

A Novel Spatial Data Pipeline for Orchestrating Apache NiFi/MiNiFi

Chase D. Carthen, University of Nevada, Reno, USA*

 <https://orcid.org/0009-0006-7027-5212>

Araam Zaremehjardi, University of Nevada, Reno, USA

Vinh Le, University of Nevada, Reno, USA

Carlos Cardillo, University of Nevada, Reno, USA

Scotty Strachan, Nevada System of Higher Education, USA

Alireza Tavakkoli and Maketitle, University of Nevada, Reno, USA

Frederick C. Harris Jr., University of Nevada, Reno, USA

 <https://orcid.org/0000-0002-0857-6931>

Sergiu M. Dascalu, University of Nevada, Reno, USA

ABSTRACT

In many smart city projects, a common choice to capture spatial information is the inclusion of lidar data, but this decision will often invoke severe growing pains within the existing infrastructure. In this article, the authors introduce a data pipeline that orchestrates Apache NiFi (NiFi), Apache MiNiFi (MiNiFi), and several other tools as an automated solution to relay and archive lidar data captured by deployed edge devices. The lidar sensors utilized within this workflow are Velodyne Ultra Puck sensors that produce 6-7 GB packet capture (PCAP) files per hour. By both compressing the file after capturing it and compressing the file in real-time; it was discovered that GZIP and XZ both saved considerable file size being from 2-5 GB, 5 minutes in transmission time, and considerable CPU time. To evaluate the capabilities of the system design, the features of this data pipeline were compared against existing third-party services, Globus and RSync.

KEYWORDS

Big Data, Data Pipeline, Data Transfer, Edge Computing, IOT, Lidar, MiNiFi, NiFi, PCAP, Smart City, Spatial Data

INTRODUCTION

As cities begin employing more and more complex sensing devices to either conduct traffic analysis or provide a measure of infrastructure, creating a system for data transferal becomes a crucial challenge. For smart city projects, spatial information such as Light Detection and Ranging (lidar) is especially a concern. Due to the massive amount of data generated by lidar point clouds, data

DOI: 10.4018/IJSI.333164

*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

collection and transferal from edge device to central repository tends to suffer from bottle-necking issues, such as low throughput networking, high latency, and packet-loss. These constraints must be considered as most cities in the United States may have difficulty placing fiber optic infrastructure in their cities (Cooper, 2022).

As part of ongoing smart city developments in the city of Reno, Nevada, the work presented within this paper involves a 100 mbps fiber network provided by the city of Reno. While this network was deployed to specifically address the cyber-infrastructure needs within the city of Reno, this called for the development of a Software Data Pipeline (SDP) that could enable reliable data transformation, transferal, and logging between edge computers and the fog computing network.

In this paper, the authors developed an SDP that leverages NiFi/MiNiFi to facilitate the movement of lidar data generated at various edge computing locations placed around the city of Reno, specifically the Virginia Street corridor. This data is relayed to the fog computing network located at the University of Nevada, Reno (UNR), which is then finally piped towards its destination, UNR's Pronghorn High Performance Computing Cluster, for archival storage. The software on the edge environments use Docker Compose with MiNiFi to hook into the NiFi-based data pipeline in which the lidar point-clouds are compressed and then transmitted off. The software within the UNR Data Center uses Kubernetes to scale up NiFi hosts and receive the lidar point clouds, which are then processed for storage. To ease any confusion, the name "UNR-Virginia SDP" was chosen as the colloquial name to refer to the SDP approach presented in this paper.

The UNR-Virginia SDP does offer some insights for those interested in establishing a scalable pipeline for spatial data collection within smart city infrastructure (Duygan et al., 2022). With the increasing interest in smart city development, the UNR-Virginia SDP provides a template so that other cities with similar network infrastructure may easily incorporate lidar data collection as part of their normal workflow. Due to the versatility of lidar data, lidar collection presents more opportunities for cities to better utilize big data methodologies for effective planning or the establishment of new data-driven solutions (McCrae & Zakhori, 2020; Zhao et al., 2019).

As a form of evaluation for the UNR-Virginia SDP, the authors conducted an analysis of different compression algorithms, compared the discussed approach with the present network bandwidth, and finally performed a feature comparison with major established third-party services, RSync (Davison, 2023) and Globus (Foster et al., 2012). Furthermore, additional metrics gathered from the UNR-Virginia SDP were recorded, such as the bandwidth usage, resource usage on edge devices, and recording time for message transfer. To elaborate, this involved testing different compression methods in terms of resource usage, average CPU usage, average memory usage, total duration time, and size of messages. As part of the feature comparison, RSync and Globus were compared against the UNR-Virginia SDP for basic functionality of data transmission and receiving, load balancing, parallel streaming support, the customization of data flow, and file verification.

The remainder of this paper is structured as follows: the first section presents background information of the technologies explored and used by the UNR-Virginia SDP, the second section describes the design of the UNR-Virginia SDP with considerations and expected requirements of the data pipeline, the third section details the resulting implementation of the planned design and data flow, the fourth section presents the overall performance evaluation of the software data pipeline with benchmarks and comparisons of other methods, and the final section discusses possible uses of the data pipeline and outlines future work to extend its functionality.

BACKGROUND AND RELATED WORKS

Data Pipeline Approaches

With the increasing interest in both cloud and fog computing, different approaches have been explored to facilitate streams of data that require high throughput, intense bandwidth usage, and consistent access across different network scenarios. In general, the very nature of these data streams presents

certain difficulties to system architects when designing the software and may require advanced big data techniques. In response to this, SDPs are often presented as solutions to abstracting data streams by utilizing either custom software or preexisting suites of third-party tools. One such SDP approach differs from the UNR-Virginia SDP presented in this paper and explored a combination of MQTT and Apache Kafka for processing data streams originating from industrial IoT devices. Both Helu et al. (2020) and Raptis et al. (2023) have used MQTT and Apache Kafka in conjunction with NiFi for industrial replated applications. Another similar SDP approach involved the usage of NiFi and MiNiFi to process spatial information captured from Twitter streams while doing some analysis on the acquired data. Kim et al. (2019) designed a custom NiFi processor for extracting spatial data from data sources such as Twitter. Pandya et al. (2019) made use of this approach by extracting data from Twitter with NiFi and MiNiFi and doing sentiment analysis on the acquired data.

NiFi/MiNiFi

A major portion of the UNR-Virginia SDP relies on NiFi and MiNiFi. Together, NiFi and MiNiFi are especially viable for SDPs in that they enable APIs to build data transformers, loggers, and other data flow measures. NiFi is composed of components called “Processors”, and the data passed between these processors, dubbed “Flow Files”, contain the data being transferred and additional metadata pertaining to the transfer. These processors used in a NiFi pipeline allow one to create, remove, modify, or inspect the contents of a flow file. Additionally, NiFi/MiNiFi allows users to create their own processors, which opens the door for greater control and manipulation of the data within a pipeline. NiFi operates with data producers and data consumers, dubbed “Agents”, and alongside the flow files and processors, represent the abstract building blocks to an SDP within NiFi. However, NiFi has the capability of over-bloating a system with tools that are not necessary for remote systems with constrained resources. While NiFi offers the full-suite of tools for building data pipelines, MiNiFi addresses the previous concerns by providing a bare-bones version made to run on resource-constrained edge devices and relay back information to a NiFi-based pipeline. This makes NiFi and MiNiFi ideal to be used for scenarios in which an SDP would be developed to collect spatial information and then process that data within a fog or cloud-based environment.

Data Pipeline With Lidar

As part of the work presented in this paper, NiFi (Apache Software Foundation, 2023) and MiNiFi (Apache Software Foundation, 2023) were used to create an SDP that allows for the collection and storage of lidar data being gathered from sensors installed within the Virginia St. University corridor of Reno, Nevada. These lidar sensors are a series of Velodyne Ultra Pucks, and each can create a 360-degree point cloud of the intersections (Velodyne Lidar Inc., 2023). Each intersection in this space has two lidar sensors installed diagonally northeast and southwest to establish a consistent setup. The Ultra Puck sensors each produce approximately 300,000 points per second, which roughly equates to about 6 GBs of data produced per hour. To account for this big data problem in near real-time, various researchers started using Apache Cassandra and Spark to compute digital terrain maps or other data stream processing frameworks (Deibe et al., 2020; Isah et al., 2019). However, the UNR-Virginia SDP would be more like a system developed by Marosi et al. (2022) where the authors created a data pipeline for handling different types of data flows in varying scenarios with analytics applications. Unlike the method described above, the UNR-Virginia SDP presented in this paper does not perform any computation with the underlying data from the lidar.

Globus

As part of the evaluation and validation section for the work presented in this paper, Globus was chosen as a suitable system to compare the features of the UNR-Virginia SDP against, due to its widespread popularity among big data researchers. Globus is a PaaS (Platform as a Service) created by the University of Chicago, now operating as a non-profit service, used to store and transmit data. The

platform allows developers to use either a software development kit or REST APIs to create Flows within Globus. Flows are the basic-building blocks used to generate SDPs within the Globus system, with Action Providers allowing users to extend the data pipeline. Action Providers open the door for some customization of the basic data pipeline used to transfer files in Globus. Globus provides a wide berth of features that includes not only the ability to send and receive data but also the ability to load balance, transmit data in parallel streams, and validate incoming and/or outgoing files.

RSync

In the same line as Globus, RSync was also chosen as a system for comparison, due to its ease of use and consistent use among researchers working with big data. RSync is a command-line utility that prioritizes performance over usability to transfer files between a source and destination host while offering features for advanced customization of a data pipeline built using the technology. This utility, while offering limited functionality, still can robustly send and receive data and provide a decent measure of file verification. This makes it ideal for creating data pipelines that continuously stream a directory from the source host to the destination host. However, if the needs of a data pipeline were to evolve to include different data processing, logging, or transformation measures, this would entail stringing multiple RSync applications together to create a more customized data pipeline.

Design of the Data Pipeline

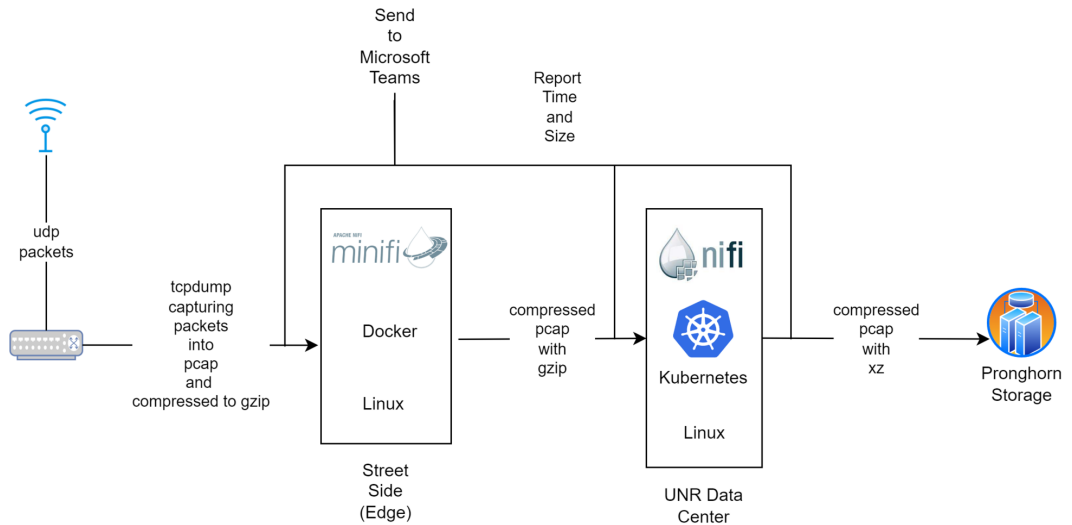
The UNR-Virginia SDP was designed with the consideration that the raw lidar data would be stored into some archive where researchers could use the data for post analysis. To ensure that this process was possible, the software pipeline needed to both be scalable and conserve resources, such as bandwidth, CPU, and storage. In response to this need, the UNR-Virginia SDP consists of several different components running on three different hosts. These three different hosts are: the edge computers at the Virginia streetside, the UNR data center serving as the centralized hub, and the UNR Pronghorn HPC Cluster. Below are some requirements that encompass the design of the data pipeline:

1. The data pipeline shall automate the process of sending lidar data as compressed files with set name schema.
2. The data pipeline shall compress data to minimize the amount of network bandwidth and storage used at the edge and on Pronghorn storage.
3. The data pipeline shall report the amount of time taken to save.
4. The data pipeline shall use compression algorithms to minimize the amount of CPU used at the edge and maximize the amount of compression at the Pronghorn storage.
5. The data shall be archived on the Pronghorn HPC Cluster.
6. The data pipeline shall be scalable and easily orchestrated.

The data pipeline was designed to keep these requirements in mind. A store and forward approach was chosen as the requirements included only archival. The store and forward approach required the authors to consider how much space was available on the device and how much bandwidth was available on the network. In this case, the design was based on a 100 mbps network. This network limited how much data that could be transferred, as it was shared across six different intersections from the edge to the data center, so a major challenge was to figure out ways to cut down on bandwidth by compressing the data. Even with an expansion of network bandwidth capabilities, a key goal was to minimize the data sent and have some form of quality of service in place to allow further applications to also run on the UNR-Virginia SDP.

As shown in Figure 1 there is a high level overview of the UNR-Virginia SDP, which denotes the software stack, alongside an illustration of the flow of data. The reporting of time and the size of data along the pipeline is especially critical and is tracked thoroughly at each step. The spatial data is

Figure 1. High level diagram of the proposed pipeline architecture showing the flow of the PCAP from the edge to Pronghorn



generated by a Velodyne Ultra Puck sensor emitting UDP packets that are then acquired by an edge computer connected to a city streetlight. The UDP packets are captured using tcpdump and stored into a PCAP file. The PCAP file with GZIP at this point is either compressed later or compressed in real-time, then sent from the edge to the data center. At the data center, the compressed PCAP file is uncompressed from a GZIP file to a XZ file and then sent to Pronghorn for archival storage.

At the edge, Docker Compose was used to orchestrate the setup of tcpdump and MiNiFi. Docker Compose was chosen as the setup of this infrastructure because it could be easily replicated on any machine that has Docker Compose installed. Docker Compose allows for rapid modification of the configurations of MiNiFi and tcpdump to try different versions of the software and different configurations without changing the underlying operating system. It also handles the setup of networking between any software that is used.

At the data center, Kubernetes was used to orchestrate the setup of NiFi and all the components that it needs to run in a cluster setup. Kubernetes was chosen because it can setup or scale many distinct types of software across multiple machines with ease. The cluster version of NiFi was selected to allow for the load balancing features to be used and enabled NiFi to scale for other future projects.

Implementation of the Data Pipeline

At the edge, the SPD setup included a DS-1200 embedded machine with Ubuntu 18.04 installed, allocated with a total of 16 GB for RAM and Intel CPU i7-8700T. At the data center, the hardware consisted of an 8-node Kubernetes cluster spread across two four-unit machines. Those machines were a Supermicro SYS-6029TP-HTR and X11DPT-PS. The Kubernetes cluster was set up to handle the workflow of this project and other projects on campus. Additionally, a Supermicro X11DPH-T with 20 TB of allocated storage was put in place to serve as an intermediate storage for the 8-node Kubernetes cluster. This intermediate storage was used as a staging area before sending it off to Pronghorn. All nine of these machines sit in the same rack at the data center located on the UNR campus. Each machine also used Ubuntu 18.04 for the operating system. Rancher RKE 1.0 for the Kubernetes distribution was chosen for the implementation of this project.

NiFi, MiNiFi, and tcpdump were all placed into Docker containers. Both NiFi and MiNiFi were configured with yaml files to set the settings of the software and data pipeline. The tcpdump container

was designed and constructed in such a way that the collection interval and the name of the file can be changed to include descriptive labeling such as: intersection names, what corner the lidar sensor is on, and the municipal location of the sensor. Regarding the collection of data, it was decided to have the tcpdump container collect data every hour and roll it to the next file to collect in another hour.

NiFi was set up as a five-node cluster that is coordinated and orchestrated by ZooKeeper. It was designed this way to scale for the increase of data coming from intersections in the future. Apache NiFi can handle scheduling and load balancing of any data being sent to it from the edge thanks to its Site-To-Site protocol. Configuring the edge is made simpler in that any new intersections or edge installations can be configured to point to this Apache NiFi cluster with a yaml file. Within this configuration the flow can be configured to different input ports with differently named ports that NiFi supports. Also, NiFi in conjunction with other tools can be used to create configurations for MiNiFi like as shown in Figure 2.

Edge to Edge Implementation

At five different intersections that make up the Virginia St. Corridor, the two Ultra Puck sensors collect at a rate of 10 revolutions per second. After being fully collected in an hour a PCAP file is moved into a directory where MiNiFi will place the file into a queue to be sent off to NiFi. When NiFi receives the file from the edge, it will place the file into a storage server within the UNR data center. Inside of NiFi, it will queue any files that land on the storage server and start to process the PCAP file by compressing it into a XZ file. Before it is compressed into XZ, it will check to see if the file has already been compressed with GZIP and if it has been compressed, then it will decompress the file. After the file has been compressed with XZ, it will finally transfer the file off to Pronghorn. During this entire process to keep track of time, NiFi is configured to send completion time statistics

Figure 2. A screenshot of a MiNiFi configuration file

```
Remote Process Groups:
- id: c545c9a6-535d-3f7a-0000-000000000000
  name: ''
  url: http://ncar-da-1.rc.unr.edu:30001/nifi
  comment: ''
  timeout: 30 sec
  yield period: 10 sec
  transport protocol: RAW
  proxy host: ''
  proxy port: ''
  proxy user: ''
  proxy password: ''
  local network interface: ''
Input Ports:
- id: 6e0d39e2-d461-1789-0000-000071bc1644
  name: From Pcap
  comment: ''
  max concurrent tasks: 1
  use compression: false
```

to a Microsoft Teams channel setup between the Edge and NiFi as well as between the storage server at the UNR Data Center and Pronghorn.

As shown in Figure 3 demonstrates a sequence diagram of the flow of lidar data in the data pipeline from the edge to the archive in sequential steps. It further elucidates the details of the data flow from Figure 1. In this figure, the size and timing information are logged to Microsoft Teams exactly two times. The first time represents how long it took for the PCAP to reach the data center along with any overhead from NiFi. The second time represents how long it took for the file to be compressed to XZ and sent off to the archive. The boxes at the top of Figure 3 indicate where the data processing is occurring.. For instance, the transformation of the PCAP from GZIP to XZ is shown below the Data Center.

Figure 4 shows a screenshot of NiFi, a part of this paper's workflow. A user of NiFi can specify their workflow in the user interface. NiFi was chosen due to the ease of use and being able to specify data flows with a GUI. This screenshot demonstrates where NiFi grabs data from MiNiFi and stores it onto a storage chassis in the UNR data center, while statistics about the whole process are sent off to Microsoft Teams.

The approach adopted for the UNR-Virginia SDP regarding the transferring of raw lidar data utilizes NiFi's and MiNiFi's Site-to-Site protocol, which in turn makes use of the RAW transport

Figure 3. A sequence diagram showing the flow of the Lidar data from the edge to the archive

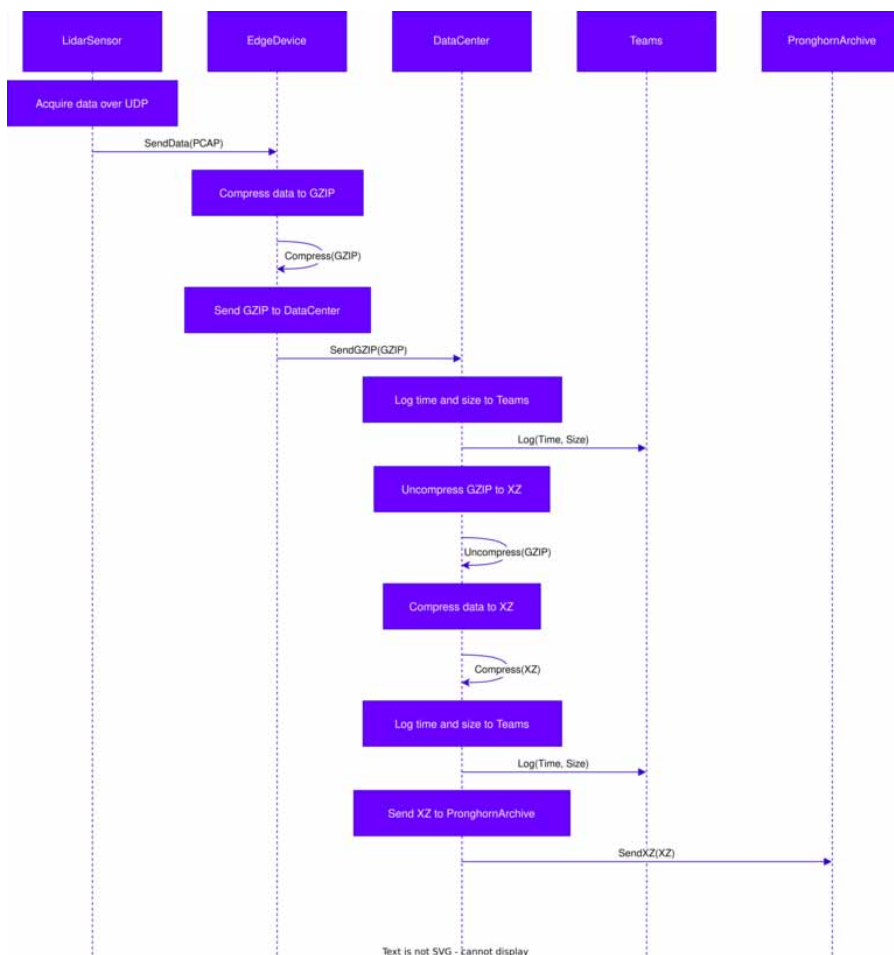
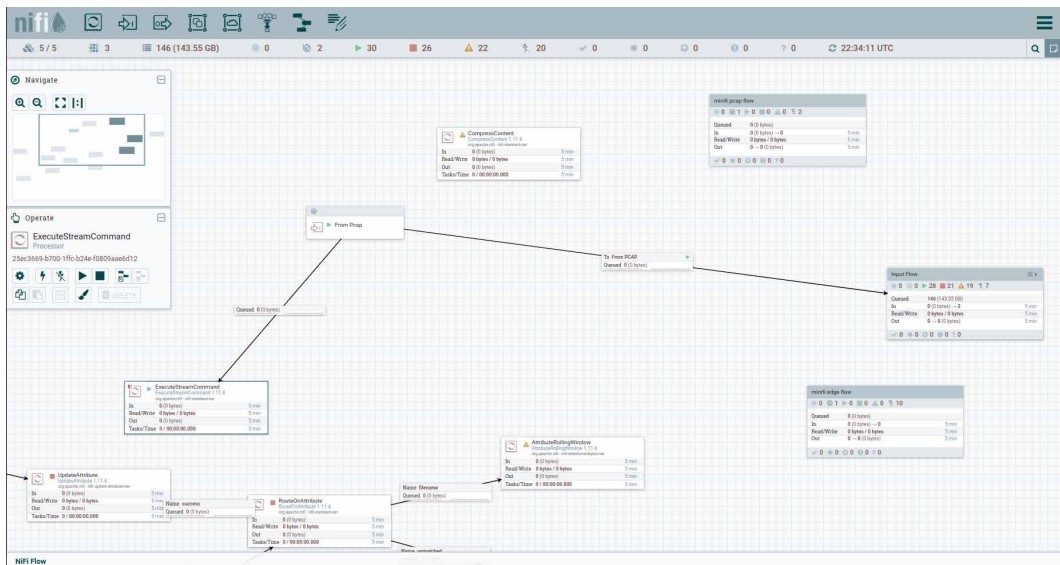


Figure 4. Screenshot of NiFi showing a portion of the data flow from the edge to the data center



protocol setting. NiFi's Site-to-Site Protocol allows for any number of MiNiFi instances to connect to the clustered instance of NiFi. This allows for MiNiFi to be scaled up for multiple different intersections. The MiNiFi flow was designed within NiFi and exported from it for the deployment at the edge.

Choice of Compression Algorithm at the Edge

This PCAP file was compressed in two diverse ways at the edge. The first method involved compressing the file after it had been fully captured. The second method was compressing the file in real-time. The first method is the easiest to implement and provides a good comparison for the second method. The second method was implemented because it saves the time of compressing the file afterward, but at the cost of extra computation due to compressing while recording. The lz4, bzip, XZ, GZIP, and zstd compression algorithms were tested. GZIP was picked as the compression algorithm between the edge and data center communication. For archival purposes, XZ was picked as the compression algorithm at Pronghorn.

Metrics Implementation

Microsoft Teams is used for metrics collection and notifications for the software data pipeline in a team's channel. Teams provide a webhooks interface that allows external applications to push messages that can be posted to the team's channel. The webhook is configured to accept an incoming HTTP POST request made from the data center using the Apache NiFi pipeline. The pipeline uses metadata (called FlowFiles) attached to the transmitted PCAP file to gather the following metrics about the incoming file from the edge: the filename, the file size, and the transmission time. The transmission time begins when a PCAP arrives at the data center to-be converted from GZIP compression format to a XZ compression format and ends after the PCAP is sent to Pronghorn. A preprocessor takes the metadata within the FlowFile and creates a formatted string that is sent via a HTTP POST message to a team's incoming webhook endpoint. Once the endpoint receives the message, it is then posted to a team's channel used by members of the university to monitor the data pipeline.

RESULTS AND DISCUSSION

A subset of results was collected from Microsoft Teams and used to find out how long the overall transfer time takes. It was found that NiFi and MiNiFi both send information at the maximum network bandwidth supported by the 100 mbps network. The bandwidth could not exceed 100 mbps as the information from all intersections flows through a single interconnect switch back into campus. Table 1 shows that the file size of the compression shaves off about 2 GB and saves about 3-4 minutes of transmission time. This paper's results show that NiFi adds about 10-12 seconds on transfer time overhead while under normal conditions. This test was conducted to ensure that the implementation of this software does indeed use the file network with no large bottleneck on the part of the software itself.

Both methods of compression from the third section were compared and measured using the ps command from Linux. Both methods were run on two different PCAP files that were recorded for one hour. Table 2 demonstrates the results for the first method where a file is compressed after being collected from lidar sensors.

Table 3 demonstrates the results for the other method where the file is compressed as it is collected from the lidar sensor. In Table 3, the duration within the table represents the amount of

Table 1. The bandwidth usage of original and compressed data

Method	Size	Bandwidth	Time
Original	7 GB	11 MB/sec	10-12 min.
Original With Compression	5 GB	11 MB/sec	7-8 min.

Table 2. Non-real-time comparison

Compress Method	Average CPU %	Average % Mem.	Total Duration (hh:mm:ss)	Size (GB)	Compression Ratio
lz4	96.12	0.097	0:30.0	3.9	0.19
bzip2	99.49	0	14:32.0	3.2	0.33
XZ	99.94	0.010	32:53.0	1.9	0.6
GZIP	99.25	0	03:56.0	3.3	0.31
LZMA	99.92	0.010	32:28.0	1.9	0.6
Zstd	98.84	0	01:09.0	3.4	0.29
Original	-	-	-	4.8	-

Table 3. Real-time comparison

Compress Method	Average CPU %	Average % Mem.	Total Duration (hh:mm:ss)	Size (GB)	Compression Ratio
lz4	0	0	1:06.0	5.2	0.19
bzip	27.14	0	20:01.0	3.4	0.47
XZ	0.010	0.010	38:01.0	2.5	0.61
GZIP	0.0006	0	07:40.0	3.9	0.39
LZMA	9.86	0	38:31.0	2.5	0.61
zstd	1.94	0	03:49.0	4.4	0.31
Original	-	-	-	6.4	-

time the program ran on the CPU, while the PCAP file was captured for one hour as reported by the ps command. All compression algorithms included in this experiment were used with their lowest and fastest setting.

Comparing the two tables, both XZ and LZMA have the best compression ratio but take the most time in comparison to the other compression algorithms. Examining the second method in comparison to the first method, the second method doesn't nearly use as much CPU when a file is being recorded. Looking at GZIP and lz4 for the second method, both compress the original file to 3.9 GB and 5.2 GB from 6.4 GB. Out of these two compression algorithms GZIP compresses better and only at a slightly higher CPU usage. Based on these results, GZIP was chosen to be the compression algorithm for the edge to data center communication. XZ was chosen to be the compression algorithm due to its high compression ratio for storage on Pronghorn.

Table 4 shows a feature comparison across Globus, RSync, and the UNR-Virginia SDP with NiFi and MiNiFi. Globus makes use of GridFTP to send data from one data source to another. Globus is typically used for larger files and supports parallel network streams when sending files. This allows Globus to send files much faster in comparison to RSync and UNR-Virginia. All three approaches support performing file verification or some form of check summing, but the UNR-Virginia SDP would have to specifically implement it within the data flow of NiFi and MiNiFi. Globus allows universities to scale up their end points and set up data transfers between two different data sources. NiFi can be scaled up due to being able to be clustered with the help of Zookeeper. RSync only supports a straight end-to-end connection from one host to another host and does not perform load balancing or scalability. The best that could be used with RSync is starting up multiple instances of RSync.

Neither RSync nor Globus can support custom data flows like NiFi or MiNiFi. As explained before, NiFi allows for the users to send their data to many diverse types of options like a database, another NiFi, a web service, and many others. This flexibility allows for the UNR-Virginia SDP to potentially send the data to other sources, for instance to cold storage or other collaborators who want a live copy of the data. Also, this approach allows for further analytics to be made as data moves through NiFi. Both Globus and RSync are good for sending data from one source to another and have some analytics, but both lack the flexibility that NiFi gives with creating data flows inside a user interface.

Globus provides data flow capabilities, but it is best used for point-to-point transfers involving large files. NiFi has the functionality to transform the underlying data as it flows from point-to-point. These transformations can be anything like adding extra metadata, compressing, decompressing, appending on new data from other sources, and other capabilities. Globus does not provide these transformation capabilities, but it does provide overall more performance in the case of transferring a file. NiFi was chosen for its data transformation capabilities over Globus's data transfer performance.

However, NiFi may take some time to set up and lacks some of the ease of use of Globus's user interface to send files within their platform. RSync is readily available on Linux and can be utilized by installing it as a package. While NiFi may take some setup effort and require some specific configuration, the ability to alter the data flow with a user interface makes it easier to visualize the flow of the data.

Table 4. A feature comparison of this paper's approach, the UNR-Virginia SDP, vs. Globus and RSync

Features	Globus	RSync	UNR-Virginia
Send/Receive Data	X	X	X
Load Balancing	X		X
Send Data in Parallel Streams	X		
Customizable Data Flow			X
File Verification	X	X	X

CONCLUSION AND FUTURE WORK

In this paper, it was found that using NiFi and MiNiFi produced a promising solution for transferring lidar data. Additionally, XZ was found to be the best compression algorithm for archiving onto the UNR's HPC cluster due to its high compression ratio. Furthermore, compressing the PCAP file in real-time used only a minimal amount of CPU power in comparison to recording the PCAP file and then compressing it afterward. This compression technique reduced the transmission time by about 5 minutes and saved almost 2 GB in file size. Through the feature comparison, it was discovered that this paper's approach, UNR-Virginia SDP, covers a significant breadth of service among similar systems but still lacks certain advanced features, such as enabling parallel streams. Additionally, under this evaluation, it was found that the UNR-Virginia SDP had greater flexibility and ease of use due to the user interface provided by NiFi.

As part of the future work, the author's plan is to expand this method by exploring new avenues, such as sending the PCAP to a web service to convert the raw data within the PCAP into point cloud format. This would then be stored in a database with NiFi. NiFi allows for different quality control (QC) and quality assurance (QA) implementations to be tested. These implementations could be applied, for example, when the file comes from the edge to the data center or when the file is placed into the archive at the data center. Finally, the approach presented in this paper could be potentially adapted with additional types of devices along the Virginia St. Corridor, such as video cameras or various time-series sensors.

ACKNOWLEDGMENT

The authors would like to acknowledge the support of Research & Innovation and the Cyberinfrastructure Team in the Office of Information Technology at the University of Nevada, Reno for facilitation and access to the Pronghorn High-Performance Computing Cluster. The authors would like to acknowledge the City of Reno. This material is based in part upon work supported by Washoe County Regional Transportation Commission (RTC) [grant number AWD- 01-00002406]. It is also based in part upon work supported by the National Science Foundation [grant number OAC-2209806, OIA-2019609, and OIA-2148788]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Washoe County RTC or The National Science Foundation.

REFERENCES

- Apache Software Foundation. (2023). *Apache MiNiFi*. Apache Software Foundation. <https://nifi.apache.org/minifi/>
- Apache Software Foundation. (2023). *Apache NiFi*. Apache Software Foundation. <https://nifi.apache.org/>
- Cooper, T. (2022). Municipal broadband 2022: Barriers remain an issue in 17 states. *Broadband Now*. <https://broadbandnow.com/report/municipal-broadband-roadblocks/>
- Davison, W. (2023). *RSync*. Samba. <https://rsync.samba.org/>
- Deibe, D., Amor, M., & Doallo, R. (2020). Big data geospatial processing for massive aerial lidar datasets. *Remote Sensing (Basel)*, 12(4), 719. doi:10.3390/rs12040719
- Duygan, M., Fischer, M., Pärli, R., & Ingold, K. (2022). Where do smart cities grow? The spatial and socio-economic configurations of smart city development. *Sustainable Cities and Society*, 77, 103578. doi:10.1016/j.scs.2021.103578
- Foster, I., Kettimuthu, R., Martin, S., Tuecke, S., Milroy, D., Palen, B., Hauser, T., & Braden, J. (2012). Campus bridging made easy via Globus services. In *Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond. XSEDE12: 2012 eXtreme Science and Engineering Discovery Environment 2012*. ACM. doi:10.1145/2335755.2335847
- Helu, M., Sprock, T., Hartenstine, D., Venkatesh, R., & Sobel, W. (2020). Scalable data pipeline architecture to support the industrial internet of things. *CIRP Annals*, 69(1), 385–388. doi:10.1016/j.cirp.2020.04.006
- Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., & Khan, S. (2019). A survey of distributed data stream processing frameworks. *IEEE Access : Practical Innovations, Open Solutions*, 7, 154300–154316. doi:10.1109/ACCESS.2019.2946884
- Kim, S.-S., Lee, W.-R., & Go, J.-H. (2019). A Study on Utilization of Spatial Information in Heterogeneous System Based on Apache NiFi. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. doi:10.1109/ICTC46691.2019.8939734
- Marosi, A. C., Emödi, M., Farkas, A., Lovas, R., Beregi, R., Pedone, G., Németh, B., & Gáspár, P. (2022). Toward reference architectures: A cloud-agnostic data analytics platform empowering autonomous systems. *IEEE Access : Practical Innovations, Open Solutions*, 10, 60658–60673. doi:10.1109/ACCESS.2022.3180365
- McCrae, S., & Zakhor, A. (2020). 3D object detection for autonomous driving using temporal lidar data. *2020 IEEE International Conference on Image Processing (ICIP)*, (pp. 2661-2665). IEEE. doi:10.1109/ICIP40778.2020.9191134
- Microsoft Corporation. (2023). Microsoft Teams – Group chat software. *Microsoft*. <https://www.microsoft.com/en-us/microsoft-teams/group-chat-software>
- Pandya, A., Kostakos, P., Mehmood, H., Cortes, M., Gilman, E., Oussalah, M., & Pirttikangas, S. (2019). Privacy preserving sentiment analysis on multiple edge data streams with Apache NiFi. In *2019 European Intelligence and Security Informatics Conference (EISIC)*. IEEE. doi:10.1109/EISIC49498.2019.9108851
- Raptis, T. P., Cicconetti, C., Falelakis, M., Kalogiannis, G., Kanellos, T., & Lobo, T. P. (2023). Engineering resource-efficient data management for smart cities with Apache Kafka. *Future Internet*, 15(2), 43. doi:10.3390/fi15020043
- Velodyne Lidar, Inc. (2023). Ultra puck surround view lidar sensor. *Velodyne Lidar, Inc.* <https://velodynelidar.com/products/ultra-puck/>
- Zhao, J., Xu, H., Liu, H., Wu, J., Zheng, Y., & Wu, D. (2019). Detection and tracking of pedestrians and vehicles using roadside lidar sensors. *Transportation Research Part C, Emerging Technologies*, 100, 68–87. doi:10.1016/j.trc.2019.01.007

Chase Carthen is currently both a PhD Student in Computer Science, and an Administrative Faculty for the Office of Information Technology at the University of Nevada, Reno. Professionally, Chase develops data-intensive pipelines for various NSF-funded research projects and has previously worked in industry developing full stack applications as a full stack developer. Academically, Chase focuses on the areas of Cyberinfrastructure, Machine Learning, and Data Engineering. Chase aspires to one day become a tenure-track professor at a research institution. In his personal life, Chase enjoys playing Dungeons and Dragons, spending time with his family, learning more about machine learning approaches, and riding his motorcycle around Lake Tahoe.

Araam Zaremehrijardi is a master's graduate student in the Department of Computer Science and Engineering at the University of Nevada, Reno and is under the advisement of Dr. Sergiu Dascalu. He is expected to earn his master's degree in May 2025. He completed his B.S. in Computer Science and Engineering with a Minor in Mathematics at the University of Nevada, Reno in 2023. He is a member of the Software Systems Laboratory and the Nevada Center of Applied Research. His academic interests encompass Human-Computer Interaction and Software Data Pipelines, alongside a focus on Smart Cities and Smart Infrastructure, particularly within transportation systems.

Vinh Le is a PhD student under the advisement of Dr. Sergiu Dascalu and Dr. Frederick Harris, Jr. at the University of Nevada, Reno (UNR). His interests lie primarily in Human-Computer Interaction, Software Systems, and Web Development. Vinh defended his Master's Thesis on the topic of Microservice Architecture for Envirosensing Projects. He has also worked professionally as a Senior Developer working on cyberinfrastructure in state licensing for over 20 states, including Nevada, Texas, Louisiana, and California. Currently, Vinh is one of three Capstone Instructors for the Computer Science Department at UNR. Vinh has great passion for teaching and research, and aspires to one day be a professor at a research university.

Carlos Cardillo is the Executive Director of the Nevada Center for Applied Research at the University of Nevada, Reno. He received his AS in Computer Programming and BSc in Systems Engineering from the National University of the Northeast, Argentina (1988 and 1990); MSc in Biostatistics from New York Medical College (2001) and a PhD in Health Science Research from Trident University International, CA in 2012. He is Project Manager for the Intelligent Mobility initiative, a User-Centered, Open-Innovation Living-Lab Ecosystem for Automated and Connected Vehicles in Nevada and Director of Nevada Autonomous, the FAA-Designated UAS Test Site. As a PI/Co-PI, he developed, implemented, and successfully executed large research programs sponsored by the US Department of Transportation (DOT) and the US Department of Defense (DOD), including projects with Special Operation Command (SOCOM) the Warfighter Enhancement Program Office and Human Performance Training and Education program sponsored by the Office of Naval Research (ONR).

Scotty Strachan serves as Principal Research Engineer for the Nevada System of Higher Education, the state's university and community college system. He specializes in IT strategy and R&D for research-driven STEM education. Scotty received his Ph.D. in Geography from the University of Nevada, Reno in 2016 while developing and deploying globally-unique hydroclimate monitoring infrastructure in the intermountain western United States. He is currently the cyberinfrastructure lead and co-PI of a \$20MM NSF fire science project in Nevada, and lead for the Research Engineering group at NevadaNet, the state's primary WAN provider for education and government internet connectivity and long-distance transport. Scotty is also a Research Affiliate with the Department of Energy ESNet at Lawrence Berkeley National Laboratory. He actively collaborates with research technology leaders across the country to develop cyberinfrastructure best practices, author national guidance reports to funding agencies, and create interdisciplinary engineering teams to solve emerging science problems.

Alireza Tavakkoli is an Associate Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno. He received his BSc and MSc degrees in Electrical Engineering from the Sharif University of Technology in 2001 and 2004, and MSC and PhD degrees in Computer Science from the University of Nevada, Reno in 2006 and 2009. He is the Director of the Human Machine Perception Lab at UNR. His main interests are in visual computing, artificial intelligence, and perception. His research projects are funded by federal agencies such as NSF, NASA, NIH and DoD. He has published over 100 peer-reviewed articles and occasionally serves as a grant review panelist as well as a reviewer for several journals and conferences. He is a senior member of the IEEE and currently the chief guest editor of a special research topic in the journal *frontiers in ophthalmology*.

Frederick C. Harris, Jr. is a Foundation Professor in the Department of Computer Science and Engineering and Associate Dean in the College of Engineering at the University of Nevada, Reno. He received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, and went on and received his MS and Ph.D. degrees in Computer Science from Clemson University. He is co-Director of the Software Systems Lab at UNR. He is also the Nevada State EPSCoR Director and the Project Director for Nevada NSF EPSCoR. He has published more than 300 peer-reviewed journal and conference papers along with several book chapters and has edited or co-edited 14 books. He has had 14 PhD students and 81 MS Thesis students finish under his supervision. His research interests are in parallel computation, simulation, computer graphics, and virtual reality. He is a Senior Member of the ACM.

Sergiu Dascalu is a Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno. He received a Master's degree in Automatic Control and Computers from the Polytechnic of Bucharest, Romania (1982) and a PhD in Computer Science from Dalhousie University, Canada (2001). He is the co-Director of the Software Systems Lab at UNR. His main interests are in software engineering, human-computer interaction, and data science. He has worked on many research projects funded by federal agencies such as NSF, NASA, and DoD-ONR. He has published over 250 peer-reviewed papers. Also, he received several awards, including the 2011 UNR F. Donald Tibbitts Distinguished Teacher of the Year Award and the 2019 UNR Vada Trimble Outstanding Graduate Mentor Award. He has been a panelist for several NSF program solicitations as well as reviewer for over 15 journals. He is a senior member of the ACM.