PoC Project name:

RDMA over Open APN PoC Report

Classification: IOWN Global Forum Recognized PoC

Stage: SSF PoC Report

Confidentiality: Public

Version: 1.0

June 6, 2024

RDMA over Open APN PoC Report

Table of Contents

RDMA over Open APN PoC Report

List of Figures

List of Tables

1. PoC Project Completion Status

- Overall PoC Project Completion Status: COMPLETED

2. RDMA over APN PoC Project Participants

- PoC Project Name: RDMA over Open APN PoC
- Member A: NTT________________________ Contact: Kazuaki OBANA______
- Member B: ____________________________ Contact: Kiwami Inoue________
- Member C: ____________________________ Contact: Takeshi Kugimoto___________
- Member D: ____________________________ Contact: Junki Ichikawa________
- Member E: ____________________________ Contact: Hitoshi Masutani______

3. Confirmation of PoC Demonstration

This PoC had been conducted in NTT Network Innovation Laboratories, Yokosuka, JAPAN.

4. PoC Goals Status Report

At first, the goal of this PoC is to achieve the objectives written in the "RDMA over Open APN PoC Reference" document.

- PoC Project Goal #1: to gather real-world experience by implementing the proposed technology Goal Status (Met) Please see Appendix F.
- PoC Project Goal #2: to discover best practices for tuning algorithm parameters to achieve highest performance Goal Status (Met) Please see Appendix F
- PoC Project Goal #3: to determine the dependence of maximum bandwidth on distance of the RDMA-over-Open-APN technology Goal Status (Met) Please see from 5.2.1 to 5.2.3

And all of the measurements have been executed as same steps as written in the Table 1 of the PoC Reference. Please see from 5.2.1 to 5.2.3.

Also all of the measurements have been executed under conditions that meet the requirements written in the section 3 of the PoC Reference. Please see from 5.2.1 to 5.2.3.

Finally, all of the benchmarks written in the section 4 of the PoC Reference have been executed and met the goal in certain conditions. Please see from 5.2.1 to 5.2.3.

Generally, RDMA is developed for relatively short distance communication, for example inside of a data center. If you want to meet throughput requirement even though in long distance communication, please consider to extend queue-pair parameter like Appendix F or apply accelerator like Appendix G.

5. PoC Technical Report

5.1. Implemented System

Basic configuration

The following figure is the basic server configuration to evaluate the performance.

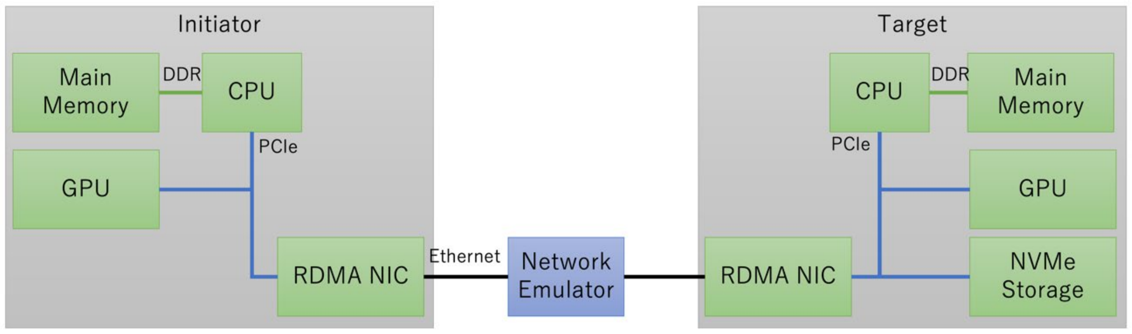


Figure 1: Basic configuration

Each step requires different hardware: Step.1 includes only the RDMA NIC, Step.2 includes the RDMA NIC and GPU, and Step.3 includes the RDMA NIC and NVMe devices.
The network emulator inserts various delays according to settings and we only consider the transmission delay of the fiber over the distance. The distance range is determined from the conditions described in CPS AM PoC reference document.

Hardware setup configuration

The hardware configuration that this report used shows the following table.

Table 1: Hardware configuration

Item	Description
Server Platform	HPE ProLiant DL380 Gen10 Plus (Both Sender & Receiver)
CPU	2 CPU sockets, Intel® Xeon® Gold 6354 CPU @ 3.00GHz Number of cores 18 per socket, number of threads 2 per core. Microcode: 0xd000389
Memory	32 x 32GB SK Hynix HMA84GR7DJR4N-XN Total of 1024GB
iLO firmware version	2.63 Jan 2 2022
BIOS	1.58
Operating System	Ubuntu 22.04.2 LTS
Linux vernel version	5.15.0-64-generic
NIC	NVIDIA ConnectX-6(MT28908) 2x 200GbE, 1 NIC per server, PCIe Slot Gen4 Firmware version: 20.35.2000 Interface MTU = 4200, IBV MTU = 4096
NIC driver version	MLNX_OFED_LINUX-5.8-1.1.2.1 [NOTE]Not installed only when evaluating software RoCE.
GPU	NVIDIA A100 40GB PCIe (Both Sender & Receiver)
GPU driver version	525.85.12 (Open Kernel version) CUDA Toolkit 12.0
Storage	SPDK Storage Target (Receiver side): 4 x Samsung SSD 970 EVO Plus 2TB (on CPU NUMA Node 1. PCIe Gen3) RAID Card: HighPoint SSD7101A-1 M.2
Network Emulator	Spirent Attero-100G

BIOS settings

The BIOS settings for this configuration are shown below.

Table 2: Test systems BIOS settings

Item	Description

Hyper threading	Enabled
	`# dmidecode -t processor | grep -E '(Core Count|Thread Count)'`
	`Core Count: 18`
	`Thread Count: 36`
	`Core Count: 18`
	`Thread Count: 36`

CPU Power and Performance Policy	Workload profile: High Performance Compute (HPC)
CPU C-state	No limit
CPU P-state	Enabled
Intel(R) Turbo Boost Technology	Enabled

Software configuration

The software used in this evaluation is shown below.

Table 3: Software information for each evaluation

Item	Evaluation Step	Description
Linux RDMA (RDMA-core)	Step.1	rdma-core-43.1
Infiniband Verbs Performance Tests (Perftest)	Step.1 Step.2	linux-rdma/perftest: Infiniband Verbs Performance Tests (github.com) tag: v4.5-0.20
GPUDirect RDMA	Step.2	Mellanox/nv_peer_memory (github.com) tag: 1.3-0
SPDK	Step.3	spdk/spdk: Storage Performance Development Kit (github.com) tag: v22.09
FIO	Step.3	axboe/fio: Flexible I/O Tester (github.com) tag: fio-3.33 (NOT plugin of SPDK)
powertop	All	fenrus75/powertop: The Linux PowerTOP tool tag: v2.15

5.2. Measurement Method

5.2.1. Step.1: Main memory to main memory

Objective Id:	Main memory to main memory	
Description:	The performance measurements for configuration "main memory to main memory". PoC reference defines the following key benchmarks. Benchmark 1: throughputs for data transferring Benchmark 2: latency between RDMA endpoints Benchmark 3: power consumption	
Pre-conditions	None	
Procedure:	1	Measure both Software RoCE and NIC-offloaded RoCE and compare the two.
	2	We improve NIC-offloaded-RoCE performance by tuning.
	3	We get the effect of reducing power consumption by NIC-offloaded RoCE.

Finding Details:	We clarified the RDMA performance in the distance of 10 km ~ 1534 km, which is more than 1 km, which is the distance within a general data centre. We also clarified the difference in performance, latency and power consumption between Software RoCE and NIC-offloaded RoCE.
Lessons Learnt & Recommendations	NIC-offload RoCE can achieve much better performance than software RoCE. RDMA Write and Send operations in NIC-offload RoCE can achieve better performance by tuning queue depth of NIC. However, RDMA Read operation can achieve it by increasing queue pairs in addition to NIC's queue depth. Regarding the measurement of power consumption, it is possible to get the effect of NIC-offload for RDMA Write and Send operations, but it is difficult to get it for RDMA Read operation.

This step evaluates the performance of RDMA data transfer between the sender main memory and the receiver main memory. In this step, we measure two typical RDMA transfer techniques, Software RoCE and Offload RoCE.

There are three types of measurement indicators: transfer performance, latency, and power consumption. To measure these indicators, we connected the Attero-100G as network emulator between the Mellanox ConnectX-6 on sender and receiver, and adjusted the latency on the Attero-100G, assuming some long-distance communications.

Software RoCE

We measured Software RoCE with three types of operations: SEND, RDMA WRITE, and RDMA READ. The measurement software used is Linux-RDMA perftest, and the throughput and latency measurement tool used in each operation are shown in Table 29.

For each operation, we measured the communication distance with the Attero-100G set to the following 7 types.

Communication Distance: Direct, 10km, 20km, 100km, 200km, 1000km, 1534km

We also set the queue depth of NVIDIA ConnectX-6 to the default value of 128 and also set the Tx depth (the argument "-t" of perftest) to 128. If the Tx depth is much less than queue depth of NIC, it will occur performance degradation due to overhead data transfer. Therefore, the Tx depth should be more than the queue depth of NIC. The queue pair (QP: the argument "-q" of perftest) is set to 1, the number of iterations (the argument "-n" of perftest) is set to 15000 to improve measurement accuracy. Power consumption was measured by running powertop during each operation and monitoring the power consumption of the perftest process.

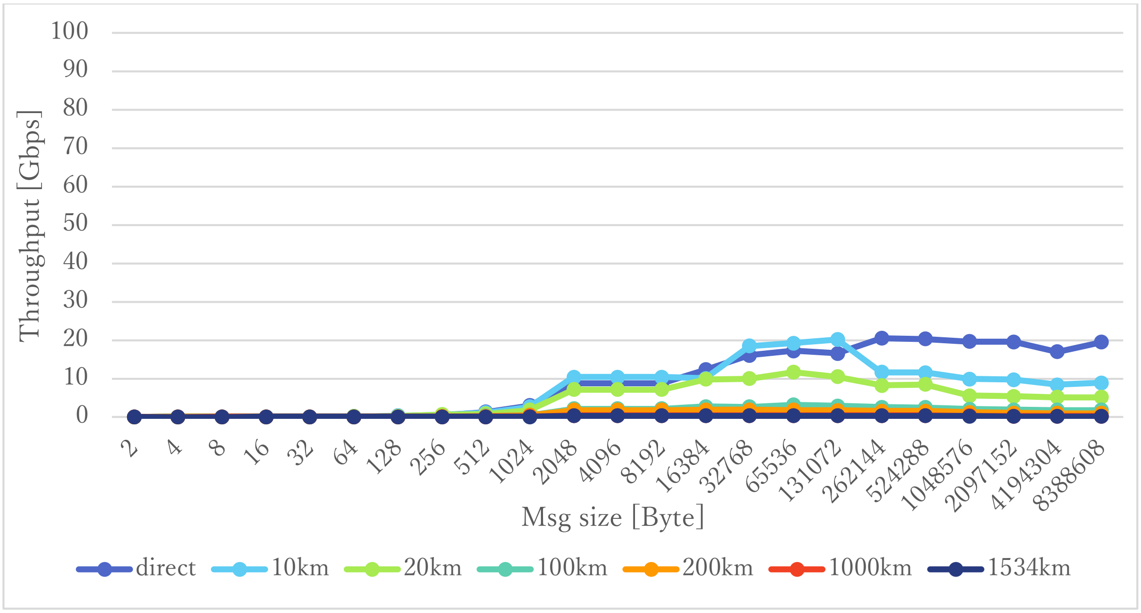


Figure 2: WRITE operation throughput using Software RoCE at Main-to-Main

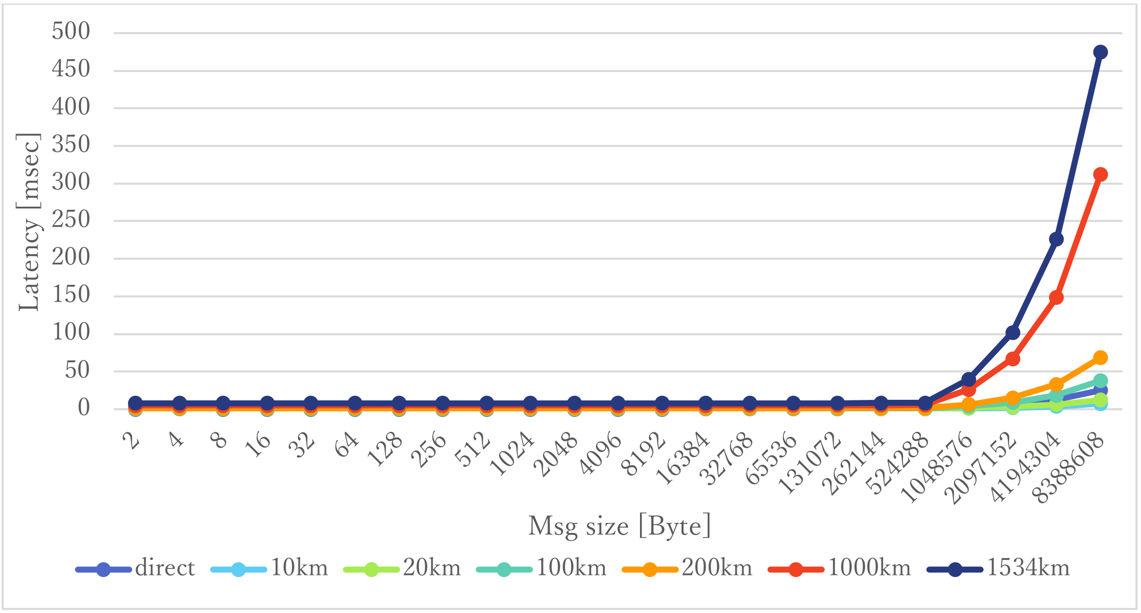


Figure 3: WRITE operation latency using Software RoCE at Main-to-Main

Table 4: Power consumption of WRITE operation using Software RoCE at Main-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.45	1.54	1.51	1.48	1.55	1.52	1.47
Sender[W]	1.46	1.53	1.49	1.5	1.52	1.53	1.45

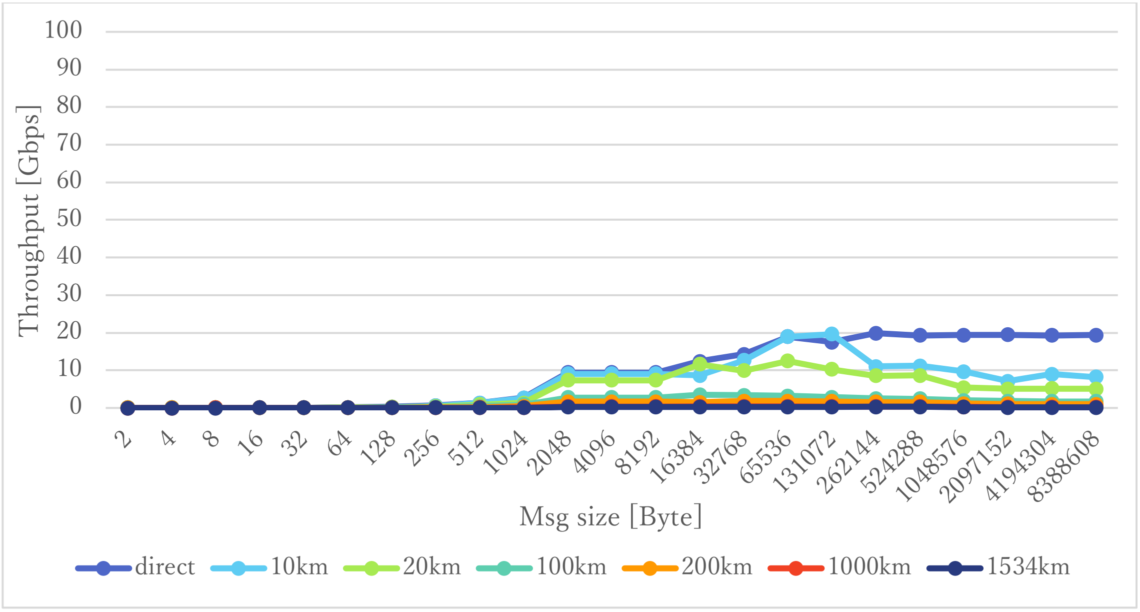


Figure 4: SEND operation throughput using Software RoCE at Main-to-Main

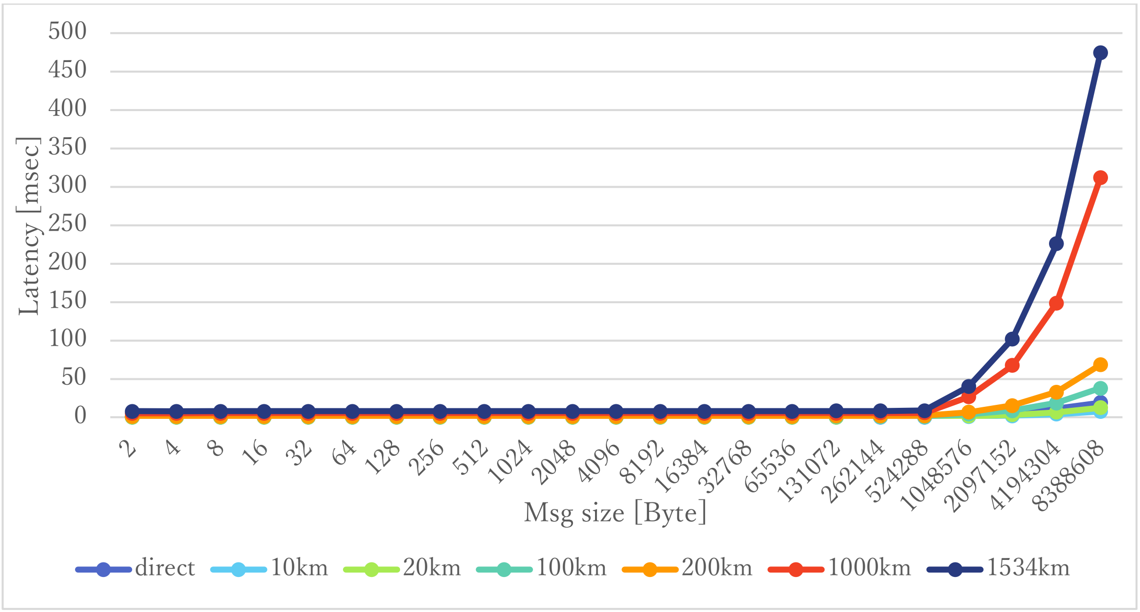


Figure 5: SEND operation latency using Software RoCE at Main-to-Main

Table 5: Power consumption of SEND operation using Software RoCE at Main-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.35	1.45	1.51	1.54	1.53	1.54	1.53
Sender[W]	1.35	1.54	1.51	1.52	1.44	1.47	1.52

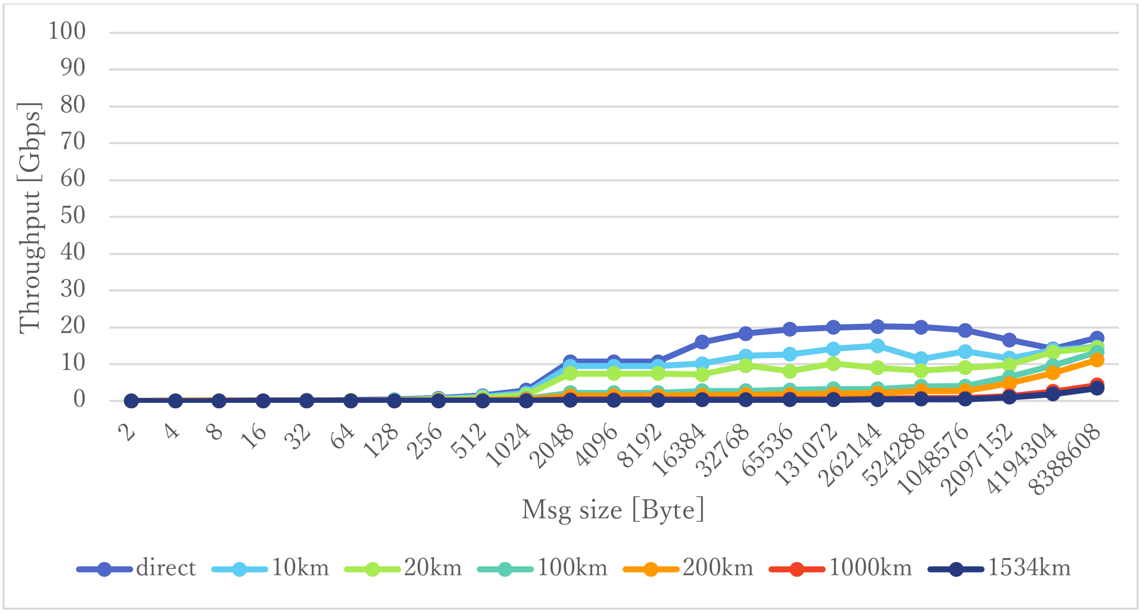


Figure 6: RDMA READ throughput using Software RoCE at Main-to-Main

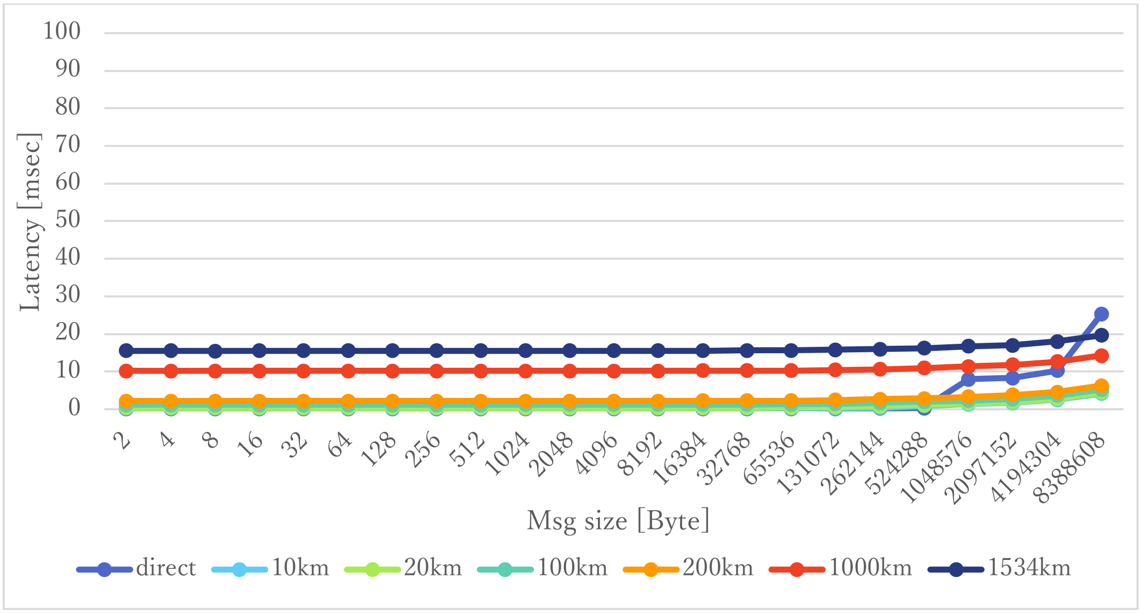


Figure 7: RDMA READ latency using Software RoCE at Main-to-Main

NIC-offloaded RoCE

We measured NIC-offloaded RoCE with three types of operations: SEND, RDMA WRITE, and RDMA READ. The measurement software used is Linux-RDMA perftest, and the measurement commands for throughput and latency is shown in Table 30.

For each operation, we measured the communication distance with the Attero-100G set to the following 7 types.

Communication Distance: Direct, 10km, 20km, 100km, 200km, 1000km, 1534km

We extended the queue depth of NVIDIA ConnectX-6 to 16384 and also set the Tx depth (the perftest's argument "-t") to 2,048. Setting 127 of Tx depth was sufficient for maximum throughput under the setting 16,384 of queue depth. In RDMA WRITE/SEND operations, the number of iterations (the perftest's argument "-n") is set to 15,000 to improve measurement accuracy and the queue pair (QP: the perftest's argument "-q") to 1. On the other hand, in RDMA READ operation which differs from RDMA WRITE/SEND, the QP is set to 4096, the number of iteration and the Tx depth is set to 16. Power consumption was measured by running powertop during each operation and monitoring the power consumption of the perftest process, similar to Software RoCE measurements.

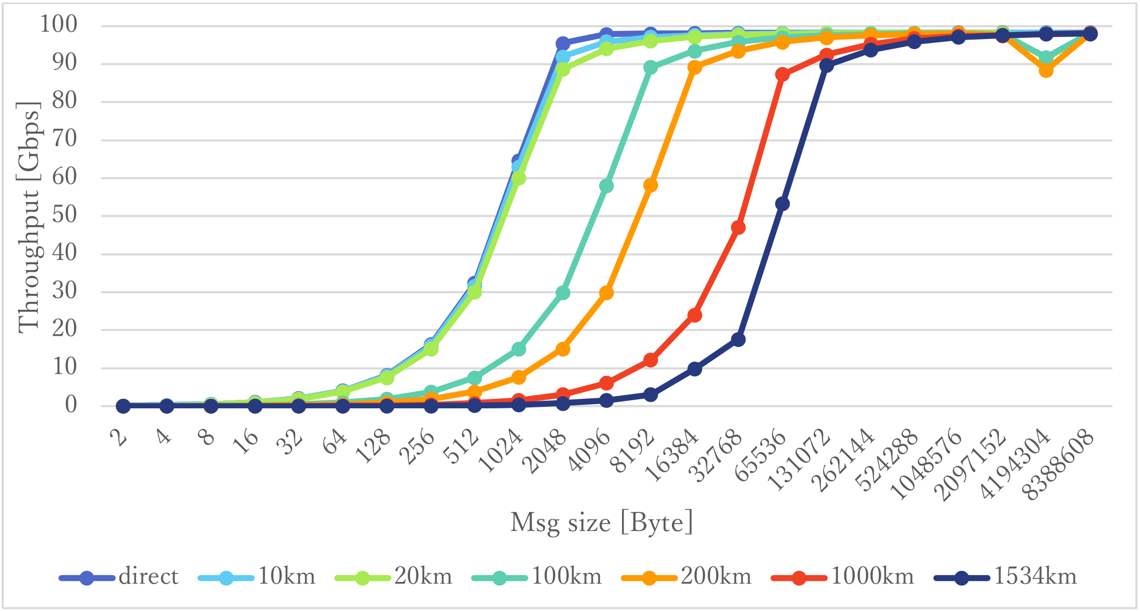


Figure 8: RDMA WRITE throughput using NIC-offload RoCE at Main-to-Main

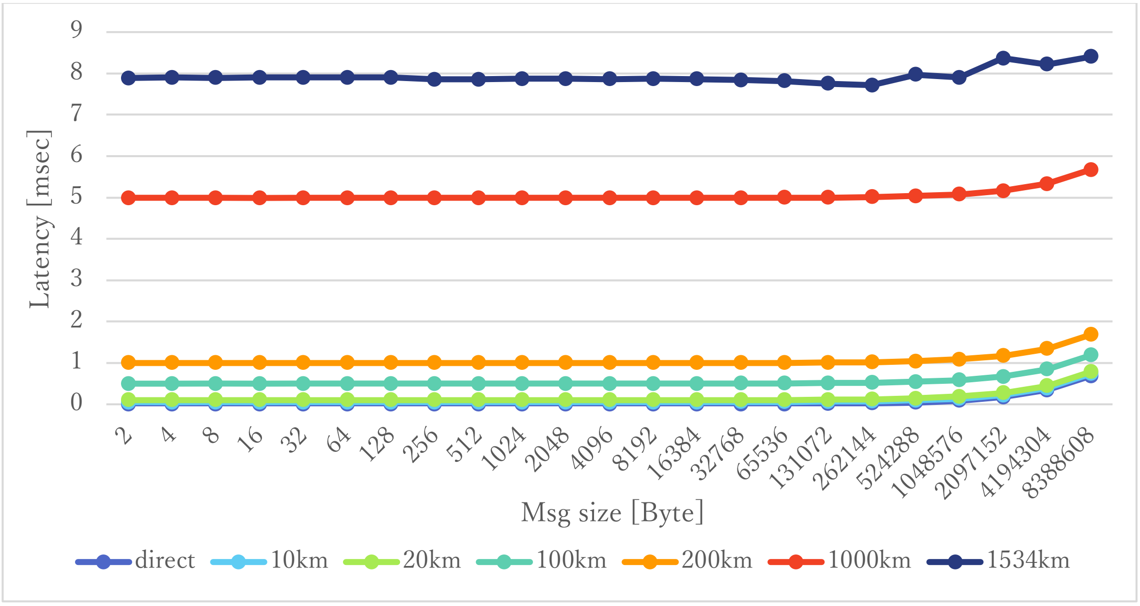


Figure 9: RDMA WRITE latency using NIC-offloaded RoCE at Main-to-Main

:

Table 6: Power consumption of RDMA WRITE using NIC-offloaded RoCE at Main-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.46	1.44	1.44	1.45	1.52	1.5	1.48
Sender[W]	1.46	1.46	1.45	1.46	1.54	1.55	1.44

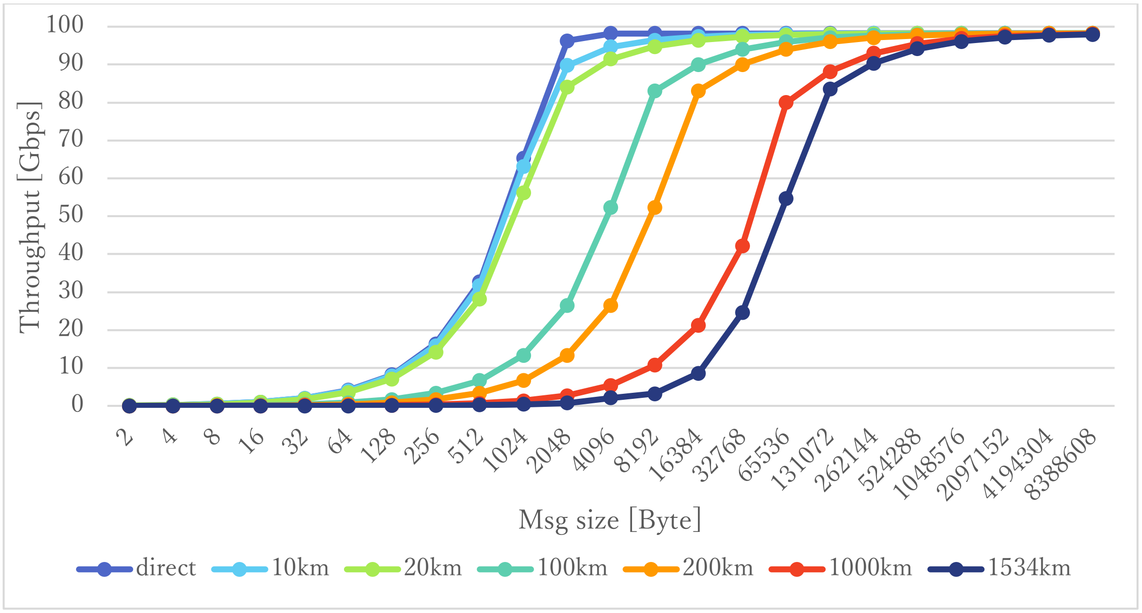


Figure 10: RDMA SEND throughput at NIC-offloaded RoCE at Main-to-Main

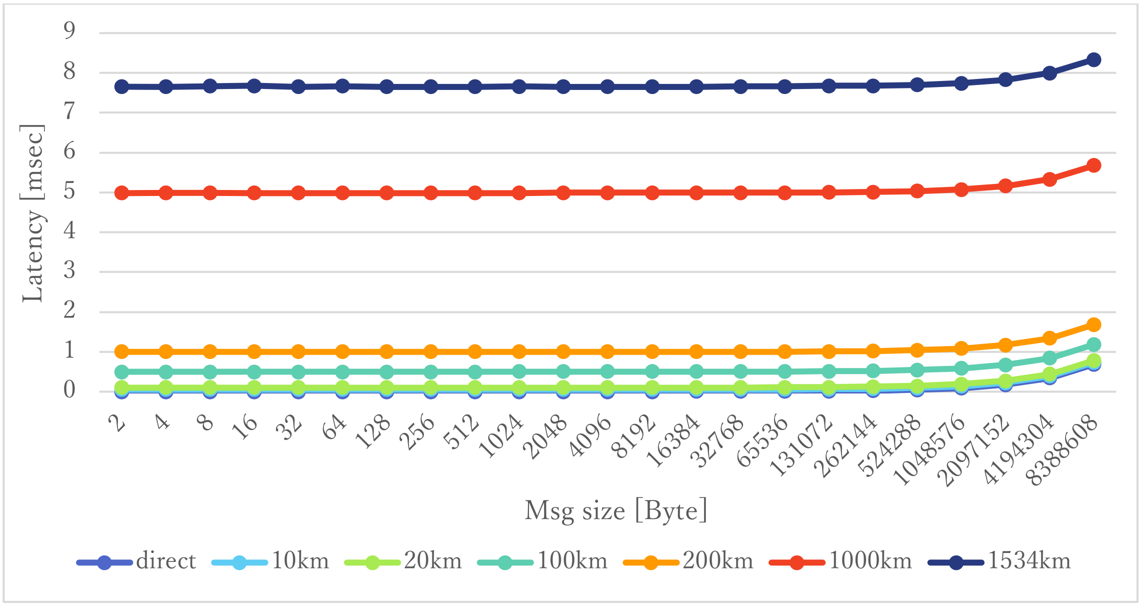


Figure 11: RDMA SEND latency using NIC-offloaded RoCE at Main-to-Main

Table 7: Power consumption of SEND using NIC-offloaded RoCE at Main-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.38	1.4	1.4	1.46	1.52	1.26	1.4
Sender[W]	1.4	1.4	1.41	1.39	1.52	1.26	1.44

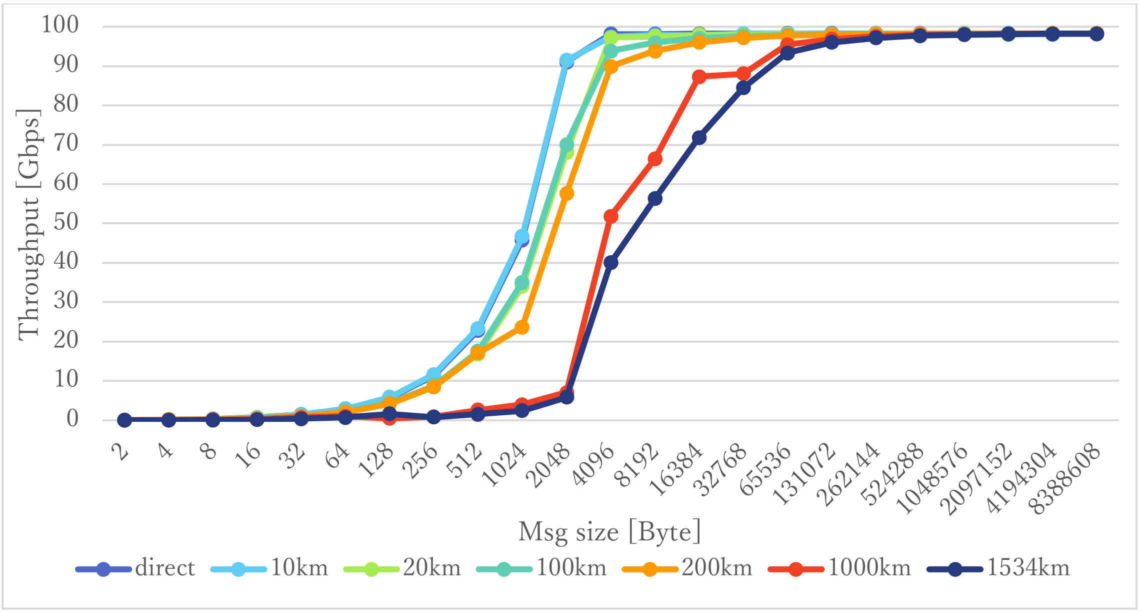


Figure 12: RDMA READ throughput using NIC-offloaded RoCE(QP=4096) at Main-to-Main

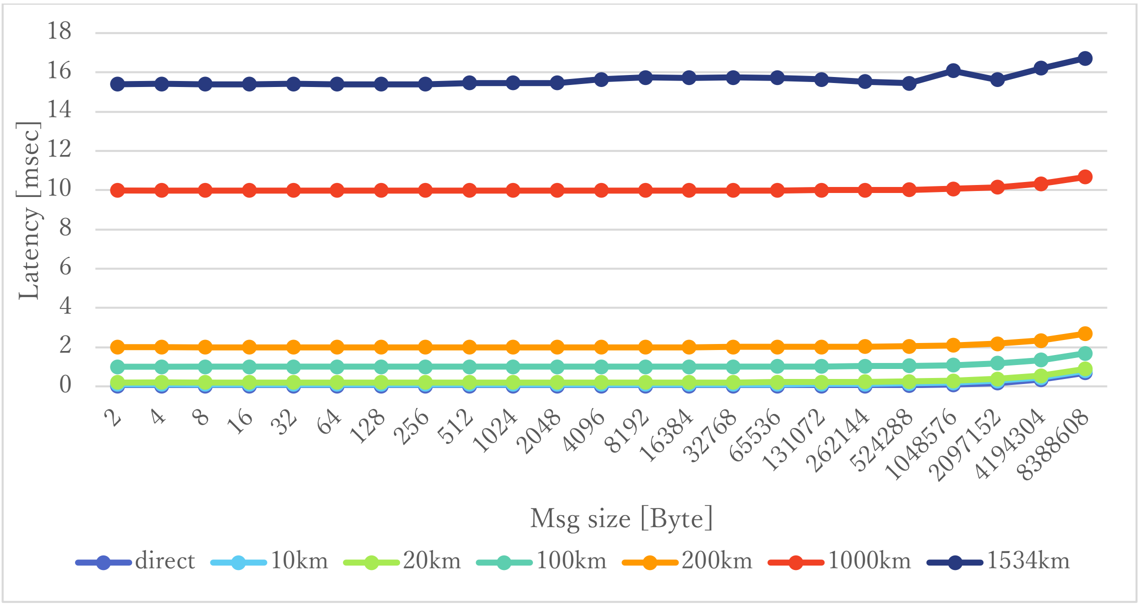


Figure 13: RDMA READ latency using NIC-offloaded RoCE at Main-to-Main

Table 8: Power consumption of READ using NIC-offloaded RoCE at Main-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.4	1.52	1.49	1.38	1.49	1.55	1.44
Sender[W]	1.35	1.52	1.55	1.54	1.5	1.54	1.51

The effect of reducing power consumption (consideration)

According to the powertop measurement results, there is the difference in throughput between Software RoCE and NIC-offloaded RoCE, but there is no difference in power consumption, both about 1.5W. To clarify this cause, we analyzed the CPU utilization of the library functions by Linux perf.

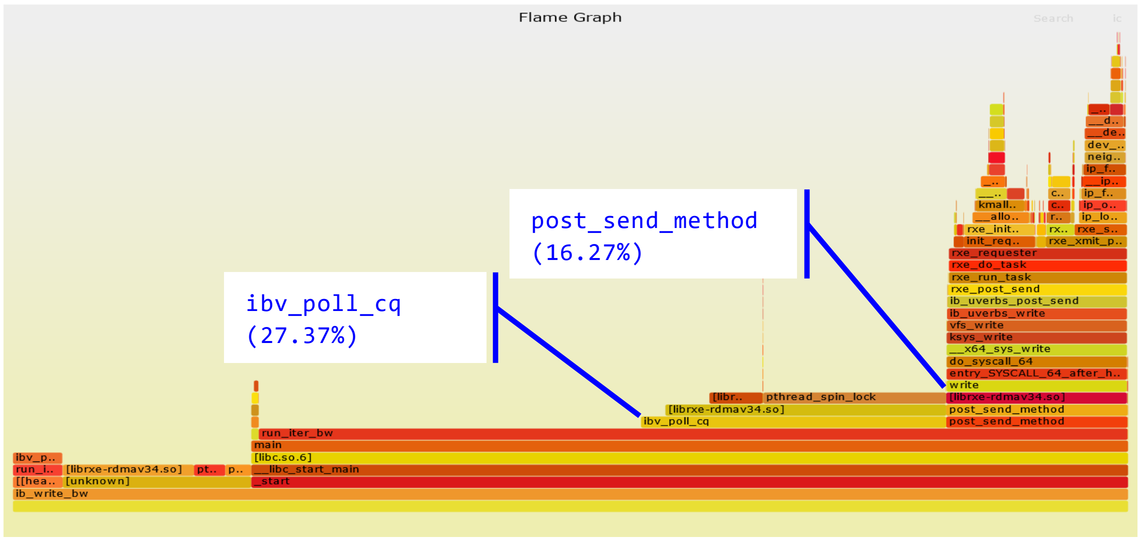


Figure 14: Linux perf sampling result of Software RoCE (Client side) at Main-to-Main (Operation=RDMA WRITE, 100km, msg size=4096B, duration(-d)=240[sec])

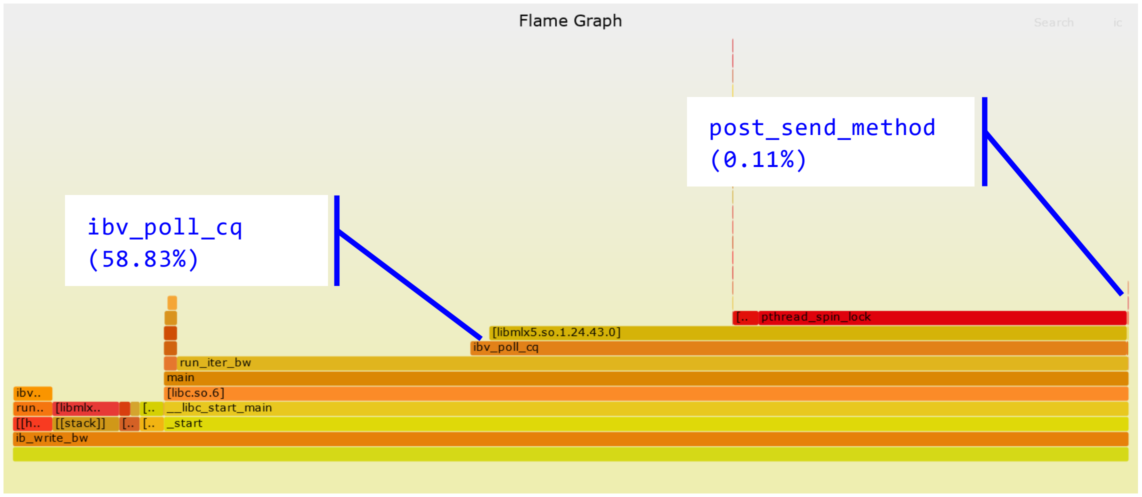


Figure 15: Linux perf sampling result of NIC-offloaded RoCE(Client side) at Main-to-Main (Operation=RDMA WRITE, 100km, msg size=4096B, duration=240[sec])

As shown in Figure 14 & Figure 15, according to Linux perf analyzing results at client side, we found that the functions "ibv_poll_cq" and "post_send_method" (equivalent to "ibv_post_send") libraries mainly accounted for the majority . Because the function "ibv_poll_cq" is polling process to extract Work Completion from CQ by host side, it is not possible to compare the amount of processing of Software RoCE and NIC-offloaded RoCE by the function "ibv_poll_cq". Therefore, we focused on the function "post_send_method" who executes RDMA protocol processing, we measured its processing time and took the method of estimating power consumption. Results of processing time and calculated power consumptions are shown in Table 9.

NIC Power Consumption is assumed as 23.6W by specification. While software function is in operation, NIC is still working even less power consumption but we cannot measure it in detail. Thus, power consumption is assumed as same as summation of powertop result (e.g. 1.4W in WRITE) and NIC spec. (e.g. 23.6W).

Table 9: Effect of reducing power consumption

	Software RoCE			NIC-offload RoCE			Reducing effect[%]
	Power [W]	Duration [sec]	PxD [Wsec]	Power [W]	Duration [sec]	PxD [Wsec]	
WRITE	1.4+23.6	117.86	**2946.50**	1.4+23.6	0.0802	**2.01**	**99.93**
SEND	1.54+23.6	114.33	**2874.26**	1.54+23.6	0.0865	**2.17**	**99.92**
READ	1.41+23.6	7.43	**185.82**	1.41+23.6	0.0813	**2.03**	**98.91**

Step.1: Conclusions

1. In the case of RDMA WRITE operation, NIC-offloaded RoCE greatly improved the maximum throughput from 20.49 Gbps to 98.27 Gbps compared to software RoCE. By increasing the queue depth of CX-6, it was obtained that ramp-up of the throughput was faster. The latency of software RoCE exponentially increased from around 1 MiB of message size, reaching 474 msec at the transmission distance of 1534 km, while it was found that the latency of NIC-offloaded RoCE was suppressed to 8.5 msec even under long message size and long transmission distance. In addition, the effect of the reducing power consumption was 99.93% compared to software RoCE, meaning a sufficient reduction.
2. Also in the case of SEND operation, NIC-offloaded RoCE greatly improved the maximum throughput from 19.44 Gbps to 98.27 Gbps compared to software RoCE. By increasing queue depth of CX-6, it was obtained that ramp-up of the throughput was faster. The latency of software RoCE exponentially increased from around 1 MiB of message size, reaching 474 msec at the transmission distance of 1534 km, while it was found that the latency of NIC-offloaded RoCE was suppressed to 8.33 msec even under long message size and long transmission distance. In addition, the effect of the reducing power consumption was 99.92% compared to software RoCE, meaning a sufficient reduction.
3. Also in the case of RDMA READ operation, NIC-offloaded RoCE greatly improved the maximum throughput from 20.04 Gbps to 98.27 Gbps compared to software RoCE. Although it was obtained that the ramp-up of the throughput was faster, this result can be obtained by increasing the QP in addition to increasing the queue depth of CX-6. If the QP is set to 1, it was found that the ramp-up of throughput was slower under the long transmission distance (see the detail in Appendix C).

The reason for the slow ramp-up of throughput with QP=1 is considered to be the Packet Sequence Number (PSN) field of the Base Transport Header in RDMA protocol. We also attached the capture data measured at QP=1 in Appendix C. According to the captured data, it was observed that the sender stopped sending RDMA Read Requests many times and it turned out that the delta of PSN was always 15 when the above phenomenon occurred. This may be a specification that the sender stops RDMA Read Request packets if the delta of PSN reaches 15. Therefore, by increasing the QP, PSN are assigned to each QP, so it is presumed that the delta of PSN will be less likely to reach 15, and the ramp-up of throughput will be faster. For RDMA Read operation, in addition to increasing queue depth of CX-6, increasing QP is an important factor in optimizing throughput.

Although the effect of the reducing power consumption in READ operation was 98.91% compared to software RoCE, its effect was smaller than that of WRITE and SEND operations. This is because RDMA WRITE and SEND operations have process that reads data from the sender memory, while RDMA READ operation has process that reads data from the receiver memory. Therefore, it is considered that the time to issue the Request packets is shorter than RDMA WRITE and SEND operations. Conversely, we also considered a method to measure the processing time for reading data at the server side. However, in perftest, it is impossible to measure it at server side because the processing of reading data is hidden by the function "ibv_poll_cq". At this time, we recognize that it is difficult to accurately calculate the reducing effect in RDMA READ operation using perftest.

RDMA Write is also called one-sided RDMA, it provides generally high performance, low CPU overhead. On the other hand, SEND is also called two-sided RDMA, it simplifies application programming and is less affected by hardware restrictions. When we conducted throughput tests with these operation types, the results were subtly different. Such difference between the throughput results on two operations needs further study.

5.2.2. Step.2: Main/GPU memory to Main/GPU memory

Objective Id:	Main/GPU memory to main/GPU memory	
Description:	The performance measurements for configuration “main/GPU memory to main/GPU memory”. PoC reference defines the following key benchmarks. Benchmark 1: throughputs for data transferring Benchmark 2: latency between RDMA endpoints (Benchmark 3: power consumption)	
Pre-conditions	None	
Procedure:	1	There are 2 types for this configuration of RDMA (GPUDirect and DMA-Buf). We demonstrate GPUDirect RDMA in terms of performance, latency.
	2	We improve GPUDirect RDMA performance by tuning.
	(3)	We verify what difference GPUDirect and DMA-Buf have in RDMA transfer performance.
Finding Details:	Objective demonstrated.	
Lessons Learnt & Recommendations	[Lessons Learnt] 1. It was found that the configuration using GPU memory has almost the same performance as the configuration using only host memories. 2. If the server side uses GPU memory, the peak performance may drop by 0.96%. [Recommendations] For better performance, verification by the configuration that both the NIC and GPU are physically placed on the same PCI switch.	

This step evaluates the performance of RDMA data transfer between the main memory and the GPU memory. We use GPUDirect RDMA which is one of typical RDMA transfer techniques. There are three configurations for RDMA transfers using GPUs: Main memory to GPU memory, GPU memory to main memory, and GPU memory to GPU memory. Here we report the evaluation results for each configuration.

As in Step.1, there are three measurement indicators: transfer performance, latency, and power consumption. To measure these indicators, we connected the Attero-100G between the NVIDIA ConnectX-6 on the sender/receiver servers and adjusted the communication distance on the Attero-100G, assuming some long-distance communication.

As in Step.1, we measured the communication distance by setting the following 7 types in the Attero-100G.

Communication Distance: Direct, 10km, 20km, 100km, 200km, 1000km, 1534km

Based on the same idea at NIC-offloaded RoCE in Step.1, we set the ConnectX-6’s queue depth to 16384 and perftest’s Tx-depth to 2048. Power consumption was also measured in the same way of Step.1.

NVIDIA ConnectX-6 and NVIDIA A100 GPU were installed in different PCIe slots in the same CPU socket. Although NVIDIA recommends installing them in the same PCI switch, we can't do that due to the hardware constraints. Therefore, in the future, it is necessary to review the environment and evaluate again.

5.2.2.1. Main memory to GPU memory

We measured the following operations according to the Linux-RDMA perftest version constraints used:

Table 10: Evaluation software of each operation type for Main-to-GPU

RDMA operation type	Perftest program
RDMA WRITE	ib_write_bw
RDMA READ	ib_read_bw, ib_read_lat

RDMA WRITE operation results

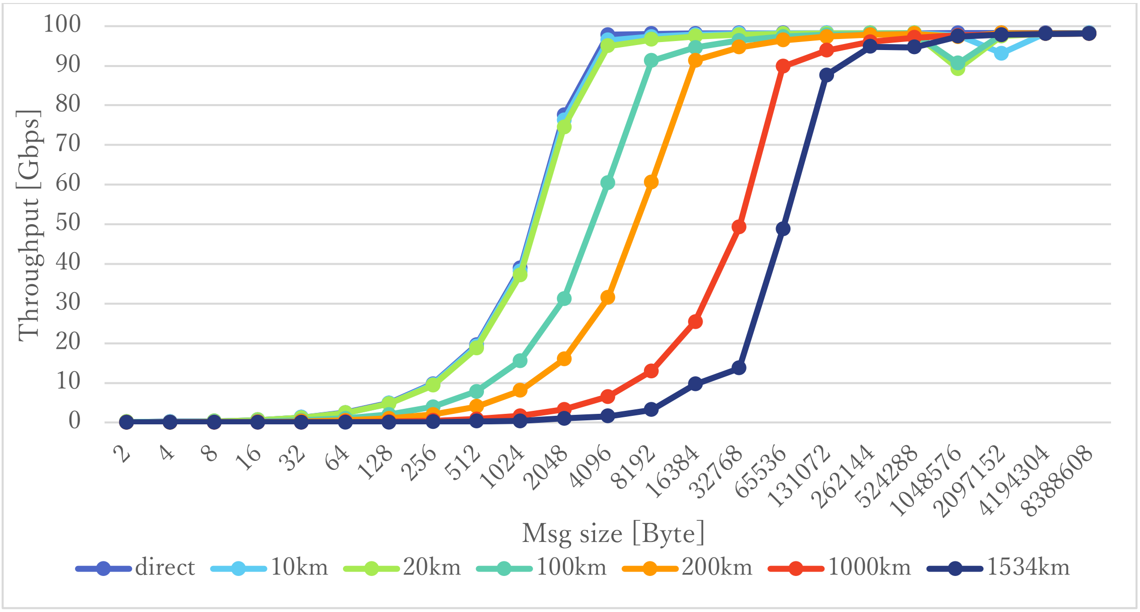


Figure 16: RDMA WRITE throughput at Main-to-GPU

Table 11: Power Consumption of RDMA WRITE at Main-to-GPU

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.53	1.26	1.28	1.26	1.44	1.36	1.54
Sender[W]	1.55	1.25	1.23	1.24	1.46	1.37	1.45

RDMA READ operation results

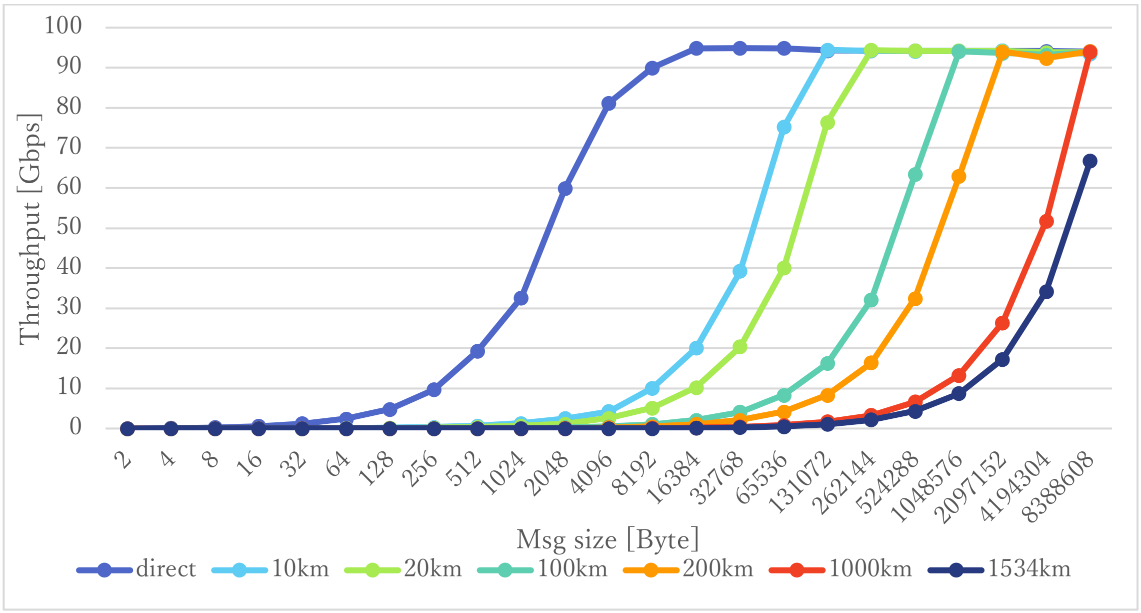


Figure 17: RDMA READ throughput at Main-to-GPU

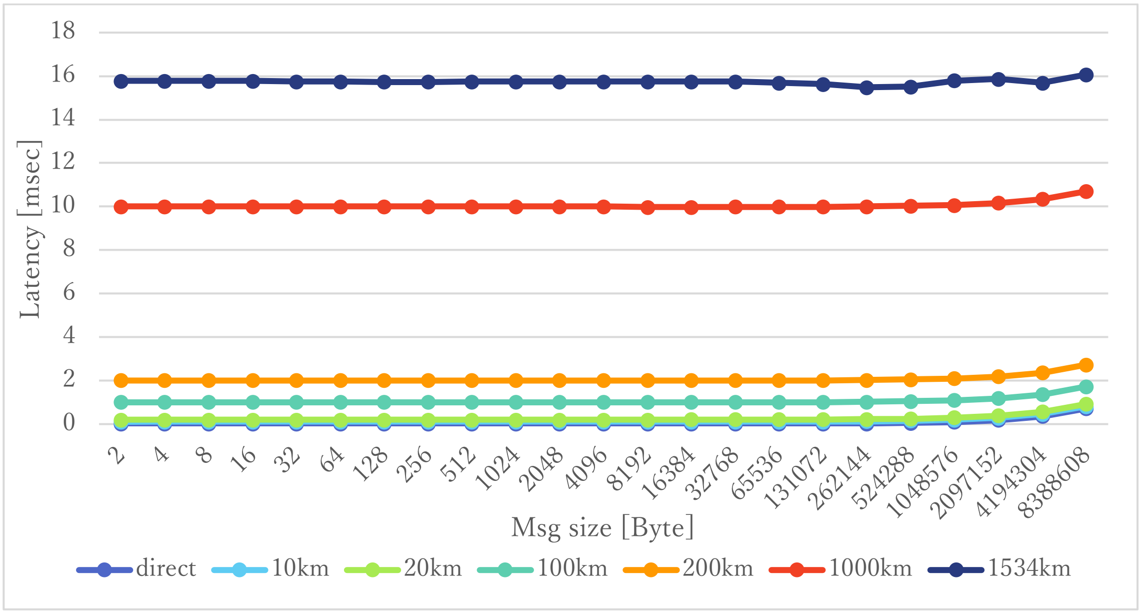


Figure 18: RDMA READ latency at Main-to-GPU

Table 12: Power Consumption of RDMA READ operation at Main-to-GPU

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.27	1.35	1.45	1.46	1.51	1.5	1.46
Sender[W]	1.48	1.34	1.45	1.52	1.51	1.53	1.53

5.2.2.2. GPU memory to main memory

We measured the following operations according to the Linux-RDMA perftest version constraints used:

Table 13: Evaluation software of each operation type for GPU-to-Main

RDMA operation type	Perftest program
RDMA WRITE	ib_write_bw
RDMA READ	ib_read_bw, ib_read_lat
SEND	ib_send_bw

RDMA WRITE operation results

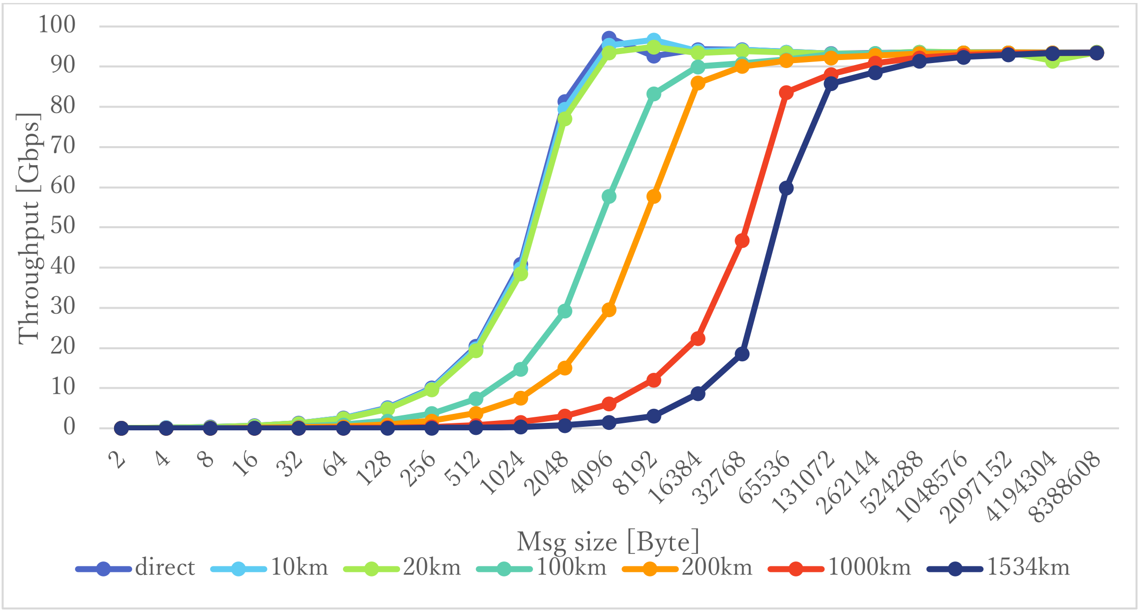


Figure 19: RDMA WRITE throughput at GPU-to-Main

Table 14: Power Consumption of RDMA WRITE at GPU-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.5	1.31	1.32	1.32	1.26	1.34	1.12
Sender[W]	1.43	1.37	1.33	1.33	1.38	1.45	1.14

RDMA READ operation results

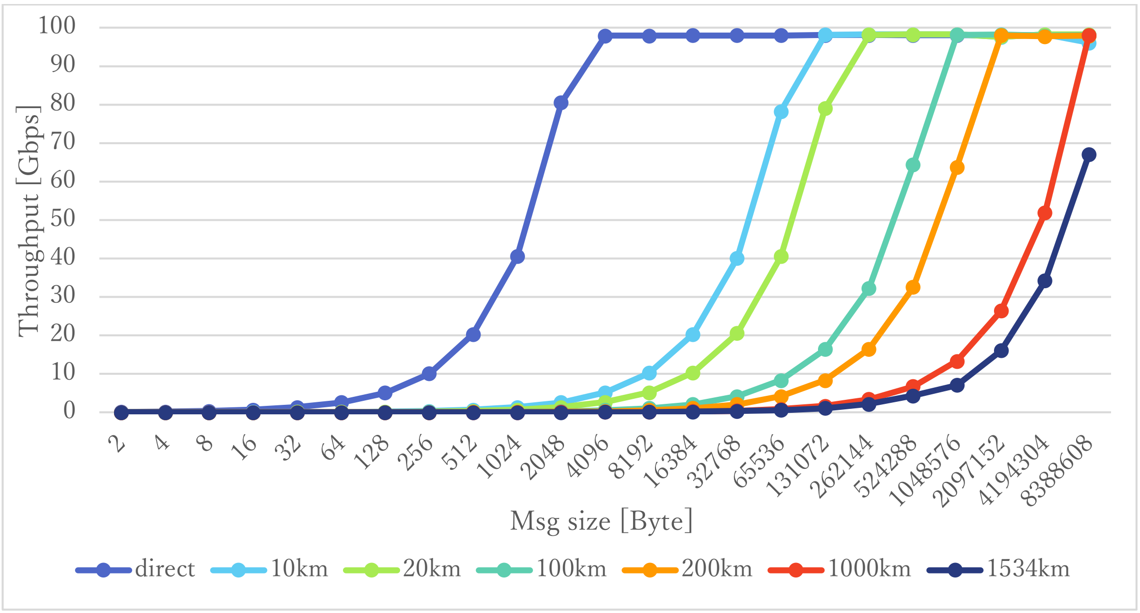


Figure 20: RDMA READ throughput at GPU-to-Main

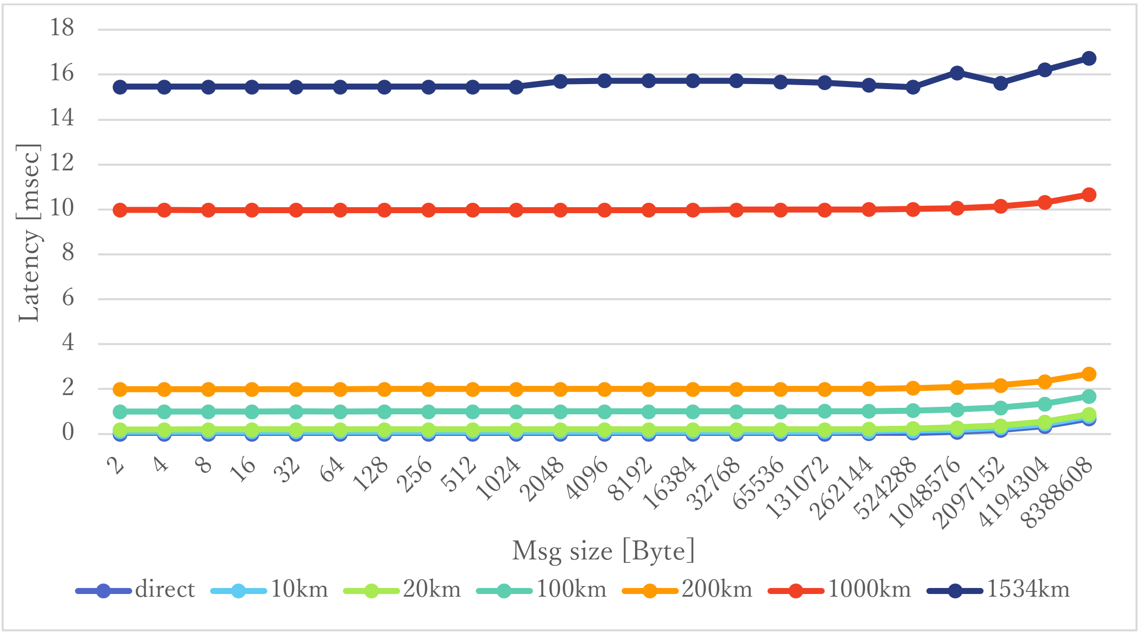


Figure 21: RDMA READ latency at GPU-to-Main

Table 15: Power Consumption of RDMA READ operation at GPU-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.52	1.27	1.32	1.55	1.55	1.53	1.51
Sender[W]	1.48	1.32	1.43	1.51	1.47	1.5	1.51

RDMA SEND operation results

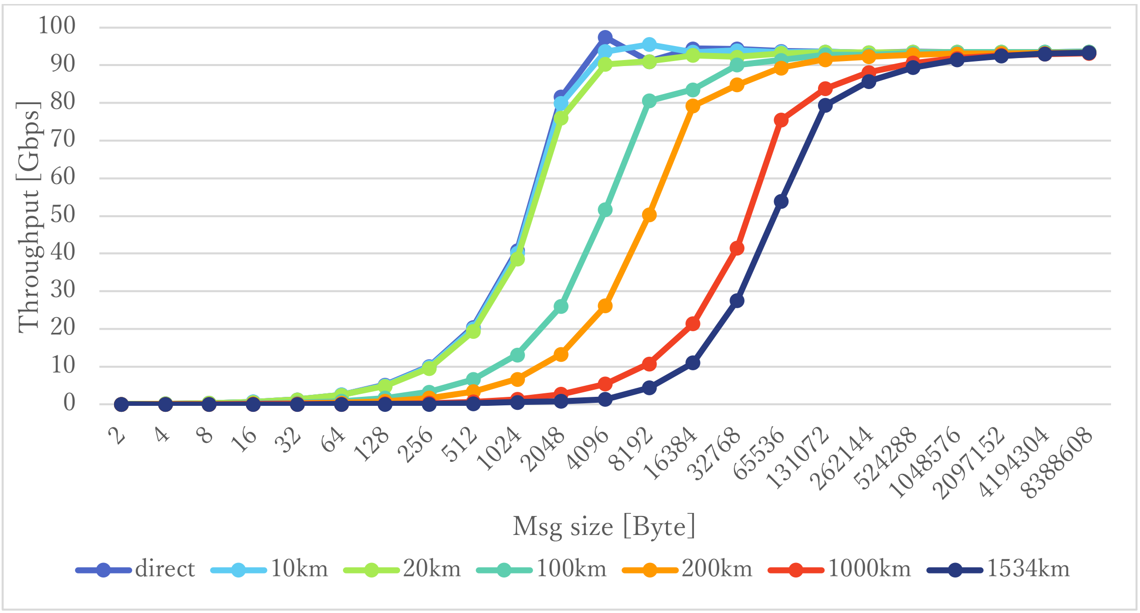


Figure 22: RDMA SEND throughput by GPUDirect RDMA at GPU-to-Main

Table 16: Power Consumption of SEND operation at GPU-to-Main

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.52	1.27	1.32	1.55	1.55	1.53	1.51
Sender[W]	1.48	1.32	1.43	1.51	1.47	1.5	1.51

5.2.2.3. GPU memory to GPU memory

We measured the following operations according to the Linux-RDMA perftest version constraints used:

Table 17: Evaluation software of each operation type for GPU-to-Main

RDMA operation type	Perftest program
RDMA WRITE	ib_write_bw
RDMA READ	ib_read_bw, ib_read_lat

RDMA WRITE operation results

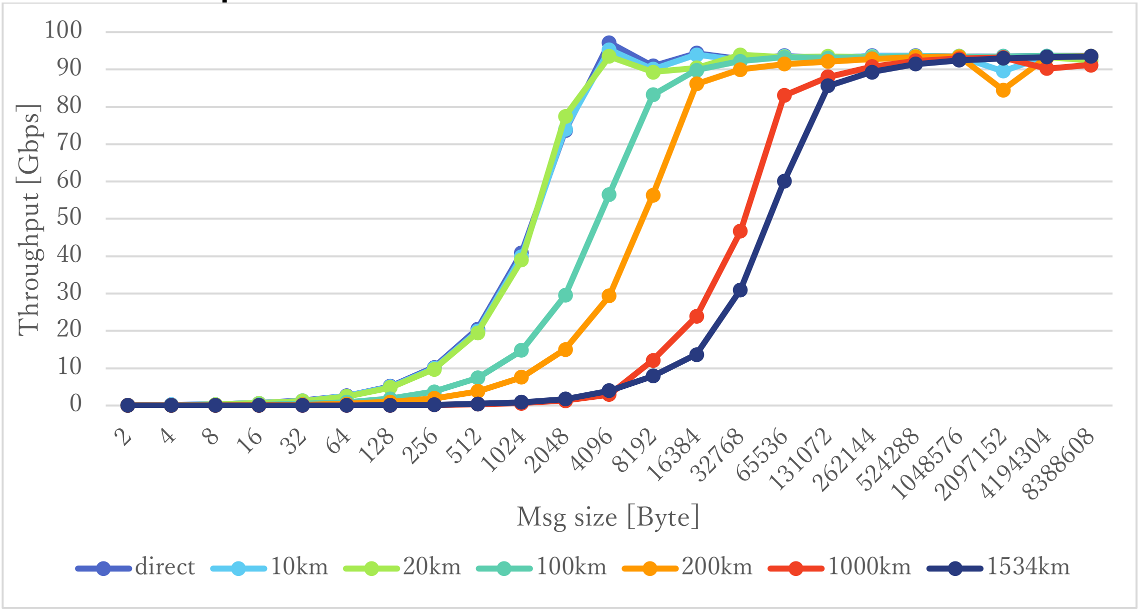


Figure 23: RDMA WRITE throughput by GPUDirect RDMA at GPU-to-GPU

Table 18: Power Consumption of RDMA WRITE at GPU-to-GPU

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.54	1.09	1.08	1.08	1.09	1.13	1.49
Sender[W]	1.54	1.11	1.1	1.13	1.1	1.17	1.51

RDMA READ operation results

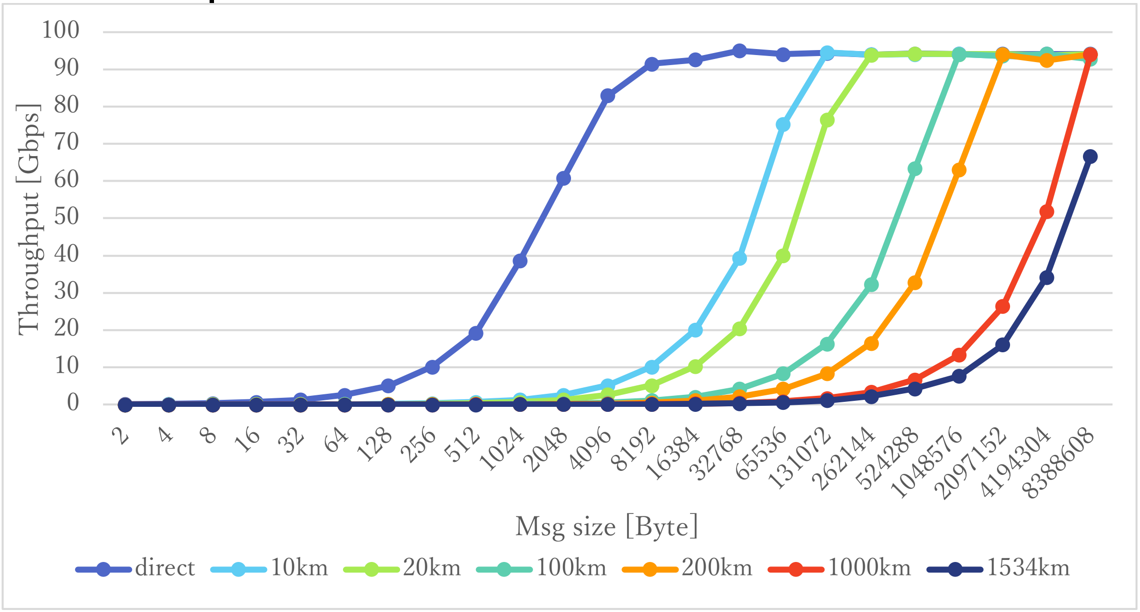


Figure 24: RDMA READ throughput by GPUDirect RDMA at GPU-to-GPU

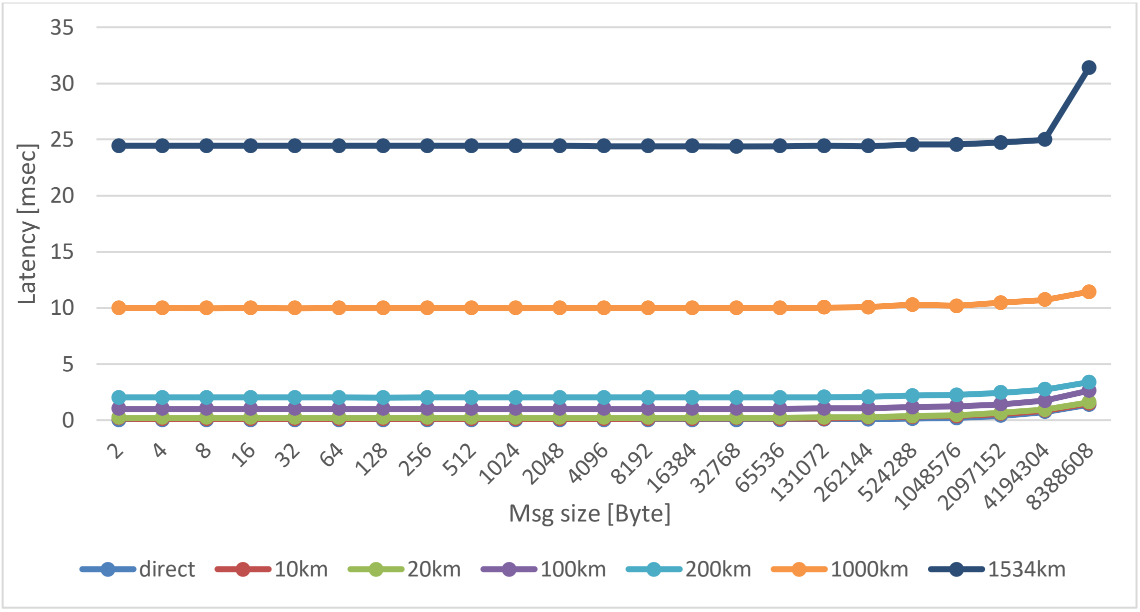


Figure 25: RDMA READ latency by GPUDirect RDMA at GPU-to-GPU

Table 19: Power of Consumption READ by GPUDirect RDMA at GPU-to-GPU

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Receiver[W]	1.55	1.46	1.53	1.53	1.54	1.48	1.58
Sender[W]	1.46	1.53	1.54	1.53	1.52	1.53	1.59

Step.2: Conclusions

1. In RDMA WRITE operation, there was no significant difference in ramp-up of throughput compared to the results of NIC-offloaded RoCE in Step.1. However, when the receiver side is GPU memory, the peak of throughput was obtained at

around 4096B of message size and decrease by 0.96% after 8192B of message size.

2. In RDMA READ operation, we measured with QP=1, there was no significant difference in ramp-up of throughput compared to the results of same QP settings at NIC-offloaded RoCE in Step.1. It was also found that the ramp-up of throughput at QP=1 was slower compared to RDMA WRITE/SEND operations. Regarding this, it is necessary to increase the QP and measure again.
   As well as RDMA WRITE operation, when the receiver side is GPU memory, it was obtained that the peak of throughput decrease by 0.96%. The latency was 9 msec higher than NIC-offloaded RoCE in Step.1 under the transmission distance of 1534 km.
3. In SEND operation, it was obtained that the peak of throughput was at around 4096B and decreased by 0.96% after 8192B of message size.
4. In this step, because it was not possible to mount the CX-6 and GPU on the same PCI switch due to the hardware restrictions, we think that the better performance could not be obtained. Therefore, in the future, it is necessary to review the environment and evaluate again.

5.2.3. Step.3: Main memory to NVMe

Objective Id:	Main memory to NVMe devices	
Description:	The performance measurements for configuration "main memory to main/GPU memory". PoC reference defines the following key benchmarks. Benchmark 1: throughputs for data transferring Benchmark 2: latency between RDMA endpoints	
Pre-conditions	None	
Procedure:	**1**	There are 2 types of configurations in NVMe-oF RDMA, Linux Kernel and SPDK. We demonstrate the performance, latency (, power) of NVMe-oF RDMA configured with SPDK.
	2	We confirm that CPU core scaling of SPDK improves the performance of NVMe-oF RDMA.
	(3)	(We confirm that the tuning of SPDK improves the transfer performance of NVMe-oF RDMA.)
Finding Details:	Maximum transfer performance of 90.4 Gbps was obtained in Sequential Read and Sequential Write. By scaling CPU cores, the performance exceeding 50 Gbps was obtained at distance of 100km, but it was not able to exceed 50 Gbps at more than distance of 200km. In the case of obtaining 50 Gbps at 100 km, the latency was suppressed less than 5 msec. But in the case of more than 200 km, the latency was over 10 msec. In order to obtain sufficient performance (50Gbps over & less than 10 msec) in long-distance communication in a configuration using Gen3 NVMe SSD, it was necessary to allocate 16 or more CPU cores at both Initiator and Target, and set I/O depth to 128 or more, and set the block size to 16384 or more. In general, with a locally mounted NVMe SSD, Sequential Read performs better than Sequential Write. But it was obtained that Sequential Write performs better than Sequential Read in NVMe-oF RDMA configuration.	
Lessons Learnt & Recommendations	[Lessons Learnt] 1. We have to investigate why Sequential Write performs better than Sequential Read in NVMe-oF RDMA configuration. [Recommendations] 1. Performance optimization by tuning SPDK. Because we applied the SPDK with default settings. 2. It is necessary to verify the performance configured with Gen4 NVMe SSD.	

This step evaluates the performance of RDMA data transfer between the main memory and the NVMe device. The one of typical RDMA transfer techniques in this step are NVMe-oF transfer by SPDK. In this report, NVMe-oF transfers with SPDK were measured to evaluate better NVMe-oF performance by software.

There are three types of performance indicators: throughput, latency, and power. To measure these indicators, we connected the Attero-100G between the NVIDIA ConnectX-6 on the initiator/target and adjusted the range on the Attero-100G, assuming some long-distance communications.
The SPDK Target consisted of four NVMe-oF subsystems, each of which was started with a separate block device assigned to each NVMe SSD. The FIO Initiator connected to four target NVMe-oF subsystems and measured transfer performance and latency.
The range pattern for the Attero-100G is similar to Step.1 and Step.2.

In this step, FIO was used as the measurement software instead of the perftest in Step.1 and Step.2.
First, we evaluate in advance whether sufficient performance could be obtained at the transmission distance of 100 km by tuning parameters such as IO depth and block size of FIO.

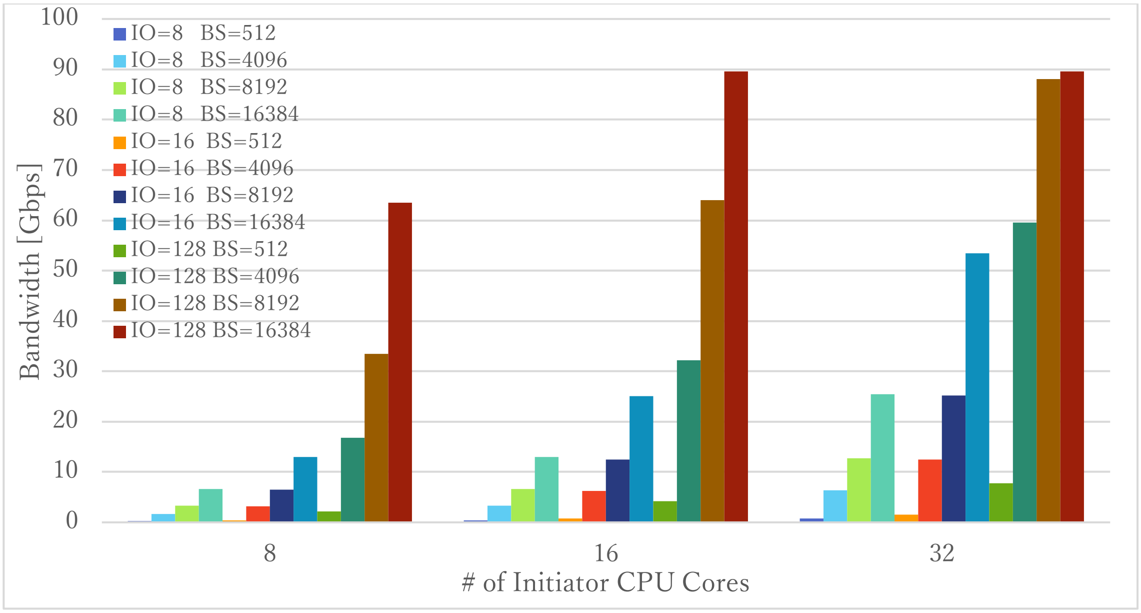


Figure 26: Initiator core scaling results at Main-to-NVMe (Bandwidth [Gbps]) (Sequential Read, Distance=100km, SPDK Target=24core, File size = 2MiB)

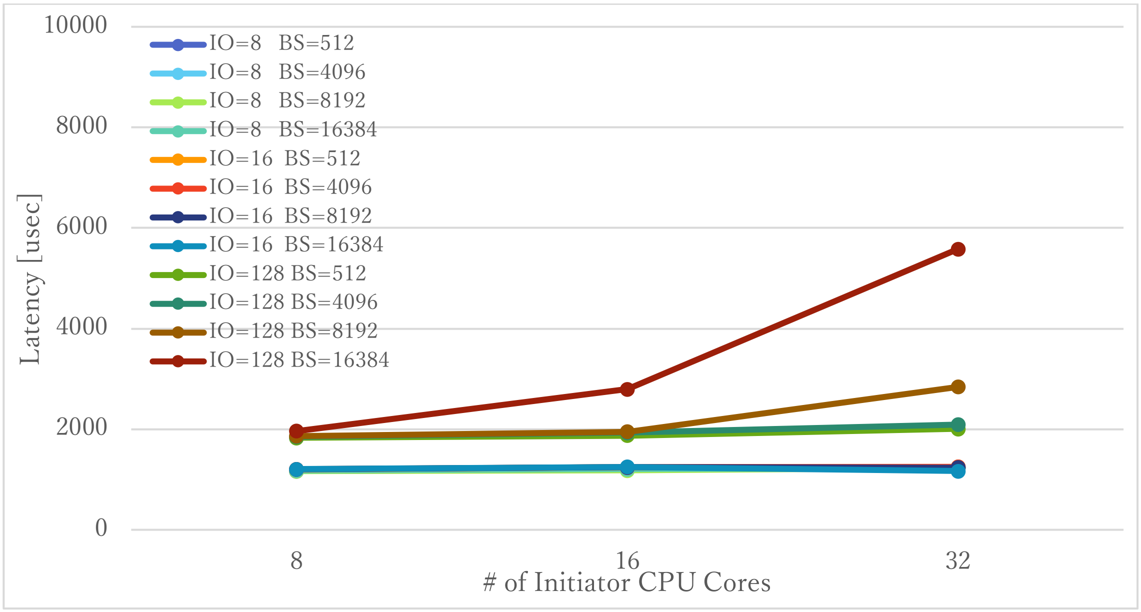


Figure 27: Initiator core scaling results at Main-to-NVMe (Latency [usec]) (Sequential Read, Distance=100km, SPDK Target=24core, File size = 2MiB)

Figure 26 and Figure 27 show the performance result of Sequential Read according to the number of cores assigned to Initiator. Throughput improves as the number of allocated CPU cores increased and as IO depth and block size increased. With 16 CPU cores or more, 128 IO depth, and a block size of 8192 or more, sufficient throughput exceeding 50 Gbps are obtained. Up to 4 msec increase in latency was observed.

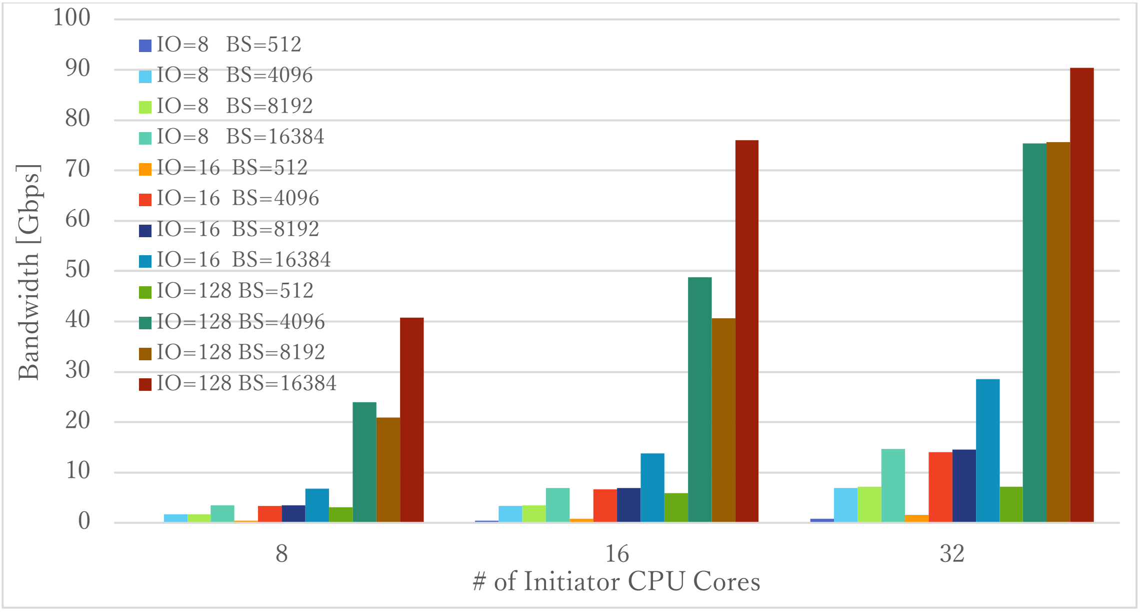


Figure 28: Initiator core scaling results at Main-to-NVMe (Bandwidth [Gbps]) (Sequential Write, Distance=100km, SPDK Target=24core, File size = 2MiB)

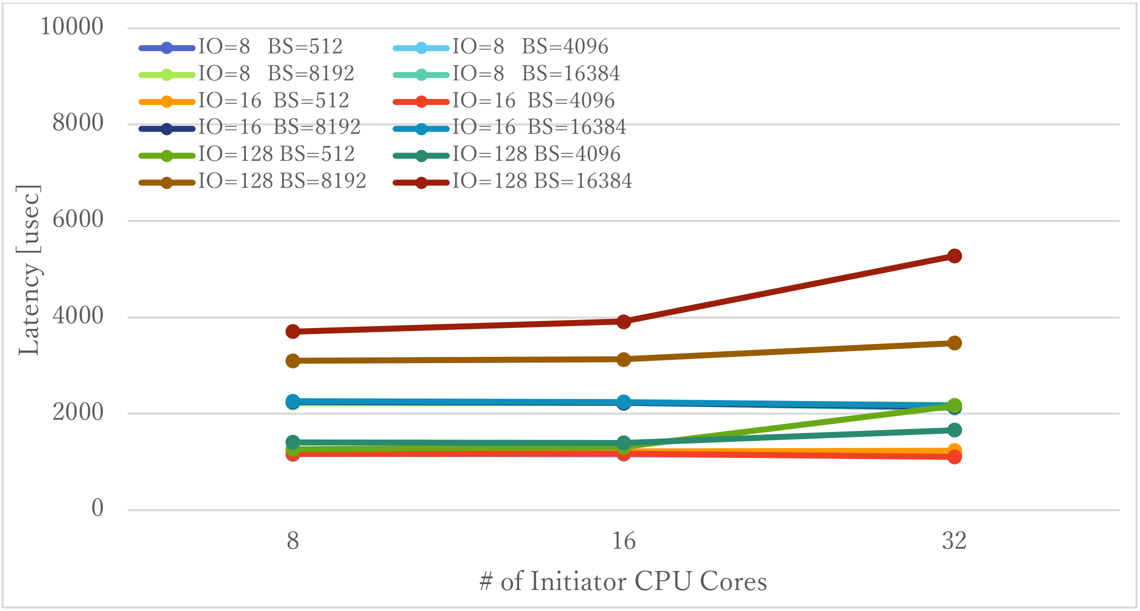


Figure 29: Initiator core scaling results at Main-to-NVMe (Latency [usec]) (Sequential Write, Distance=100km, SPDK Target=24core, File size = 2MiB)

Figure 28 and Figure 29 show the performance result of sequential write according to the number of cores assigned to the initiator. Throughput improves as the number of allocated CPU cores increased and as IO depth and block size increased. With 16 CPU cores or more, 128 IO depth, sufficient throughput exceeding 50 Gbps are obtained. Up to 2 msec increase in latency was observed.

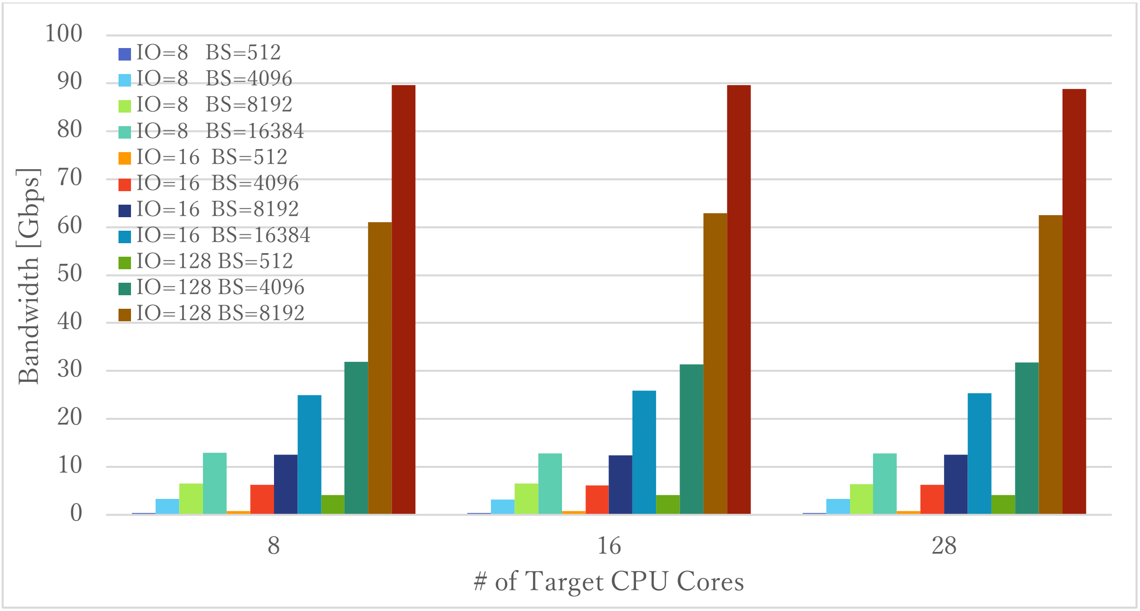


Figure 30: Target core scaling results at Main-to-NVMe (Bandwidth [Gbps])
(Sequential Read, Distance=100km, FIO jobs=16thread, File size = 2MiB)

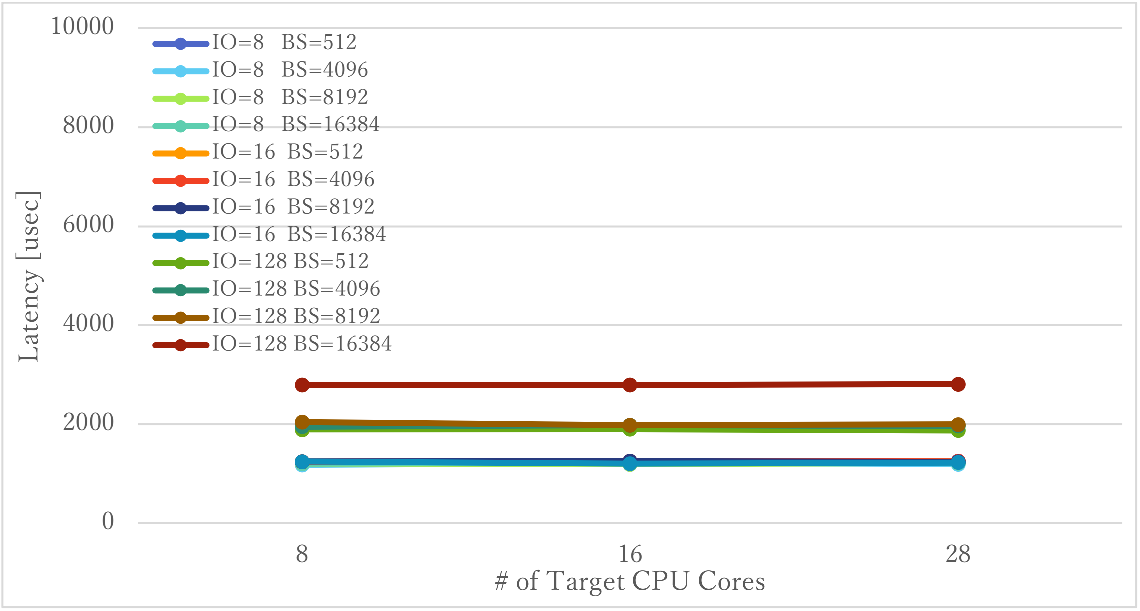


Figure 31: Target core scaling results at Main-to-NVMe (Latency[usec])
(Sequential Read, Distance=100km, FIO jobs=16thread, File size = 2MiB)

Figure 30 and Figure 31 show the performance result of Sequential Read according to the number of cores assigned to the SPDK Target. Throughput improves as the number of allocated CPU cores increased and as IO depth and block size increased. With 8 CPU cores or more, 128 IO depth, and block size of 8192 or more, sufficient throughput exceeding 50 Gbps are obtained. No increase in latency was observed.

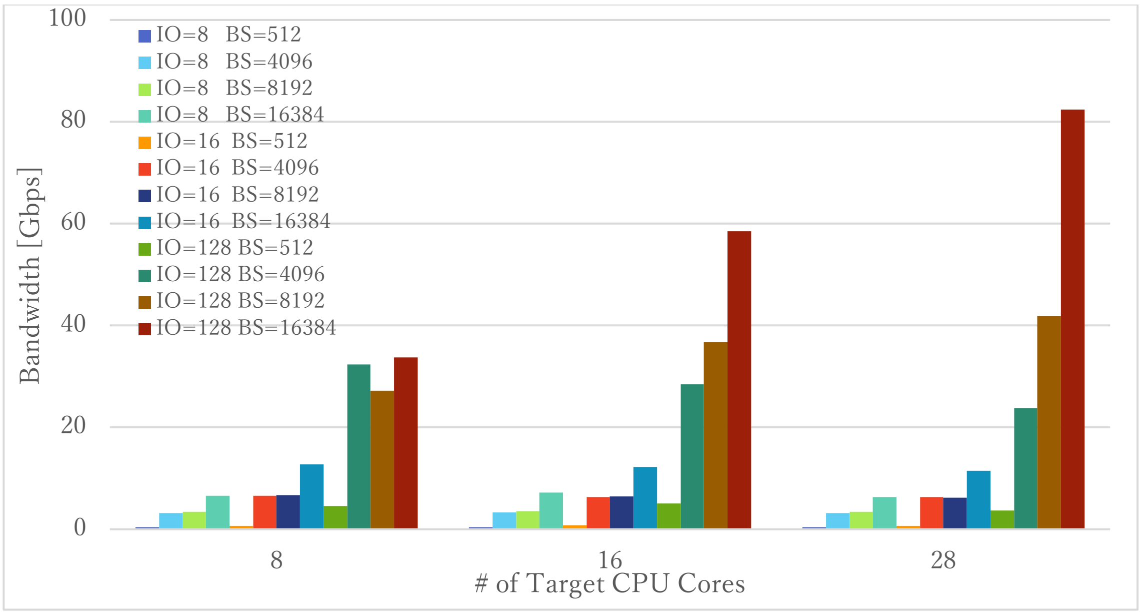


Figure 32: Target core scaling results at Main-to-NVMe (Bandwidth [Gbps])
(Sequential Write, Distance=100km, FIO jobs=16thread, File size = 2MiB)

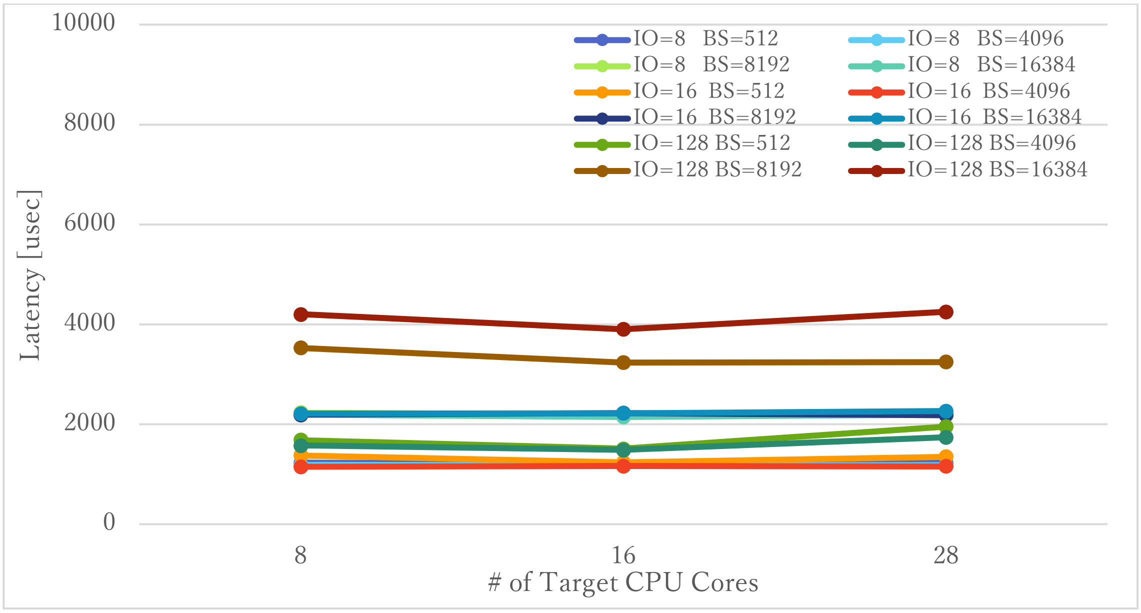


Figure 33: Target core scaling results at Main-to-NVMe (Latency [usec])
(Sequential Write, Distance=100km, FIO jobs=16thread, File size = 2MiB)

Figure 32 and Figure 33 show the performance result of sequential write according to the number of cores assigned to the SPDK target. Throughput improves as the number of allocated CPU cores increased and as IO depth and block size increased. With 16 CPU cores or more, 128 IO depth, sufficient throughput exceeding 50 Gbps are obtained. No increase in latency was observed.

From Figure 26 to Figure 33, we decide setting IO depth to 128, block size to 16384, and setting the core allocation of Initiator/Target to 16 or more as parameters to obtain sufficient throughput (50 Gbps or more) at a transmission distance of 100 km.

Table 20 shows the parameters for performance evaluating using FIO and SPDK.

Table 20: Patterns of each parameter for evaluating storage performance

Parameters	Patter values
Workload	Sequence Read/Sequence Write
I/O depth	128
I/O Block size	16384
filesize	I/O Block size ~ 2MiB
CPU cores for SPDK Target	16/28 (If fixing for initiator scaling, 16 cores)
CPU cores for Initiator	16/32 (If fixing for target scaling, 24 cores)

5.2.3.1. Initiator Scaling

Initiator Scaling evaluates how throughput changes with the number of CPU cores allocated to the initiator. SPDK target starts with 16 CPU cores and we measure the performance when FIO initiator CPU cores were allocated up to 32.

Please see measuring commands of SPDK target and FIO in Appendix E.

Sequential Read results

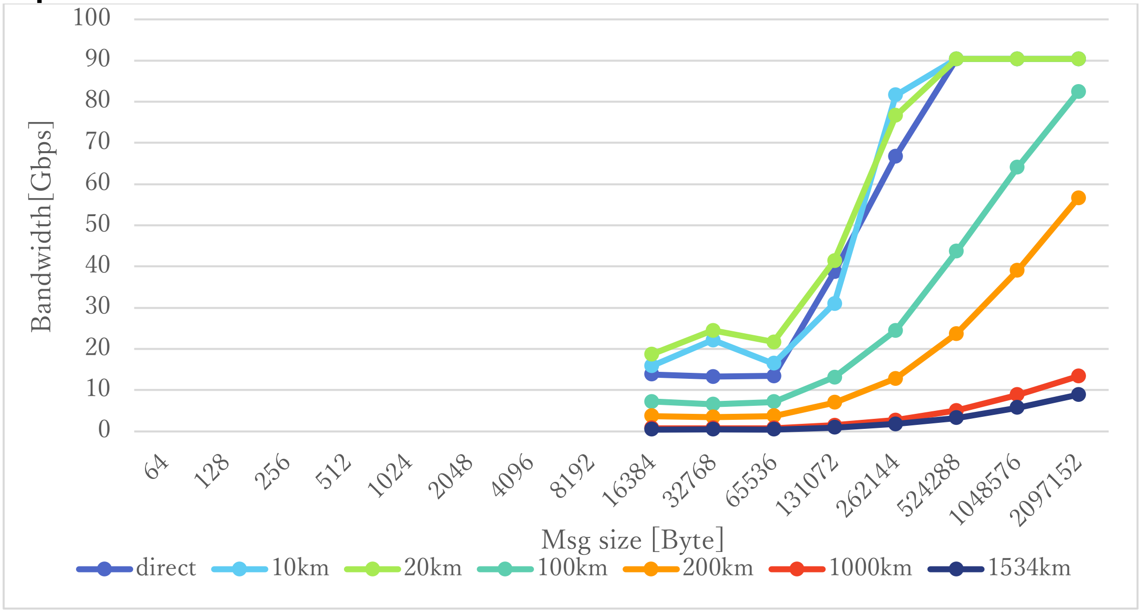


Figure 34: Sequential Read throughput at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

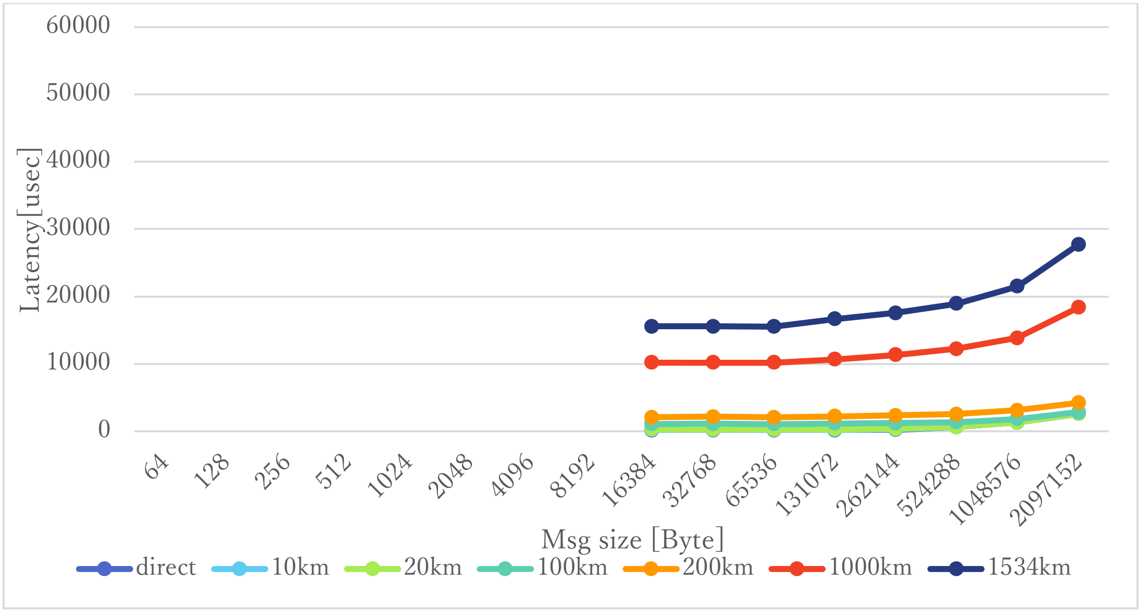


Figure 35: Sequential Read latency at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

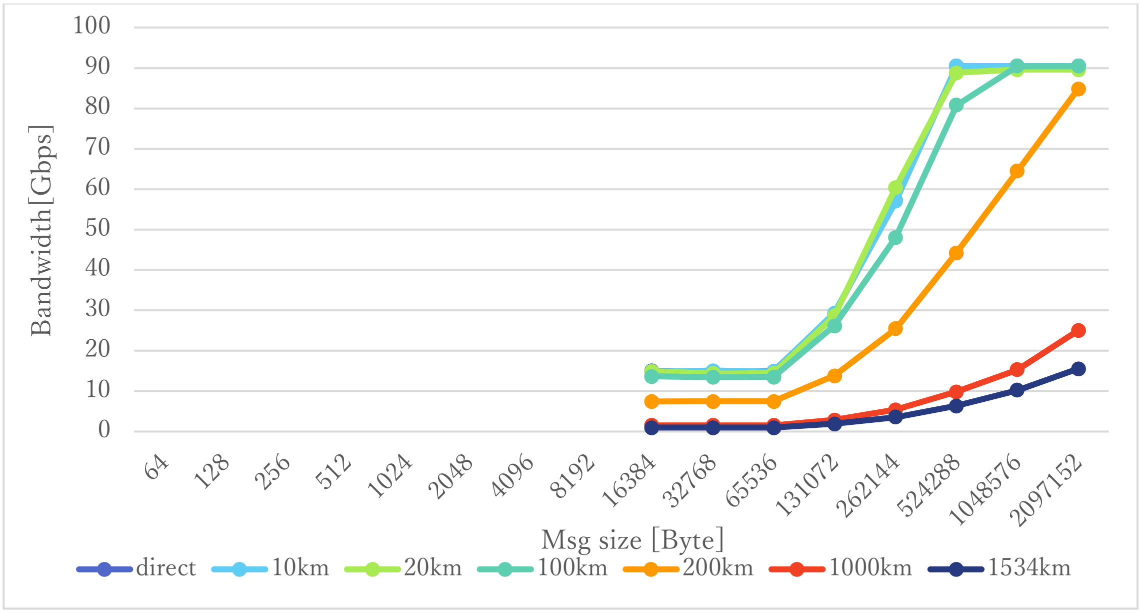


Figure 36: Sequential Read throughput at CPU 32 cores at Main-to-NVMe (BS=16384, QD=128)

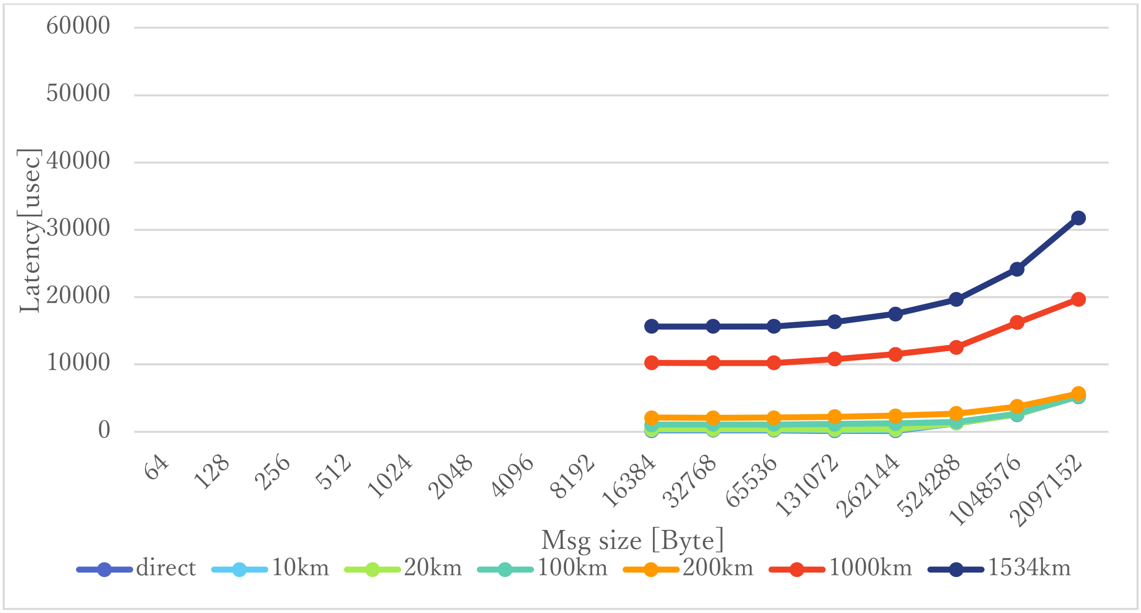


Figure 37: Sequential Read latency at CPU 32 cores at Main-to-NVMe (BS=16384, QD=128)

Sequential Write results

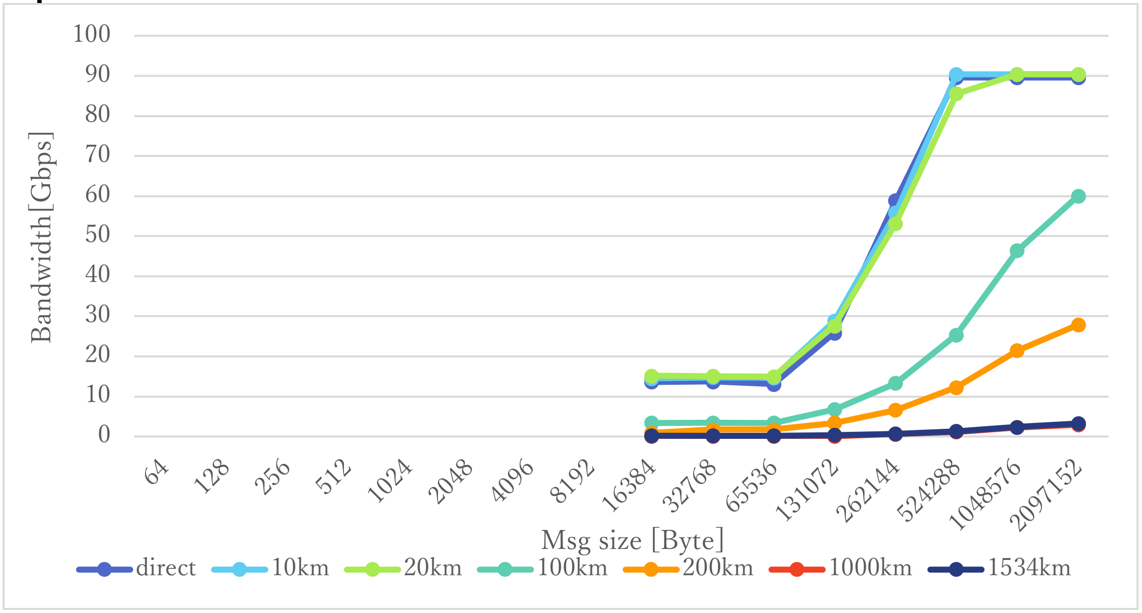


Figure 38: Sequential Write throughput at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

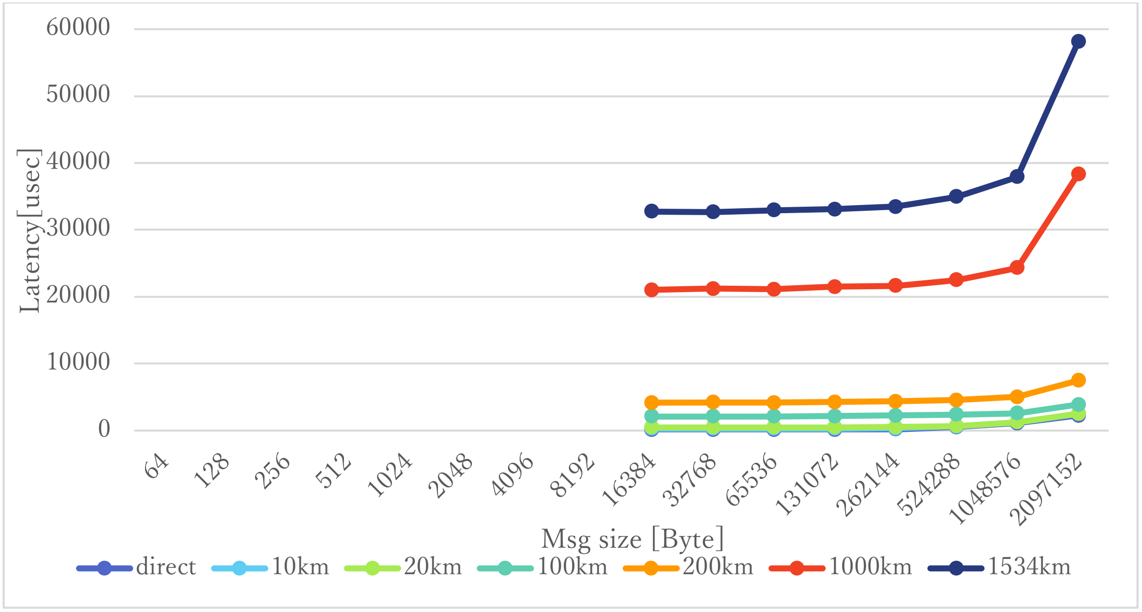


Figure 39: Sequential Write latency at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

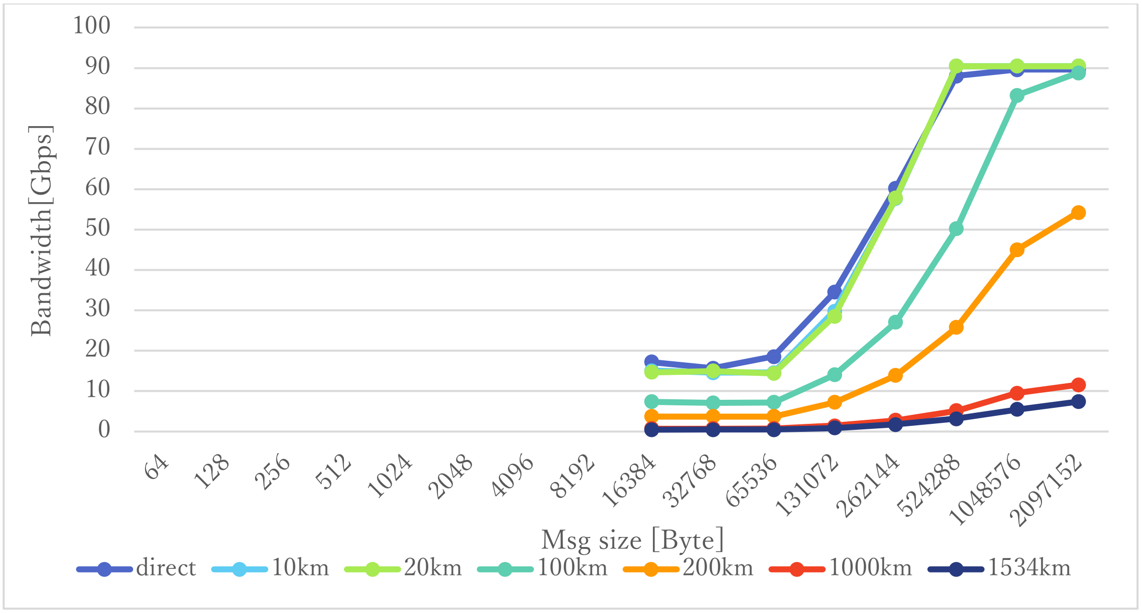


Figure 40: Sequential Write throughput at CPU 32 cores at Main-to-NVMe (BS=16384, QD=128)

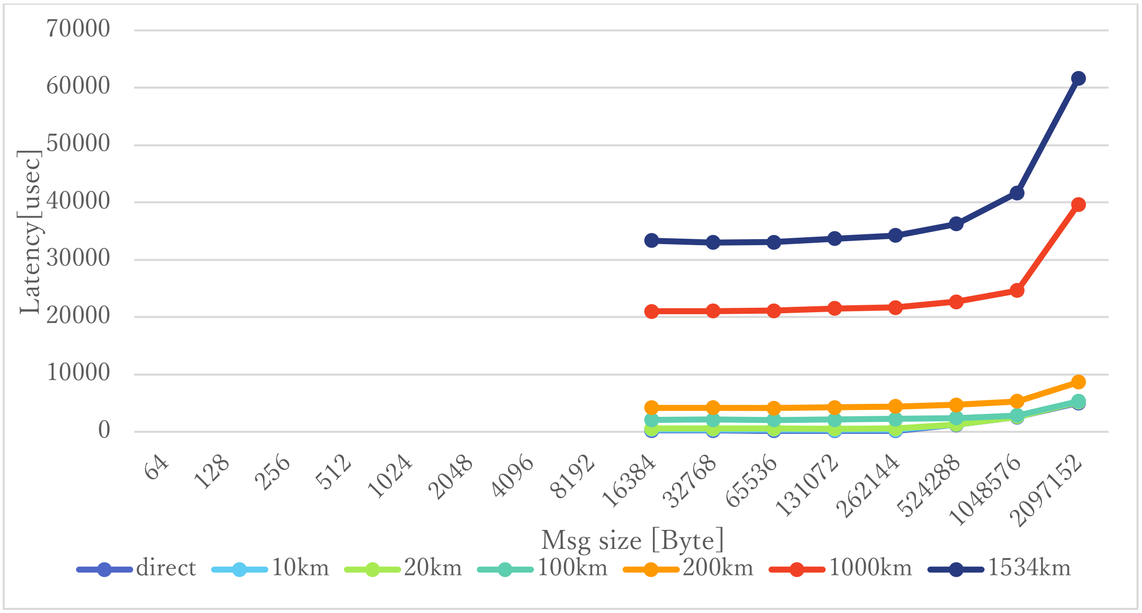


Figure 41: Sequential Write latency at CPU 32 cores at Main-to-NVMe (BS=16384, QD=128)

Table 21: Power Consumption of Sequential Read in Initiator Scaling at Main-to-NVMe (16core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	7.60	11.57	20.45	20.61	17.39	16.08	3.60
Target[W]	30.67	30.39	30.43	30.61	31.22	30.76	30.74

Table 22: Power Consumption of Sequential Read in Initiator Scaling at Main-to-NVMe (32core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	15.76	16.76	13.09	17.66	17.48	31.55	5.15
Target[W]	30.40	30.60	30.77	30.67	30.72	30.334	30.90

Table 23: Power Consumption of Sequential Write in Initiator Scaling at Main-to-NVMe (16core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	7.45	10.20	11.67	20.98	12.55	1.51	1.84
Target[W]	30.33	30.53	30.87	30.58	31.49	30.78	30.63

Table 24: Power Consumption of Sequential Write in Initiator Scaling at Main-to-NVMe (32core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	6.42	9.07	10.40	17.52	12.34	2.67	3.35
Target[W]	30.22	30.41	29.91	31.01	30.78	30.35	30.96

5.2.3.2. Target Scaling

Target Scaling evaluates how throughput changes with the number of CPU cores allocated to the SPDK target. SPDK target starts with CPU cores allocations starting in two patterns from 16 to 28. And we measure the performance when FIO initiator starts with the fixed number of CPU cores.

Please see the measuring commands of SPDK target and FIO in Appendix E.

Sequential Read results

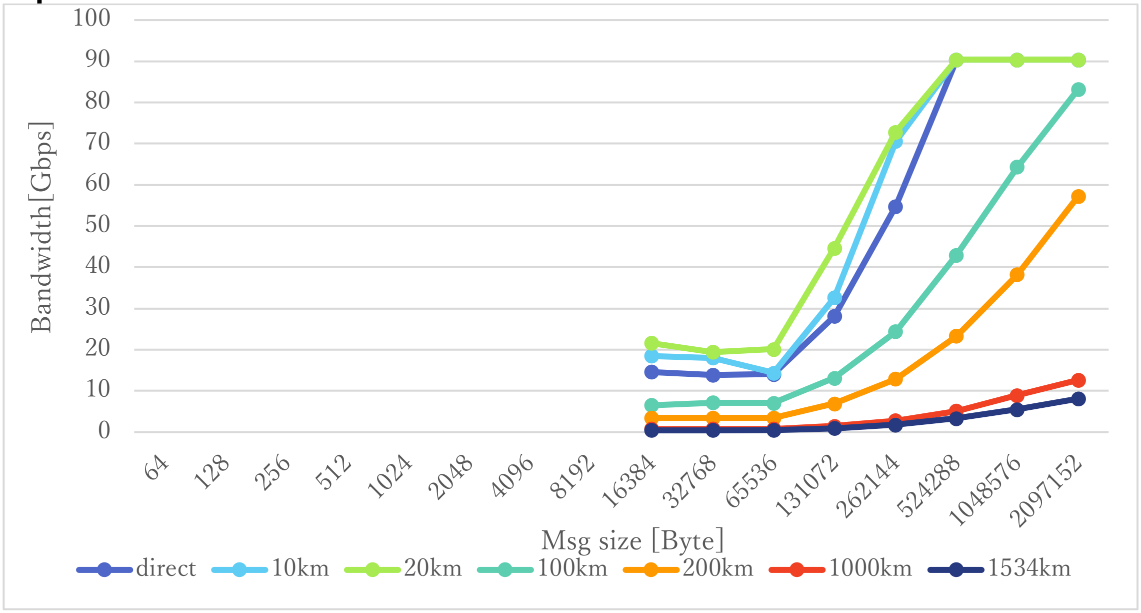


Figure 42: Sequential Read throughput at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

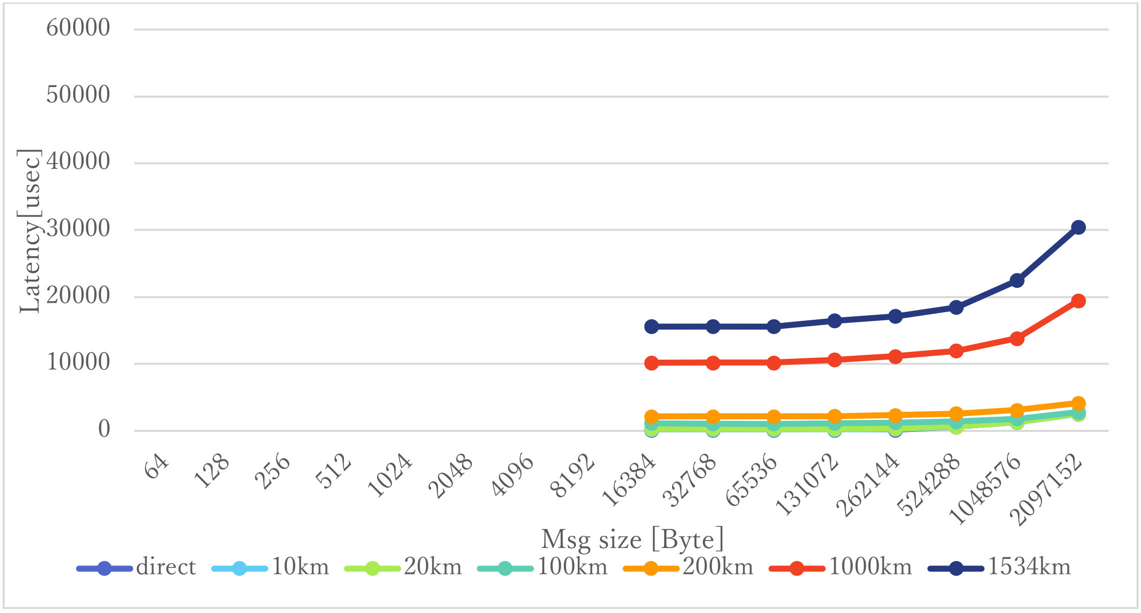


Figure 43: Sequential Read latency at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

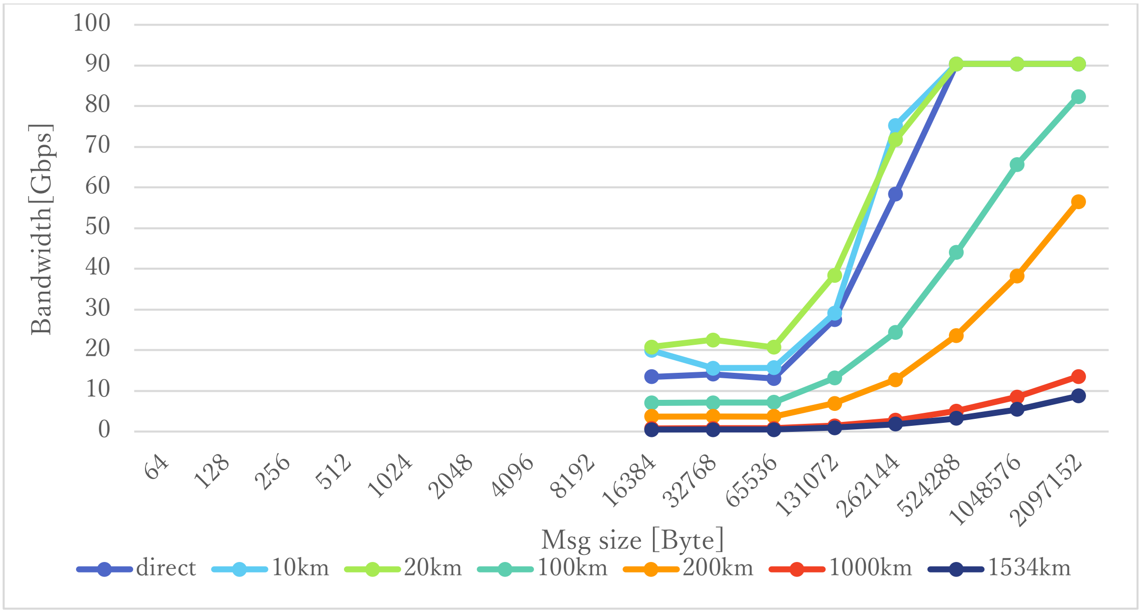


Figure 44: Sequential Read throughput at CPU 28 cores at Main-to-NVMe (BS=16384, QD=128)

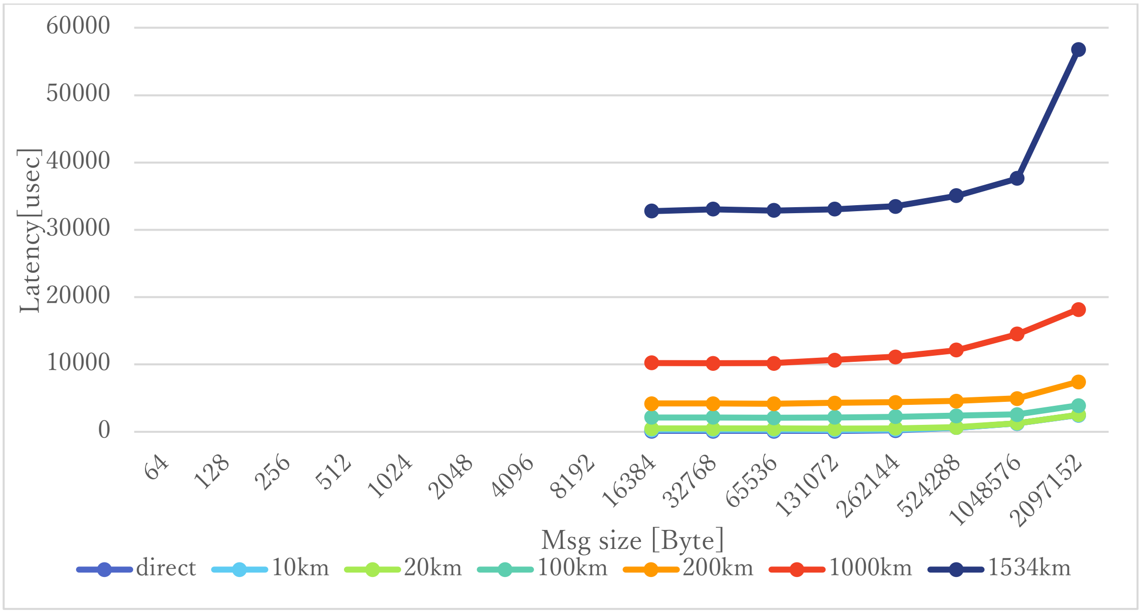


Figure 45: Sequential Read latency at CPU 28 cores at Main-to-NVMe (BS=16384, QD=128)

Sequential Write results

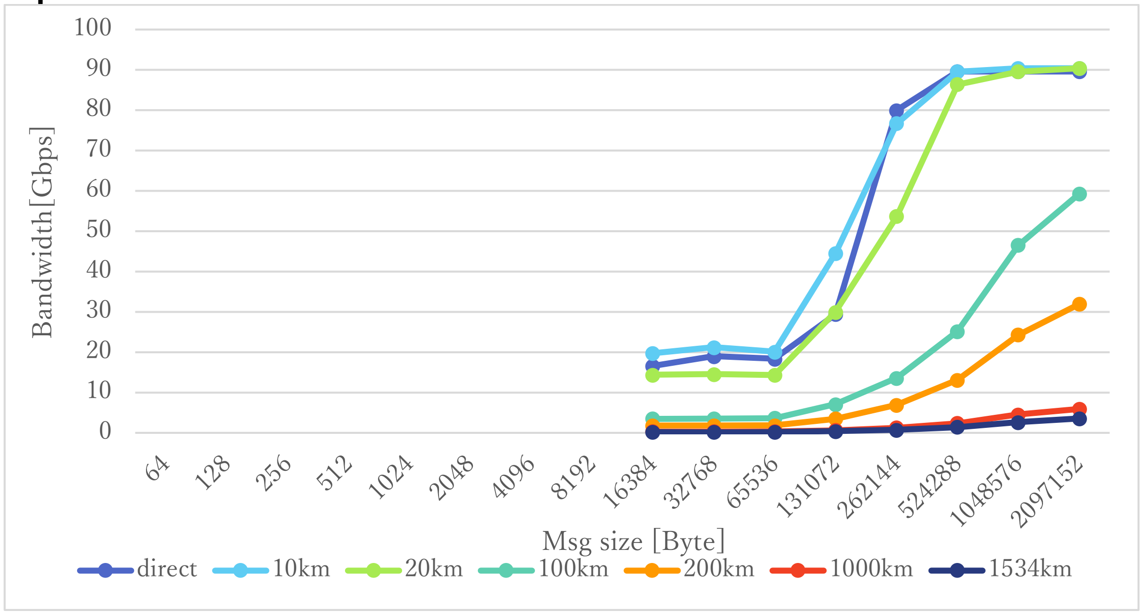


Figure 46: Sequential Write throughput at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

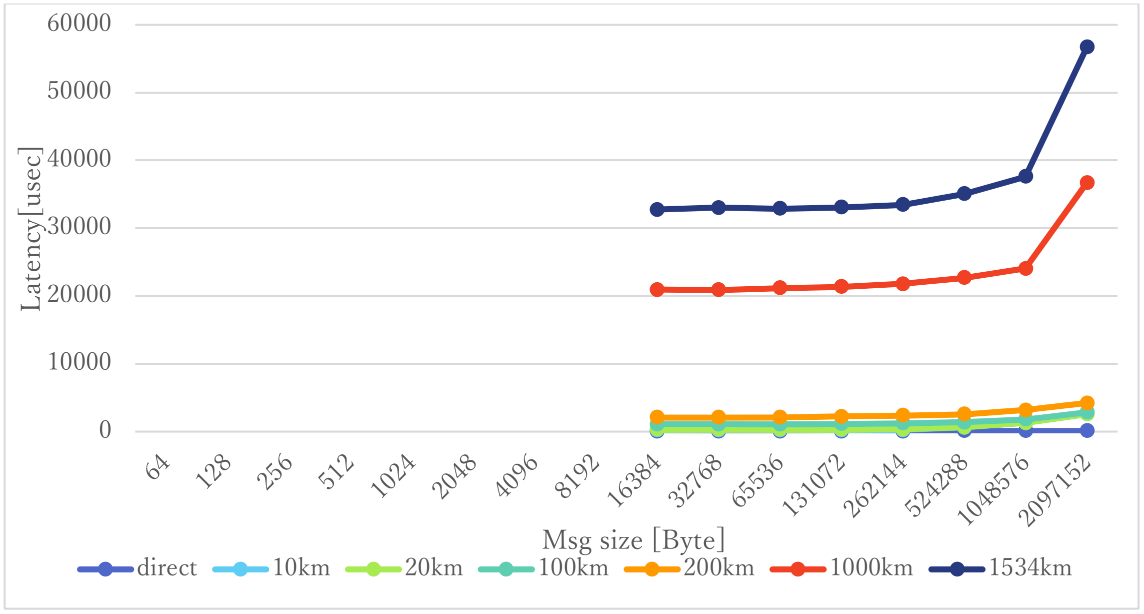


Figure 47: Sequential Write latency at CPU 16 cores at Main-to-NVMe (BS=16384, QD=128)

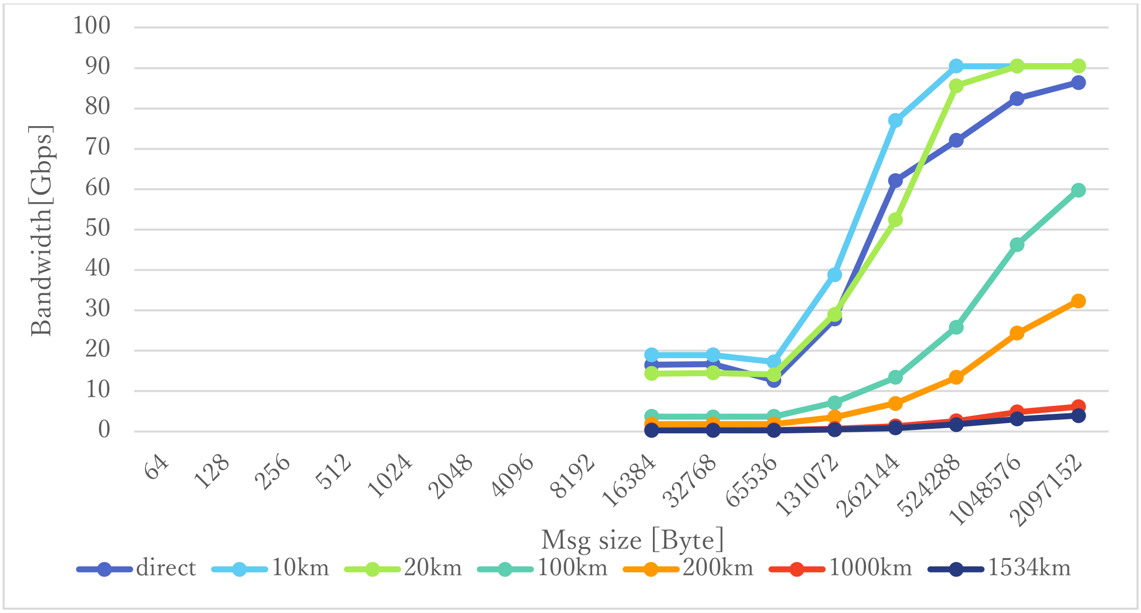


Figure 48: Sequential Write throughput at CPU 28 cores at Main-to-NVMe (BS=16384, QD=128)

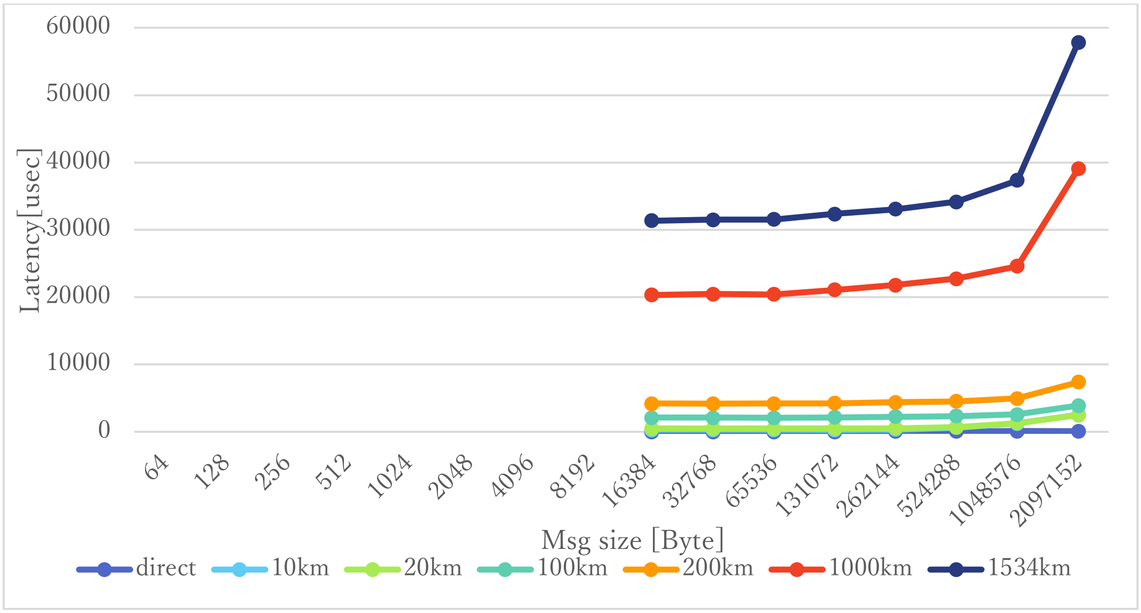


Figure 49: Sequential Write latency at CPU 28 cores at Main-to-NVMe (BS=16384, QD=128)

Table 25: Power Consumption of Sequential Read in Target Scaling at Main-to-NVMe (16core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	5.37	19.08	14.58	20.80	23.74	18.73	3.074
Target[W]	20.33	20.63	21.14	20.69	20.06	20.51	20.36

Table 26: Power Consumption of Sequential Read in Target Scaling at Main-to-NVMe (28core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	20.14	9.74	6.51	21.87	3.27	2.85	2.26
Target[W]	35.37	35.61	36.06	35.97	35.55	35.56	35.68

Table 27: Power Consumption of Sequential Write in Target Scaling at Main-to-NVMe (16core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	6.96	10.16	9.33	17.00	2.71	3.58	1.67
Target[W]	20.04	20.16	20.18	20.17	20.43	20.08	20.11

Table 28: Power Consumption of Sequential Write in Target Scaling at Main-to-NVMe (28core)

Distance	Direct	10km	20km	100km	200km	1000km	1534km
Initiator[W]	4.71	8.09	13.87	10.86	3.46	2.47	2.24
Target[W]	35.18	35.34	35.63	35.90	36.47	35.93	35.31

Step.3: Conclusions

1. In the case of Initiator Scaling, we obtained a maximum throughput of 90.4 Gbps at both Sequential Read and Sequential Write. At a transmission distance of 100 km, the maximum throughput was also 90.4 Gbps at both workloads, and the latency was suppressed to within 10 msec. However, when the transmission distance is 1000 km or more, the maximum throughput was 25 Gbps at Sequential Read and the latency exceeded 10 msec.
   Throughput improved as the number of allocated CPU cores on the Initiator increased. Since the number of allocated CPU cores on the Initiator is equivalent to the number of QP of RDMA, it can be said that it matches the performance by QP as in Step.1.
2. In the case of Target Scaling, we obtained the maximum throughput of 90.4 Gbps at both Sequential Read and Sequential Write. At a transmission distance of 100 km, the maximum throughput was 83.2 Gbps at Sequential Read and 59.7 Gbps at Sequential Read respectively, and the latency was suppressed to within 10 msec. However, when the transmission distance is 1000 km or more, maximum throughput was 13.5 Gbps and the latency exceeded 10 msec.
   Throughput improvement by the number of allocated CPU cores on the SPDK Target was not observed remarkably. To clarify the improvement of throughput depending on Target Scaling, we believe that it is necessary to finely set the number of CPU cores allocations in patterns with a small number and verify the performance again.
3. The maximum transfer performance according to the catalog value of the NVMe device used in this report is 3,500 MB/s at Sequential Read and 3,300 MB/s at Sequential Write per one SSD, respectively. In this evaluation, it was found that 80 % or more throughput was achieved.
4. As for characteristics of NVMe-oF RDMA performance, it was found that the larger the message size, the higher throughput, and the higher latency at both Sequential Read and Sequential Write. At this point, it was similar to Step.1.

5.3. Summary

This report showcased performance results of RDMA transfer with the following test cases:

- Step.1: Main memory to main memory
- Step.2: Main/GPU memory to GPU/main memory
- Step.3: Main memory to NVMe device

By comparing the performance of Software RoCE and NIC-offloaded RoCE, it was demonstrated that NIC-offloaded RoCE has the following advantages.

- ✓ Improve the maximum throughput.
- ✓ Faster the ramp-up of the throughput due to the message size.
- ✓ Possible to suppress the latency within 10 msec.
- ✓ Possible to reduce the power consumption by 98% or more.

In addition, in the case of evaluation using GPU memory, since the performance was almost equivalent to NIC-offloaded RoCE in Step.1, therefore the following effect has been demonstrated:

- ✓ Possible to maintain the same performance even if it is transferred directly to GPU memory without passing through main memory.

Additionally, in the case of evaluation of NVMe-oF RDMA using NVMe device, the following has been demonstrated.

- ✓ Possible to achieve the performance close to the physical rate (100 Gbps) at a transmission distance of 100 km.
- ✓ Possible to suppress the latency to within 10 msec. This can be said to meet storage performance requirements.

This report provided lots of information regarding methodologies and practices while benchmarking performance over several transmission distance of RDMA transfers using commodity NIC, GPU, and NVMe device. It should be noted that the performance data showcased in this report is based on specific hardware and software configurations and that performance results may vary depending on different hardware and software configurations.

Appendix A. Step.1: NVIDIA NIC configuration

Item	Description

ibv_devinfo	
	```
	$ ibv_devinfo
	hca_id: mlx5_0
	        transport:                      InfiniBand (0)
	        fw_ver:                         20.35.2000
	        node_guid:                      88e9:a4ff:ff29:1810
	        sys_image_guid:                 88e9:a4ff:ff29:1810
	        vendor_id:                      0x02c9
	        vendor_part_id:                 4123
	        hw_ver:                         0x0
	        board_id:                       MT_0000000594
	        phys_port_cnt:                  1
	                port:   1
	                        state:                  PORT_ACTIVE (4)
	                        max_mtu:                4096 (5)
	                        active_mtu:             4096 (5)
	                        sm_lid:                 0
	                        port_lid:               0
	                        port_lmc:               0x00
	                        link_layer:             Ethernet

	hca_id: mlx5_1
	        transport:                      InfiniBand (0)
	        fw_ver:                         20.35.2000
	        node_guid:                      88e9:a4ff:ff29:1811
	        sys_image_guid:                 88e9:a4ff:ff29:1810
	        vendor_id:                      0x02c9
	        vendor_part_id:                 4123
	        hw_ver:                         0x0
	        board_id:                       MT_0000000594
	        phys_port_cnt:                  1
	                port:   1
	                        state:                  PORT_ACTIVE (4)
	                        max_mtu:                4096 (5)
	                        active_mtu:             4096 (5)
	                        sm_lid:                 0
	                        port_lid:               0
	                        port_lmc:               0x00
	                        link_layer:             Ethernet
	```

mlxconfig	
	```
	Device #1:
	----------

	Device type:    ConnectX6
	Name:           MCX653106A-HDA_HPE_Ax
	Description:    HPE InfiniBand HDR/Ethernet 200Gb 2-port QSFP56 PCIe4 x16
	MCX653106A-HDAT Adapter
	Device:         mlx5_0

	Configurations:                                      Next Boot
	         MEMIC_BAR_SIZE                              0
	         MEMIC_SIZE_LIMIT                            _256KB(1)
	         HOST_CHAINING_MODE                          DISABLED(0)
	         HOST_CHAINING_CACHE_DISABLE                 False(0)
	         HOST_CHAINING_DESCRIPTORS                   Array[0..7]
	         HOST_CHAINING_TOTAL_BUFFER_SIZE             Array[0..7]
	         FLEX_PARSER_PROFILE_ENABLE                  0
	         FLEX_IPV4_OVER_VXLAN_PORT                   0
	         RoCE_NEXT_PROTOCOL                          254
	         ESWITCH_HAIRPIN_DESCRIPTORS                 Array[0..7]
	         ESWITCH_HAIRPIN_TOT_BUFFER_SIZE             Array[0..7]
	         PF_BAR2_SIZE                                0
	         PF_NUM_OF_VF_VALID                          False(0)
	         NON_PREFETCHABLE_PF_BAR                     False(0)
	         VF_VPD_ENABLE                               False(0)
	         PF_NUM_PF_MSIX_VALID                        False(0)
	         PER_PF_NUM_SF                               False(0)
	         STRICT_VF_MSIX_NUM                          False(0)
	         VF_NODNIC_ENABLE                            False(0)
	```

```
         NUM_PF_MSIX_VALID                           True(1)
         NUM_OF_VFS                                  0
         NUM_OF_PF                                   2
         PF_BAR2_ENABLE                              False(0)
         SRIOV_EN                                    True(1)
         PF_LOG_BAR_SIZE                             5
         VF_LOG_BAR_SIZE                             1
         NUM_PF_MSIX                                 63
         NUM_VF_MSIX                                 11
         INT_LOG_MAX_PAYLOAD_SIZE                    AUTOMATIC(0)
         PCIE_CREDIT_TOKEN_TIMEOUT                   0
         PHY_COUNT_LINK_UP_DELAY                     DELAY_NONE(0)
         ACCURATE_TX_SCHEDULER                       False(0)
         PARTIAL_RESET_EN                            False(0)
         RESET_WITH_HOST_ON_ERRORS                   False(0)
         DISABLE_SLOT_POWER_LIMITER                  True(1)
         ADVANCED_POWER_SETTINGS                     True(1)
         CQE_COMPRESSION                             BALANCED(0)
         IP_OVER_VXLAN_EN                            False(0)
         MKEY_BY_NAME                                False(0)
         PRIO_TAG_REQUIRED_EN                        False(0)
         UCTX_EN                                     True(1)
         PCI_ATOMIC_MODE
PCI_ATOMIC_DISABLED_EXT_ATOMIC_ENABLED(0)
         TUNNEL_ECN_COPY_DISABLE                     False(0)
         LRO_LOG_TIMEOUT0                            6
         LRO_LOG_TIMEOUT1                            7
         LRO_LOG_TIMEOUT2                            8
         LRO_LOG_TIMEOUT3                            13
         LOG_TX_PSN_WINDOW                           7
         LOG_MAX_OUTSTANDING_WQE                     14
         RoCE_ADAPTIVE_ROUTING_EN                    False(0)
         TUNNEL_IP_PROTO_ENTROPY_DISABLE             False(0)
         ICM_CACHE_MODE                              DEVICE_DEFAULT(0)
         HAIRPIN_DATA_BUFFER_LOCK                    False(0)
         TX_SCHEDULER_BURST                          0
         LOG_MAX_QUEUE                               17
         CRYPTO_POLICY                               UNRESTRICTED(1)
         LOG_DCR_HASH_TABLE_SIZE                     11
         MAX_PACKET_LIFETIME                         0
         DCR_LIFO_SIZE                               16384
         LINK_TYPE_P1                                ETH(2)
         LINK_TYPE_P2                                ETH(2)
         RoCE_CC_PRIO_MASK_P1                        255
         RoCE_CC_PRIO_MASK_P2                        255
         CLAMP_TGT_RATE_AFTER_TIME_INC_P1            True(1)
         CLAMP_TGT_RATE_P1                           False(0)
         RPG_TIME_RESET_P1                           300
         RPG_BYTE_RESET_P1                           32767
         RPG_THRESHOLD_P1                            1
         RPG_MAX_RATE_P1                             0
         RPG_AI_RATE_P1                              5
         RPG_HAI_RATE_P1                             50
         RPG_GD_P1                                   11
         RPG_MIN_DEC_FAC_P1                          50
         RPG_MIN_RATE_P1                             1
         RATE_TO_SET_ON_FIRST_CNP_P1                 0
         DCE_TCP_G_P1                                1019
         DCE_TCP_RTT_P1                              1
         RATE_REDUCE_MONITOR_PERIOD_P1               4
         INITIAL_ALPHA_VALUE_P1                      1023
         MIN_TIME_BETWEEN_CNPS_P1                    4
         CNP_802P_PRIO_P1                            6
         CNP_DSCP_P1                                 48
         CLAMP_TGT_RATE_AFTER_TIME_INC_P2            True(1)
         CLAMP_TGT_RATE_P2                           False(0)
         RPG_TIME_RESET_P2                           300
         RPG_BYTE_RESET_P2                           32767
         RPG_THRESHOLD_P2                            1
         RPG_MAX_RATE_P2                             0
```

```
RPG_AI_RATE_P2                              5
RPG_HAI_RATE_P2                             50
RPG_GD_P2                                   11
RPG_MIN_DEC_FAC_P2                          50
RPG_MIN_RATE_P2                             1
RATE_TO_SET_ON_FIRST_CNP_P2                 0
DCE_TCP_G_P2                                1019
DCE_TCP_RTT_P2                              1
RATE_REDUCE_MONITOR_PERIOD_P2               4
INITIAL_ALPHA_VALUE_P2                      1023
MIN_TIME_BETWEEN_CNPS_P2                    4
CNP_802P_PRIO_P2                            6
CNP_DSCP_P2                                 48
LLDP_NB_DCBX_P1                             True(1)
LLDP_NB_RX_MODE_P1                          ALL(2)
LLDP_NB_TX_MODE_P1                          ALL(2)
LLDP_NB_DCBX_P2                             True(1)
LLDP_NB_RX_MODE_P2                          ALL(2)
LLDP_NB_TX_MODE_P2                          ALL(2)
DCBX_IEEE_P1                                True(1)
DCBX_CEE_P1                                 True(1)
DCBX_WILLING_P1                             True(1)
DCBX_IEEE_P2                                True(1)
DCBX_CEE_P2                                 True(1)
DCBX_WILLING_P2                             True(1)
KEEP_ETH_LINK_UP_P1                         True(1)
KEEP_IB_LINK_UP_P1                          False(0)
KEEP_LINK_UP_ON_BOOT_P1                     True(1)
KEEP_LINK_UP_ON_STANDBY_P1                  False(0)
DO_NOT_CLEAR_PORT_STATS_P1                  False(0)
AUTO_POWER_SAVE_LINK_DOWN_P1                False(0)
KEEP_ETH_LINK_UP_P2                         True(1)
KEEP_IB_LINK_UP_P2                          False(0)
KEEP_LINK_UP_ON_BOOT_P2                     True(1)
KEEP_LINK_UP_ON_STANDBY_P2                  False(0)
DO_NOT_CLEAR_PORT_STATS_P2                  False(0)
AUTO_POWER_SAVE_LINK_DOWN_P2                False(0)
NUM_OF_VL_P1                                _4_VLs(3)
NUM_OF_TC_P1                                _8_TCs(0)
NUM_OF_PFC_P1                               8
VL15_BUFFER_SIZE_P1                         0
QOS_TRUST_STATE_P1                          TRUST_PCP(1)
NUM_OF_VL_P2                                _4_VLs(3)
NUM_OF_TC_P2                                _8_TCs(0)
NUM_OF_PFC_P2                               8
VL15_BUFFER_SIZE_P2                         0
QOS_TRUST_STATE_P2                          TRUST_PCP(1)
DUP_MAC_ACTION_P1                           LAST_CFG(0)
MPFS_MC_LOOPBACK_DISABLE_P1                 False(0)
MPFS_UC_LOOPBACK_DISABLE_P1                 False(0)
UNKNOWN_UPLINK_MAC_FLOOD_P1                 False(0)
SRIOV_IB_ROUTING_MODE_P1                    LID(1)
IB_ROUTING_MODE_P1                          LID(1)
DUP_MAC_ACTION_P2                           LAST_CFG(0)
MPFS_MC_LOOPBACK_DISABLE_P2                 False(0)
MPFS_UC_LOOPBACK_DISABLE_P2                 False(0)
UNKNOWN_UPLINK_MAC_FLOOD_P2                 False(0)
SRIOV_IB_ROUTING_MODE_P2                    LID(1)
IB_ROUTING_MODE_P2                          LID(1)
PHY_AUTO_NEG_P1                             DEVICE_DEFAULT(0)
PHY_RATE_MASK_OVERRIDE_P1                   False(0)
PHY_FEC_OVERRIDE_P1                         DEVICE_DEFAULT(0)
PHY_AUTO_NEG_P2                             DEVICE_DEFAULT(0)
PHY_RATE_MASK_OVERRIDE_P2                   False(0)
PHY_FEC_OVERRIDE_P2                         DEVICE_DEFAULT(0)
PF_TOTAL_SF                                 0
PF_SF_BAR_SIZE                              0
PF_NUM_PF_MSIX                              63
RoCE_CONTROL                                RoCE_ENABLE(2)
PCI_WR_ORDERING                             per_mkey(0)
```

	```
	        MULTI_PORT_VHCA_EN                          False(0)
	        PORT_OWNER                                  True(1)
	        ALLOW_RD_COUNTERS                           True(1)
	        RENEG_ON_CHANGE                             True(1)
	        TRACER_ENABLE                               True(1)
	        IP_VER                                      IPv4(0)
	        BOOT_UNDI_NETWORK_WAIT                      0
	        UEFI_HII_EN                                 True(1)
	        BOOT_DBG_LOG                                False(0)
	        UEFI_LOGS                                   DISABLED(0)
	        BOOT_VLAN                                   1
	        LEGACY_BOOT_PROTOCOL                        PXE(1)
	        BOOT_RETRY_CNT                              NONE(0)
	        BOOT_INTERRUPT_DIS                          False(0)
	        BOOT_LACP_DIS                               True(1)
	        BOOT_VLAN_EN                                False(0)
	        BOOT_PKEY                                   0
	        P2P_ORDERING_MODE                           DEVICE_DEFAULT(0)
	        ATS_ENABLED                                 False(0)
	        DYNAMIC_VF_MSIX_TABLE                       False(0)
	        EXP_ROM_UEFI_ARM_ENABLE                     True(1)
	        EXP_ROM_UEFI_x86_ENABLE                     True(1)
	        EXP_ROM_PXE_ENABLE                          True(1)
	        ADVANCED_PCI_SETTINGS                       False(0)
	        SAFE_MODE_THRESHOLD                         10
	        SAFE_MODE_ENABLE                            True(1)
	```

ifconfig settings	
	```
	#!/bin/bash

	IF_0="ens5f0np0"
	IF_1="ens5f1np1"
	IP_0="192.168.100.10/24"
	IP_1="192.168.101.10/24"
	MTU=4200

	sudo ip link set $IF_0 up
	sudo ip link set $IF_1 up
	sudo ip link set $IF_0 mtu $MTU
	sudo ip link set $IF_1 mtu $MTU
	sudo ip addr add $IP_0 dev $IF_0
	sudo ip addr add $IP_1 dev $IF_1
	```

Appendix B. Step.1: Evaluation program of Perftest

Table 29: Evaluation program of each operation type for Software RoCE

Operation	Test type	Perftest commands
WRITE	Bandwidth	[Server] ./ib_write_bw -d rxe0 -R -t 128 -a -n 15000 [Client] ./ib_write_bw -d rxe0 -R -t 128 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
	Latency	[Server] ./ib_write_lat -d rxe0 -R –t 128 – a -n 15000 [Client] ./ib_write_lat -d rxe0 -R -t 128 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
SEND	Bandwidth	[Server] ./ib_send_bw -d rxe0 -R –t 128 – a -n 15000 [Client] ./ib_send_bw -d rxe0 -R -t 128 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
	Latency	[Server] ./ib_send_lat -d rxe0 -R –t 128 – a -n 15000 [Client] ./ib_send_lat -d rxe0 -R -t 128 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
READ	Bandwidth	[Server] ./ib_read_bw -d rxe0 -R -t 128 -a -n 15000 [Client] ./ib_read_bw -d rxe0 -R -t 128 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
	Latency	[Server] ./ib_read_lat -d rxe0 -R –t 128 – a -n 15000 [Client] ./ib_read_lat -d rxe0 -R -t 128 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits

Table 30: Evaluation program of each operation type for NIC-offloaded RoCE

Operation	Test type	Perftest commands
WRITE	Bandwidth	[Server] ./ib_write_bw -d mlx5_0 -R -t 2048 -a -n 15000 [Client] ./ib_write_bw -d mlx5_0 -R -t 2048 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
	Latency	[Server] ./ib_write_lat -d rxe0 -R –t 2048 – a -n 15000 [Client] ./ib_write_lat -d rxe0 -R -t 2048 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
SEND	Bandwidth	[Server] ./ib_send_bw -d mlx5_0 -R –t 2048 – a -n 15000 [Client] ./ib_send_bw -d mlx5_0 -R -t 2048 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
	Latency	[Server] ./ib_send_lat -d rxe0 -R –t 2048 – a -n 15000 [Client] ./ib_send_lat -d rxe0 -R -t 2048 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits
READ	Bandwidth	[Server] ./ib_read_bw -d mlx5_0 -R -t 16 -a -q 4096 -n 16

		[Client] ./ib_read_bw -d mlx5_0 -R -t 16 -a -q 4096 \ > 192.168.101.10 -n 16 -F --report_gbits
	Latency	[Server] ./ib_read_lat -d mlx5_0 -R –t 2048 – a -n 15000 [Client] ./ib_read_lat -d mlx5_0 -R -t 2048 -a \ > 192.168.101.10 -n 15000 \ > -F --report_gbits

Appendix C. Step.1: Analyzing RDMA Read operation at QP=1

Here, we provide more details about our thoughts on why RDMA Read operation at QP=1 had a slower ramp-up of throughput compared to RDMA Write/Send operations.

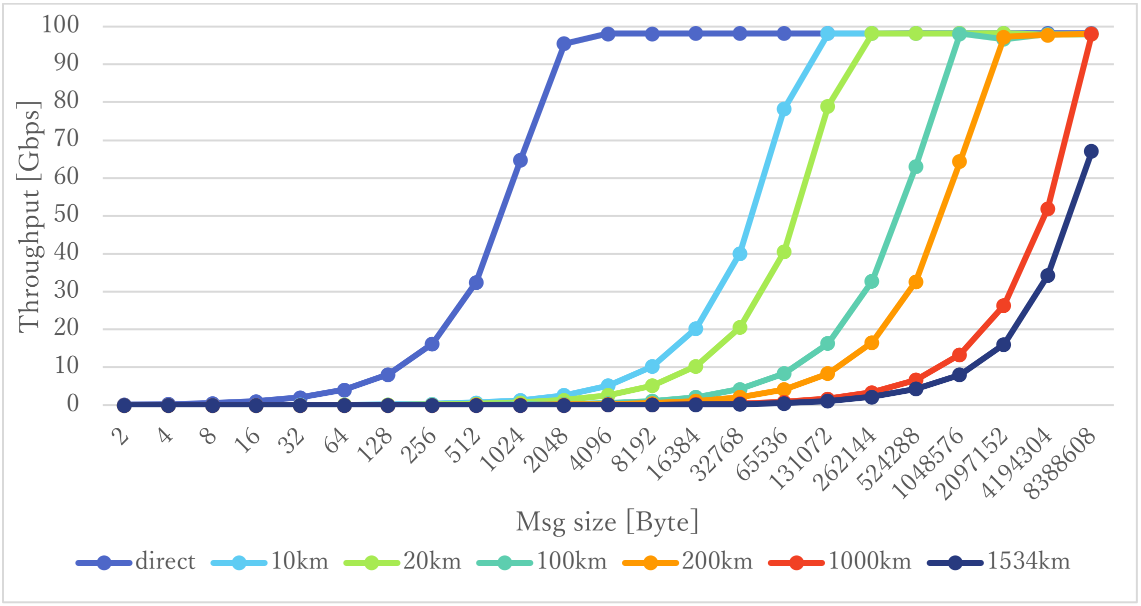
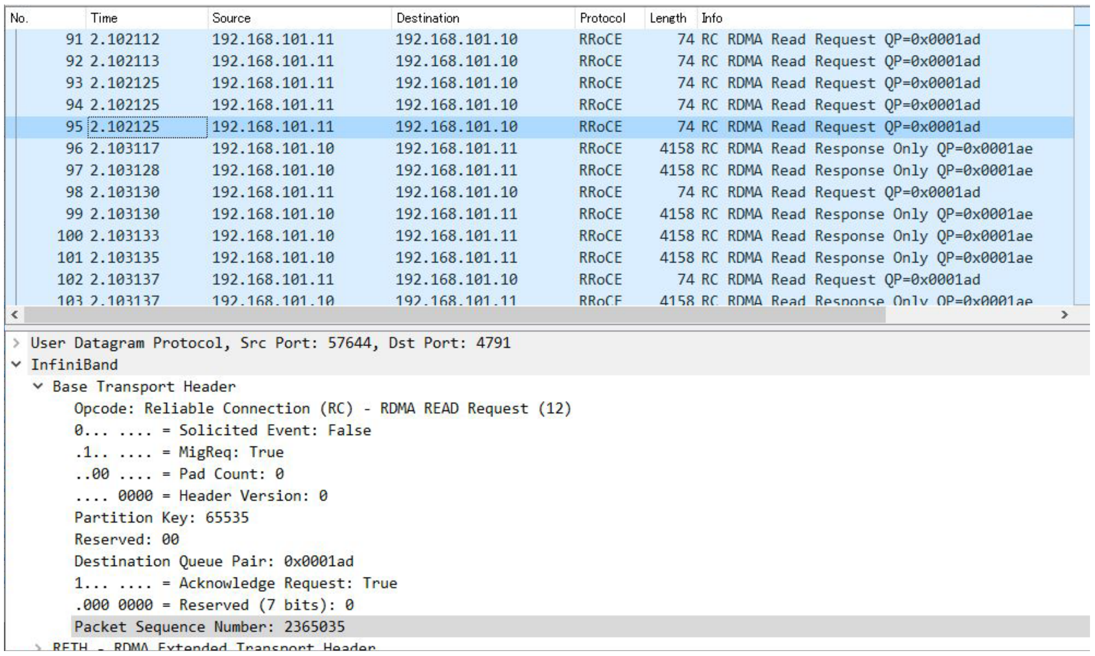


Figure 50: READ operation throughput at NIC-offload RoCE(QP=1)

Figure 50: READ operation throughput at NIC-offload RoCE(QP=1)shows the result of throughput that measured RDMA Read operation of QP=1at NIC-offloaded RoCE in Step.1. Focusing on the measurement result for a transmission distance of 100 km, it is found that the ramp-up point of the throughput is around 32KiB. Focusing on the measurement results for a transmission distance of 100 km in Figure 8 and Figure 10, where QP=1, the ramp-up point of the throughput is 256B. From the above result, it is clear that RDMA Read operation is slower the ramp-up point of the throughput than RDMA Write/Send operations.

To find this reason, we captured RDMA packets while performing RDMA Read operation with QP=1. Figure 51 and Figure 52 show the captured data when the transmission distance is 100 km and the message size is 4096B.

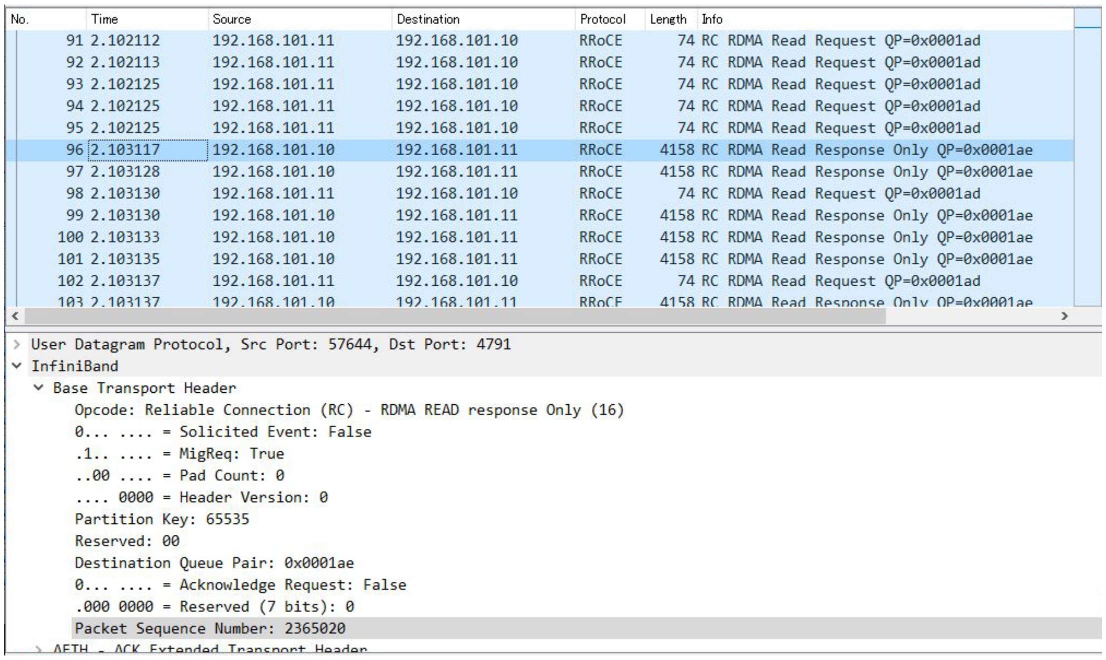
According to the captured data, after sending RDMA Read Requests, it was observed that the transmission of RDMA Read Request stopped nearly 1 msec until receiving RDMA Read Response. Comparing the RDMA Read Request packet just before the RDMA Read Request transmission stops and the RDMA Read Response received after 1 msec has passed, it was confirmed that the difference in the Packet Sequence Number (PSN) of the Base Transport Header is 15. This phenomenon of not receiving RDMA Read Response for 1 millisecond was observed many times, and the PSN difference at that time was always 15.



```
No.   Time       Source           Destination      Protocol  Length  Info
  91 2.102112   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  92 2.102113   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  93 2.102125   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  94 2.102125   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  95 2.102125   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  96 2.103117   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
  97 2.103128   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
  98 2.103130   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  99 2.103130   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
 100 2.103133   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
 101 2.103135   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
 102 2.103137   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
 103 2.103137   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae

> User Datagram Protocol, Src Port: 57644, Dst Port: 4791
v InfiniBand
  v Base Transport Header
      Opcode: Reliable Connection (RC) - RDMA READ Request (12)
      0... .... = Solicited Event: False
      .1.. .... = MigReq: True
      ..00 .... = Pad Count: 0
      .... 0000 = Header Version: 0
      Partition Key: 65535
      Reserved: 00
      Destination Queue Pair: 0x0001ad
      1... .... = Acknowledge Request: True
      .000 0000 = Reserved (7 bits): 0
      Packet Sequence Number: 2365035
  > RETH - RDMA Extended Transport Header
```

Figure 51: Captured data just before the Read Request stops (Read operation, distance = 100km, message size = 4096B)



```
No.   Time       Source           Destination      Protocol  Length  Info
  91 2.102112   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  92 2.102113   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  93 2.102125   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  94 2.102125   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  95 2.102125   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  96 2.103117   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
  97 2.103128   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
  98 2.103130   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
  99 2.103130   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
 100 2.103133   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
 101 2.103135   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae
 102 2.103137   192.168.101.11   192.168.101.10   RRoCE       74 RC RDMA Read Request QP=0x0001ad
 103 2.103137   192.168.101.10   192.168.101.11   RRoCE     4158 RC RDMA Read Response Only QP=0x0001ae

> User Datagram Protocol, Src Port: 57644, Dst Port: 4791
v InfiniBand
  v Base Transport Header
      Opcode: Reliable Connection (RC) - RDMA READ response Only (16)
      0... .... = Solicited Event: False
      .1.. .... = MigReq: True
      ..00 .... = Pad Count: 0
      .... 0000 = Header Version: 0
      Partition Key: 65535
      Reserved: 00
      Destination Queue Pair: 0x0001ae
      0... .... = Acknowledge Request: False
      .000 0000 = Reserved (7 bits): 0
      Packet Sequence Number: 2365020
  > AETH - ACK Extended Transport Header
```

Figure 52: Captured data of the Read Response received just after the Read Request stops (Read operation, distance = 100km, message size = 4096B)

In the above case, the delta of PSN is 2365035 - 2365020 = 15.
In all other cases, the delta was also 15.

Appendix D. Step.2: NVIDIA A100 configuration

Item	Description

System	
	```
	# nvidia-smi

	+-----------------------------------------------------------------------------+
	| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
	|-------------------------------+----------------------+----------------------+
	| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
	| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
	|                               |                      |               MIG M. |
	|===============================+======================+======================|
	|   0  NVIDIA A100-PCI...  Off  | 00000000:84:00.0 Off |                    0 |
	| N/A   41C    P0    39W / 250W |    433MiB / 40960MiB |      0%      Default |
	|                               |                      |             Disabled |
	+-------------------------------+----------------------+----------------------+

	+-----------------------------------------------------------------------------+
	| Processes:                                                                  |
	|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
	|        ID   ID                                                   Usage      |
	|=============================================================================|
	|    0   N/A  N/A     17429      C   ...xtec/perftest/ib_read_lat      430MiB |
	+-----------------------------------------------------------------------------+
	```

GPU Topology	
	```
	# nvidia-smi topo -mp
	        GPU0    NIC0    NIC1    CPU Affinity    NUMA Affinity
	GPU0     X      NODE    NODE    18-35,54-71     1
	NIC0    NODE     X      PIX
	NIC1    NODE    PIX      X

	Legend:

	  X    = Self
	  SYS  = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
	  NODE = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
	  PHB  = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
	  PXB  = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
	  PIX  = Connection traversing at most a single PCIe bridge

	NIC Legend:

	  NIC0: mlx5_0
	  NIC1: mlx5_1
	```

Appendix E. Step.3: NVMe Initiator/Target configuration

1. SPDK Target Settings

Item	Description

Run SPDK	
	```
	#!/bin/bash
	set -eu

	STEP3_DIR=/root/step3
	SPDK_DIR=${STEP3_DIR}/spdk
	SPDK_SET=${SPDK_DIR}/scripts/setup.sh
	TGT_BIN=${SPDK_DIR}/build/bin/nvmf_tgt
	TGT_JSN=${STEP3_DIR}/spdk_bdev.json

	#CORE_MASK="[18-25]"          # 8 core
	CORE_MASK="[18-33]"          # 16 core
	#CORE_MASK="[18-35,54-63]"    # 28 core

	${SPDK_SET}      # nvme to uio
	${TGT_BIN} -c ${TGT_JSN} -m ${CORE_MASK}
	```

BDEV config (spdk_bdev.json)	(Excerpt)
	```
	{
	  "method": "nvmf_create_transport",
	  "params": {
	    "trtype": "RDMA",
	    "max_queue_depth": 128,
	    "max_io_qpairs_per_ctrlr": 64,
	    "in_capsule_data_size": 4096,
	    "max_io_size": 131072,
	    "io_unit_size": 8192,
	    "max_aq_depth": 128,
	    "num_shared_buffers": 8192,
	    "buf_cache_size": 32,
	    "dif_insert_or_strip": false,
	    "zcopy": false,
	    "max_srq_depth": 4096,
	    "no_srq": false,
	    "acceptor_backlog": 100,
	    "no_wr_batching": false,
	    "abort_timeout_sec": 1
	  }
	},

	...

	{
	  "method": "nvmf_create_subsystem",
	  "params": {
	    "nqn": "nqn.2014-08.io.spdk:cnode1",
	    "allow_any_host": true,
	    "serial_number": "SPDK001",
	    "model_number": "SPDK bdev Controller",
	    "max_namespaces": 8,
	    "min_cntlid": 1,
	    "max_cntlid": 65519,
	    "ana_reporting": false
	  }
	},
	{
	  "method": "nvmf_subsystem_add_ns",
	  "params": {
	    "nqn": "nqn.2014-08.io.spdk:cnode1",
	    "namespace": {
	      "nsid": 1,
	      "bdev_name": "Nvme0n1",
	      "nguid": "9FC0252D00484421B0D3591DC2D1D787",
	      "uuid": "9fc0252d-0048-4421-b0d3-591dc2d1d787"
	    }
	  }
	},
	```

```
      {
        "method": "nvmf_subsystem_add_listener",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode1",
          "listen_address": {
            "trtype": "RDMA",
            "adrfam": "IPv4",
            "traddr": "192.168.100.10",
            "trsvcid": "4420"
          }
        }
      },
      {
        "method": "nvmf_create_subsystem",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode2",
          "allow_any_host": true,
          "serial_number": "SPDK002",
          "model_number": "SPDK bdev Controller",
          "max_namespaces": 8,
          "min_cntlid": 1,
          "max_cntlid": 65519,
          "ana_reporting": false
        }
      },
      {
        "method": "nvmf_subsystem_add_ns",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode2",
          "namespace": {
            "nsid": 1,
            "bdev_name": "Nvme1n1",
            "nguid": "74241B53C6EA46989CB20361A2D0B1CC",
            "uuid": "74241b53-c6ea-4698-9cb2-0361a2d0b1cc"
          }
        }
      },
      {
        "method": "nvmf_subsystem_add_listener",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode2",
          "listen_address": {
            "trtype": "RDMA",
            "adrfam": "IPv4",
            "traddr": "192.168.100.10",
            "trsvcid": "4420"
          }
        }
      },
      {
        "method": "nvmf_create_subsystem",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode3",
          "allow_any_host": true,
          "serial_number": "SPDK003",
          "model_number": "SPDK bdev Controller",
          "max_namespaces": 8,
          "min_cntlid": 1,
          "max_cntlid": 65519,
          "ana_reporting": false
        }
      },
      {
        "method": "nvmf_subsystem_add_ns",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode3",
          "namespace": {
            "nsid": 1,
            "bdev_name": "Nvme2n1",
            "nguid": "0B8CE9AF245B42459776863985B10C2E",
            "uuid": "0b8ce9af-245b-4245-9776-863985b10c2e"
          }
        }
      },
      {
        "method": "nvmf_subsystem_add_listener",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode3",
          "listen_address": {
            "trtype": "RDMA",
            "adrfam": "IPv4",
            "traddr": "192.168.100.10",
            "trsvcid": "4420"
          }
        }
      },
      {
        "method": "nvmf_create_subsystem",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode4",
          "allow_any_host": true,
          "serial_number": "SPDK004",
          "model_number": "SPDK bdev Controller",
          "max_namespaces": 8,
          "min_cntlid": 1,
          "max_cntlid": 65519,
          "ana_reporting": false
        }
      },
      {
        "method": "nvmf_subsystem_add_ns",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode4",
          "namespace": {
            "nsid": 1,
            "bdev_name": "Nvme3n1",
            "nguid": "9D47FCBC8F93401690485389FBB36F1A",
            "uuid": "9d47fcbc-8f93-4016-9048-5389fbb36f1a"
          }
        }
      },
      {
        "method": "nvmf_subsystem_add_listener",
        "params": {
          "nqn": "nqn.2014-08.io.spdk:cnode4",
          "listen_address": {
            "trtype": "RDMA",
            "adrfam": "IPv4",
            "traddr": "192.168.100.10",
            "trsvcid": "4420"
          }
        }
      }
```

2. FIO

Item	Description

Run FIO	`#!/bin/bash`
	`set -eu`
	
	`FIO=/root/step-3/fio/fio`
	`NVME_SSD=/dev/nvme4n1:/dev/nvme5n1:/dev/nvme6n1:/dev/nvme7n1`
	`FIO_CORES=16`
	`WORKLOAD=read`
	`BLK_SIZE=16384`
	`IO_Q=128`
	`FILE_SIZE=2097152`
	`TEST_NAME="test-0-0"`
	`RESULT="test-result.txt"`

	```
	$FIO --group_reporting=1 \
	    --direct=1 --ioengine=libaio \
	    --filename=$NVME_SSD \
	    --time_based=1 --ramp_time=20 --runtime=60 \
	    --thread=1 --numjobs=${FIO_CORES} \
	    --bs=${BLK_SIZE} --iodepth=${IO_Q} --rw=${WORKLOAD} \
	    --name=${TEST_NAME} --size=${FILE_SIZE} \
	    --output=./${RESULT}
	```

Appendix F. Performance Tuning for meeting requirements of some use cases.

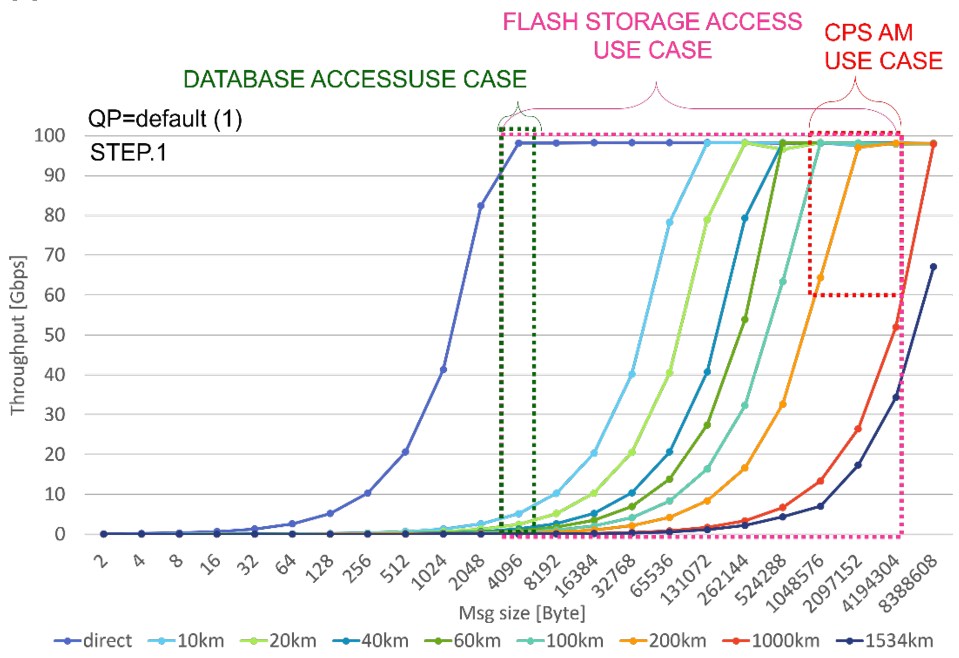


Figure 53: Step.1 results READ operation (QP=1)

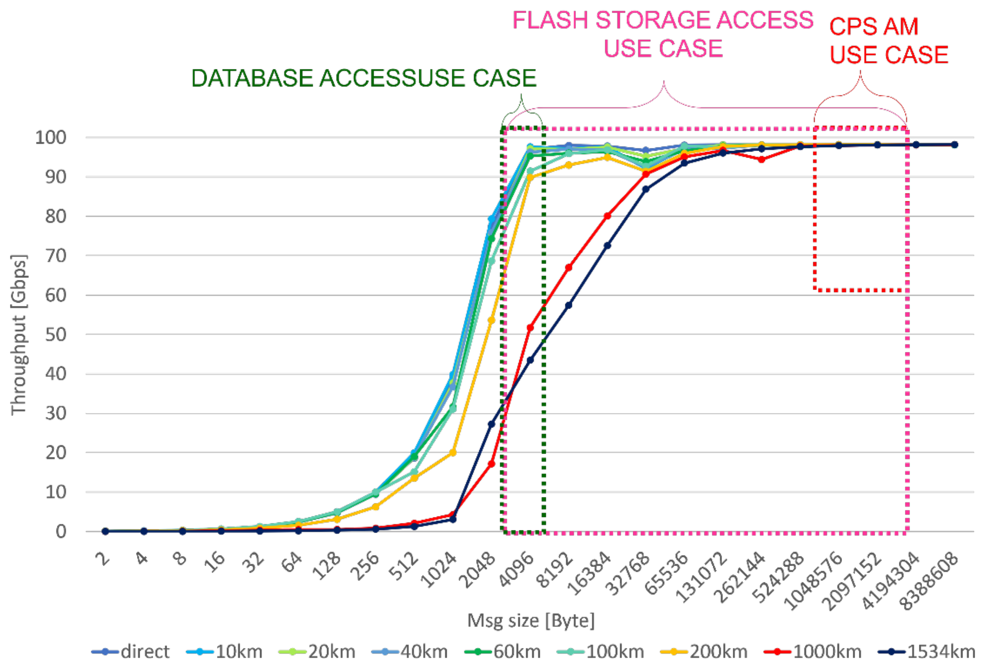


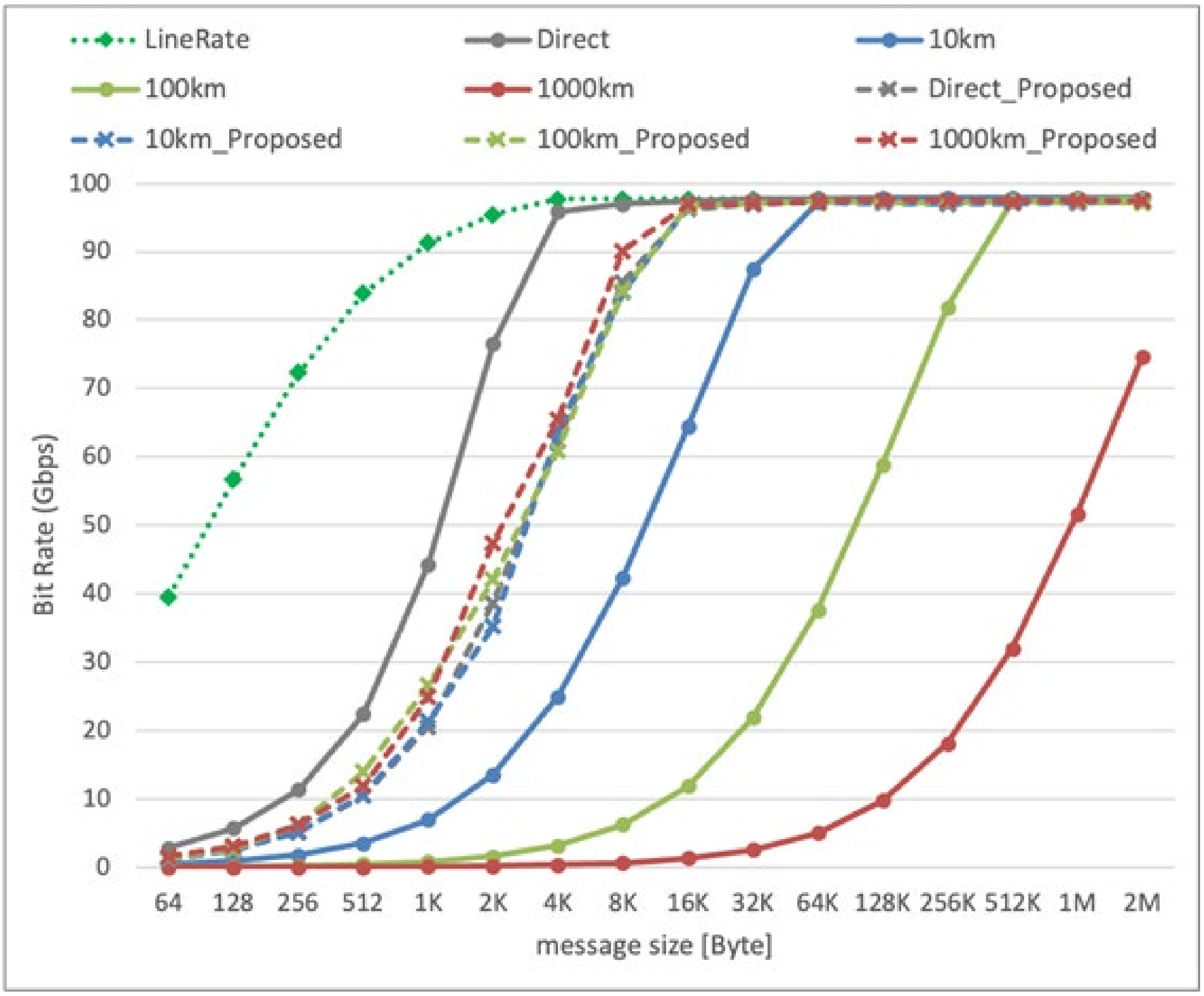
Figure 54: Step.1 results READ operation (QP=4096)

As already described in Appendix C, RDMA READ operation at QP=1 had a slower ramp-up of throughput compared to RDMA Write/Send operations. RDMA WRITE operation and SEND operation are obtained that ramp-up the throughput was faster by increasing queue depth of CX-6, and setting QP is 1. For RDMA READ operation, in addition to increasing queue depth of CX-6, increasing QP is an important factor in optimizing throughput. For example, we suppose three use case. Each use case could change the requirements.

CPS AM use case require message size from 1MB to 4MB. In this use case, RDMA READ op. can meet requirements if QP is increased to 4096 and queue depth of CX-6 is setting to 16384.
Flash Storage access use case require message size from 4KB to 4MB. In this use case, RDMA READ op. can meet requirements if QP is increased to 4096 and queue depth of CX-6 is setting to 16384.
For database access, the message size can be up to a few MB or more depending upon usage scenarios such as bulk insert. The figure 53 and 54 show one example. In this example, RDMA READ op. can get more than 40Gbps throughputs even long-distance transmission (1534km) if QP is increased to 4096 and queue depth of CX-6 is setting to 16384.

Appendix G. RDMA WAN Accelerator

In the previous section, the RDMA over Open APN PoC reports shows better performance than standard RDMA with expanding queue pairs. However, since the large queue pair/depth configuration can consume many memory resources on the SmartNICs. it might lead to long queuing latency or unstable behavior because of lack of memory resources for other network protocol processing. A transparent accelerator can be another approach to tackle these issues which are both low network throughput of standard RDMA in long distance and resource consumption on the SmartNICs. NTT proposed an RDMA WAN accelerator that creates pseudo-ACK messages to avoid waiting for ACK message by monitoring RDMA messages. Our proposed RDMA WAN accelerator shows that it can achieve better network throughput even though in case of long distance RDMA communication. This result indicates that ACK messages should be handled properly in such case.


LineRate
Direct
10km
100km
1000km
Direct_Proposed
10km_Proposed
100km_Proposed
1000km_Proposed
Bit Rate (Gbps)
1K
2K
4K
8K
16K
32K
64K
128K
256K
512K
1M
2M
message size [Byte]