memory available for computation because the pinned buffer is never swapped out. As the number of the cores in one chip increases, the total memory required for communication is increasing but the total size of memory is still constant, which makes overlapping communication and computation difficult [18]. Second, a complex mechanism for reusing pin-down caches lead to complicated programming model and cause less productivity for RDMA-based systems [17].

Recently, Mellanox has addressed these issues by introducing On-Demand Paging (ODP) [17, 19, 20, 23] into InfiniBand. ODP is a feature to page-in/-out communication buffers automatically depending on the situation. It frees developers from manual memory registrations and management of pin-down caches and achieves automatic memory management. With ODP, when the required memory is not mapped into the physical memory, Host Channel Adapter (HCA) asks the OS to cause page faults and provide the physical memory and its mapping. ODP also allows the communication buffers to be swapped out without deregistration, which can ensure enough memory for computation and, therefore, achieves good communication-computation overlap. Fur-

[1] The University of Tokyo
[a)] fukuoka@eidos.ic.i.u-tokyo.ac.jp
[b)] sato.shigeyuki@mi.u-tokyo.ac.jp
[c)] tau@eidos.ic.i.u-tokyo.ac.jp
[*1] https://www.top500.org/lists/

thermore, the experimental reports [17, 20] said that it took an acceptable cost of only several hundred of microseconds on page faults. Therefore, the ODP feature seems to be a promising approach to taking both performance and productivity in IB networks.

On the contrary to the existing reports, in this work, we reveal that ODP has awful performance pitfalls. First, we experimentally demonstrate that, even in a simple condition with a micro-benchmark, ODP can perform badly and, surprisingly enough, cause several hundred *milliseconds* of latency. We also provide a detailed analysis of this phenomenon with the help of ibdump, a packet capturing tool for InfiniBand and then, identify that the root cause is packet drops and subsequent timeout with sufficient proof. Finally, we explore how to cope with the performance pitfalls through experiments and observations from various angles. Because ODP is currently employed in production-grade communication software such as MVAPICH2-X[*2], UCX [30], and libfabric [6] with several microseconds of peer-to-peer latency, tackling these pitfalls should significantly influence people who require high performance communication.

This paper is organized as follows. Section 2 provides the basic knowledge of InfiniBand and illustrates how badly the timeout mechanism works with the measurement of a variety of InfiniBand adapter cards. Section 3 describes how ODP is implemented using Reliable Connection (RC) transport and depicts the behavior of RDMA READ operations based on the packets captured with ibdump. Section 4 provides experiments in a simple condition and show the performance pitfalls first. We also analyze in what condition the performance degradation occurs precisely. In Section 5, we explore ways to reduce the impact of the pitfalls from the insight gained from the previous section. Section 6 reviews the existing work of performance evaluation of ODP, memory registration methods, and reliability of RC which support ODP. Finally, Section 7 presents the conclusion and future work.

## 2. InfiniBand

InfiniBand is a popular switched interconnection standard widely used in high-performance computing systems [31]. With InfiniBand, computing nodes are connected through the switched fabric and channel adapters. InfiniBand verbs are provided as an interface for the host operating systems to communicate with host channel adapters (HCAs).

InfiniBand supports remote direct memory access (RDMA) technology and achieves ultra-low latency and high bandwidth. RDMA enjoys these benefits by enabling communication without involving remote CPUs and directly transferring the data between user-level buffers and NICs. These characteristics are called kernel-bypassing and zero-copy transfer, and they are supported through address translation tables in NICs. Since users specify a virtual address when posting communication, NICs should be able to translate from a virtual address to a physical addres. For this mechanism to work well, you need to make sure that the physical addresses of communication buffers always exist during the communication.

Memory registration is a process to pin down communication buffers and it should be done before issuing communications operations. It proceeds with following three steps: (1) Checking whether the physical address of a target communication buffer exists and if not, a new page is allocated. (2) Pinning down the buffer to the physical memory to avoid swapping it out and make it always accessible from remote CPUs. (3) Updating the translation table in NICs. Memory deregistration is the opposite operation and it unpins the buffer and removes the translation entry in NICs to free resources. These operations are expensive operations with a latency of tens of microseconds and the bandwidth would increase by 3.3 times without them [5, 24].

### 2.1 Memory Semantics and Channel Semantics

There are two kinds of communication semantics in InfiniBand, channel semantics and memory semantics. Channel semantics includes SEND and RECEIVE operations where both sender and receiver should involve in the communication. On the other hand, memory semantics includes one-sided RDMA READ and WRITE operation to the remote memory, where the remote side is not aware of the communication. Regardless of the semantics you use, all memory buffers for communication should be registered beforehand.

### 2.2 QP Model

The verbs interface provides Queue Pair (QP) and Completion Queue (CQ) models. QP is composed of two sets of queues: Send Queue (SQ) and Receive Queue (RQ). When issuing communication, an application posts a Work Request (WR) to the corresponding queue through verbs and it is managed as Work Queue Element (WQE). When the WR is processed by HCA and it completes, Completion Queue Entry (CQE) appears in CQ which is associated with the QP to notify the application. Even if the operation fails, CQE appears in the CQ, and in this case, CQE includes the corresponding error code.

### 2.3 Transport Types

InfiniBand fabric offers four types of transport modes: Reliable Connection (RC), Reliable Datagram (RD), Unreliable Connection (UC), and Unreliable Datagram (CD), among which RC and UD are commonly used. Reliable transport assures the transmission of packets and conducts retransmission if an error occurs. While connection-oriented transport requires the establishment of the connection between two end-nodes, datagram transport enables for one end-node to communicate with the other end-node without a one-to-one connection. Depending on the transport type you use, the number of QPs required in each end-nodes differs. While RC needs $O(N^2)$ QPs for each connection where N represents the number of end-nodes, UD only requires $O(N)$ QPs because it does not create connections.

### 2.4 Retransmission Handling in RC

To assures reliable transport, RC supports detecting packet loss by implementing the timeout of requests. When QP detects packet loss, it retransmits the packet and assures the transmission.

[*2] https://mvapich.cse.ohio-state.edu/

**Table 1** InfiniBand systems and details on their HCAs, where Reedbush-H/L, ABCI, and ITO are supercomputing systems.

| System name | PSID | Model name | Driver version | Firmware version | Lower limit of $C_{\mathrm{ACK}}$ | Minimum $T_o$ [ms] |
|---|---|---|---|---|---|---|
| Private servers A | MT_1100120019 | ConnectX-3 56Gbps FDR | 5.0-2.1.8.0 | 2.42.5000 | 16 | 530 |
| Private servers B | MT_2170111021 | ConnectX-4 56Gbps FDR | 5.0-2.1.8.0 | 12.27.1016 | 16 | 517 |
| Reedbush-H[*3] | MT_2160110021 | ConnectX-4 56Gbps FDR | 4.5-0.1.0 | 12.24.1000 | 16 | 518 |
| Reedbush-L[*3] | MT_2180110032 | ConnectX-4 100Gbps EDR | 4.5-0.1.0 | 12.24.1000 | 16 | 508 |
| ABCI[*4] | MT_0000000095 | ConnectX-4 100Gbps EDR | 4.4-1.0.0 | 12.21.1000 | 16 | 505 |
| ITO[*5] | FJT2180110032 | ConnectX-4 100Gbps EDR | 4.4-1.0.0 | 12.23.1020 | 16 | 508 |
| Azure VM HCr Series | MT_0000000010 | ConnectX-5 100Gbps EDR | 4.7-3.2.9 | 16.26.0206 | 12 | 32 |
| Azure VM HBv2 Series | MT_0000000223 | ConnectX-6 200Gbps HDR | 5.0-2.1.8.0 | 20.26.6200 | 16 | 508 |

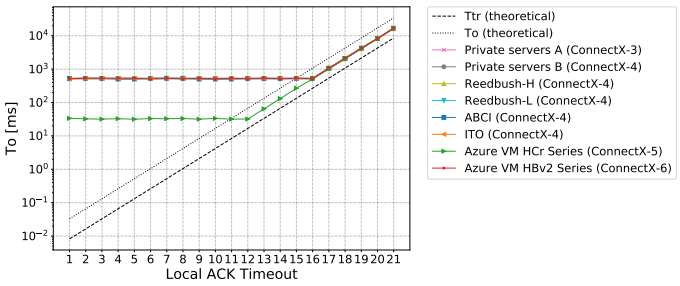


**Fig. 1** Measurement of timeout condition $T_o$ of a variety of InfiniBand adapter cards varying $C_{\mathrm{ACK}}$. The tilts of the graphs change at the point of the practical lower limit of $C_{\mathrm{ACK}}$. Only Azure VM HCr Series (ConnectX-5) has relatively smaller lower limit of 12. The shapes of all the other systems are almost the same and the lower limit is 16.

You can control configurations of the timeout by changing two parameters of QPs: Local ACK Timeout $C_{\mathrm{ACK}}$ and Retry Count $C_{\mathrm{Retry}}$. $C_{\mathrm{ACK}}$ is a 5-bits counter to define the timeout interval $T_{tr}$ with zero meaning that the timer is disabled and $T_{tr}$ is calculated as $4.096 \cdot 2^{C_{\mathrm{ACK}}}$ µs. The period during which each QP detects the timeout condition is called $T_o$ and defined as $T_{tr} \leq T_o \leq 4T_{tr}$. The other parameter, $C_{\mathrm{Retry}}$ represents the maximum number of retransmission and the process aborts with `IBV_WC_RETRY_EXC_ERR` when the count of retransmission exceeds the value.

You might think you can minimize $T_{tr}$ to 8.192 µs with $C_{\mathrm{ACK}} = 1$, but it does not work in this way. That is because the minimum acceptable value of $C_{\mathrm{ACK}}$ shall be defined by the CA vendor according to the specification [9]. If you set the value less than the lower limit to $C_{\mathrm{ACK}}$, $T_{tr}$ is calculated with the lower limit in instead. The specification also mentions that timing outstanding requests with a high degree of precision is not possible or desirable because of many reasons. Indeed, few web articles complain about the long delay to detect the anomaly of InfiniBand [8, 27]. It would not be surprising that the lower limit of $C_{\mathrm{ACK}}$ and $T_{tr}$ is moderately high.

To confirm this hypothesis, we measured actual $T_o$ with a variety of InfiniBand adapters as it practically represents the cost of the timeout and retransmission. In this experiment, we gave a wrong destination LID to QP in the initialization phase so that we deliberately caused packet loss while with no QP error. We set $C_{\mathrm{Retry}} = 7$ and varying $C_{\mathrm{ACK}}$ and measured the time from the issue of the first request to the abortion of the process with `IBV_WC_RETRY_EXC_ERR`. $T_o$ was calculated as the measurement time divided by $C_{\mathrm{Retry}} + 1 = 8$.

**Table 1** describes the detailed information of the machines and InfiniBand adapters used in this measurement and **Fig. 1** shows the result. It shows in seven adapters, the lower limit of $C_{\mathrm{ACK}}$ is 16 and minimal $T_o$ is around 500 ms. Only one ConnectX-5 adapter has relatively small lower limit of 12 and minimum $T_o$ is around 30 ms. Given that round trip latency of InfiniBand is usually several µs, there is no doubt that the timeout can horribly degrade performance in both cases. The notable points of the result are that the high cost of timeout is ubiquitous in major supercomputing systems and it remains even in the latest Mellanox card, ConnectX-6 200Gbps HDR.

We would like to assure you that the occurrence of the timeout is thought to be very rare with general usage of InfiniBand for the first place and it usually dose not lead to any major problems. That is because InfiniBand employs lossless link-level flow control and therefore packets are never lost due to buffer overflows in switches, a principle cause of transport-layer retransmissions [22]. Koop at al. [16] reported network packet loss seldom appeared with UD on a 4096-core system. Some systems [11, 13] even chose the design sacrificing the reliability for fast common case performance at the cost of rare retries.

## 3. On-Demand Paging

Mellanox introduced On-Demand Paging (ODP) to InfiniBand recently to ease memory management for RDMA operations. ODP is a feature to page-in/-out communication buffers automatically depending on the situation. It frees users from manual memory registrations and management of pin-down caches, which makes programming simpler. It also allows the communication buffers to be swapped out without deregistration, which can ensure enough memory for computation and, therefore, achieves good communication-computation overlap. Currently, ODP is employed in production-grade communication software such as MVAPICH2-X, UCX, and libfabric with several microseconds of peer-to-peer latency.

ODP can be divided into two kinds of categories: Explicit ODP and Implicit ODP. Explicit ODP specifies the memory regions used for ODPs while Implicit ODP allows the whole address space to be used for ODP.

### 3.1 Basic Implementation

ODP is implemented with the help of network page fault support for InfiniBand [17]. When HCA tried to access the required memory, it checks whether the requested virtual memory is mapped into the physical memory. If not, it asks the driver to cause an interrupt to query the OS. Then, the OS searches the

[*3] `https://www.cc.u-tokyo.ac.jp/en/supercomputer/reedbush/service/`
[*4] `https://abci.ai/`
[*5] `https://www.cc.kyushu-u.ac.jp/scp/eng/`

```
Time [s]
0.000000    RDMA Read Request
0.000141    Acknowledge (RNR NAK)
0.000538    RDMA Read Response
0.004565    RDMA Read Request
0.004639    RDMA Read Response
```

```
Time [s]
0.000000    RDMA Read Request
0.000005    RDMA Read Response
0.000527    RDMA Read Request
0.000620    RDMA Read Response
```

```
Time [s]
0.000000    RDMA Read Request
0.000215    Acknowledge (RNR NAK)
0.000533    RDMA Read Response
0.004047    RDMA Read Request
0.004163    RDMA Read Response
0.004747    RDMA Read Request
0.004812    RDMA Read Response
```

**Fig. 2** Packets obtained with ibdump while handling one RDMA READ operation with remote, local, and remote-local ODP

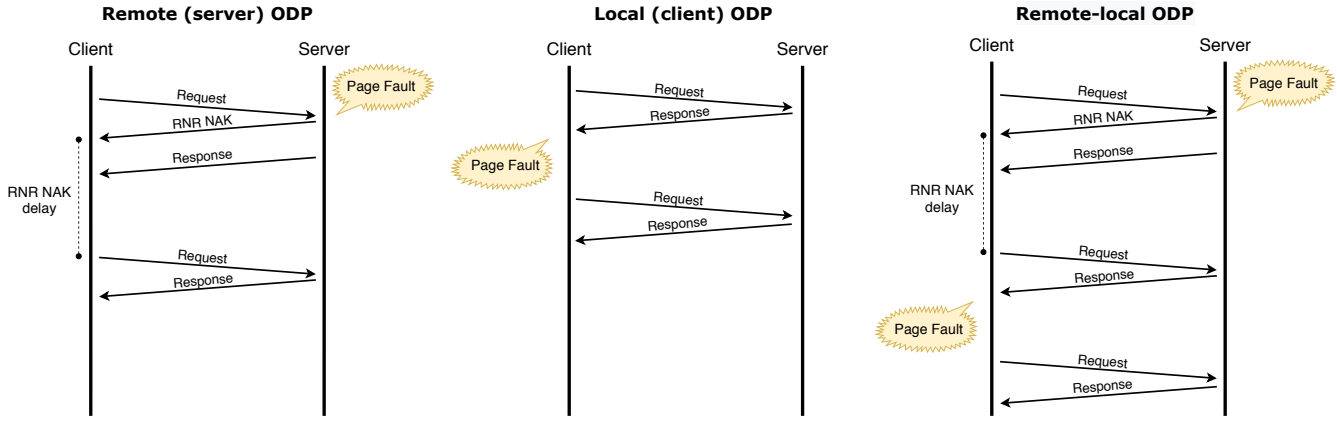


**Fig. 3** Illustration of how ODP worked with one RDMA READ operation based on the packets described in Fig. 2.

physical address, and if needed, allocates pages or retrieves them from secondary storage. Finally, the information is passed to the HCA through the driver, and virtual-physical memory mapping of HCA is updated.

When the OS wants to release pages that have been registered by the HCA, it should invalidate the page before that. Pages invalidation works oppositely as the page faults. When this invalidation event happens, the OS notifies the driver of the invalidation and the driver. Then, the driver looks for the page mapping which should be invalidated and tell HCA to flush the hardware caches. Finally, the completion of invalidation is notified to the OS through the driver.

### 3.2 Utilizing RC Support

However, this flow does not work practically for page faults on the receiver side. This limitation comes from the small memory of HCA, which makes it difficult for the HCA on the receiver side to keep received packets until the resolution of page faults. In order to cope with this problem, ODP relies on retransmission support in RC service. When the receiver encounters a page fault, it sends back RNR NAK (Receiver-Not-Ready Negative-Acknowledgement) packet to ask the sender to retransmit the original packet after a while. The minimal delay period can be designated by a software parameter and we call it as minimal RNR NAK delay in this paper. This solution works fine with Send/Receive operations, but RDMA operations have more to be considered because RNR NAK cannot be issued in some situations. One such case is page faults on the side of the initiator of a remote read request. This leads to packet drops until the page fault resolved, but we can solve the problem by starting the communication all over again from the beginning. Even if some data could be dropped for other reasons, RC should assure the transmission of packets because retransmission is executed in any case after waiting for at least a period specified as $C_{\mathrm{ACK}}$ of RC.

To understand the behavior of ODP fully, we observed one RDMA READ operation with three conditions: remote (server) ODP, local (client) ODP, and remote-local ODP. The experimental environment is the same as the one described in Section 4 and we set 1.28 ms as minimal RNR NAK delay. Figure 2 shows the InfinIBand packets captured with ibdump, and **Fig. 3** illustrate what was happening based on the information of page fault count on both sides. In the case of remote ODP, RNR NAK was sent back to the client in the face of page faults through the server need not keep the information in the packet. It took about 4.6 ms for the communication to complete and most of the time was consumed by RNR NAK delay. It should be noted that actual RNR NAK delay and minimal RNR NAK delay can be different. In this case, actual RNR NAK delay was about 3 times longer than minimal RNR NAK delay but it was not strange. In the case of local ODP, the local page fault was handled by retransmission of the request as we mentioned earlier. The overall time was nearly equal to the time for resolving the local page fault, which was about 500 ms. In the case of local-remote ODP, remote ODP and local ODP were handled in sequence and it took three round trips.

## 4. Experiments

In this section, we conduct experiments to explore the conditions in which ODP degrades performance. We find the performance pitfalls of ODP and provide the proof with a simple micro-benchmark. We also analyze the root cause by dumping InfiniBand traffic which flows to and from Mellanox adapter cards by ibdump and discuss the ways to avoid it.

```
size = 100;
local_buf = memalign(4096, size * num_ops);

init(qp, local_buf, remote_buf, ...);

for (i = 0; i < num_ops; i++) {
  local  = &local_buf[size * i];
  remote = &remote_buf[size * i];
  post_rdma_read(qp, size, local, remote);
  usleep(interval);
}
wait();
```

**Fig. 4** Our micro-benchmark in simplified C code, which allows us to specify `num_ops` as a number of RDMA READ operations and `interval` as time intervals between communications. The `wait` function polls the CQ to check whether all communications have finished.

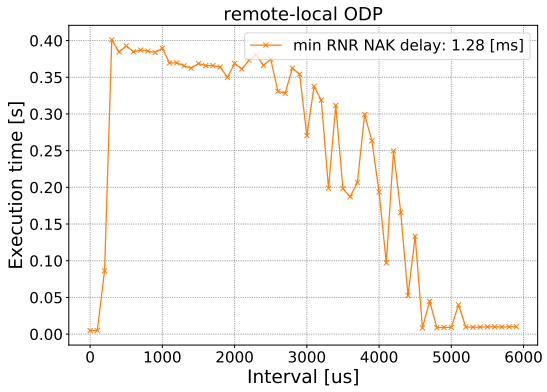


**Fig. 5** Average execution time of our micro-benchmark out of 10 trials varying the interval between two communications. Two RDMA READ operations are issued and ODP is used both on the server and client side. Minimal RNR NAK delay is set to 1.28 ms.

For our experiments, two Ubuntu 18.04 machines connected by InfiniBand were used. Each machine has a Xeon Phi CPU 7250 with 272 logical cores operating at 1.40 GHz and memory of PC4-19200 196GB and MCDRAM 16GB. They are equipped with one Mellanox MCX456A-FCAT ConnectX-4 VPI adapter card listed in Table 1 as private servers B, whose minimal timeout condition $T_o$ is about 500 ms. We used RC as transport type with $C_{\mathrm{ACK}} = 1$ and $C_{\mathrm{Retry}} = 7$.

We create a micro-benchmark with InfiniBand verbs in which two processes are created in each machine and the client process issues multiple RDMA READ operations to the server process as shown in **Fig. 4**. We insert a sleep function to control the interval of communications. At the end of the benchmark, a wait block is inserted so that we confirm all the communications are finished. The communication buffer is allocated with 4096 bytes boundaries considering the page size.

### 4.1 Performance Pitfalls with Two RDMA READ Operations

First, let's see the execution time of the micro-benchmark with two RDMA READ operations in **Fig. 5**. Although one normal round trip takes several microseconds and the resolution of one page fault takes several hundred microseconds, the execution time took several hundred *milliseconds* with 100 to 4500 µs of the interval. While the result of the figure is in the case of remote-local ODP, the execution time also becomes long even with remote ODP or local ODP.

To analyze this phenomenon, we captured InfiniBand packets using ibdump with three conditions: remote ODP, local ODP, and remote-local ODP. Figure 6 shows the InfiniBand packets and **Fig. 7** illustrates the conjecture about what happened based on the information of the page fault counts on both sides. From these information, we can say that the long latency comes from the timeout of the second RDMA READ operation. The response of the second READ seemed to disappear for some reasons with no NAK and it forces the client to wait for the timeout to expire. This phenomenon seemed to be caused by consecutive packets requests followed by the second READ operation posted while the client was waiting for something: RNR NAK delay or the resolution of the local page fault.

To verify this hypothesis, we measured the probability of timeout in three ODP conditions with changing the value of minimal RNR NAK delay. The left one in **Fig. 8** shows the probability of the timeout of remote ODP and you can see the range of interval works together with the minimal RNR NAK delay. The result shows that in the case of 1.28 ms of minimal RNR NAK delay, the timeout occurs up to around 4500 µs of the interval, which is exactly corresponds to the actual RNR NAK delay measured in Section 3. The middle one in Fig. 8 is the result of local ODP. It shows the timeout occurs up to around 500 µs of the interval, which corresponds to the time for the page fault to resolve. Finally, the right one in Fig. 8 shows the result of remote-local ODP and the shape of the graph looks like remote ODP and local ODP joined together. With minimal RNR NAK delay is equal to 0.01 ms, you can see clearly the graph has two waves for each waiting period: RNR NAK delay and the resolution of the local page fault.

### 4.2 Performance with more than Two RDMA READs

**Figure 9** shows the probability of the timeout varying the number of RDMA READ operations in the benchmark. It shows that increasing the number of READ operations surprisingly narrows down the range of intervals. This phenomenon can be explained by PSN (Packet Sequence Number) management in RC. All the packets contain PSN to detect missing or out-of-order packets. When the responder detects a packet with unexpected PSN, it returns the NAK packet to the requester with PSN sequence error.

Figure 11 shows the packets captured with ibdump in this situation with three READ operations and **Fig. 10** illustrates what happened. Even after the packet loss of the second READ operation, the server was expecting the second READ request to come. Therefore, if the request of the third READ request was issued from the client after the packet loss, the server acknowledged the third request as an unexpected one and returns NAK with PSN sequence error. After the client receives NAK, it retransmitted all the packets whose response the server has not received so far. In this time, retransmission was conducted for the second and third READ operations and the timeout never happened. If the interval is small enough for all the READ operations to fit into the waiting period, however, the timeout still happened with more than two operations.

```
Time [s]
0.000000    RDMA Read Request (1st)
0.000140    Acknowledge (RNR NACK)
0.000451    RDMA Read Response (1st)
0.003968    RDMA Read Request (1st)
0.003970    RDMA Read Request (2nd)
0.004051    RDMA Read Response (1st)
0.541748    RDMA Read Request (2nd)
0.541755    RDMA Read Response (2nd)
```

```
Time [s]
0.000000    RDMA Read Request (1st)
0.000005    RDMA Read Response (1st)
0.000778    RDMA Read Request (1st)
0.000782    RDMA Read Request (2nd)
0.000861    RDMA Read Response (1st)
0.435909    RDMA Read Request (2nd)
0.435912    RDMA Read Response (2nd)
```

```
Time [s]
0.000000    RDMA Read Request (1st)
0.000170    Acknowledge (RNR NAK)
0.000510    RDMA Read Response (1st)
0.004471    RDMA Read Request (1st)
0.004473    RDMA Read Request (2nd)
0.004558    RDMA Read Response (1st)
0.005171    RDMA Read Request (1st)
0.005174    RDMA Read Request (2nd)
0.005215    RDMA Read Response (1st)
0.451931    RDMA Read Request (2nd)
0.451934    RDMA Read Response (2nd)
```

**Fig. 6** Packets obtained with ibdump while handling two RDMA READ with remote, local, and remote-local ODP. The interval is set to 3000 µs for remote ODP and remote-local ODP, and to 500 µs for local ODP respectively.

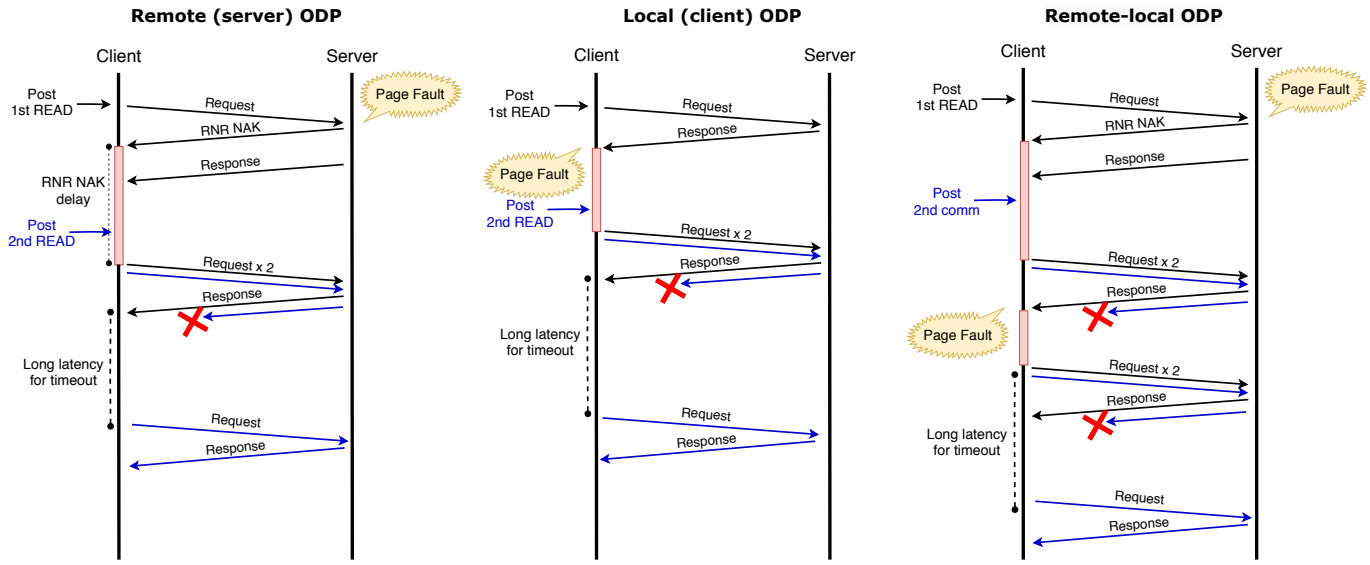


**Fig. 7** Illustration of how ODP works with two RDMA READ operations based on the packets described in Fig. 6. Red lines represent waiting periods which can cause packet loss.

### 4.3 Experiments with Other Conditions

In order to explore more about this packet loss and subsequent timeout, we conducted experiments with other conditions and discovered the following.

- This phenomenon arose independently of other QPs. The QP still continued to wait for the timeout and caused the long latency even if new operations were posted in other QPs.
- It occurred regardless of whether the communication buffer in each communication operation was on the same page or not.
- It was not related to the page fault on the second (or later) communication. We confirmed it occurred even if we had used and touched all the communication buffers except for the one for the first communication in advance.
- It was irrelevant to the size of the communication buffer. It occurred even with the buffer with a larger size than the page size.
- It occurred in a variety of systems with bare metal servers include Reedbush-H/L, ABCI, and ITO (see Table 1 for more information of these systems).

## 5. Discussions

Our experiments and analysis in the previous section show that the timeout of ODP must be ubiquitous because

- page fault is an indispensable feature for ODP
- only two communication operations are needed
- the range of intervals in which the timeout occurs is quite wide

As we described in Section 3, most InfiniBand adapter cards have an unbelievably rough resolution time for the detection of the packet loss and timeout. Although InfiniBand is equipped with numerous features to avoid the usage of timeouts such as lossless link-level flow control and PSN management, we revealed that the current implementation of ODP highly depends on this mechanism, which spawns performance pitfall unacceptable for high-performance communication.

However, we can also derive clues from our experiments for mitigating this performance degradation. First, setting the smallest value to minimal RNR NAK delay can narrow down the range of the intervals in which the timeout occurs. As previous research shows [20], this setting can also reduce the time of local ODP and, therefore, we should actively adopt it. Second, there is a lot of potentials to handle this phenomenon with a little help of software. Our experiment showed that increasing the number of communication operations provide more chances for the responder to detect PSN Sequence Error and, subsequently, reduce the possibility of the timeout. It means that the long stall caused by packet loss can be avoided by posting additional communica-

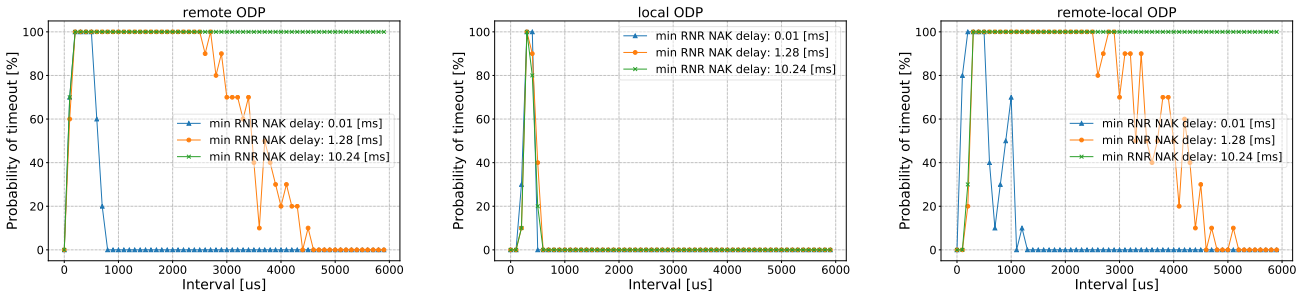


**Fig. 8** Probability of timeout calculated with 10 trials varying the interval between two communications with remote, local, and remote-local ODP. Two RDMA READ operations is issued with remote ODP. Minimal RNR NAK delay is set to 1.28 ms.

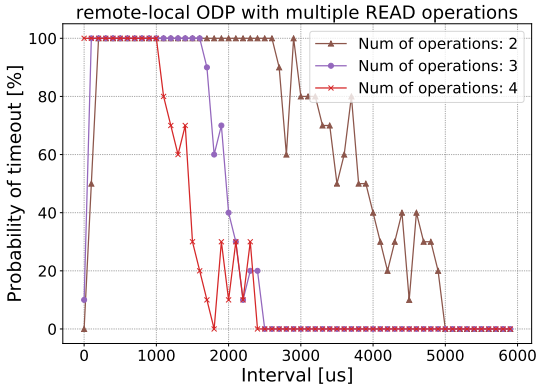


**Fig. 9** Probability of timeout with multiple READ operations calculated with 10 trials varying the interval between each communication. The number of RDMA READ operations are 2, 3 and 4 and ODP is used both on the server and client side. Minimal RNR NAK is set to 1.28 ms.

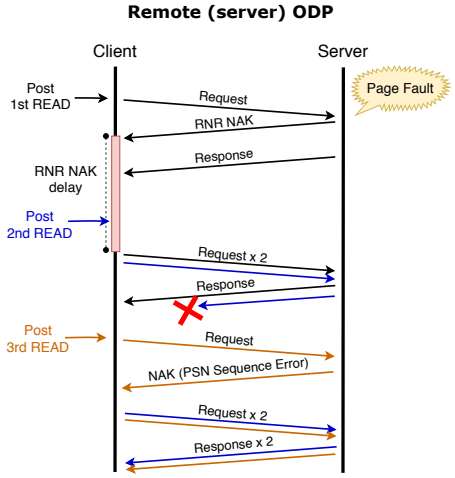


**Fig. 10** Illustration of how ODP works with three RDMA READ operations based on the packet capturing described in Fig. 11

tion if needed. The software can provide this functionality easily by implementing a timer with appropriate granularity to poll the outstanding communication. Even coarse-grained software timer should take effect considering the fact that the existing hardware is equipped with a timer with an awful resolution of several hundred milliseconds. We believe this approach should be promising and practical because it should work with no modification to the existing hardware.

```
Time [s]
0.000000    RDMA Read Request (1st)
0.000143    Acknowledge (RNR NAK)
0.000464    RDMA Read Response (1st)
0.003913    RDMA Read Request (1st)
0.003915    RDMA Read Request (2nd)
0.004005    RDMA Read Response (1st)
0.006414    RDMA Read Request (3rd)
0.006415    Acknowledge (NAK)
0.006615    RDMA Read Request (2nd)
0.006616    RDMA Read Request (3rd)
0.006617    RDMA Read Response (2nd)
0.006618    RDMA Read Response (3rd)
```

**Fig. 11** Packets obtained with idbump while handling three RDMA READ with remote-local ODP. The interval is set to 3000 µs

## 6. Related Work

### 6.1 Performance Analysis of On-Demand Paging

Only a few studies analyzed the performance characteristics of ODP because it is a relatively new feature. Lesokhin et al. [17] proposed network page fault support for InfiniBand for the first time. They evaluated the overhead of page faults and invalidation. Their breakdowns showed the overhead of page fault was dominated by the hardware including triggering interrupt and resuming transmission while the invalidation spends most of its time on the update of page tables. Li at al. [19] gave a thorough analysis of the performance of Explicit ODP in order to design a memory-efficient MPI library. They compared Explicit ODP with pin-down cache in latency and bandwidth and revealed that the performance of Explicit ODP degraded with page faults whereas it little degraded without page faults. Their results also indicated that the characteristics of page faults on the sender-side were different from those on the receiver side and prefetching on the receiver side worked well. In their later work [20] on Implicit ODP, they revealed that the overhead of page faults was able to mitigate through an elaborate tuning of RNR NACK timer.

### 6.2 Memory Registration Methods

Many efforts to eliminate costly memory registration of RDMA by reusing pinned-down buffers were made. Tezuka et al. [32] proposed the basic idea of pin-down cache for the first time, which reused registered buffers by delaying deregistration. The registered buffers worked as a cache; when the total size of registered buffers exceeded the maximum size, the deregistration of a buffer occurred in the least recently used (LRU) order. Zhou et al. [39] also reused pinned buffers for dynamic registered re-

gions and introduced batched deregistration to reduce the average cost of deregistering memory. Ou at al. [28, 29] proposed Memory Registration Region Cache and a replacement algorithm based on region size and recency. They also proposed a communication scheme between an RDMA client and server that overlapped memory registrations. Wu et al. [35] proposed a two-level architecture called Fast Memory Registration and Deregistration scheme, which adopted batched deregistration. It also made use of the Mellanox fast memory region registration extension to the InfiniBand-Verbs API.

Even in the presence of pin-down cache, several performance tradeoffs are known to exist. The Unifier [37] caching system addressed a tradeoff between (de)registration cost and spatial cost. Larger pinned down buffers suppress on-demand pinning in dynamic registration, whereas they more occupy physical memory, which other computations should have used. Frey and Alonso [5] argued a tradeoff between pinning (i.e., newly registering) and copying. They experimentally showed that for 256-KB or larger regions, it was more efficient to newly register by order of magnitude.

Some studies dealt with memory registration methods aimed at specific situations. Firehose algorithm [1] was designed for global address space over distributed memory, which used reference counting of global address regions for (de)registration. Wu et al. [36] designed a new scheme, Optimistic Group Registration for RDMA Gather/Scatter for noncontiguous data transmission. Li et al. [18] implemented dynamic memory management efficiently on multicore platforms by employing a dedicated helper thread to which all the dynamic memory (de)registration requests are offloaded.

The breakdown of the memory registration process itself is known to be important for efficient memory management. Mietke et al. [24] analyzed the registration process inside the Mellanox InfiniBand driver and provided clues to improving it. For example, it drastically reduced the cost of `mlock()` by making a separate kernel thread zero-fill pages when they were not present.

### 6.3 Reliability of InfiniBand

The reliability of InfiniBand has attracted attention in a lot of the literature. Koop et al. [15] evaluated and compared the cost of hardware-based reliability and software-based one by taking InfiniBand as a case study. They implemented MPI over the UC transport and experimentally compared it with MPI over the RC and UD transports. They showed a software-based approach was not only feasible but might provide higher performance because of its minimal memory consumption. Later, they developed MVAPICH-Aptus [14], which implemented MPI over multiple types of transport and was able to select transport protocols and message channels dynamically. Jose at al. [10] made a similar attempt to address the memory consumption and scalability issue of RC transport and implemented memcached with a hybrid use of RC and UD for improving performance.

## 7. Conclusions and Future Work

In this work, we have found ODP of InfiniBand has awful performance pitfalls, which are never acceptable in high-performance communication. We have reported that the long stall of the requester arises easily with our simple micro-benchmark which contains only two RDMA READ operations. We have analyzed this phenomenon using ibdump, a packet capturing tool for InfiniBand, and identified that it comes from the packet drop and timeout of the request through ODP handling. We also have shown that the performance degradation should happen in many systems including famous supercomputing systems by measuring the timeout resolution period in a variety of InfiniBand adapter cards.

We have two tasks for future work. One is to implement a communication software layer that covers the performance pitfalls. We believe that it can be achieved by implementing the software timer to poll outstanding communication requests and post a dummy request to avoid the stall of the requester. The other is to evaluate the performance pitfalls empirically through applications.

### References

[1] Bell, C. and Bonachea, D.: A new DMA registration strategy for pinning-based high performance networks, *Proceedings - International Parallel and Distributed Processing Symposium, IPDPS 2003*, Institute of Electrical and Electronics Engineers Inc., (online), DOI: 10.1109/IPDPS.2003.1213363 (2003).

[2] Chen, Y., Wei, X., Shi, J., Chen, R. and Chen, H.: Fast and general distributed transactions using RDMA and HTM, *Proceedings of the 11th European Conference on Computer Systems, EuroSys 2016*, (online), DOI: 10.1145/2901318.2901349 (2016).

[3] Dongarra, J., Heroux, M. A. and Luszczek, P.: High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems, *The International Journal of High Performance Computing Applications*, Vol. 30, No. 1, pp. 3–10 (online), DOI: 10.1177/1094342015593158 (2016).

[4] Dragojević, A., Narayanan, D., Hodson, O. and Castro, M.: FaRM: Fast remote memory, *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014*, pp. 401–414 (2014).

[5] Frey, P. W. and Alonso, G.: Minimizing the Hidden Cost of RDMA, *2009 29th IEEE International Conference on Distributed Computing Systems*, IEEE, pp. 553–560 (online), DOI: 10.1109/ICDCS.2009.32 (2009).

[6] Grun, P., Hefty, S., Sur, S., Goodell, D., Russell, R. D., Pritchard, H. and Squyres, J. M.: A Brief Introduction to the OpenFabrics Interfaces - A New Network API for Maximizing High Performance Application Efficiency, *Proceedings - 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects, HOTI 2015*, Institute of Electrical and Electronics Engineers Inc., pp. 34–39 (online), DOI: 10.1109/HOTI.2015.19 (2015).

[7] Guo, C., Wu, H., Deng, Z., Soni, G., Ye, J., Padhye, J. and Lipshteyn, M.: RDMA over commodity ethernet at scale, *SIGCOMM 2016 - Proceedings of the 2016 ACM Conference on Special Interest Group on Data Communication*, New York, New York, USA, Association for Computing Machinery, Inc, pp. 202–215 (online), DOI: 10.1145/2934872.2934908 (2016).

[8] Hudzia, B.: On allowing shorter timeout on Mellanox cards and other tips and tricks, `https://www.reflectionsofthevoid.com/2014/02/on-allowing-shorter-timeout-on-mellanox.html` (2014).

[9] InfiniBand℠Trade Association: *InfiniBand Architecture Specification Volume 1*, release 1.4 edition (2020). `https://cw.infinibandta.org/document/dl/8567`.

[10] Jose, J., Subramoni, H., Kandalla, K., Wasi-Ur-Rahman, M., Wang, H., Narravula, S. and Panda, D. K.: Scalable memcached design for InfiniBand clusters using hybrid transports, *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, pp. 236–243 (online), DOI: 10.1109/CCGrid.2012.141 (2012).

[11] Kalia, A., Kaminsky, M. and Andersen, D. G.: Using RDMA efficiently for key-value services, *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, Vol. 44, No. 4, New York, New York, USA, ACM Press, pp. 295–306 (online), DOI: 10.1145/2619239.2626299 (2014).

[12] Kalia, A., Kaminsky, M. and Andersen, D. G.: Design guidelines for high performance RDMA systems, *Proceedings of the 2016 USENIX Annual Technical Conference, USENIX ATC 2016*, pp. 437–450 (2016).

[13] Kalia, A., Kaminsky, M. and Andersen, D. G.: FaSST: Fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs, *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pp. 185–201 (2016).

[14] Koop, M. J., Jones, T. and Panda, D. K.: MVAPICH-Aptus: Scalable high-performance multi-transport MPI over InfiniBand, *2008 IEEE International Symposium on Parallel and Distributed Processing*, IEEE, pp. 1–12 (online), DOI: 10.1109/IPDPS.2008.4536283 (2008).

[15] Koop, M. J., Kumar, R. and Panda, D. K.: Can software reliability outperform hardware reliability on high performance interconnects?, *Proceedings of the 22nd annual international conference on Supercomputing - ICS '08*, New York, New York, USA, ACM Press, p. 145 (online), DOI: 10.1145/1375527.1375551 (2008).

[16] Koop, M. J., Sur, S., Gao, Q. and Panda, D. K.: High performance MPI design using unreliable datagram for ultra-scale InfiniBand clusters, *Proceedings of the 21st annual international conference on Supercomputing - ICS '07*, New York, New York, USA, ACM Press, p. 180 (online), DOI: 10.1145/1274971.1274997 (2007).

[17] Lesokhin, I., Eran, H., Raindel, S., Shapiro, G., Grimberg, S., Liss, L., Ben-Yehuda, M., Amit, N. and Tsafrir, D.: Page Fault Support for Network Controllers, *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. Part F1271, New York, NY, USA, ACM, pp. 449–466 (online), DOI: 10.1145/3037697.3037710 (2017).

[18] Li, D., Cameron, K. W., Nikolopoulos, D. S., de Supinski, B. R. and Schulz, M.: Scalable memory registration for high performance networks using helper threads, *Proceedings of the 8th ACM International Conference on Computing Frontiers - CF '11*, New York, New York, USA, ACM Press, p. 1 (online), DOI: 10.1145/2016604.2016652 (2011).

[19] Li, M., Hamidouche, K., Lu, X., Subramoni, H., Zhang, J. and Panda, D. K.: Designing MPI Library with On-Demand Paging (ODP) of InfiniBand: Challenges and Benefits, *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, Vol. 0, IEEE, pp. 433–443 (online), DOI: 10.1109/SC.2016.36 (2016).

[20] Li, M., Lu, X., Subramoni, H. and Panda, D. K.: Designing Registration Caching Free High-Performance MPI Library with Implicit On-Demand Paging (ODP) of InfiniBand, *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*, Vol. 2017-Decem, IEEE, pp. 62–71 (online), DOI: 10.1109/HiPC.2017.00017 (2017).

[21] Lu, Y., Shu, J., Li, T. and Chen, Y.: Octopus: An RDMA-enabled distributed persistent memory file system, *Proceedings of the 2017 USENIX Annual Technical Conference, USENIX ATC 2017*, pp. 773–785 (2019).

[22] MacArthur, P., Liu, Q., Russell, R. D., Mizero, F., Veeraraghavan, M. and Dennis, J. M.: An Integrated Tutorial on InfiniBand, Verbs, and MPI, *IEEE Communications Surveys and Tutorials*, Vol. 19, No. 4, pp. 2894–2926 (online), DOI: 10.1109/COMST.2017.2746083 (2017).

[23] Mellanox: *Understanding On Demand Paging (ODP)* (2019). `https://community.mellanox.com/s/article/understanding-on-demand-paging--odp-x`.

[24] Mietke, F., Rex, R., Baumgartl, R., Mehlan, T., Hoefler, T. and Rehm, W.: Analysis of the Memory Registration Process in the Mellanox InfiniBand Software Stack, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 4128 LNCS, Springer Verlag, pp. 124–133 (2006).

[25] Mitchell, C., Geng, Y. and Li, J.: Using one-sided RDMA reads to build a fast, CPU-efficient key-value store, *Proceedings of the 2013 USENIX Annual Technical Conference, USENIX ATC 2013*, pp. 103–114 (2013).

[26] Mittal, R., Shpiner, A., Panda, A., Zahavi, E., Krishnamurthy, A., Ratnasamy, S. and Shenker, S.: Revisiting network support for RDMA, *SIGCOMM 2018 - Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, Vol. 18, pp. 313–326 (online), DOI: 10.1145/3230543.3230557 (2018).

[27] Nakamura, M.: Understanding Retransmission Control in InfiniBand, `http://www.nminoru.jp/~nminoru/network/infiniband/iba-retransmission.html` (2014). In Japanese.

[28] Ou, L., He, X. and Han, J.: MRRC: An effective cache for fast memory registration in RDMA, *IEEE Symposium on Mass Storage Systems and Technologies* (2006).

[29] Ou, L., He, X. and Han, J.: An efficient design for fast memory registration in RDMA, *Journal of Network and Computer Applications*, Vol. 32, No. 3, pp. 642–651 (online), DOI: 10.1016/j.jnca.2008.07.008 (2009).

[30] Shamis, P., Venkata, M. G., Lopez, M. G., Baker, M. B., Hernandez, O., Itigin, Y., Dubman, M., Shainer, G., Graham, R. L., Liss, L., Shahar, Y., Potluri, S., Rossetti, D., Becker, D., Poole, D., Lamb, C., Kumar, S., Stunkel, C., Bosilca, G. and Bouteiller, A.: UCX: An Open Source Framework for HPC Network APIs and Beyond, *2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*, IEEE, pp. 40–43 (online), DOI: 10.1109/HOTI.2015.13 (2015).

[31] Technologies, M.: Introduction to InfiniBand, *Technical Report*, pp. 1–20 (2003).

[32] Tezuka, H., O'Carroll, F., Hori, A. and Ishikawa, Y.: Pin-down cache: A virtual memory management technique for zero-copy communication, *Proceedings of the 1st Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing, IPPS/SPDP 1998*, Vol. 1998-March, Institute of Electrical and Electronics Engineers Inc., pp. 308–314 (online), DOI: 10.1109/IPPS.1998.669932 (1998).

[33] Wei, X., Dong, Z., Chen, R., Chen, H. and Tong, S. J.: Deconstructing RDMA-enabled Distributed Transactions: Hybrid is Better!, *13th USENIX Symposium on Operating Systems Design and Implementation OSDI 18* (2018).

[34] Wei, X., Shi, J., Chen, Y., Chen, R. and Chen, H.: Fast in-memory transaction processing using RDMA and HTM, *SOSP 2015 - Proceedings of the 25th ACM Symposium on Operating Systems Principles*, pp. 87–104 (online), DOI: 10.1145/2815400.2815419 (2015).

[35] Wu, J., Wyckoff, P. and Panda, D.: PVFS over InfiniBand: Design and performance evaluation, *Proceedings of the International Conference on Parallel Processing*, Vol. 2003-Janua, Institute of Electrical and Electronics Engineers Inc., pp. 125–132 (online), DOI: 10.1109/ICPP.2003.1240573 (2003).

[36] Wu, J., Wyckoff, P. and Panda, D.: Supporting efficient noncontiguous access in PVFS over InfiniBand, *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, Vol. 2003-Janua, Institute of Electrical and Electronics Engineers Inc., pp. 344–351 (online), DOI: 10.1109/CLUSTR.2003.1253333 (2003).

[37] Wu, J., Wyckoff, P., Panda, D. and Ross, R.: Unifier: Unifying cache management and communication buffer management for PVFS over InfiniBand, *2004 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2004*, pp. 523–530 (online), DOI: 10.1109/ccgrid.2004.1336646 (2004).

[38] Xue, J., Miao, Y., Chen, C., Wu, M., Zhang, L. and Zhou, L.: Fast distributed deep learning over RDMA, *Proceedings of the 14th EuroSys Conference 2019*, Vol. 19, New York, New York, USA, ACM Press, pp. 1–14 (online), DOI: 10.1145/3302424.3303975 (2019).

[39] Zhou, Y., Bilas, A., Jagannathan, S., Dubnicki, C., Philbin, J. F. and Li, K.: Experiences with VI communication for database storage, *Conference Proceedings - Annual International Symposium on Computer Architecture, ISCA*, pp. 257–268 (online), DOI: 10.1109/ISCA.2002.1003584 (2002).

[40] Zhu, Y., Zhang, M., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S. and Yahia, M. H.: Congestion Control for Large-Scale RDMA Deployments, *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15* (Uhlig, S., Maennel, O., Karp, B. and Padhye, J., eds.), Vol. 45, No. 4, New York, New York, USA, ACM Press, pp. 523–536 (online), DOI: 10.1145/2785956.2787484 (2015).