AN INTELLIGENT PARALLEL DISTRIBUTED STREAMING FRAMEWORK(IPDSF) FOR NEAR REAL-TIME SCIENCE SENSORS AND HIGH-RESOLUTION MEDICAL IMAGES

Welcome to DUAC 2021 Workshop

Co-located with ICPP 2021: https://www.icpp-conf.org

Acknowledgement

This project is funded by NASA grant number NNH16ZDA001N-AIST16-0091

Brightest minds of University of Maryland Baltimore County are investing on this project.

IPDSF Team

Samit Shivadekar
PhD candidate in Computer Science at UMBC

Jayalakshmi Mangalagiri
PhD candidate in Computer Science at UMBC

Rahul Gite
Master's in Data Science
Graduate Research Assistant at UMBC

Dr. Milton Halem
Research Professor at UMBC

Dr. Phuong Nguyen
Research Assistant Professor at UMBC

Introduction

An Intelligent Parallel Distributed Streaming Framework(IPDSF)

- ✓ To explore the potential of stream processing with data parallelism methodologies and to provide the most effective data processing architecture

- ✓ Address challenges such as latency, scalability, throughput and heterogeneous data sources of streaming analytics and deep learning pipelines in science sensors and medical imaging applications

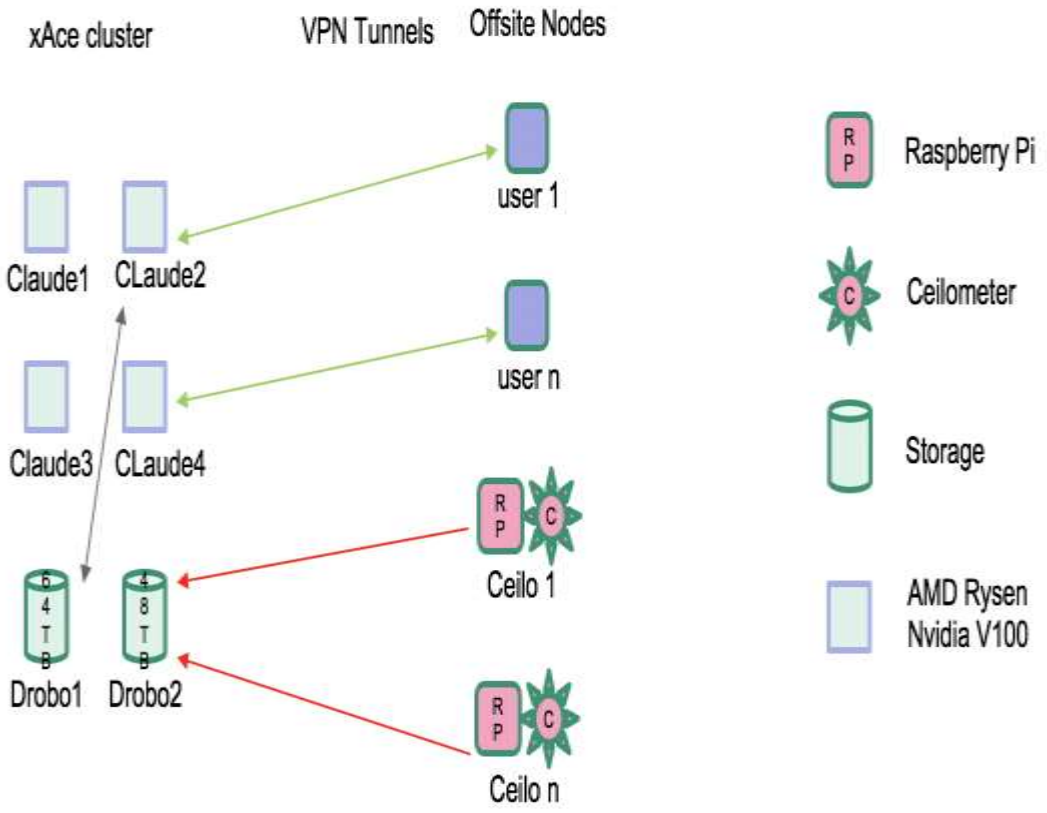
- ✓ Designed to run streaming Artificial Intelligent (AI) analytic tasks using data parallelism including partitions of multiple streams of short time sensing data and high-resolution 3D medical images, and fine grain tasks distribution.

IPDSF HARDWARE ARCHITECTURE

Physically deployed using the University of Maryland Baltimore County (UMBC) Advanced Information System Technology (AIST) compute cluster consisting of the following nodes

- ✓ 2 dual 32-Core AMD CPUs with 32GB RAM, CPUs.
- ✓ 4 Nvidia GPU Geforce 2080 Ti GPUs with 11GB GPU memory and high bandwidth connections.
- ✓ additional AMD CPUs and 2 Nvidia A100 GPUs servers.
- ✓ cluster contains a DROBO storage array consisting of 64 TB of high-speed disks for data archiving.
- ✓ additional IBM Minsky Power8 with 4 GPU servers.
- ✓ Access to the cluster of GPU Servers is maintained under a VPN firewall.

AEROSOL CYBERINFRASTRUCTURE ECOSYSTEM (XACE)



Intra-net Security

- VPN access security (user unique certificate & password)
- Node security (VPN access & user unique password)
- Connections in are secure
- Connections out are open

Once connected to Intra-net:

- Access from any machine to any other machine with valid user account
  - User workstation (laptop, desktop)
  - Compute nodes
  - Instrument node

Instrument node (~ $50 Raspberry Pi)

- Local data backup from instrument (up to 3 yrs)
- Periodically passes data on to xAce cluster database
- Can send data to other offsite nodes/organizations (future)

Tools and Technologies

KAFKA – DATA STREAMING

PYTHON – DATA PROCESSING

TENSORFLOW – ARTIFICIAL INTELLIGENCE SYSTEM

HOROVOD - DISTRIBUTED DEEP LEARNING TRAINING

CONTRIBUTION

Designing IPDSF to use data and task parallelism methodologies including partitions of multiple streams of short time sensing data and fine grain task partitions of high-resolution medical images using cGAN model

Implementation and validation IPDSF for two real world applications using distributed nodes

IPDSF applications scale to process thousands of near real time streaming science sensors and generate full size realistic synthetic high resolution 3D CT images

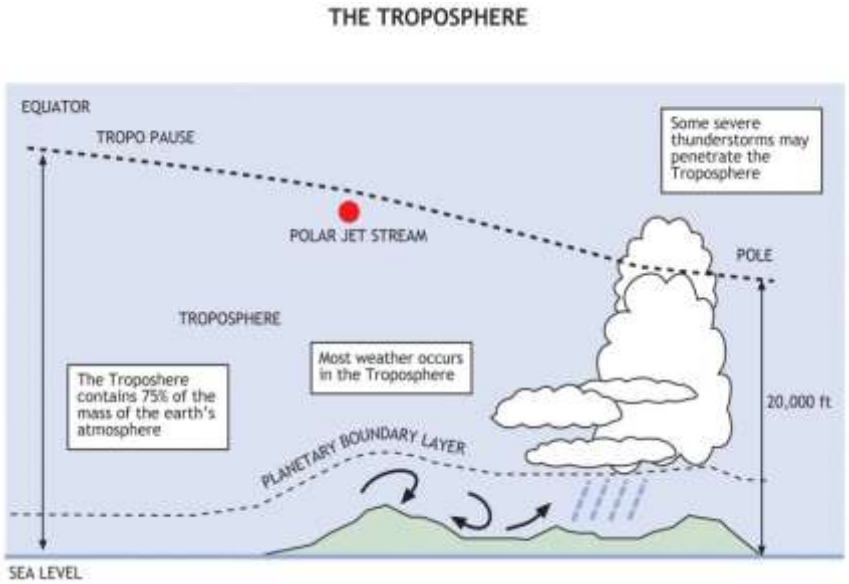
CASE STUDIES

Air quality Index

➢1) Ceilometer measures the Air quality and the layer of air close to the earth's surface is known as the planetary boundary layer (PBL)

➢ 2) Thousand CEIL (Ceilometer instrument) sites operating distributed over the US by different organizations.

➢3) We use IPDSF to show how it scales to process thousands of CEIL streams at the same time



High Resolution Medical Imaging

1) Thousands of Covid-19 patient's Computed Tomography CT scans are collected

2) The high-resolution CT scans have resolution from 100 to 600 images of 512x512 pixels up to 1024x1024 pixels. The goals of this application are to generate high resolution synthesis images from those CT scans for public usage employing Deep Learning application to detect, track, and study Covid-19 infection

3) It is impractical to train GANs for generating high resolution 3D scans 512x512x600 pixels using current common GPU memory of 10GB-32GB. In addition, training GANs models take several days using a single node of GPUs.

4)Using IPDSF for data partitioning, parallel fine grain tasks and training the GANs model in parallel distributed fault tolerant manner

Overview Of IPDSF

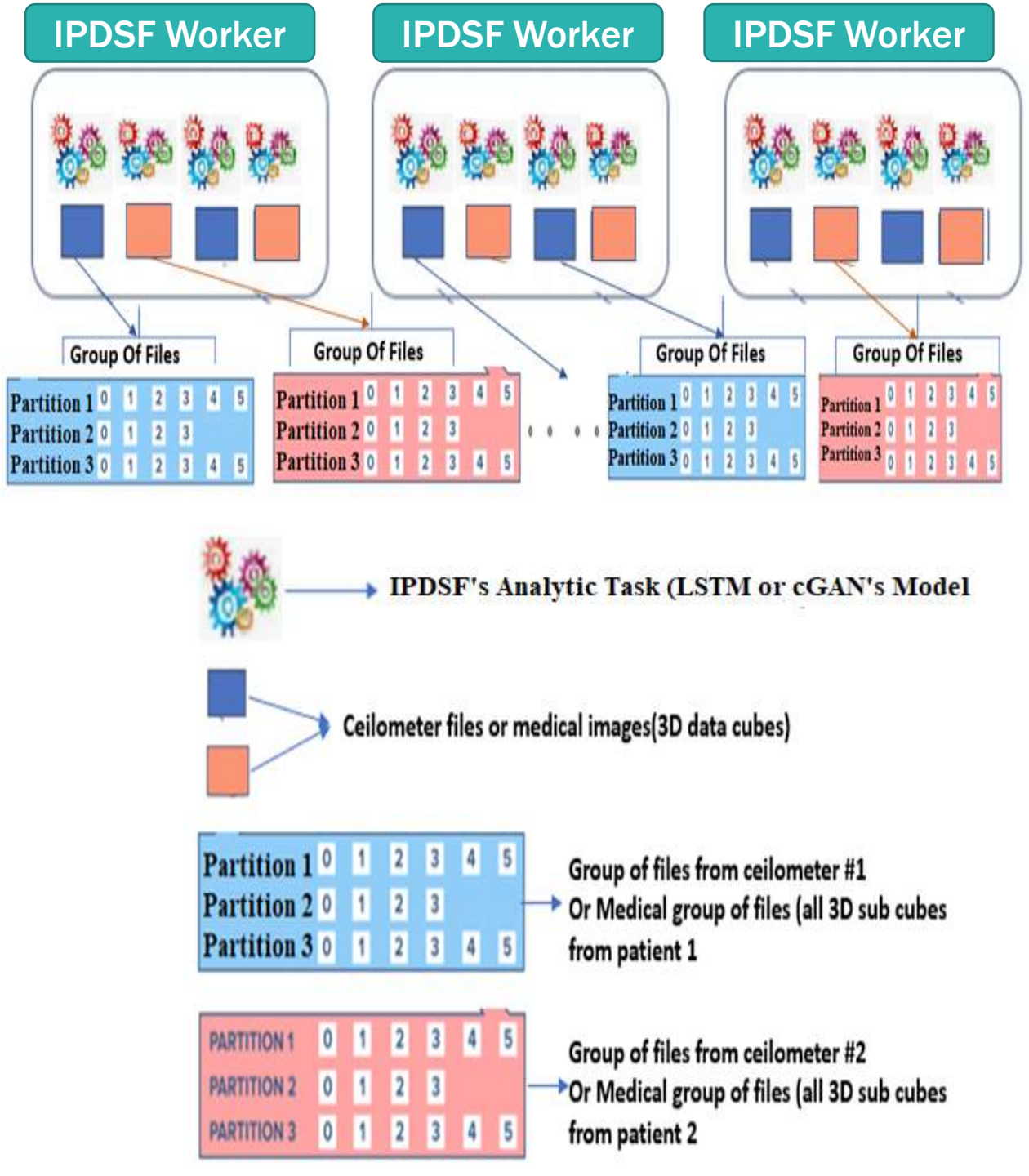
✓ Data Parallelism

$$\Theta \leftarrow F\left(\Theta, \Delta_L\left(\Theta, X_{p1}\right), \ldots, \Delta_L\left(\Theta, X_{pM}\right)\right) \quad (1)$$

where the AI analytic tasks ΔL is replicated across M number of IPDSF workers, and multiple data partitions $\{X_{p1}\}_{m=1}^{M}$ are processed in parallel on each worker

Data parallelism supports synchronization to collect the updates generated at each worker ΔL(Θ,Xpm), then applies them together to aggregate (F function) the model parameter (Θ) during training AI analytic tasks in distributed parallel mode.

✓ AI analytic tasks using the partitions are time series analytic science sensors data or medical imaging analytics using deep learning models (Long Short-Term Memory neural network, edge detection, GAN etc..).



IPDSF Validation and Performance Metrics

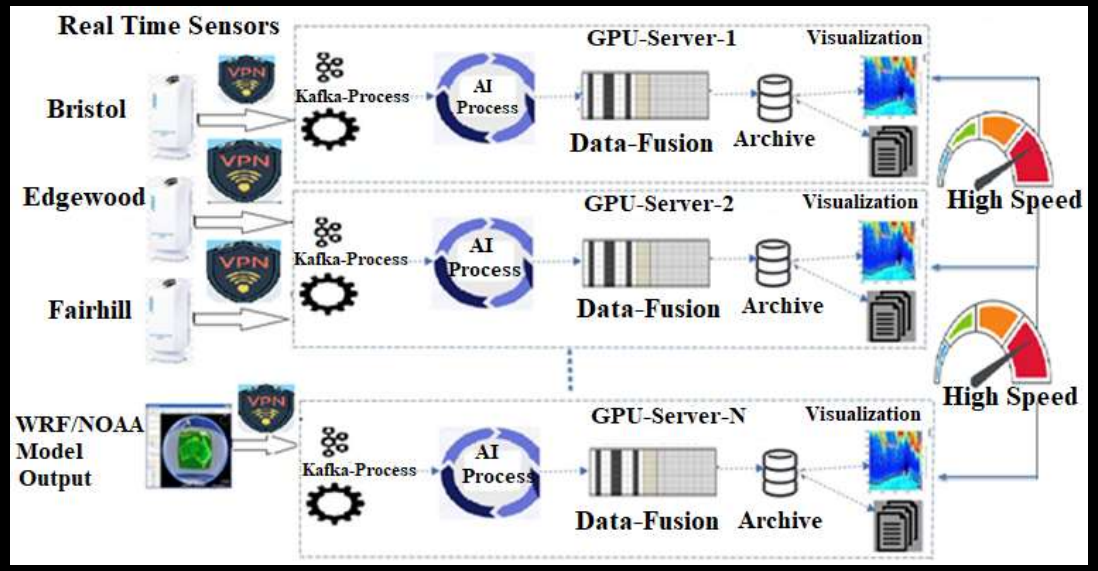
System throughput/latency

- measure the number of streams (files)
- how long it takes for IPDSF to process simultaneously multiple streams

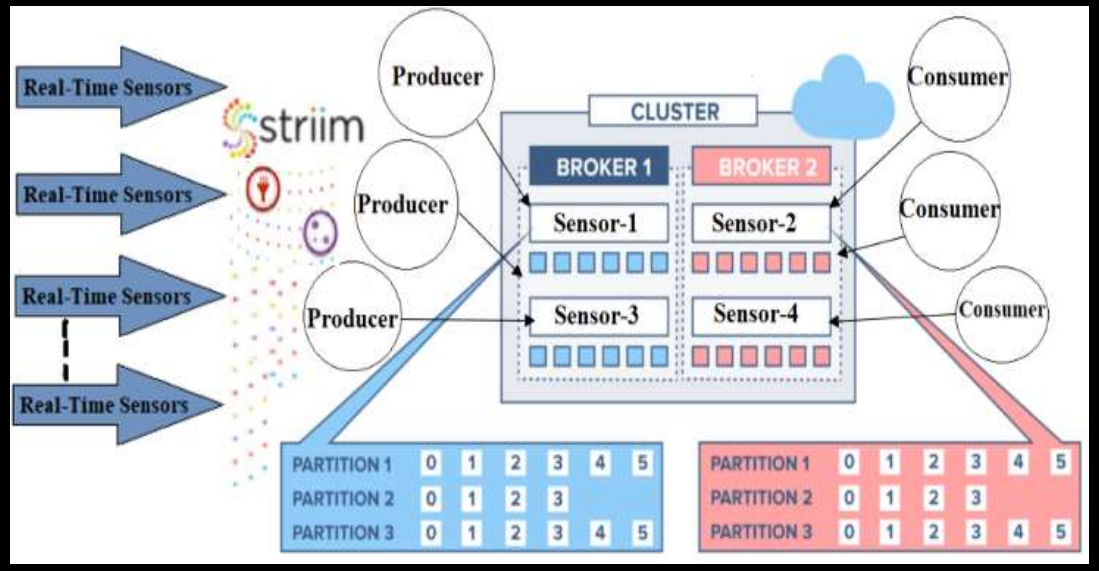
Performance Metrics

- ✓ Peak Signal to Noise Ratio (PSNR) which is designed to get approximations of human perception of the image reconstruction quality which usually varies from 30-50 dB for a 8-bit data representation and 60-80 dB for a 16-bit data representation
- ✓ Structural Similarity Index (SSIM) that is another metric to estimate the quality of images by measuring the similarity between original and the reconstructed image. SSIM varies from -1 to +1 where +1 indicates that the two images are identical

IPDSF system architecture applied for science sensor application



IPDSF distributed data streaming processing for science sensor application

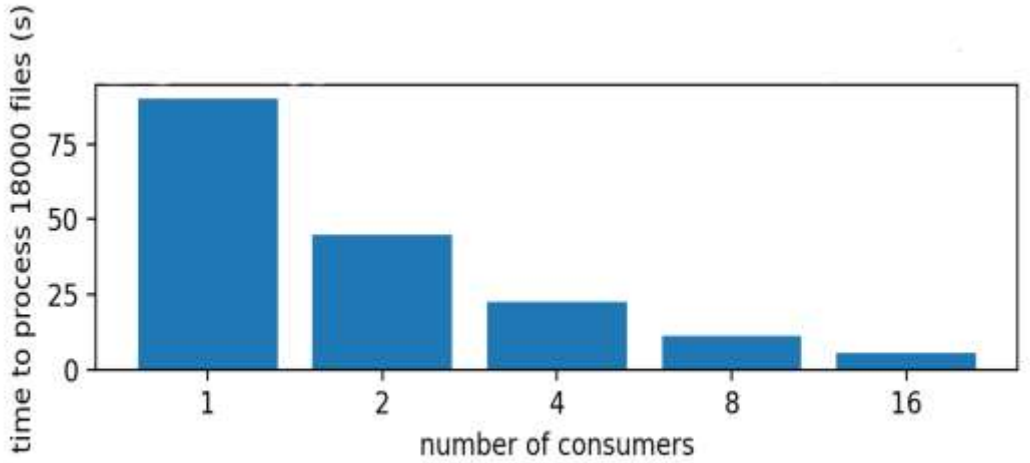


- ✓ The multiple science sensors (Ceilometers) are connected to lowcost computers (i.e., Raspberry PIs) with allocated VPN addresses for network access to the CPU/GPU server nodes for streaming the data into the Servers.
- ✓ Each ceilometer produces backscatter profile data every 15 seconds, stores them, and then streams the backscatter profile data files to the IPDSF system every 1 to 5 minutes

- ✓ IPDSF is implemented using the Apache Kafka streaming library
- ✓ Kafka producers write files to stream's partitions and consumers are set up to run IPDSF's analytical tasks which read from partitions. We have used 50, 100, and 150 partitions per Ceilometer's stream
- ✓ Producers send data to partitions in a round-robin fashion, starting at the first partition and iterating through to the last in a loop. Consumers subscribe to all available streams, automatically being assigned stream partitions by the Kafka infrastructure.

Experiments - Latency Performance

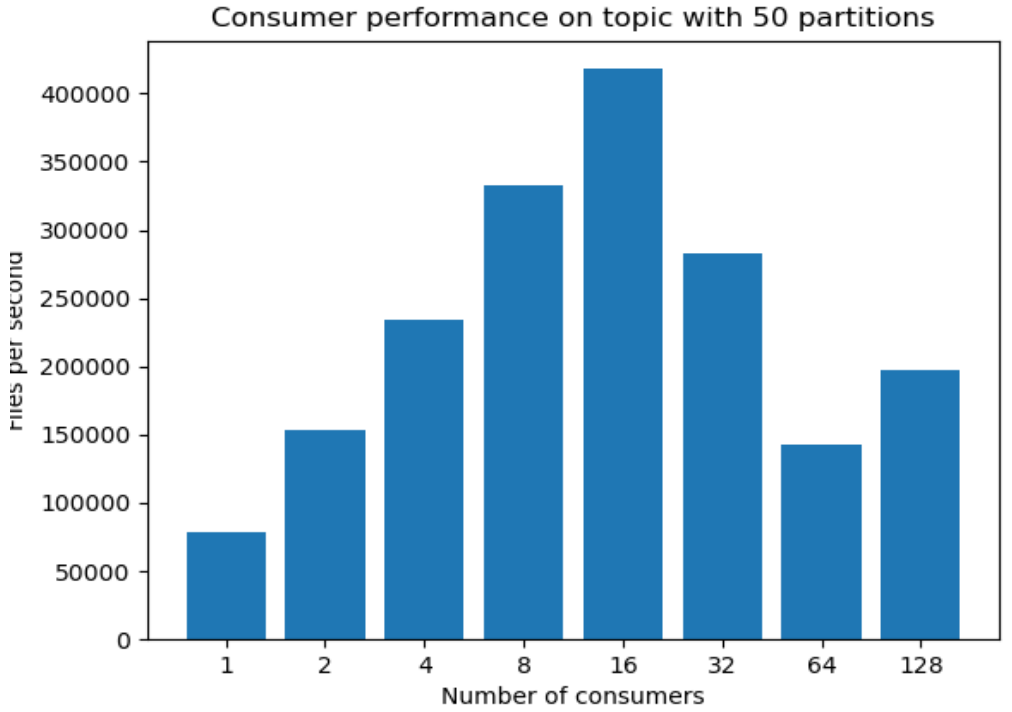
- The latency performance is recorded to measure the total time to process all streams of data.
- Horizontal axis shows the number of Consumers and the vertical axis shows time to process 18000 files.
- The fixed rate of 200 files per second are sent to the system.
- The results show the total amount of time to process all the streams is almost linearly reduced as the number of Consumers are increased using a single node.



Time to process number of files vs number of consumers using a single node (predicted speed for 18000 files at 200.0 files per second).

Experiment - Throughput Performance

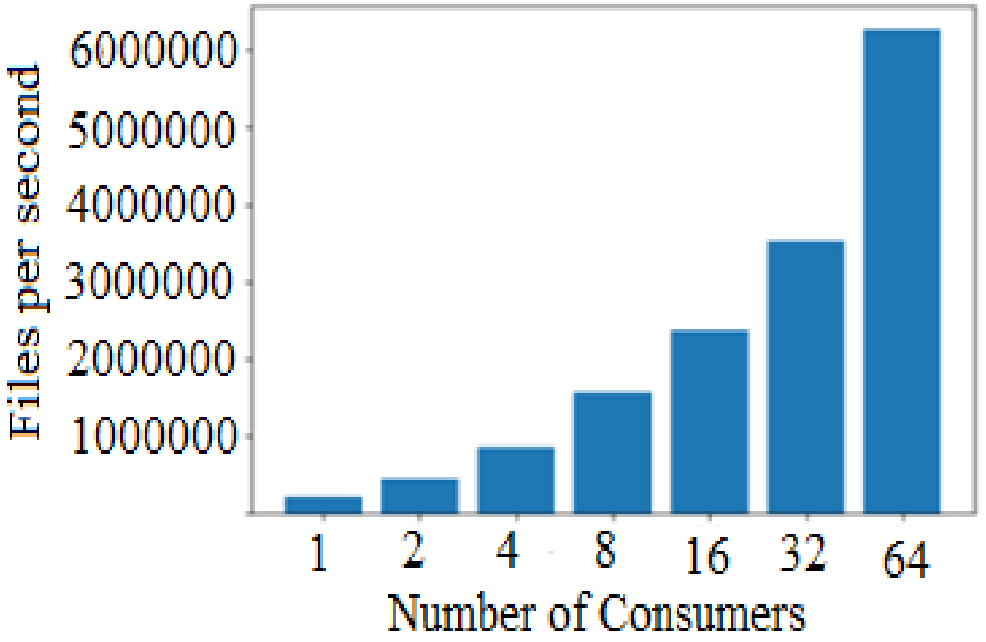
- Throughput measures number of files transfer per second through Kafka with respect to number of partitions and number of Consumers.
- The results shows the throughput increases till the number of partitions are less than number of consumer.
- The reason being all new consumers wait in idle mode until an existing consumer unsubscribes from that partition.



Throughput performance with configuration set to us
Consumers and 50 partitions using a single node.

Experiment - Throughput Performance

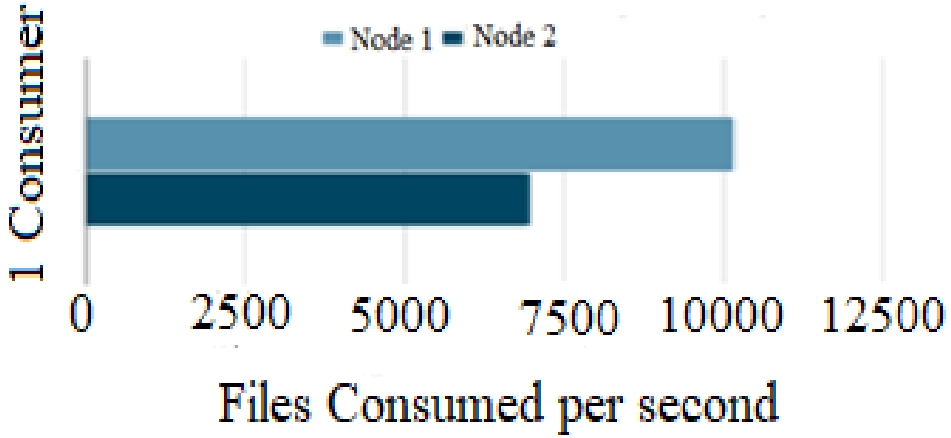
- When the number of Consumers exceeds the number of partitions in a stream, all new consumers wait in idle mode until an existing consumer unsubscribes from that partition.
- We can see that when we increase the number of partitions up to 150 then throughput increases as shown in.



Throughput performance with configuration set to use partitions using a single node

Experiment - Scalability Performance

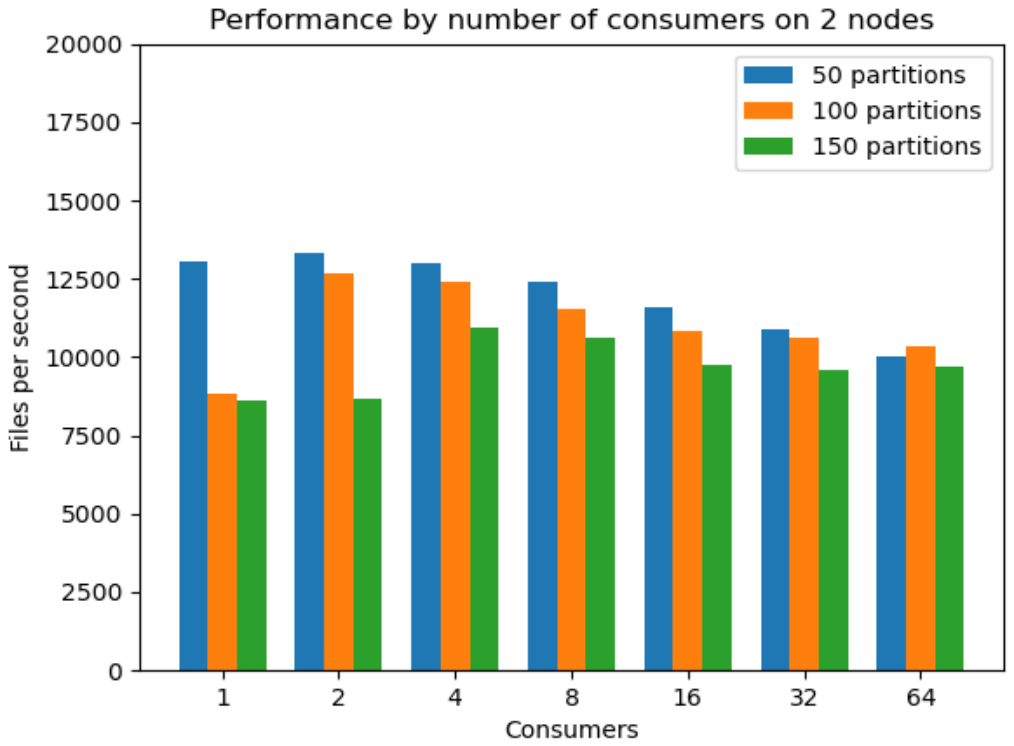
- Figure shows the throughput performance when the system uses two nodes each operates 64 Consumers, adding to totaling 128 consumers.
- The number of partitions is varied from 50, 100, 150. The performance changes significantly for two nodes. The number of CPU cores and stream partitions have less of an impact on consumer performance than the network bandwidth and latency.
- Since we are running only two brokers, we have set this replication factor to 2 to ensure 100% of messages are duplicated between nodes. This also provides fault tolerance - with the files being duplicated, nothing is lost if one broker goes down.



Throughput performance on each node using 2-node cluster (2 brokers).

Experiment - Throughput Performance on Cluster

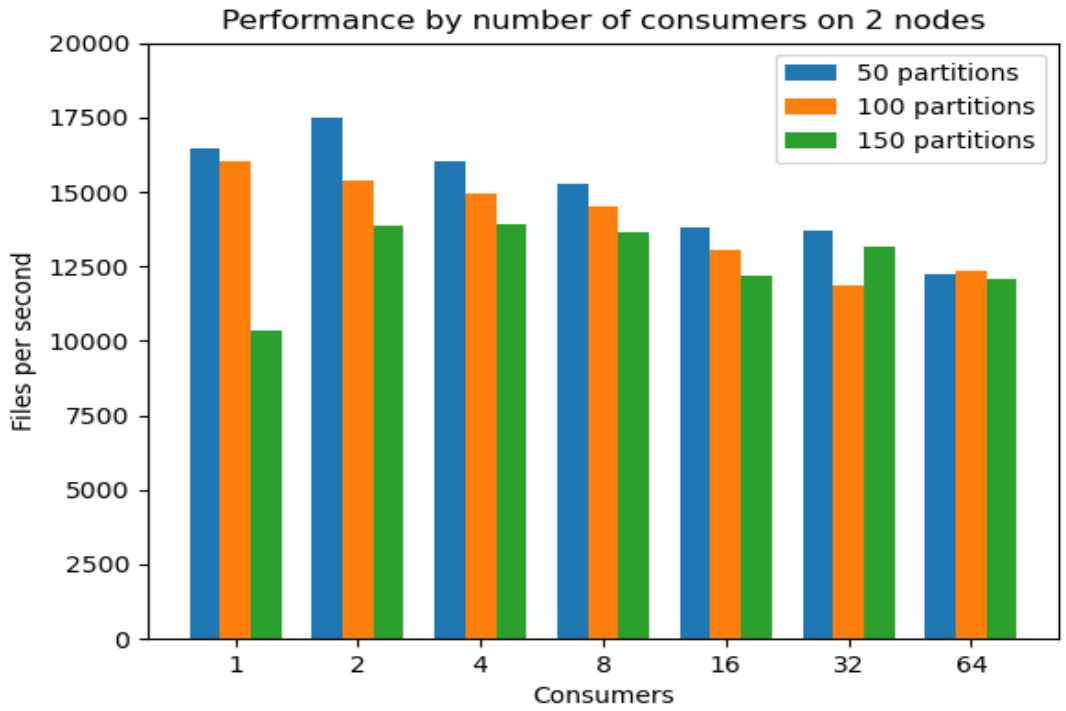
- Throughput performance when only one broker is present across a cluster of two nodes.
- Performance decreases as the number of consumers on two nodes increase.
- The reason is network bottleneck. When multiple consumers on a single node connect to a broker, the bandwidth available for each is reduced.



Throughput performance results for 2-node cluster. Brokers is running on Node 1

Experiment - Throughput Performance on Cluster

- Throughput performance when two broker are present across a cluster of two nodes.
- Much better performance compared when running single broker.
- All traffic happens between brokers on a single connection, which uses much less bandwidth because of partition replication.

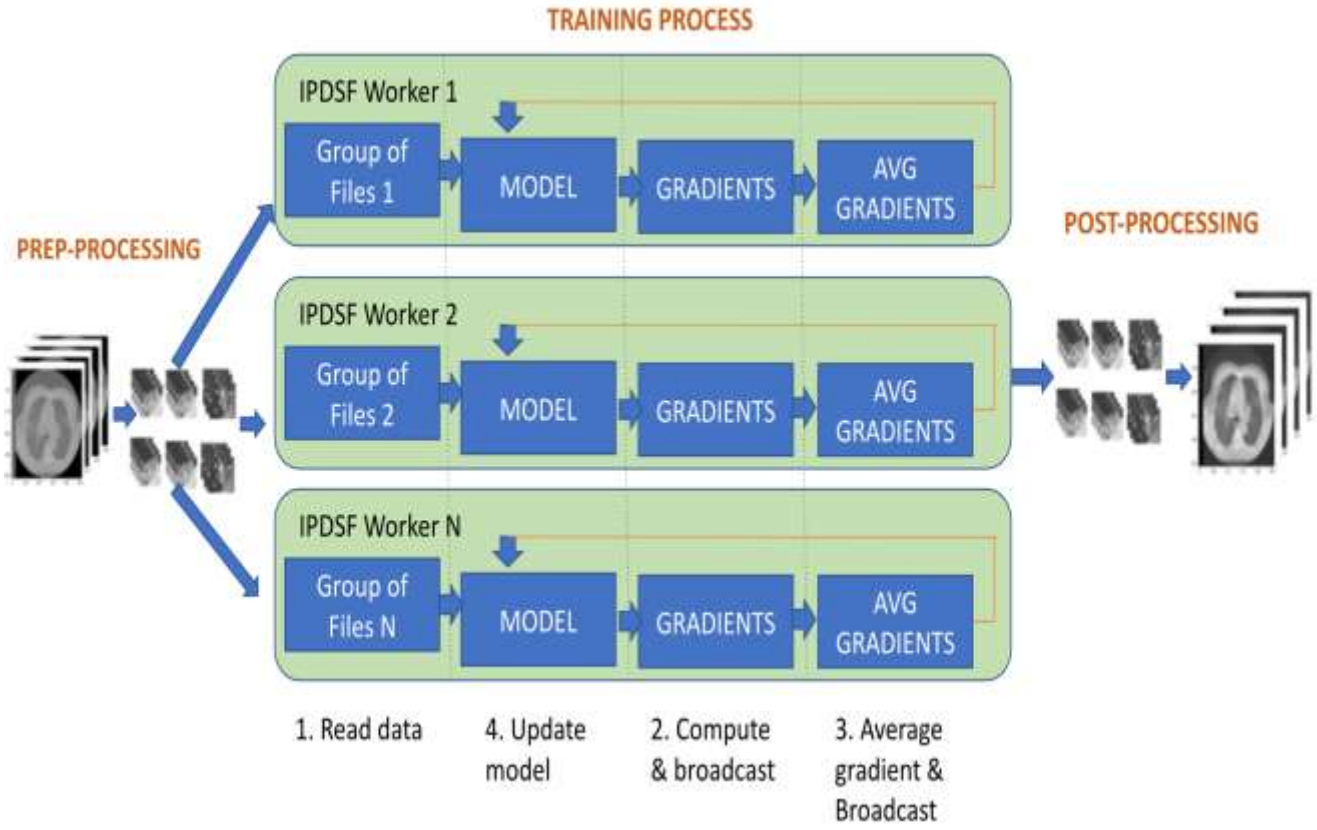


Throughput performance results for 2-node cluster. Brokers are running on Node 1 and Node 2

Evaluation of IPDSF for High Resolution Medical Imagery

- IPDSF-Horovod implementation using 4 GPUs and compared with single node implementation runs on a single GPU
- Trained our cGAN model for 100 epochs with a batch size of 50 samples and tested model by using 4 different patients and evaluated the performance of the model using two different evaluation metrics like the Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index metrics (SSIM).
- We have reconstructed our full-size high-resolution CT-Scans by conducting reverse preprocessing and then integrated the predicted cubes to an image of full-size CT-scan with a resolution of 515x512xZ where Z is the count of axial slices

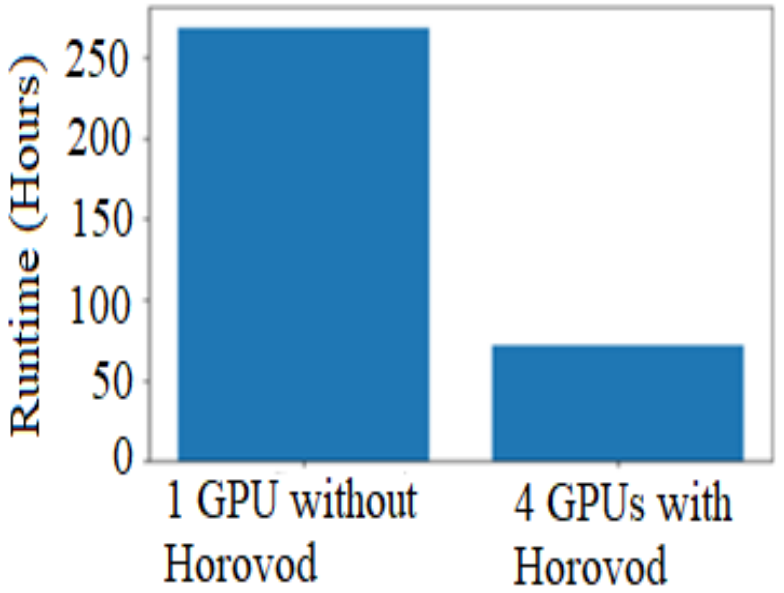
IPDSF for batch training processes



- Models are trained in parallel as distributed across multi-GPU servers by using the Horovod framework
- Horovod leverages efficient inter-GPU and inter-node communication methods such as NVIDIA Collective Communications Library (NCCL) and Message Passing Interface (MPI) to distribute and aggregate model parameters between workers.
- cGAN architecture, which generates 3D Computed Tomography scans in cubes for an autoencoder representation
- The novelty of our 3D convolutional cGAN lies in its distributed setting where we have used our IPDSF to generate scalable realistic high resolution full size synthetic 3D CT images

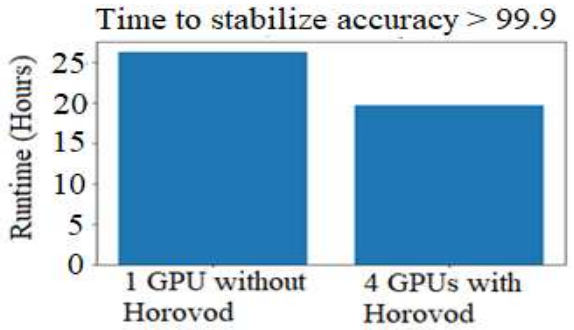
LATENCY & SCALABILITY OF CGAN MODEL

- IPDSF reduces cGAN training time by 4x factor compared with using single node implementation.
- This shows the linear scaling of using our IPDSF with respect to the number of GPUs.



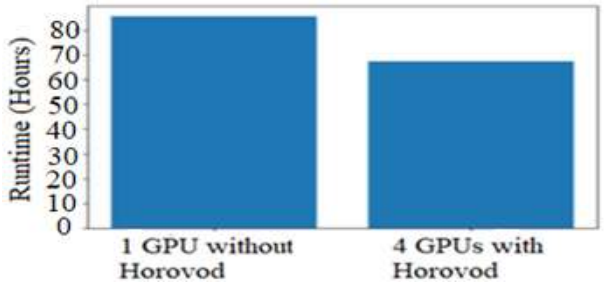
cGAN GPU runtime comparison with and without IPDSF-Horovod implementation

- GPU with IPDSF-Horovod minimized the runtime to 19 hours than the GPU that took runtime of 27 hours without IPDSF-Horovod



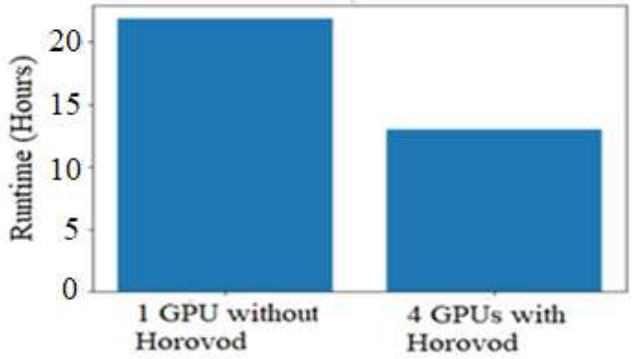
Time to stabilize accuracy of model using GPU with and without IPDSF-Horovod

- Runtime required to stabilize discriminator loss is depicted where the GPU with IPDSF-Horovod requires minimized runtime of about 65 hours while the GPU without IPDSF-Horovod requires 85 hours to stabilize the discriminator loss.



Time to stabilize discriminator loss < 0.0008 involving GPU with and without IPDSF- Horovod

- Runtime required to stabilize the generator loss is depicted where the GPU with IPDSF-Horovod requires a minimized runtime of about 13 hours while the GPU without IPDSF-Horovod requires a longer runtime of about 22 hours to stabilize the generator loss



Time to stabilize generator loss < 3.0 involving GPU with and without IPDSF- Horovod

cGAN Architecture Validation and Verification comparison

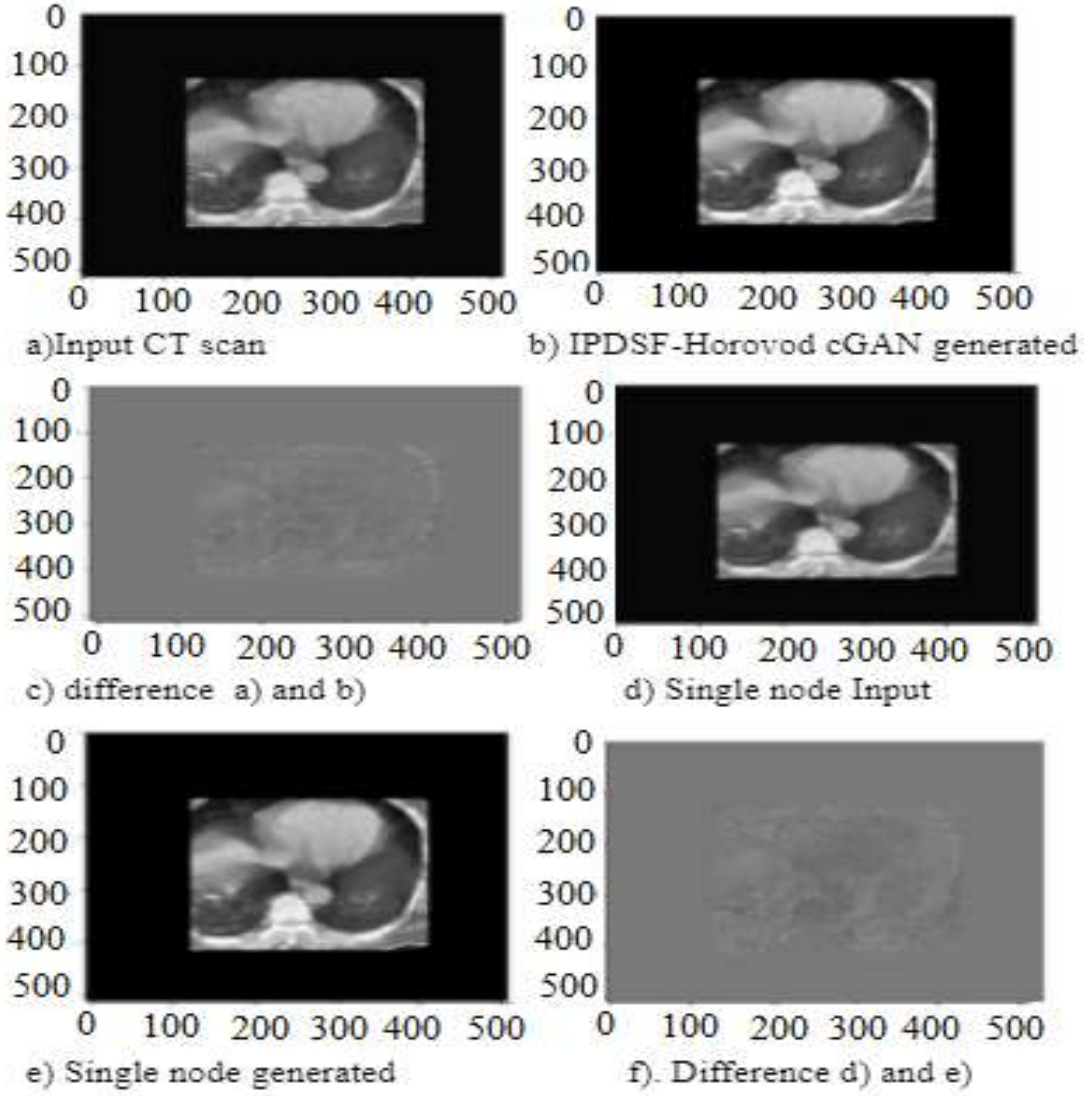
(a) We see the cGAN model using IPDSF-Horovod where the generated CT-image of a patient is shown where all the cubes are stitched back after reverse preprocessing

(b) shows the cGAN model using IPDSF-Horovod stitched cGAN predicted CT-image after reverse preprocessing of the same patient as shown in (a).

(c) the difference between the cGAN generated and the original enhanced contrast images are shown which seems to be having very little difference after subtracting the two images from (a) and (b)

(d), (e), and (f) are the images that came from the cGAN model which is trained without using IPDSF-horovod with the same patient as used in IPDSF-horovod training.

It is evident by visualizing IPDSF-Horovod with and without using IPDSF-Horovod makes no difference in terms of generating synthetic images, but it does make a drastic difference when it comes to accelerating the training time as mentioned earlier in our study.



cGAN evaluation metrics using with and without IPDSF-Horovod

Patient	PSNR (units =dB) With IPDSF-Horovod	SSIM With IPDSF-Horovod	PSNR (units =dB) Without IPDSF-Horovod	SSIM without IPDSF-Horovod
1	34	0.98	36	0.98
2	32.6	0.97	36.8	0.98
3	33.4	0.97	37.5	0.98
4	33.9	0.98	34.9	0.98

These readings clearly show that the cGAN model as trained using the IPDSF-Horovod distributed framework achieves comparable performance to the baseline implementation for all the 4 patients in the testing dataset.

Conclusion

High Performance and Scalability

- IPDSF achieve high performance and scalability while maintaining low latency, and high throughput by using both fine grain data and task partitioning parallelism.

Fault Tolerance

- Increase Fault Tolerance by including commodity hardware to duplicate data across Kafka using replication factor.

Novelty

- One of the first applications of attempting to use deep learning models to generate full size realistic synthetic high resolution 3D CT images using a distributed framework

Future Work

Incorporate

- Incorporate more data from different ceilometer sites and medical images.

Streaming Tensor Decomposition

- Evaluate IPDSF using streaming tensor decomposition tools like SPLATT

THANK YOU