IDH Primitive PoC Scenario 2 Report

Classification: IOWN Global Forum Recognized PoC

Stage: SSF PoC Report

Confidentiality: Public

Version: 1.0

March 13, 2025

Contents

1 PoC Project Completion Status

- Completion Status: Done

2 Project Participants

- PoC Project Name: IDH Primitive PoC Scenario 2 Report
- Member: NTT (Contact: Tomohiro INOUE, Hongjie Zhai)

3 Confirmation of PoC Demonstration

Since this is a primitive PoC scenario, we do not anticipate demonstrating the contents of this report. We are considering presenting the results of the advanced PoC scenario at some event.

4 PoC Goals and Status

The IOWN Global Forum Data Hub (IDH) is a data management, utilization, and sharing solution concept built on IOWN Global Forum's communication and processing infrastructure, such as Open APN and DCI. IDH enables the integration of a wide variety of geographically dispersed data, facilitating the creation of new added value through cross-industry data utilization and the resolution of global environmental issues through social data sharing.

To realize a data-driven society, it is necessary to promote the mutual utilization of diverse and large-scale data across distance and organizational boundaries. However, traditional data hubs designed for intra-organizational data sharing face significant challenges: speed is hindered by distance, and simplicity and security are compromised by organizational boundaries. What is needed is an IDH that allows various entities to utilize large amounts of geo-distributed data quickly, easily, and securely.

To prove and demonstrate such capabilities of IDH, this PoC evaluated the virtual data lake functionality according to IDH Primitive PoC Scenario 2, which describes how to accelerate data retrieval from remote IDH data service servers to IDH frontend servers. The goals of this PoC and its achievement status are described below:

Table 1. Success Criteria of IDH Primitive PoC Scenario 2

GOAL	SUCCESS CRITERIA	CORRESPONDING BENCHMARK IN POC REFERENCE	ACHIEVEMENT STATUS
Verification of the efficient data transfer mechanism for geo-distributed data hub environment	To clarify the conditions under which distributed data processing over Open APN and virtual data lake functionality are possible without performance degradation compared to local implementation. These features should be implemented: • Data federation function • Get Communications over Open APN • Data filtering and preprocessing	From Sec 4.3: • End-to-end IDH latency (time for required data transfer)	Successfully Done

5 PoC Feedback Received from Non-Member

Feedback from non-PoC members is as follows:

> It was important to show that Open APN can be used to handle remote data from IDH in the same way as local data. It would be better to apply the results to a more specific use case PoC.
>
> Software bottlenecks found in this PoC are important findings, so it is good to feed them back to the community.

6 PoC Technical Report

6.1 Positioning of the PoC

Five primitive PoC scenarios are described in the IDH PoC Reference Document, and this report covers scenario 2, which involves the IDH data service server (DS) to the frontend server (FE) GET communication with virtual data lake functionality. The positioning is shown in the figure below.

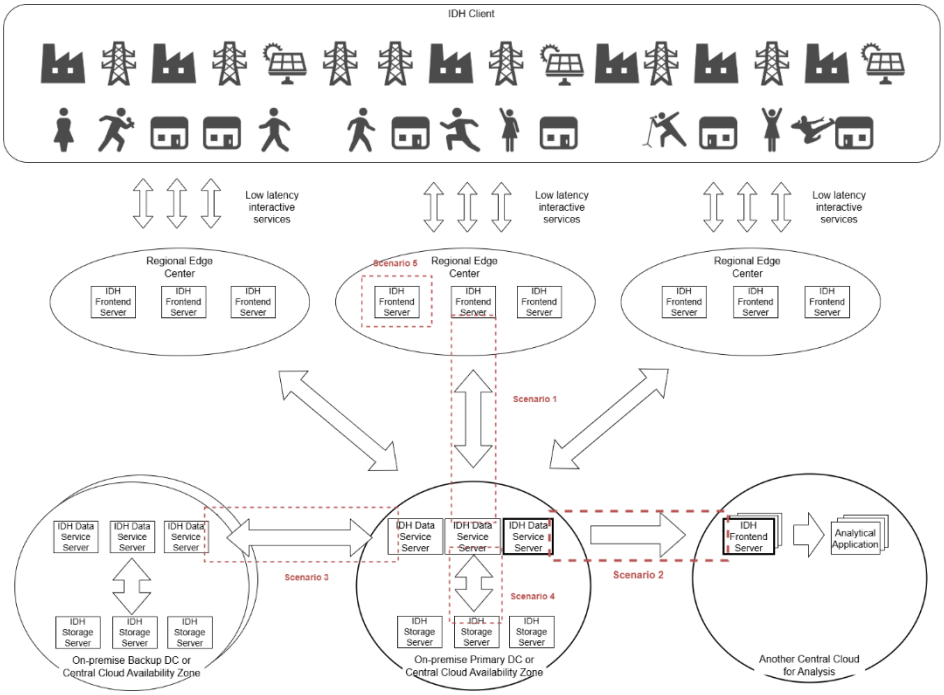

Figure 1. Scope of IDH Primitive PoCs and Scenario 2

6.2 PoC Overview

The scope of this PoC is to measure the performance of an IDH virtual data lake implementation with varying conditions such as data transfer size, and to identify the conditions under which data transfer between IDH DS and FE in a geographically distributed environment is sufficiently efficient compared to local implementations. In addition, we also aimed to:

- Analyze the performance of geo-distributed IDH: Identify and organize the parameters needed to maximize the performance of the IDH using Open APN, and determine the scenarios in which it excels or encounters challenges.

- Clarify expertise for the use of IOWN infrastructure: Identify expertise and direction of improvement to fully utilize IOWN infrastructure such as DCI in the next version of IDH.

The figure below provides an overview of the PoC Scenario 2 and the selected features.

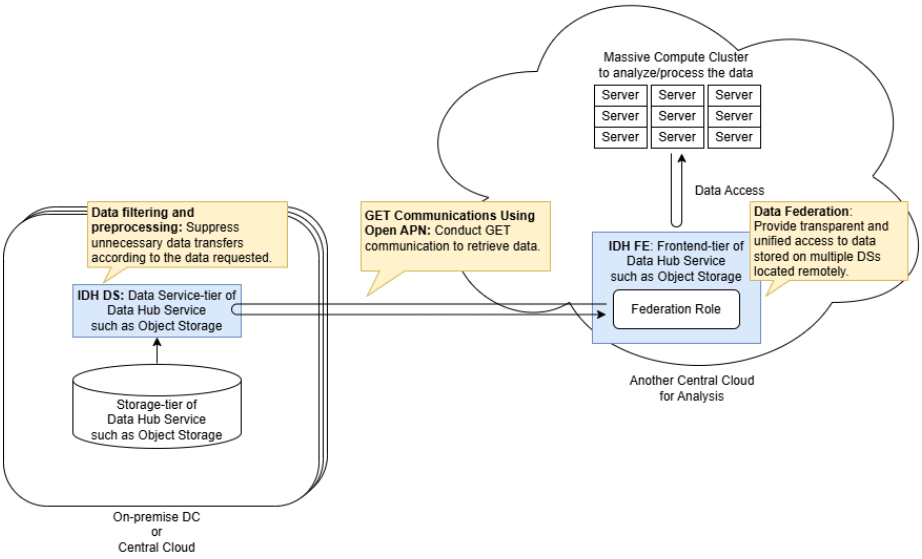

Figure 2. Overview of the PoC

6.3 System Configuration

The basic configuration of the PoC, shown in the figure below, consists of FE, a client that retrieves remote data, and DS, a data service that manages data stored in the IDH storage server. In the main test environment, FE is located at the Musashino data center (DC) and connected via Open APN (100G) to DS, located at the Akihabara DC, 30 km away from the Musashino DC. In the other comparative verification environment, FE and DS are connected via local Ethernet (1G), which simulates the case of connecting FE and DS within the same data center using existing LAN technology.

As the Open APN (100G) and Ethernet LAN (1G) environments have significantly different bandwidths, the comparison focuses on evaluating the potential performance differences between inter-DC and intra-DC communication scenarios, rather than a direct one-to-one comparison of network technologies. In addition, the throughput evaluation described later in this report was tested to ensure that the 1G bandwidth would not become a bottleneck.

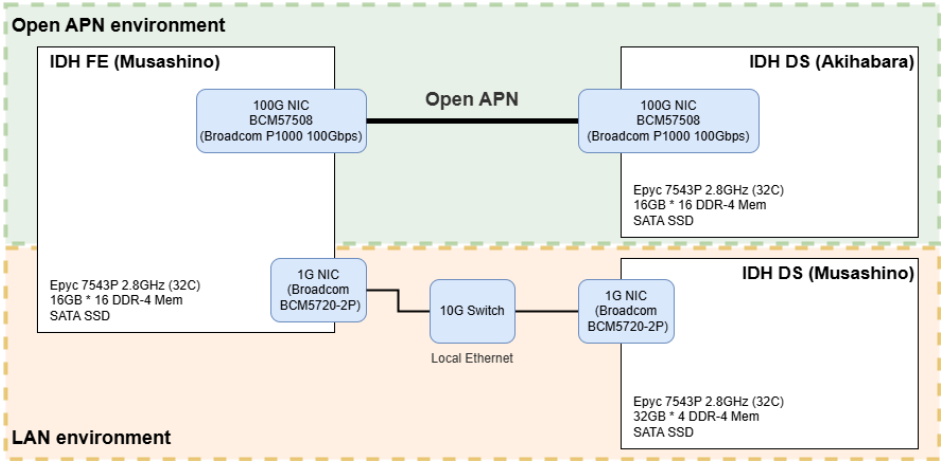

Figure 3. Test Environment

The system and software configuration is shown in Figure 4. We built the IDH as a Virtual Data Lake service class, where the IDH is configured in a geographically distributed manner, with the IDH FE located where a data consumer is located and the IDH DS located where a data provider is located.

Table 2 lists the software components and implemented features of each element. Requests sent by a test client are processed by the FE and forwarded to the DS connected via Open APN or LAN. In this experiment, DS is an object storage implemented with MinIO servers, and the actual data is stored in in-memory file systems. The data is stored in memory so that the device I/O does not become a bottleneck in this experiment.

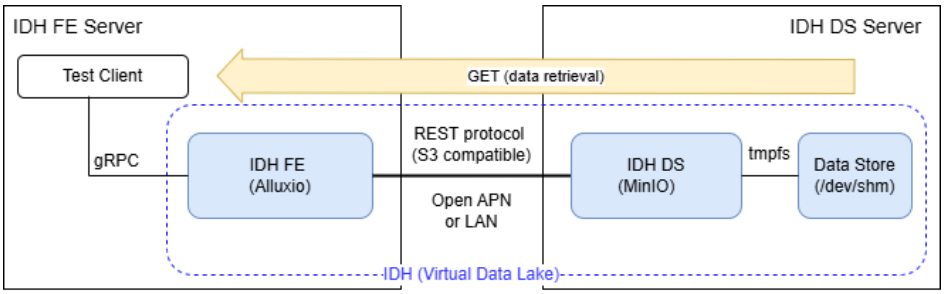

Figure 4. System Configuration

Table 2. System Components

ITEM	SOFTWARE/IMPLEMENTATION	FEATURES
Test Client	Customized Alluxio client	Test client to request data
IDH FE	Customized/modified version of Alluxio 2.9.3	Service that offers data federation capabilities, providing abstracted, unified access to data stored across multiple data backends. It provides a unified namespace, enabling seamless interaction between applications and data, regardless of its physical location.
IDH DS	MinIO 8.5.9	Object storage service with filtering capabilities that allow clients to retrieve a subset of data by using simple SQL-like expressions, providing a way to suppress unnecessary data transfers. (As this report is focused on the analysis of overall data transfer efficiency, queries that request a subset are not used.)
Data Store	/dev/shm	In-memory file systems as an actual data store

6.4 PoC Results

IDH Latency on Open APN

The first step was to measure the end-to-end latency (time required for data transfer) of the IDH and to assess the impact of each component on this measurement. The experiment demonstrated that a very low latency of 3.8 ms could be achieved for a data size of 128 kB, as shown in Table 3.

In this data flow, the data managed by the DS was sent to the remote DC via Open APN. Then, the data was delivered to the test client through the FE's data federation function, which virtually integrates DSs at multiple locations (although only one DS was used in this test configuration).

Table 3 provides a breakdown of latency in this data flow, calculated based on the time differences between timestamps recorded in logs by each component. The remote servers were precisely time-synchronized using PTP (Precision Time Protocol).

Table 3. Latency breakdown of IDH over Open APN

Item	Component	Avg. Latency (μs)
End-to-end Latency (data size: 128 kB)	**<Total>**	**3,840**
	├DS (MinIO)	622
	├Open APN transmission (incl. kernel stack)	246
	├FE (Alluxio)	2,875
	│ └ FE (storage connector only)	1,852
	└ Test client	95

The breakdown showed that most of this was due to the delay caused by the implementation of the FE software, which enables the data federation capabilities of the remote data. Notable complements to this experiment are as follows.

- The FE we used is a Java implementation, and it was responsible for much of the end-to-end latency. In particular, the storage connector has a JNI (C-lang) interface, and data is copied multiple times in the software stack, resulting in relatively high latency.
- In another part of the FE, latency is caused by waiting in the queue for parallel processing of requests, operations related to metadata, and data copying from the storage connector.
- The overhead of the in-memory file system (/dev/shm) was negligible (<0.250 μs).

The graph below illustrates the file transfer latency of IDH over Open APN for different file sizes. Latency is generally proportional to the file size, but if the file size is small, the internal processing overhead becomes non-negligible. Additionally, the impact of transfer buffer size was compared, showing that a buffer size of 8 KB or larger reduces latency.

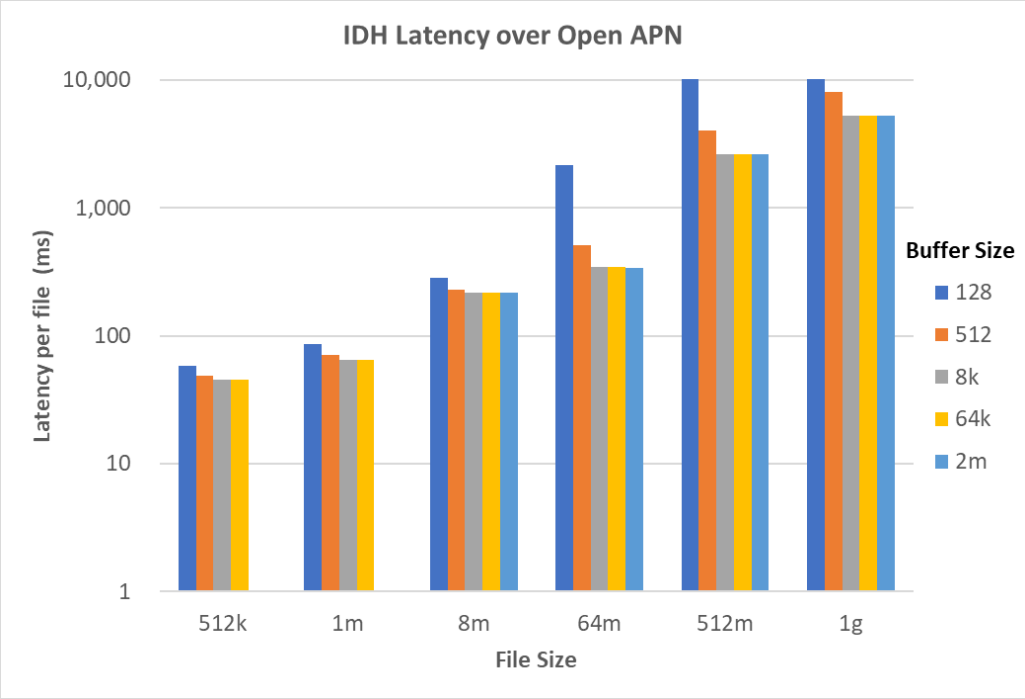

Figure 5. IDH Latency over Open APN

We compared IDH latency for various file sizes in the Open APN and the existing local network (LAN) environments. The figure below shows the results: when the ratio is greater than 1, it indicates that the local network has lower latency, while a ratio of less than 1 suggests that the Open APN has lower latency. As shown, for file transfers larger than 512 kB, the IDH is able to achieve a lower latency in an Open APN environment than in a LAN environment, despite being geographically separated by more than 30 km. This characteristic is consistent with the results of the Open APN baseline validation described in Appendix A of this report.

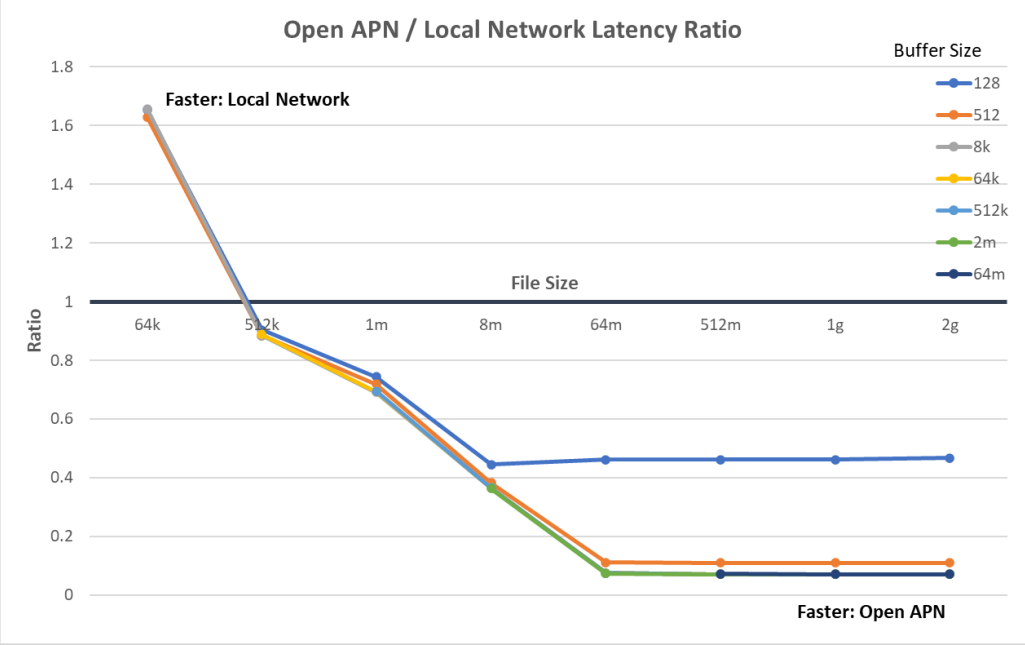

Figure 6. Open APN vs. local network for IDH latency

IDH Throughput on Open APN

Secondly, the end-to-end throughput of the IDH was evaluated. Furthermore, the raw TCP bandwidth and the throughput of the object storage (without data federation function) were also assessed as baselines. IDH achieves 50 Gbps throughput, which is about 50% of TCP bandwidth and up to 60% of object storage throughput. The following paragraphs present an analysis of the obstacles encountered in the implementation of the IDH.

In these measurements, the CPU load is consistently at 100% of the specified number of cores, indicating that the CPU is the bottleneck in this test scenario. To achieve maximum TCP throughput, a minimum of five CPU threads (equivalent to five TCP sessions) are required. However, to enhance object storage performance, a greater number of threads (corresponding to the same number of HTTP sessions) is necessary. This indicates that high-level data transfers, such as object storage, require additional CPU power and that a reduction in CPU load is also necessary to optimize the use of Open APN bandwidth.

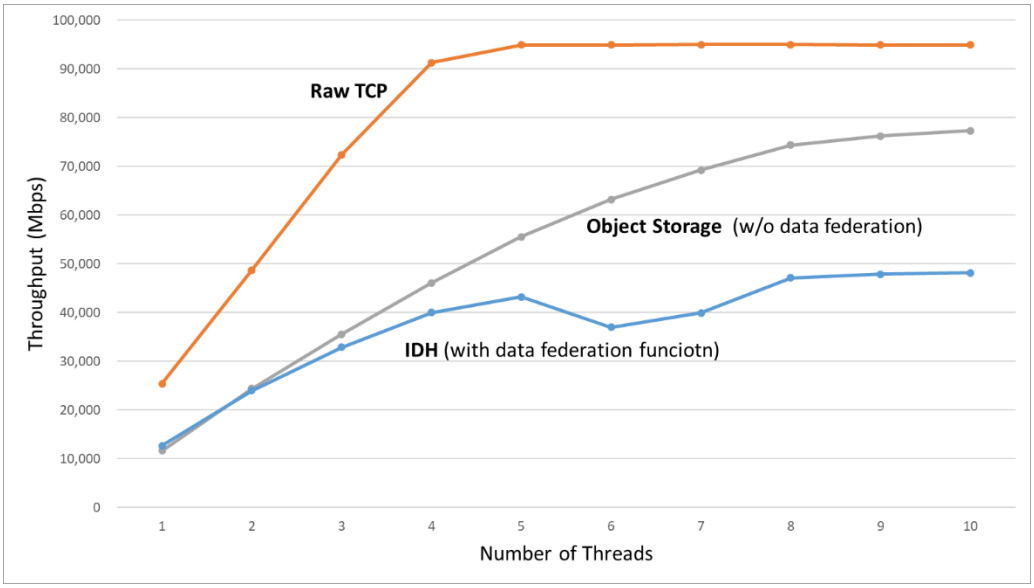

Figure 7. Total Throughput

The IDH is efficient in CPU utilization of up to 5 threads, but after that, it makes less contribution to throughput. The following figure shows the result when the number of threads used by the IDH is increased to 30, and when 10 or more threads are used, almost 100% of the CPU is consumed in areas other than transfers due to contention between tasks.

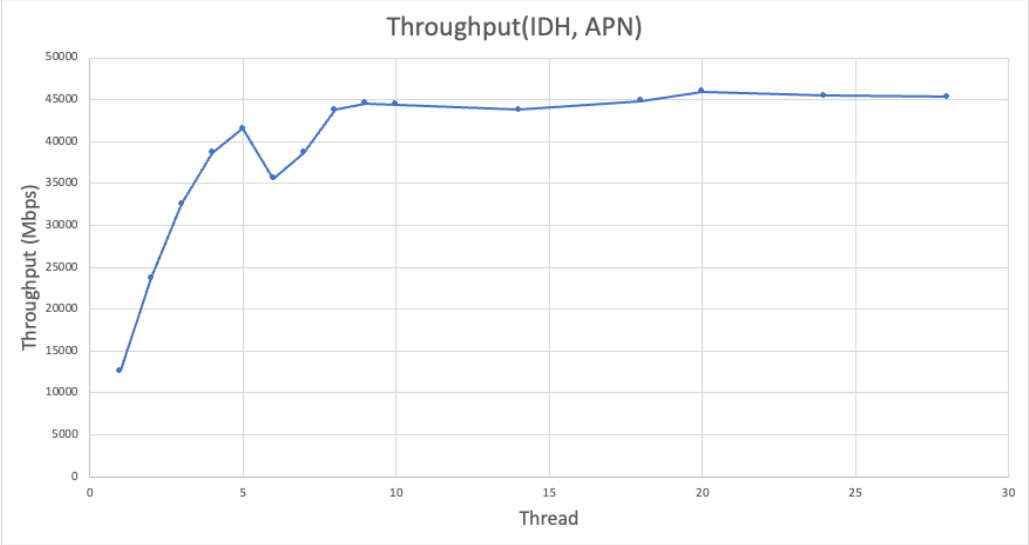

Figure 8. Throughput against CPU Thread Count

The following measurements were made for comparison with the existing local network (LAN). LAN has a bandwidth of 1G bps, so this measurement was made with a traffic load of less than 1G. The graph below shows the results as a ratio of Open APN to LAN, where ratio = 1 indicates that Open APN and LAN have the same throughput, and a ratio greater than 1 indicates that Open APN can transfer more data. We found that when the file size to be transferred exceeds 512K, the IDH has higher throughput in an Open APN environment than in a LAN, regardless of software configuration.

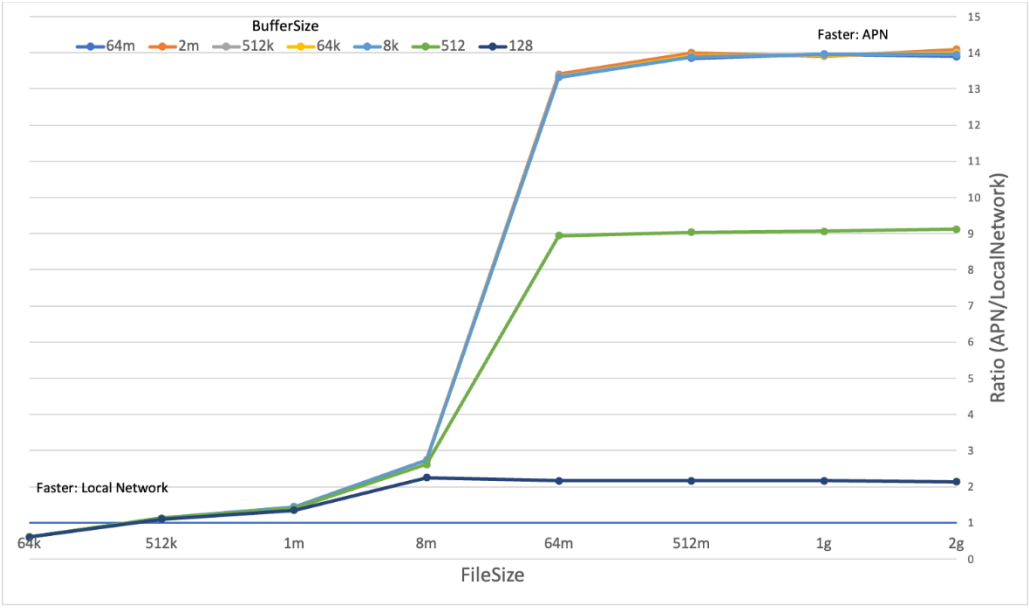

Figure 9. Open APN vs. local network for IDH throughput

6.5 Technical Findings

As shown in Figure 6, when the file transfer size is 64 MB or larger, the geo-distributed IDH over Open APN can achieve one-tenth the latency of the LAN-connected IDH.

However, to achieve this, the internal buffer size of our IDH needs to be set properly. Our FE has buffering capabilities to prevent fragmented access to object storage, and the buffer size is configurable. For this parameter, as with the transfer file size, we found that the larger the size, the better the performance. This is based on the characteristics of Open APN, as detailed in Appendix A, which has the potential to outperform the local network when transferring data in large chunks.

The following table shows the sweet spot for the parameters of our IDH that we found through experimentation. Of course, the actual configurations to be set will vary depending on the details of the FE and DS implementations, but the ideas for tuning parameters would be helpful for implementations with similar configurations. These findings also have important implications for the design of IDH and other systems that transfer data over Open APN, particularly with respect to buffer tuning.

Table 4 Parameter Tunings

SETTING ITEM	RECOMMENDED VALUE	NOTES
File Size	Minimum: 512 kB Recommended: 64 MB or more	For data less than 512 kB, performance will be gained by transferring the data in batches (see Fig. 8).
Buffer Size	Minimum: 8 kB Recommended: 32 MB	Buffer sizes smaller than 8kb limit throughput. (see Fig. 8).
Number of Transfer Threads	Recommended: 5-10	For transfers up to 100 Gbps, a single thread is not sufficient (see Fig. 8).
Cache Device	Minimum: NVMe SSD Recommended: Memory	The use of slow SSDs or HDDs increases latency that is not matched by the speed of the Open APN.

6.6 PoC Summary

We have successfully constructed and tested a geographically distributed IDH system with virtual data lake functionality. Our findings demonstrate that the system can transfer large amounts of data at a rate 10 times faster than conventional systems with one-tenth the latency if the file size is larger than 64 MB and the buffer size is configured to 8 kB or more, and can treat data from a site 30 km apart as if it were on a local network if the file size is 512 kB or bigger. This result exceeds the success criteria listed in Section 4 and achieves the PoC goal.

Our detailed analysis shows that to take advantage of Open APN characteristics, it is advisable to transfer small data in batches of at least 512 kB, preferably in MB order. In such cases, the data transfer performance exceeds even that of local networks with no distance barrier.

On the other hand, to fully utilize the 100 Gbps bandwidth of Open APN, it is crucial to optimize the entire system, including the software stack. As 100 Gbps is a very high speed, it's easy to have bottlenecks that hinder the full utilization of 100 Gbps bandwidth. For example, processing delays caused by CPU load can become significant bottlenecks, making it essential to optimize the system.

6.7 Implication of Advanced IDH implementation

As a step toward taking full advantage of Open APN, it is recommended to remove CPU bottlenecks in the current software stack. One promising way to achieve this is to transfer data using RDMA-based protocols, which has been a frequent topic of reference implementation and PoC discussions in the IOWN Global Forum.

Our experiments showed that the latency in the software stack of the IDH FE is about 2.87 ms, which accounts for about 75% of the total latency. If the delay in this part can be reduced by using the RDMA technique, the overall performance of the IDH can be greatly improved. We have done some preliminary work on this, but have only been able to confirm a partial effect, as the bottleneck in the current implementation is due to a combination of factors,

including those in the JVM and our Java implementation. The work is still ongoing, as discussed in detail in Appendix B.

7 PoC's Contribution to IOWN GF

IOWN GF Task Forces that contributed to this PoC are listed below:

Table 5. Task Force Contributions

CONTRIBUTION	TASK FORCE	STUDY ITEM OR WORK ITEM	COMMENTS
A	IDH TF	IDH Functional Architecture development	This PoC demonstrates the fundamental values of IDH functional architecture and relevant IOWN technologies, such as the IOWN GF Open APN and the new IDH FE implementation model.
B	DSDT TF	Digital Twin Architecture development (IDH as a technical enabler of it)	DTF TF's analysis is considered a referential input to drive IDH technology development.

Appendix

A. Basic performance evaluation of Open APN

As a preliminary to the IDH tuning, we performed a baseline evaluation to understand the characteristics of Open APN. Latency, throughput, and packet loss characteristics under various traffic loads were measured and compared to an existing local area network.

System Configuration

In the same test environment as Figure 3, evaluation was performed with the software configuration shown in the following figure.

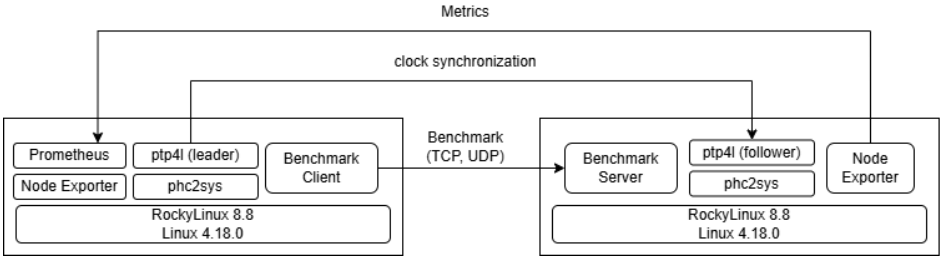

Figure A-1. System configuration of basic performance evaluation of Open APN

Table A-1. Benchmark tools

ITEM	DESCRIPTION
Time synchronization	• Time synchronization uses PTP (ptp4l, phc2sys)
Benchmark client and server	• Using a modified version of `rperf` ○ `iperf`, `netperf`, `ultra_ping`, `rperf` are tested. `rperf` and `ultra_ping` are modified to add measures and improve accuracy according to this experiment. ○ Use modified `rperf` to organize the results after verifying the consistency of the results from the above tools.

Throughput

The TCP transfer performance has been measured, achieving a throughput of 100 Gbps with approximately four CPU threads. Also noteworthy, no packet loss was observed regardless of traffic load.

Table A-2. Open APN Throughput at Various CPU Threads

THREADS	TOTAL THROUGHPUT (MBPS)	THROUGHPUT PER CORE (MBPS)	RETRANSMISSIONS (PACKETS)
1	25407	25407	0
2	48716	24358	0
3	72353	24117	0
4	91254	22813	0
5	94915	18983	0

6	94912	15818	0
7	94948	13564	0
8	94944	11868	0

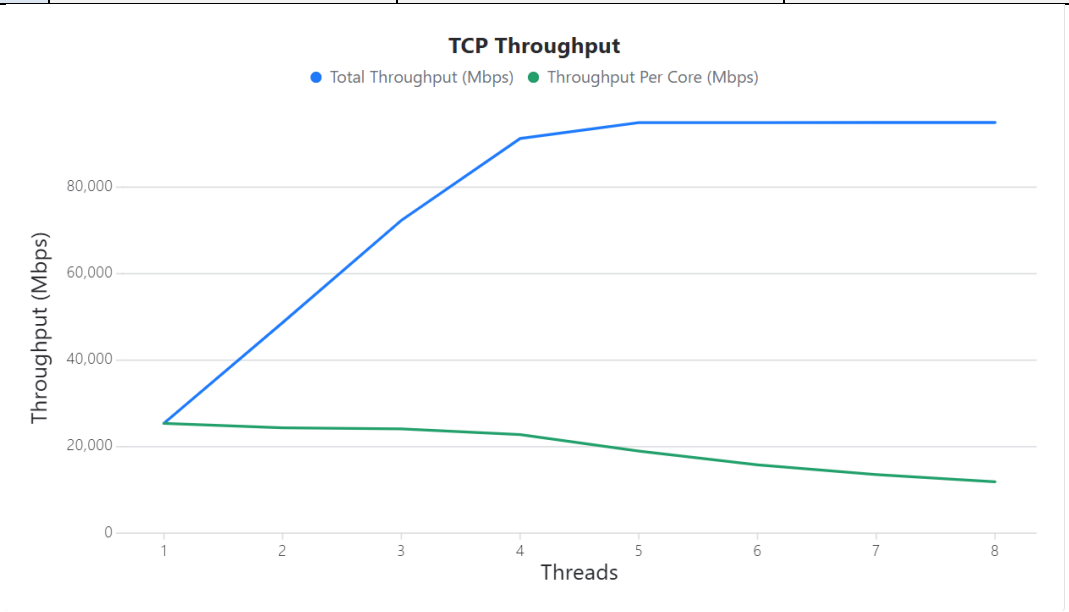

Figure A-2. TCP Throughput at Various CPU Threads

Latency

We have measured the transfer latency of Open APN is 138.6 µs, which is quite close to the theoretical value of optical transfer. The software stack adds 71.4 us of delay.

- Measured UDP packet forwarding latency at a sufficiently low load (<1%)
- Use `perf` to track the execution time of kernel functions and measure the latency at each stage

Table A-3. UDP Latency of Open APN

Item	Latency (µs)
Total	210
User Space (send)	< 0.1
Kernel (send)	12.6
Open APN	138.6
Kernel (recv)	55.8
User Space (recv)	< 0.1

Latency Comparison Between Open APN and Local Network

We have measured TCP packet latency with 0% ~ 100% traffic load on both networks. The following observations have been confirmed:

- Open APNs can maintain stable latency regardless of traffic load, whereas typical networks experience high latency when load exceeds 90%.

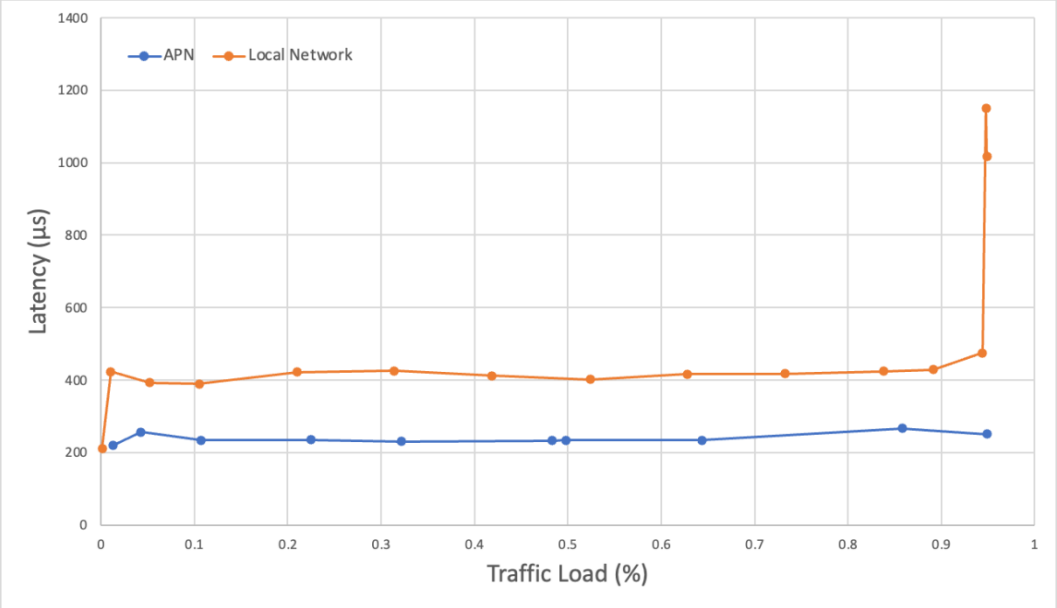

Figure A-3. TCP Latency at Various Traffic Load

- The latency of the local network is small at low traffic load (about 0.1%), but as the load increases, it becomes slower than the Open APN, which has a physical distance of 30 km longer.

- Also, the Open APN is not much affected by packet size, while the local network is greatly affected by packet size. For example, with 64B packets, the local network can demonstrate the speed of the local network with a transfer latency of 32 µs, but with 16 kB packets, it is reversed by the Open APN.

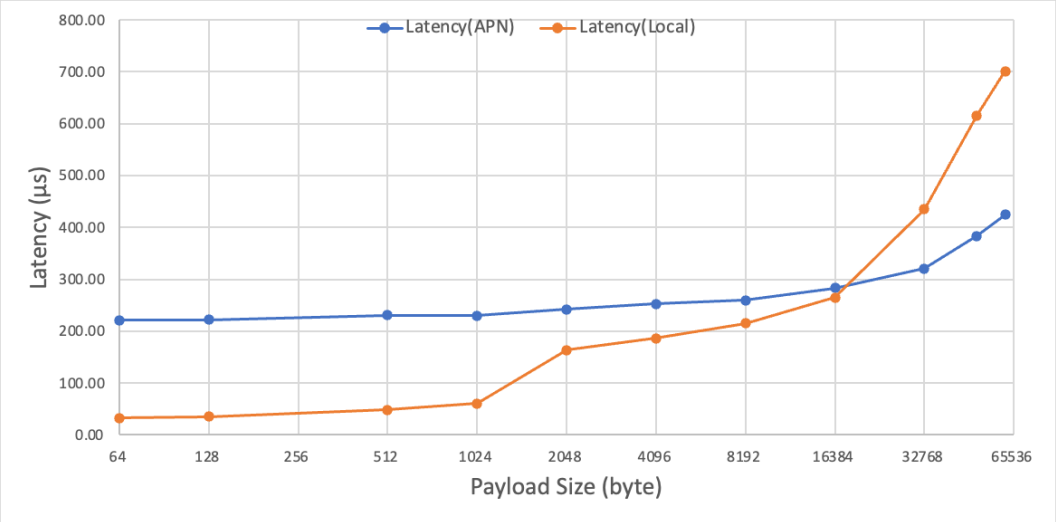

Figure A-4. TCP Latency at Various Payload Size

Latency Jitter

The latency jitter of Open APN was measured under each traffic load. The jitter is basically <10μs and lower than the local network in all cases. We confirmed the system on the Open APN can assume that it is a jitter-free network.

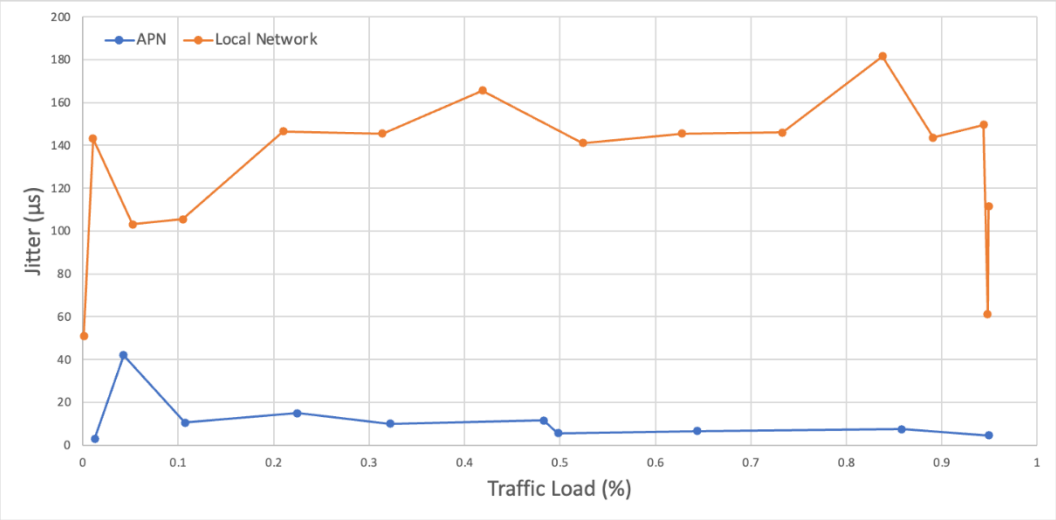

Figure A-5. Latency Jitter of Open APN

Open APN Characteristics Summary

We confirmed that Open APN will be able to handle data between remote data centers as if they were local, especially when handling data sizes larger than 16 kB.

Table A-4. Open APN Characteristics Summary

ITEM	OPEN APN
Jitter	Almost nonexistent
Throughput	Achieves stability up to 100 Gbps
Latency	Close to the theoretical value of optical transfer
Operation under high traffic load	Same as at low-traffic load
Transfer characteristic	Faster transfer (~100 Gbps) of large data (>16 kB) with lower latency than the local network
Is Open APN a high-bandwidth, low-latency version of the general network?	No. Other than bandwidth and latency, it has characteristics not seen in traditional networks. • High load tolerance • Packet Size-Independent Transfer Delay • Zero Jitter

B. Initial investigation for RDMA integration into IDH

This section shows the result of a preliminary study on incorporating an RDMA technique into IDH, although this is still a work in progress.

Characteristics of RDMA data transfer over Open APN

We measured the raw characteristics of RDMA communication between servers using OpenUCX's `ucx_perftest`, a general-purpose communication library that supports RDMA (RoCEv2).

In terms of latency, the results were almost identical to UDP with the same payload. In addition, jitter was always kept below 0.1 µs, regardless of traffic load.

Table B-1. Latency and Jitter of RDMA Data Transfer over Open APN

Item	UDP (µs)	RoCEv2 (µs)
E2E latency	210	219
Jitter	< 0.1	< 0.1

For throughput, the following results show that CPU utilization can be reduced compared to TCP.

As shown in the figure below, throughput is linear and only limited by latency up to the upper limit of the network; there is no CPU bottleneck as there was with TCP and stable performance can always be achieved regardless of traffic load.

- Because of the overhead, including RDMA headers, the maximum performance is about 8% lower than TCP (without UCX). However, when CPU utilization is low and the number of threads is small, CPU bottlenecks do not occur and performance is better than with TCP.
- The performance degradation is also seen for TCP transfers using OpenUCX, where the use of the OpenUCX library seems to have a ~5% performance impact, and thus the overhead of RDMA (RoCEv2) is considered to be ~2%.

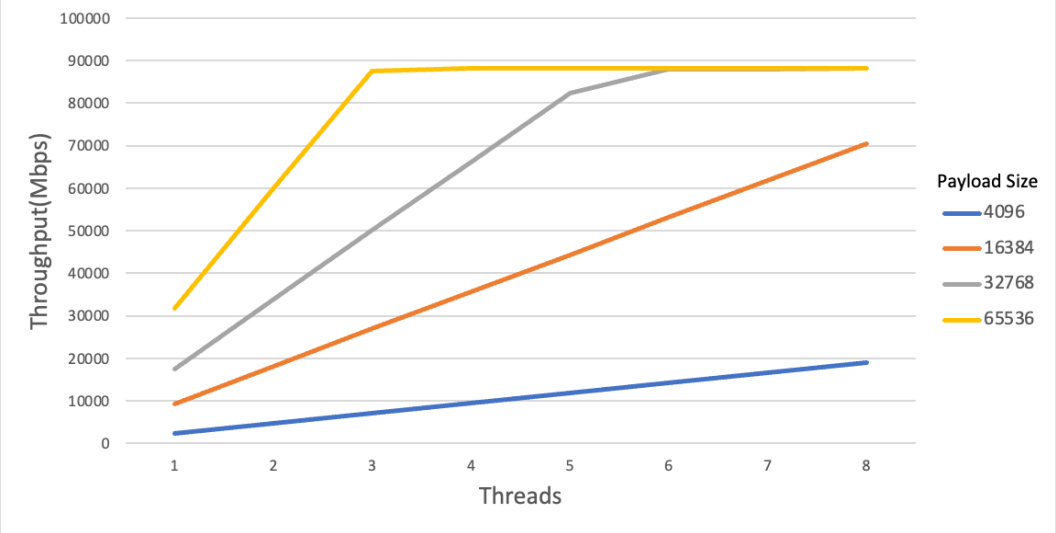

Figure B-1. Throughput using RoCEv2 over Open APN

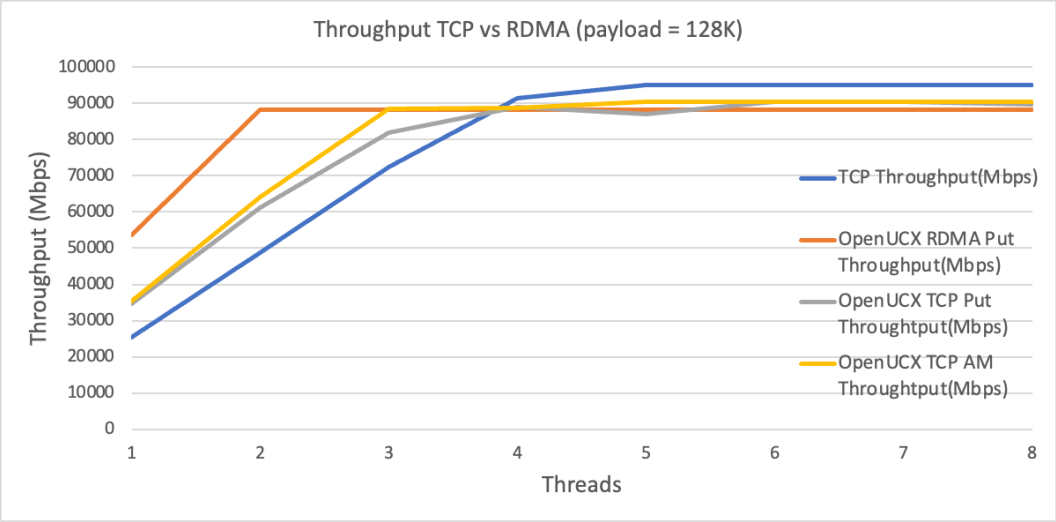

Figure B-2. Throughput comparison between RoCEv2 and TCP

With RDMA, if the payload (packet) size is large enough, it is possible to achieve nearly 100 Gbps performance even with 1 thread, as shown below. To achieve 100 Gbps, TCP requires more than 4 CPU cores, while RDMA requires less than 1 core.

Latency is generally proportional to payload size, the following parameter settings are recommended for actual operation.

- To take advantage of the Open APN bandwidth, a payload size of 512 kB or larger is recommended.
- To take advantage of the low latency of Open APN, it is recommended to calculate the payload size according to the maximum latency allowed.
- In the case of this test environment (30 km distance), if the payload size is 256 kB or less, the impact on latency is less than 10%, which is considered sufficient to demonstrate the low latency characteristics of Open APN.

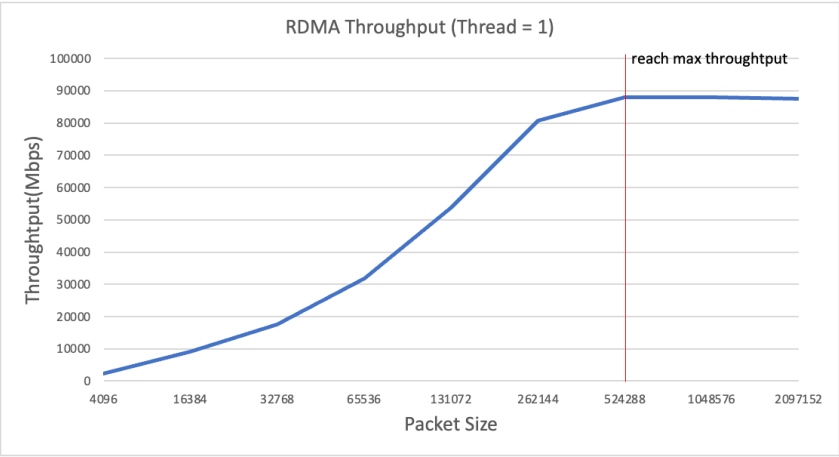

Figure B-3. RDMA Throughput on 1 CPU thread

RDMA Integration into Storage Connector

For the Storage Connector library, which was a major bottleneck in our FE implementation, we tried to speed up the data read operation from MinIO using RDMA. The figure below shows the basic design of our RDMA implementation, which aims to maintain basic compatibility with our existing IDH software.

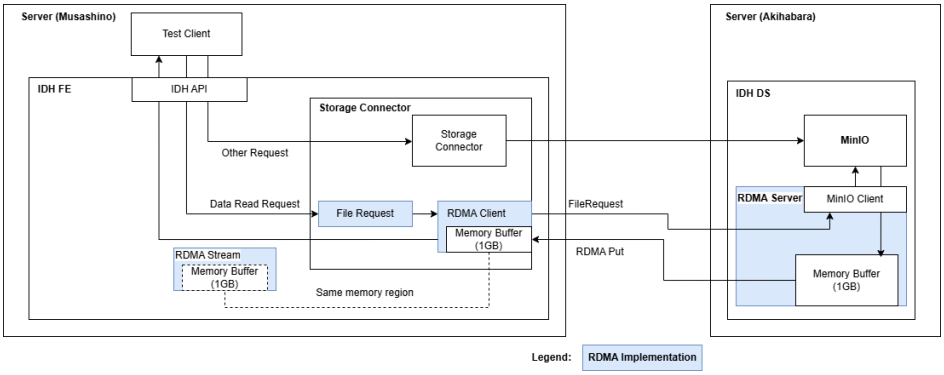

Figure B-4. RDMA implementation in Storage Connector

The following three patterns of RDMA implementations were tested and compared to the Storage Connector without RDMA modification.

- RDMA Normal: Implementation that allocates and uses a different send buffer for each request.
- RDMA /w request arena: Implementation in the client that uses the same memory to send multiple requests.
- RDMA /w server buffer arena: Uses the same memory on the server in addition to the client.

RDMA /w Server Buffer Arena (hereafter simply referred to as RDMA) only outperformed the unmodified Storage Connector, reducing latency by 36%. Upon closer analysis, this result is roughly consistent with the predicted performance gains from reducing the number of data copies.

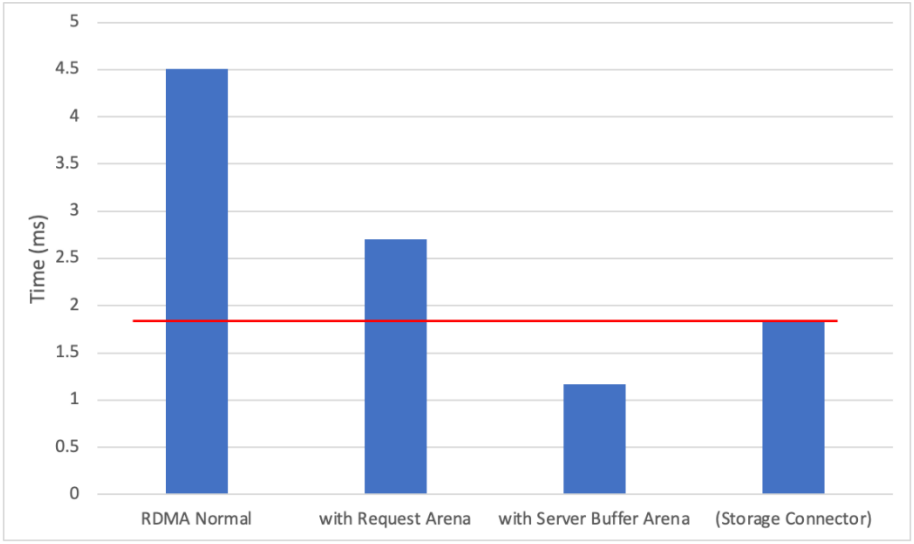

Figure B-5. Effects of RDMA Implementation of Storage Connector

From a throughput perspective, the same trend follows because throughput is dependent on latency.

Table B-2. Throughput of RDMA-Enabled Storage Connector

ITEM	STORAGE CONNECTOR (UNMODIFIED)	STORAGE CONNECTOR (RDMA)
Throughput (Block size = 1MB, Thread = 1)	1874 (MB/s)	2515 (MB/s) +34.2%

Although the RDMA implementation has improved performance, it is still very large in terms of taking full advantage of Open APN. A detailed latency analysis showed that the memory copy required 1.2ms due to processing in the JVM, which needs to be reduced for the advanced implementation. Also, the end-to-end jitter was more than 1ms, which was also generated by the JVM. In the current configuration, the JVM jitter has become the dominant term of the total jitter, negating the benefits of Open APN's zero jitter.

RDMA Integration into IDH

We also measured the performance of the IDH with the RDMA version of the Storage Connector described above and found no advantage over the unmodified version. The performance analysis showed that the reason for this is on the FE side calling the Storage Connector, and that the design of the FE also needs to be changed to account for changes in the performance characteristics and overhead of the library. We plan to work on improving this part of the design.

Summary

- Using the RDMA protocol in Open APN Environments
    - While RDMA requires a lossless network, it is difficult to utilize bandwidth in existing networks due to packet loss caused by congestion, Open APN's packet-loss-free and low-jitter characteristics make an RDMA technique easy to implement.

      Compared to TCP data transfer, CPU utilization can be reduced to less than a quarter, which is a sufficient benefit for implementing RDMA.
- Improving IDH performance with RDMA
    - By implementing RDMA in the Storage Connector library, which is responsible for data transfer, both throughput and latency in this part were improved by about 30%, confirming a certain effect. On the other hand, it was found that there were other issues to be addressed in improving the JVM-based data transfer implementation.
    - No effect was observed when the RDMA version of Storage Connector was incorporated into IDH. This is because the existing implementation does not assume the characteristics of the RDMA library.

C. Coverage of the Selected Features described in the IDH PoC Reference Document

The IDH PoC Reference document describes several features to be tested, called “selected features”. The scope of “selected features” tested and evaluated in this PoC is shown in the table below:

Table C-1. Coverage Scope of the Selected Features described in the IDH PoC Reference

SELECTED FEATURES DESCRIBED IN POC REFERENCE DOCUMENTS	COVERAGE STATUS **✓: IMPLEMENTED AND TESTED IN THIS REPORT**
Federation function	✓
Get Communications over Open APN	✓
Data filtering and preprocessing	✓* * This report leaves out the analysis of this perspective.

Document History

VERSION	DATE	BY	DESCRIPTION OF CHANGE
1.0	Feburary 20th, 2025	INOUE Tomohiro	Release version