SCinet DTN-as-a-Service Framework

Se-young Yu, Jim Chen, Fei Yeh,
Joe Mambretti and Xiao Wang
International Center for
Advanced Internet Research
Northwestern University
Chicago, USA
{young.yu, jim.chen, fyeh, j-mambretti, xiao.wang2}
@northwestern.edu

Anna Giannakou, Eric Pouyoul
Lawrence Berkeley National Lab
California, USA
agiannakou@lbl.gov, lomax@es.net

Marc Lyonnais
External Research
Ciena
Ottawa, Canada
mlyonnai@ciena.com

***Abstract*—Transferring big data over Wide Area Networks (WANs) is challenging because optimization is dependent on the specifics of multiple parameters. Network services, paths, and technologies have different characteristics, including loss rate, latency, and available capacity. Yet, frameworks currently used to configure and orchestrate transfer systems, measure performance, and analyze results have limited capabilities. We propose a framework, DTN-as-a-Service (DaaS), for high-performance network data transfers using and integration of techniques, including virtualization, network provisioning, and performance data analysis. This framework has a modular design for supporting multiple transfer tools, optimizers and orchestrators for the data transfer environment, including *Docker* and *Kubernetes*. We present a *Jupyter* based workflow for high-speed network data transfer in data-intensive science and evaluate the performance of the transfer with a simple programmable visualizer implemented in the framework. This framework has been implemented as a prototype at two recent SC supercomputing conferences. With the increase in the number and the capacity of WAN links at the conferences (multiple 100 Gbps WAN circuits), the challenges involved in setting up, testing, debugging, verifying and running applications on high-performance systems connecting to the conference SCinet WAN circuits also increase. The SCinet implementation of the DaaS framework for the conference community allowed users to control hardware, software, and network infrastructure for high-speed network data transfer, primarily for large scale applications. Through the evaluation of the framework in our test setup, we demonstrated that NVMe over Fabrics with TCP is twice as efficient compared to using conventional TCP in high-speed NVMe-to-NVMe transfers. We also implemented a 400 Gbps LAN experiment to evaluate the DaaS framework.**

***Keywords*-Data Transfer Nodes, NUMA, measurement, NVMe over Fabrics, SCinet**

I. Introduction

Transferring data among highly distributed locations is an essential part of national and international collaborative science research. Data Transfer Nodes (DTNs) play a critical role in collaborative science, by enabling optimized performance in transferring data at high-speed.

SCinet DTNs provide services to allow users to transfer data over high-capacity networks, including over 100 Gbps paths. SCinet DTN projects require a systematic approach to optimize, test, and monitor the performance of the data flows over LANs and WANs.

In our previous study, we revealed the performance variations among different NUMA and storage configuration in a specific local area network [1]. The study showed no single optimization can address all of the different sets of transfer environments to obtain optimal performance. Instead, each DTN needs to be optimized for each specific transfer. It is also essential to monitor the performance of DTNs to anticipate potential failures that may impede high-performance network data transfers.

This study complements the previous work by proposing a framework for orchestrating and measuring the performance of DTNs in high-performance network data transfers. The framework can identify and optimize configurations in DTN for OS-level virtualization technologies, orchestrating data transfer, and evaluate the performance of the transfer.

Existing frameworks orchestrate high-speed transfers and allow monitoring transfer results [2]–[4]. These frameworks optimize the underlying system using a static set of parameters or require system administrators to configure the system to fully optimize the DTNs in limited environments.

Our framework complements the existing frameworks by integrating OS, hardware, and software using a Jupyter [5] controller to evaluate the performance of DTNs for high-speed network data transfer in different containerization technologies, including *Docker* and *Kubernetes* [6]. Although our framework can be used as the basis for a high-performance network data transfer system, this framework focuses on the ability to test different optimizations in the system for high-speed data transfers and help system administrators examine optimization parameters for a specific transfer. The modular design of the framework allows integrating a broader set of optimizations, configurations, transfer tools, evaluation methods, and other variables. We describe the components of the DaaS framework and how they can be implemented, and we present our example implementation. We evaluate the framework by measuring the performance of NVMe over Fabrics (NVMEoF) with TCP over 100 Gbps WANs and discuss the implications of the results.

By providing the framework for testing high-performance

Funding Agency: NSF IRNC Grant Award 1450871

978-1-7281-6666-7/19/$31.00 ©2019 IEEE
DOI 10.1109/INDIS49552.2019.00006

network data transfers, we innovate in techniques for measuring performance of DTNs systematically and show how to analyze the result and to optimize for a specific transfer.

Our framework integrates both hardware and software optimization, configuration, testing, and evaluation of high-speed network data transfer using Jupyter, a workflow manager. Users of the DaaS framework can specify their workflow in their Jupyter controller. The controller can call modules to perform tasks that fit their needs.

Our framework ensures consistent performance of high-speed data transfers for data intensive science. It is out of the scope of this study to develop another transfer orchestrator or transfer protocols for WAN. Instead, we focus on integrating different optimizations and analyzing their impact on the underlying system using a framework to test and monitor DTNs. We are building our framework to support existing transfer orchestrators and to allow testing these transfer services with specific optimizations the framework provides. Furthermore, because of the flexibility of this design, we anticipate this framework to assist in migrating such systems from static configurations and deployments to dynamic models that enhance optimizations by adjusting to particular circumstances.

In the next section, we provide background knowledge related to our study. We provide an overview of relevant work in section III and the design rationale of the framework in section IV. We describe the implementation of our prototype of the DaaS framework in section 3. In section VI, we provide our evaluation and discussion. We conclude our study in section VII and describe future works in section VIII.

II. Background

A. DTNs

Data Transfer Nodes (DTNs) are systems specifically designed for high-performance network data transfers. Generalized systems cannot be used for these types of services. DTNs can be built with different hardware and network capabilities to allow for sharing data among DTNs in WAN. Typically DTNs run software tools for high-speed data transfers between remote endpoints, e.g., GridFTP [7], Globus [3] or XrootD [4].

DTNs are essential components for Science DMZs [8], which allow for segments of networks to be optimized for large capacity transfers. DTNs are typically connected to Science DMZ switch/routers and serve data to both local site and remote sites over the special circuit over LANs and WANs.

To achieve high-performance network data transfers in WAN with DTNs, the DTNs need to be optimized for the specific transfer. This optimization includes specialized tuning and configuration of hardware, firmware, BIOS, OS, and application-level tuning as well as using the transfer protocol optimized for the high-speed network data transfer.

B. DTN-as-a-Service

DTNs are commonly used in diverse scientific implementations in different large scale research facilities, such as

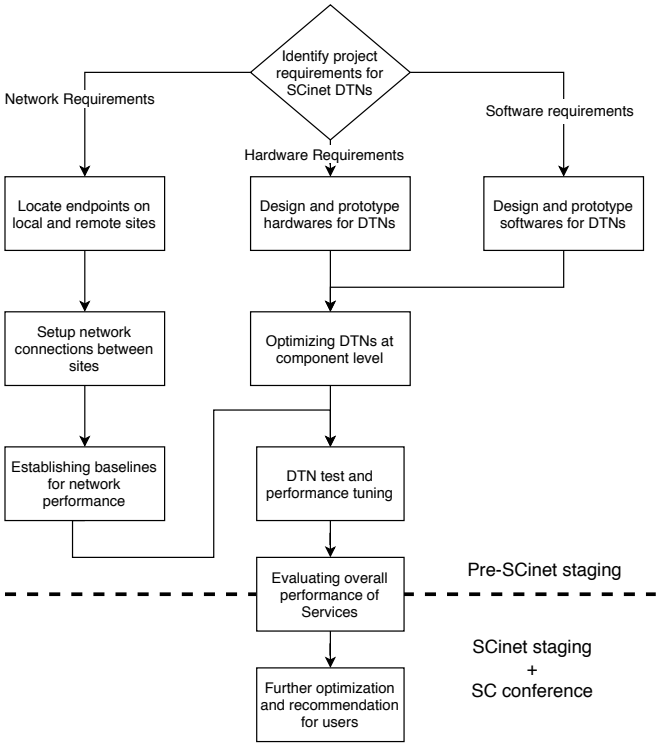

Fig. 1. SCinet DTN-as-a-service workflow

National Lawrence Berkeley Laboratory[1], DOE[2], Argonne National Laboratory[3] and those at research universities.

Each of the DTNs has different hardware, network connectivity, workloads, and software used for different projects. Consequently, no standard set of DTNs can be assumed. It is necessary to anticipate a heterogeneous environment. To support these diverse implementations, our DTN-as-a-Service provides a framework that helps to implement tools to optimize them for specific needs and establish performance baseline of the DTNs. We define a science workflow in a DTN and map the workflow with our framework to resolve DTN performance issues in high-performance network data transfers.

The DTN-as-a-Service allows a user to measure the performance of DTNs using an interactive Jupyter controller. The controller invokes functions to set up, optimize, and measure high-performance network data transfers. The controller also provides a function to evaluate and monitor each transfer.

C. SCinet DTN

Since SC17, the DaaS project has been partnering with technology companies and the SC community to define requirements, select technology components, implement new DTN functions and features, integrate new hardware and software stacks, and reach out and support to additional SC community users. These activities resulted in the development of a SCinet DTN-as-a-Service framework that meets the specific need for SCinet DTNs. A SCinet DTN was first implemented as an integrated prototype service to set up, test,

[1] http://scs.lbl.gov/dtn
[2] https://fasterdata.es.net/science-dmz/DTN/
[3] https://www.alcf.anl.gov/user-guides/data-transfer

tune, verify, debug, monitor and run data-intensive applications on high-performance systems that connect to the SCinet WAN circuits and other SCinet high-performance networks, including the conference WAN. This service demonstrated that it enabled high-performance network data transfers for data-intensive science projects before and during SC conferences. The SCinet DTN provides integrated infrastructure hardware, software stacks, and continuous DevOps integration to enable researchers to transport their big science data over network quickly, reliably, and straightforwardly.

Encouraged by the success of the prototype, we are now transitioning the SCinet DTN service to become a standard SCinet offering for future SC supercomputing conferences. The DaaS framework is implemented as shown in Figure 1. A key point is that DaaS can be implemented not only as a single static service, but as a flexible service can adjust dynamically to the requirements of multiple applications. DaaS is a platform that can support many flavors of a transfer service to ensure it can meet the precise individual requirements of the researchers and demonstrators who use them to transfer large amounts of data over high-performance networks.

D. OS-level virtualization

This approach uses OS-level virtualization, which allows multiple user-space instances to share a kernel running on a host. Compared to full virtualization, OS-level virtualization reduces the overhead by using a host OS system call from the user-space instance.

OS-level virtualization is used to provide an isolated environment for different applications. Different technologies exist to implement OS-level virtualization, such as LXC, *Docker* [9] and Singularity [10].

Some orchestrators provide a platform for managing OS-level virtualization [6]. These orchestrators control selection of virtualization technologies, allocation of resources, and deployment of user instances to specific hosts and networks.

E. NUMA

Non-Uniform Memory Architecture [11] allows multiple processors in a system to access the main memory simultaneously. The NUMA architecture reduces the memory access time for multi-processor systems when each processor accesses the local memory allocated from the main memory but causes overhead when accessing memory allocated for other processors because of the nature of cache-coherency.

NUMA in DTNs allows faster data processing with multi-processors operate on their local memory when applications run in parallel with each other. However, the applications need to access foreign memory through the processor interconnect if the NIC is located at the foreign NUMA node.

In high-speed WAN transfer, transfer applications may benefit from NUMA by avoiding use of foreign memory by binding processes to the processor the network interface is connected to. On the other hand, it is possible to use the other processors when there are not enough processing resources available in the local processor.

F. TCP

TCP [12] is the most popular transport layer protocol used on the internet. It provides reliable transport between end hosts using sliding window and acknowledgments. Many applications rely on TCP for its reliability and ordered delivery.

TCP uses congestion control to achieve max-min fairness among the flows in a shared path. It increases the congestion window size for each flow to increase the size of segments on-the-fly when the link is underutilized. TCP reduces the congestion window size when there is a signal for congestion, usually a packet loss detected by the sender.

This behavior allows TCP flows to avoid congestion collapse when many flows share a network path. At the same time, it may lead to under utilization of the network capacity of a long fat pipe when the path loses packets without congestion. Because the congestion window size increases when the sender receives an acknowledgment, it takes a longer time for flows in a high-latency path to recover the congestion window after each packet loss.

G. NVMe over Fabrics

NVMe over Fabrics (NVMeoF) [13] exports an NVMe interface over networking fabrics by mapping NVMe commands to transport layer protocols. NVMeoF allows a local machine to issue NVMe commands to storage devices attached remotely, enabling it to provide block devices to the OS and applications running on the local machine.

Unlike iSCSI, NVMeoF does not need to translate SCSI command for NVMe devices. Instead, the NVM subsystem forwards the NVMe command directly to the storage devices.

NVMeoF can be implemented in many different network fabrics, such as Fibre Channel, Infiniband, RoCE, and TCP/IP.

III. Related Works

In this section, we review the related literature and how our study complements the related works.

Several studies proposed a framework for orchestrating high-speed data transfer in WAN. Globus Striped GridFTP Framework [7] orchestrates high-speed network file transfers and manages accessing, processing, and security. The resource management of the framework relies on Globus Toolkit, which provides broader services including execution management, web information services, and the discovery of services. *GridFTP* uses Globus eXtensible I/O (XIO) system [14] to implement its transfer protocol, supporting FTP and UDT [15].

Big Data Express [2] provides a scalable system architecture and transfer orchestration for high-speed WAN transfer with resource management and modular transfer protocols. It consists of service scheduler, transfer protocol, network provisioning, and authentication.

These frameworks focus on orchestrating data transfer, optimize the underlying system and transfer protocol, resource management, and developing high-speed transfer protocol. Our framework complements these frameworks with the ability to test systems with different hardware to component level,

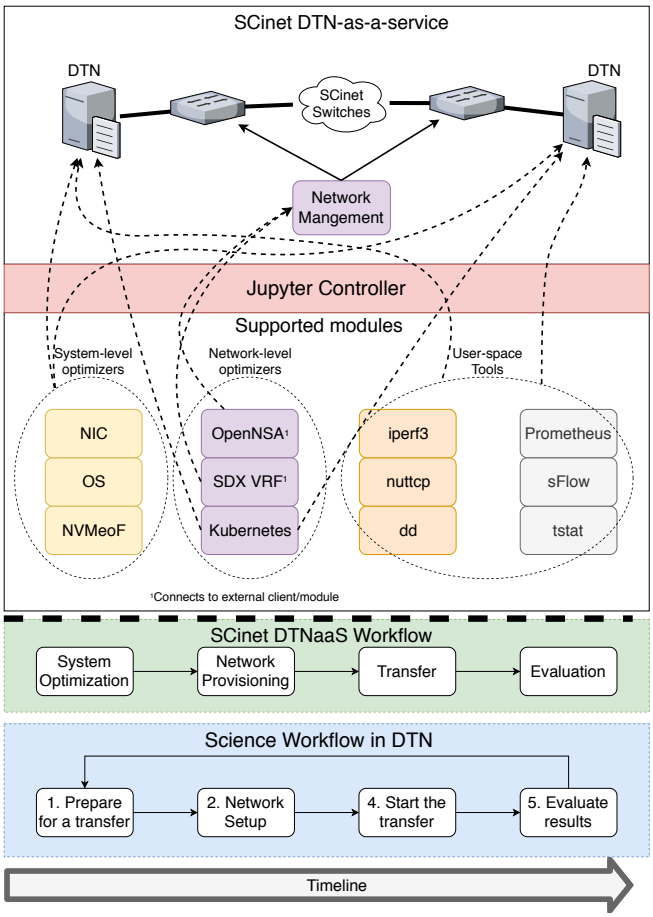

Fig. 2. Design of SCinet DTN-as-a-service framework

OS, and application settings for DTNs with different OS-level virtualization. Furthermore, the framework can implement orchestrators for existing transfer applications to benefit from their performance. Our framework is not tied with a single transfer application; instead, it focuses on orchestrating different systems and compare each other to achieve better performance in high-performance network transfers.

Kubernetes [16] provides a framework for managing the deployment of user-space instances to a cluster of systems hosting OS-level virtualization techniques. Our framework sets up an environment for transfer applications in *Kubernetes* and other orchestrators. Our framework allows configuring the most efficient networking environment with OS-level virtualization such as using host networking[4] in Docker and Kubernetes. The framework also orchestrates transfer protocols to establish a connection between two instances running in different OS-level virtualization techniques.

The DaaS framework has a modular design for easy integration with existing data-intensive science. The framework can be integrated with many SCinet DTN projects to provide better performance and analysis of DTNs.

IV. Design

Our framework is designed for systematic testing of SCinet DTNs and to comply with a SCinet DTN science workflow

[4]https://docs.docker.com/network/host/

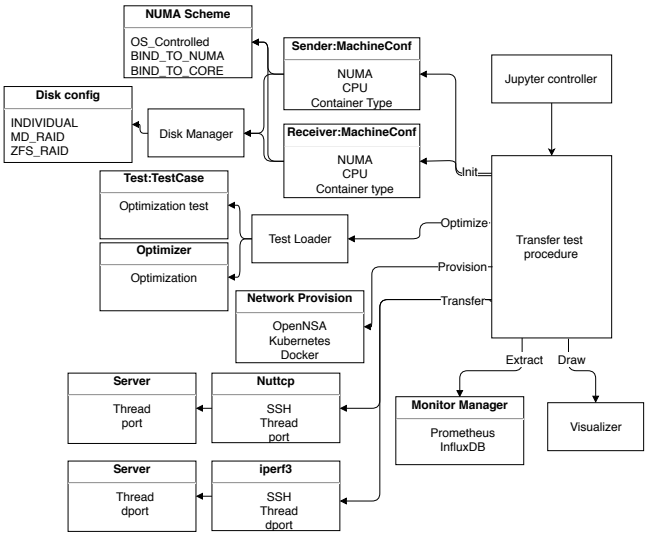

Fig. 3. Overview of SCinet DTN-as-a-service implementation

shown in the figure 2. Our framework consists of system optimization and network provisioning and result-analysis modules. Each of these modules exposes their API for a testing module specifying a sequence of tests to evaluate the performance of transfer.

Each module is mapped to a science workflow for DTNs, as shown in Figure 2. Modules mapped into the science workflow provide APIs for a testing script. The testing script specifies which modules and parameters to use in each step.

The system optimization module provides functions to optimize systems for high-performance WAN transfer. This module is prepared for system administrators with an elevated privilege to prepare an optimal environment to experiment with high-performance network transfers. These optimizations include kernel parameters, NIC ring buffer size, IRQ optimizations, and NVMeoF target and host configuration.

The network provisioning module provides functions to establish a path between two DTNs on the WAN. This module includes configuring the network for high-performance data transfers in cluster networking and requires deploying controllable network devices along the path.

The transfer module translates standard API function calls into execution functions for each transfer protocol. This function includes locating data, executing transfers, and verifying the transferred data in the destination.

An evaluation module provides functions that extract transfer data from monitoring systems and prepare data for analyzers. The module provides API to query data from monitoring systems and output results to feed to visualization.

A typical data transfer procedure follows the science workflow in SCinet DTNs shown in the figure 2.

This section describes our prototype implementation of the framework for SCinet DTNs. Figure 3 shows an overview of the implementation detail.

V. Implementation

A. Init modules

Init modules identify system hardware configuration and provide APIs for transfer modules. A *Machineconf* module is

implemented to scan NUMA configuration, specify container type and initialize. *Disk Manager* scans Storage configuration and provides API to format, mount, and build RAID.

These modules include a system setup tool to configure the NVMeoF target and initiator on the SCinet DTNs.

B. Optimization modules

The system optimization module consists of several tests and optimizers. Each test looks for values from kernel TCP buffer sizes, the Linux traffic control, Maximum Transmission Unit (MTU), CPU governor profiles, default Linux system services, NIC drivers, and PCI-express link status to check if they are optimal for high-speed WAN transfer.

These tests are implemented as a python unittest TestCase object and are called from a python TestLoader class and output the result of the test. The main module will accept the network interface names as parameters and set the baseline optimization by default. The test cases can be loaded from a testing script, and individual optimizer functions can be called from the main optimization module.

C. Network Provisioning Modules

The network provisioning module establishes a high-performance network path between DTNs to enable network data transfers. This module is responsible for identifying DTNs in the framework, finding a path between DTNs, and requesting a high-speed path for network data transfers.

The modules include cluster network orchestrator to set host network on the containers running in *Docker* and *Kubernetes* to avoid bottlenecks when using overlay network. These modules also include an NSI-compatible client to set up layer 2 circuits between two hosts running the framework. These modules require additional configuration to allow it to connect to the external network management system and communicate to virtualization orchestrators.

D. Transfer modules

Transfer modules implement common API for different transfer protocols for the testing script. The prototype implementation contains *nuttcp* and *iperf3* module that can be used.

Transfer modules are NUMA aware and implement CPU core-binding schemes for the transfer tools.

1) **Pinned to Core** scheme where each transfer process is bounded to a specific core in the local NUMA node.
2) **Pinned to NUMA** scheme where each transfer process is bounded to local NUMA node using *numactl*, which allows the O/S to schedule the process on any core in that processor but not the other.
3) **System Balanced** scheme where each transfer process is not bounded to any processor so the O/S can schedule the process in any online processor core in the system with the Completely Fair Scheduler (CFS).

We chose to implement transfer modules for *iperf3* (memory-to-memory), *nuttcp* (NVMe-to-NVMe) and *dd* (NVMeoF NVMe-to-NVMe) because of their zero-copy transfer support and they consume fewer resources compared to other tools, such as *bbcp* and *GridFTP*.

TABLE I
PARAMETERS AVAILABLE FOR TESTING PROCEDURES

Parameter	Description
Sender	*Machineconf* for sender
Receiver	*Machineconf* for receiver
num_thread	Number of transfer threads to use
cport_num	Control ports to use
dport_num	Data ports to use
disk_conf	NVMe configuration scheme used in the transfer
scheme	NUMA scheme to use
src_path	Path to directory for files to send in sender
dst_path	Path to directory for files to receive in receiver
file_size	Size of files to create in src_path

The transfer module supports two types of integrity checks on each file transferred; file size and checksum. When there is a mismatch between the original and the transferred file due to lost connection or corrupted during transmission, it adds the file to the retransmission queue and retransmits the file.

Currently, orchestrating the transfer is done using *paramiko*, an SSH library for python for *nuttcp* and *iperf3* servers. The library allows users to authenticate against the SSH server. The performance of each transfer depends on how efficient the underlying transfer tool and also the kind of congestion control the tool uses.

E. Evaluation Modules

The evaluation module implements extracting data from the monitoring system and visualization of the data. The current implementation supports *Prometheus* and *influxdb* using REST for data source through Monitor Manager module. Metrics, including processor, network, and NVMe utilization can be extracted from the data source and visualized using Visualizer. Extracted data is grouped using their NUMA binding and NVMe configuration schemes and used in the visualizer module for analyzing the statistics of the transfers. These data can also be evaluated using the external analysis modules.

F. Testing procedures

A testing procedure specifies machines to use, storage setup, transfer size, transfer protocol, and monitoring type. Table I shows the test specification and its variables.

Machine details are specified using the *Machineconf* object, which contains machine name, NUMA node number, IP address, interface name, and RSA key. The procedure module implements a function that translates testing specifications into scripts that configure storage, creates synthetic files, orchestrate servers, and export data using the modules described in the previous subsections. Evaluation module extracts OS and hardware metrics during the transfer and visualizes the data.

In the following section, we evaluate the framework design using a prototype we developed. We performed experiments to show how to optimize the system, choose the transfer protocol, and evaluate the outcome of the transfer using our prototype implementation of the framework.

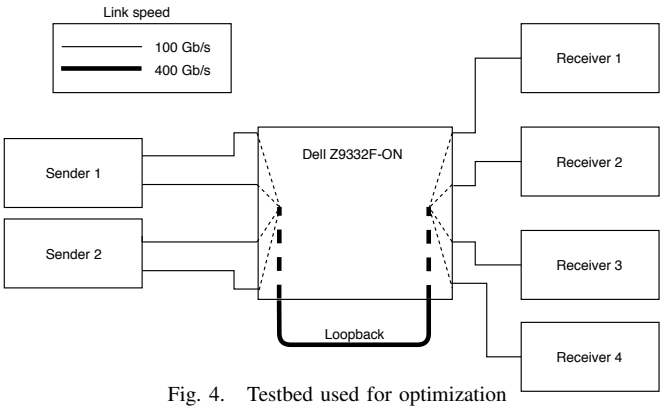

Fig. 4. Testbed used for optimization

TABLE II
SYSTEM SETUP FOR 400 GBPS LAN TRANSFER

	Sender 1,2	Receiver 1,2	Receiver 3	Receiver 4
CPU	AMD EPYC 7371 16-Core Processor	Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz	Intel(R) Xeon(R) Gold 5115 CPU @ 2.40GHz
Memory	DDR4-2666 126 GB	DDR4-2666 64 GB	DDR4-2666 96 GB	
NIC	Mellanox Technologies ConnectX-5			
OS	GNU/Linux 5.2.0			

VI. EVALUATION

A. Optimization

To support 400 Gbps SCinet projects, we evaluated the optimization provided from the framework and measured the performance of memory-to-memory transfers in a 400 Gbps LAN with multiple machines. We prepared six DTNs (two senders and four receivers) to transfer 400 Gbps from senders to receivers. Two senders have two 100 Gbps NICs where receivers have one 100 Gbps NIC. Table II shows the detail of each DTN used in this experiment.

We used a Dell Z9332F-ON to connect DTNs and set up two VLANs with a 400 Gb/s loopback connecting them. Figure 4 shows the network used in the evaluation. Senders sent TCP segments to for receivers using 32 iperf3 streams using our framework through four 100 Gbps connections.

We used the SCinet DaaS framework to optimize the DTNs to maximize their throughput as described in section Optimization modules. Through this optimization, we show our framework can optimize systems with different hardware for 100 Gbps transfers before using them for transferring over WANs to eliminate local bottlenecks.

Figure 5 shows the throughput of each 100 Gbps connection and 400 Gbps loopback connection. We were able to reach an average of 92 % utilization of 400 Gbps using four 100 Gbps connections for 300 seconds.

Each connection utilized between 90 to 94 % of 100 Gbps link, however, we saw between 0 to 500 packets lost each second. Through the monitoring system, we noticed that the receiver 1 and receiver 2 had less number of CPU cores and some of these cores are overloaded, leading to drop packets due to lack of resource to handle them.

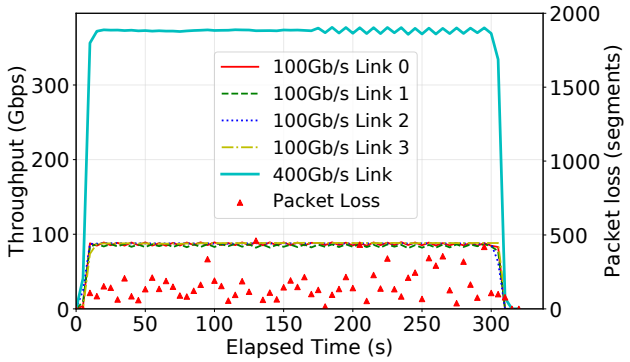

Fig. 5. 400 Gbps memory-to-memory throughput

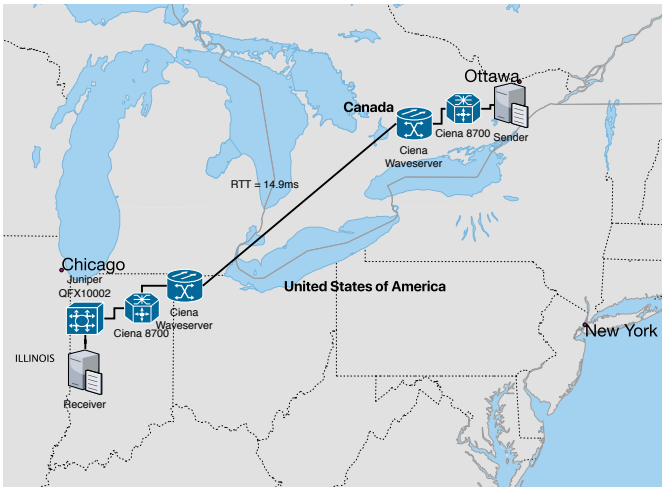

Fig. 6. Testbed used in the WAN transfer

B. WAN transfer

To evaluate the prototype implementation of the framework, we decided to emulate a WAN connection between SCinet and an international remote site. We set up two DTNs in Ottawa, Canada, and Chicago, USA. Figure 6 shows the experiment network used and Table III shows the setup of DTNs used in the experiment. Two DTNs are connected through StarLight SDX and Ciena's Network Research Platform providing 100 Gbps end-to-end network capacity. The two DTNs are separated by 2 Waveserver transponders which provide only Long Haul transport OTN protocol. Each DTN is tuned for 100 Gbps network data transfers using the optimization module, as stated in V-B.

The objective of this evaluation is to test the functionality of the framework in a WAN environment which resembles a SCinet DTN WAN service connection. This experiment allows us to test features, usability, and performance of our

TABLE III
SYSTEM SETUP FOR 100 GBPS WAN TRANSFER

	Sender	Receiver
CPU	2 * Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz	
Memory	DDR4-2666 192 GB	
NIC	Mellanox Technologies MT27800 Family [ConnectX-5]	
NVME	2 * Kingston DCP1000 (4 * 800 GB each)	8 * Samsung SSD 960 PRO 2TB
OS	GNU/Linux 5.1.0.rc4	
File System	XFS	XFS

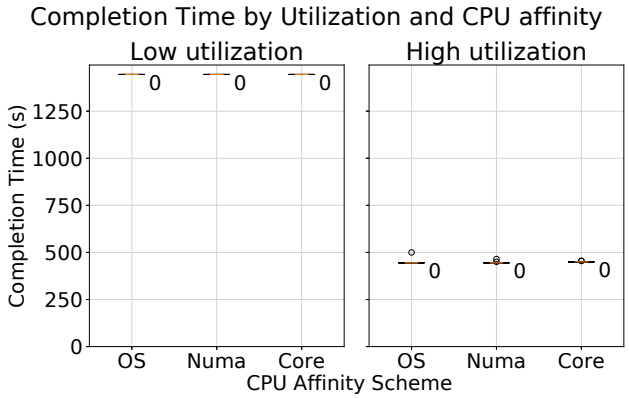

Fig. 7. Completion time of nuttcp, with range shown on the right side of each box

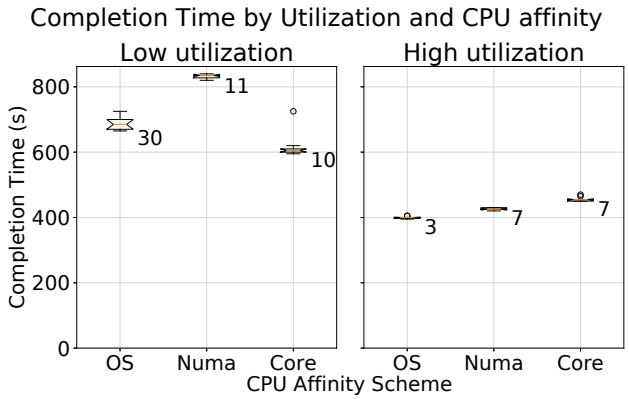

Fig. 8. Completion time of NVMeoF, with range shown on the right side of each box

framework, which we will deploy in the SCinet DTN.

In a preliminary test, we ensured there were no bottlenecks in the network capacity. We used *iperf3* to test available network capacity and achieved 98 Gbps memory-to-memory transfer using the optimizer module in our framework.

From the previous findings, we revealed that using XFS in individual storage devices provides faster performance than MD RAID and ZFS in 100 Gbps network data transfer [1]. We also found the primary bottleneck for 100 Gbps NVMe-to-NVMe transfers is NVMe I/O speed. To remedy this, we installed faster NVMe devices in the receiver to increase NVMe write speed and we manually trimmed the mounted volumes before each transfer.

Along with *nuttcp*, we used NVMeoF to demonstrate the evaluation function and post-analysis the framework provides.

In each setup, we transferred four terabytes of data in total to avoid the effects of NVMe cache in the storage controller.

Figures 7 and 8 shows the average completion time of *nuttcp* transfer and NVMeoF with TCP during the experiment, generated from the framework.

NVMeoF with TCP shows significantly faster completion time. It is more than two times faster than *nuttcp* with low utilization transfer. In high utilization transfers, the completion time gain was smaller but still more efficient. The main reason for this result is the NVMeoF has less overhead and better CPU load management compared to the nuttcp server. As shown in figures 9 and 10, NVMeoF has less average CPU

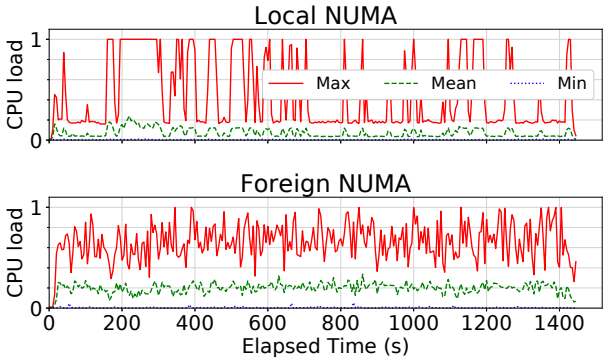

Fig. 9. Average CPU core load of nuttcp in low utilization transfer in System Balanced Scheme

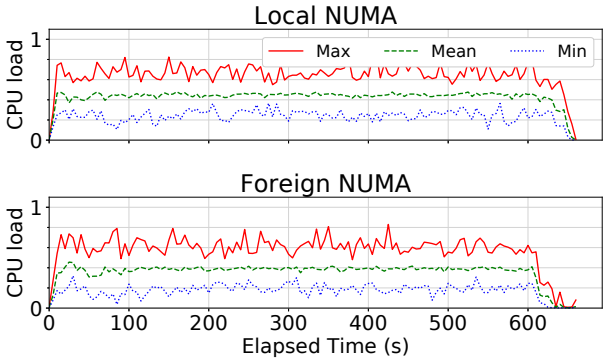

Fig. 10. Average CPU core load of NVMeoF in low utilization transfer in System Balanced Scheme

usage than nuttcp, leading to higher overall throughput. In contrast, nuttcp utilizes 100% of CPU cores and throttles the overall transfer most of the time.

Throughout the figures, one can easily compare the load distribution of different transfer tools during the experiment and notice possible bottleneck of some CPU cores.

We noticed a CPU bottleneck during low utilization transfer with nuttcp. The two CPU cores were maximized most of the time during the experiment with *nuttcp* with low utilization transfer, which is the primary bottleneck. In high-utilization transfers, we did not see a much difference because the bottleneck was moved to the NVMe devices.

Interestingly, we observed only a small variation in completion times during the experiment, showing each transfer was consistent in terms of their throughput. The consistent result shows the framework's ability to set up a consistent experiment environment. However, we noticed longer completion times in the Pinned To NUMA scheme using NVMeoF in low utilization transfers. The throughput of the Pinned To NUMA technique was lower than other schemes slightly but significantly, causing the overall transfer to take longer. This behavior suggests optimization challenges exist on the NVMeoF over the TCP module in the Linux Kernel.

In high utilization transfers with NVMeoF with TCP, the Bind to NUMA and Bind to Core schemes show longer completion time slightly, due to the CPU bottleneck. These schemes limit the number of available CPU cores compared to the System Balanced scheme. This leads to slower perfor-

mance of the two schemes compared to the System Balanced scheme, yet it is faster than the *nuttcp* because the overall overhead is smaller.

VII. Conclusion

In this section, we conclude our study by revisiting the its experimental results. We found the DaaS framework can set up and orchestrate high-speed network data transfers and analyze transfer data from the experiments.

We proposed and tested the DaaS framework for orchestrating high-speed data transfers between DTNs. We showed the framework can set up and optimize DTNs for a specific transfer and allow experimenting with different transfer environments. Different transfer modules provide APIs to use transfer applications within the framework with different NUMA schemes. The DaaS framework also allows configuring NVMeoF and transferring data from remote NVMe devices.

We implemented a prototype of the framework and evaluated high-performance network data transfers using NVMeoF over TCP in WAN. The evaluation shows the DaaS framework can establish a stable high-speed network data transfer experiment and that it is capable of identifying transfer bottlenecks from the monitoring modules. We will implement a version of the DaaS framework for SCinet 2019 to provide DaaS for national and international collaborations.

The framework does not provide the optimal solution for every possible DTNs and its transfers. It is a framework to help to optimize, test, and evaluate different configurations to find out bottlenecks of high-speed network data transfer.

VIII. Future works

The SCinet DTN-as-a-Service framework established a baseline performance for DTNs. Through its open and modular design, we plan to keep integrating new technologies such as 400 Gbps LAN, a planned 400 Gbps WAN, and NVMeoF with TCP in WAN into the framework.

This work can be extended in many directions to explore the capabilities of the framework further. We are working on integrating Globus into the framework to allow federation of users to access their existing resource. We plan to integrate more transfer protocols in the framework by implementing transfer modules for GridFTP and mdtmFTP.

To further evaluate the performance of DTNs, we plan to add support for different TCP congestion control algorithms, including TCP BBR [17]. As NVMe over Fabrics with TCP provides direct access to the remote NVMe directly, we plan to include the evaluation of different storage systems into the future development of the framework including ZFS and Intel Virtual RAID on CPU[5].

We also plan to include more OS-level virtualization and network orchestrators to find suitable configuration required for the optimum performance autonomously. Comparing the performance of different overlay networks provide insight into cloud-based networking services.

[5]https://www.intel.com/content/www/us/en/software/virtual-raid-on-cpu-vroc.html

Acknowledgment

We would like to thank our sponsors including StarLight, Metropolitan Research and Education Network (MREN), National Science Foundation (NSF), Northwestern University, Dell Technologies Inc., Ciena Inc. and more specifically Rodney Wilson who as always been a very keen supporter and contributor time and resource to this project.

References

[1] S. Yu, J. Chen, J. Mambretti, and F. Yeh, "Analysis of CPU Pinning and Storage Configuration in 100 Gbps Network Data Transfer," in *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, Nov. 2018, pp. 64–74.

[2] Q. Lu, L. Zhang, S. Sasidharan, W. Wu, P. DeMar, C. Guok, J. Macauley, I. Monga, S. Yu, J. H. Chen, J. Mambretti, J. Kim, S. Noh, X. Yang, T. Lehman, and G. Liu, "BigData Express: Toward Schedulable, Predictable, and High-Performance Data Transfer," in *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*, Nov. 2018, pp. 75–84.

[3] I. Foster, "Globus Online: Accelerating and Democratizing Science through Cloud-Based Services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, May 2011.

[4] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky, "XROOTD/TXNetFile: A Highly Scalable Architecture for Data Access in the ROOT Environment," in *Proceedings of the 4th WSEAS International Conference on Telecommunications and Informatics*, ser. TELE-INFO'05. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2005, pp. 46:1–46:6, event-place: Prague, Czech Republic. [Online]. Available: http://dl.acm.org/citation.cfm?id=1391157.1391203

[5] M. Ragan-Kelley, F. Perez, B. Granger, T. Kluyver, P. Ivanov, J. Frederic, and M. Bussonnier, "The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication." in *AGU Fall Meeting Abstracts*, 2014.

[6] R. Peinl, F. Holzschuher, and F. Pfitzer, "Docker Cluster Management for the Cloud - Survey Results and Own Solution," *Journal of Grid Computing*, vol. 14, no. 2, pp. 265–282, Jun. 2016. [Online]. Available: https://doi.org/10.1007/s10723-016-9366-y

[7] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus striped GridFTP framework and server," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2005, p. 54.

[8] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The Science DMZ: A network design pattern for data-intensive science," in *2013 SC - International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov. 2013, pp. 1–10.

[9] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[10] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLOS ONE*, vol. 12, no. 5, pp. 1–20, 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0177459

[11] N. Manchanda and K. Anand, "Non-uniform memory access (numa)," *New York University*, vol. 4, 2010.

[12] J. Postel, "Transmission Control Protocol," ietf, RFC 793, Sep. 1981.

[13] D. Minturn, "NVM Express Over Fabrics," in *11th Annual OpenFabrics International OFS Developers Workshop. Monterey, CA, USA*, 2015.

[14] R. Kettimuthu, S. Link, J. Bresnahan, M. Link, and I. Foster, "Globus xio pipe open driver: enabling gridftp to leverage standard unix tools," in *Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery*. ACM, 2011, p. 20.

[15] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks*, vol. 51, no. 7, pp. 1777 – 1799, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606003057

[16] K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running Dive into the Future of Infrastructure*, 1st ed. O'Reilly Media, Inc., 2017.

[17] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016. [Online]. Available: http://queue.acm.org/detail.cfm?id=3022184