OmniFed: A Modular Framework for Configurable Federated Learning from Edge to HPC

Sahil Tyagi*, Andrei Cozma*+, Olivera Kotevska, Feiyi Wang

Oak Ridge National Laboratory (ORNL), University of Tennessee Knoxville+

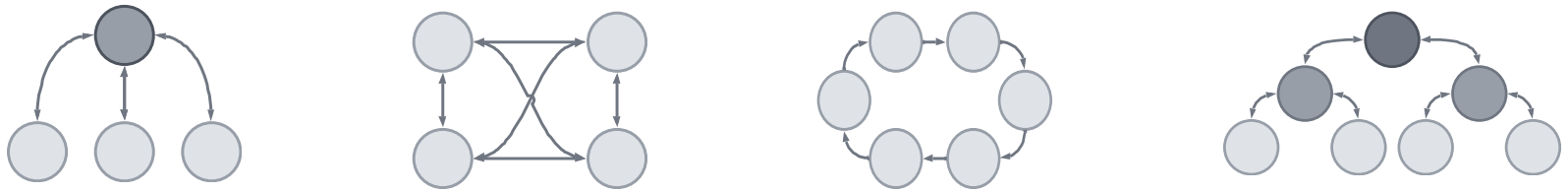
* = Equal contributors

Challenges in Federated and Collaborative Learning

- Data heterogeneity: non-IID (independent and identically distributed), restricted, unbalanced or skewed
- Systems + network heterogeneity: varying compute, latency and bandwidth on clients
- Adversarial nodes, malicious or semi-trust/untrustworthy participants
- Different arrangement or setup of clients in a federation (e.g., hospitals, financial institutions, computing and research facilities)

Motivation

- Rapid prototyping of new FL algorithms without boilerplate + easy deployment of existing ones!
- Implementing / configuring custom topologies remains complex
  - Potential to simplify / streamline
- Ability to run / deploy different cluster and training configurations
  - Think balanced/unbalanced topologies with heterogeneous islands of devices

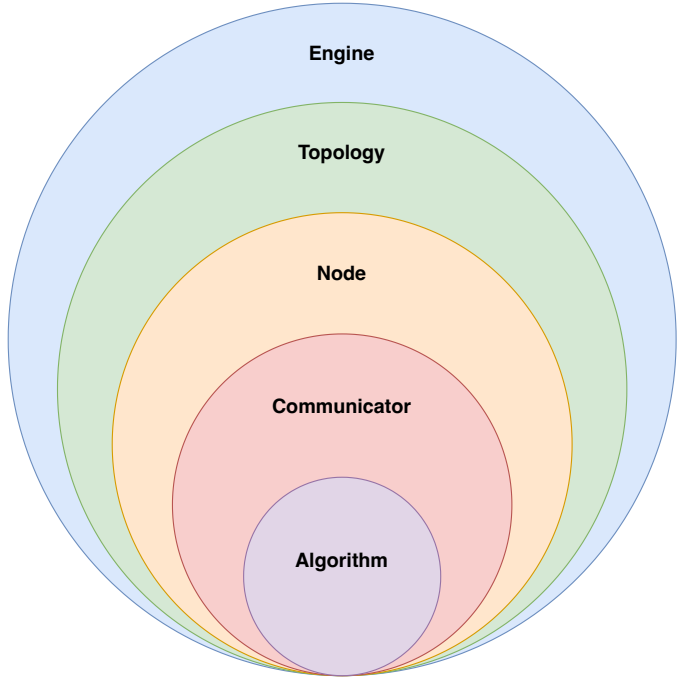


Centralized | Peer-to-Peer | Decentralized Ring | Hierarchical

- Ability to run different parallelization schemes and communication models

Federated Learning with OmniFed

- Treat modularity, flexibility and extensibility as first-class citizens!
- Developed with clear, layered abstractions and a unified API
  - Separates local computation, communication logic and algorithm control strategy
- Deploy FL jobs in a configuration-driven, schema-based workflow
- Agnostic to network topology and communication protocols
- Developer friendly with consistent and intuitive usage patterns
  - Allow users/developers to focus on what matters to them



OmniFed's design hint

Principles and Design Philosophy

- Clear separation of concerns across components
- Components expose clear, minimal interfaces
- Override-only-what-you-need paradigm.
  - e.g., add/remove privacy, compression features, etc. with minimal code change
- Single-file algorithm plugins
  - Extend and override abstract methods

```
1  defaults:
2    - override topology: centralized
3    - override algorithm: fedavg
4    - override model: resnet18
5    - override datamodule: cifar10
6
7  topology:
8    _target_: src.omnifed.topology.CentralizedTopology
9    num_clients: 8
10   inner_comm:
11     _target_: src.omnifed.communicator.GrpcCommunicator
12     master_port: 50051
13     master_addr: 127.0.0.1
14
15 algorithm:
16   _target_: src.omnifed.algorithm.FedAvg
17   lr: 0.01
18
19 global_rounds: 2
```

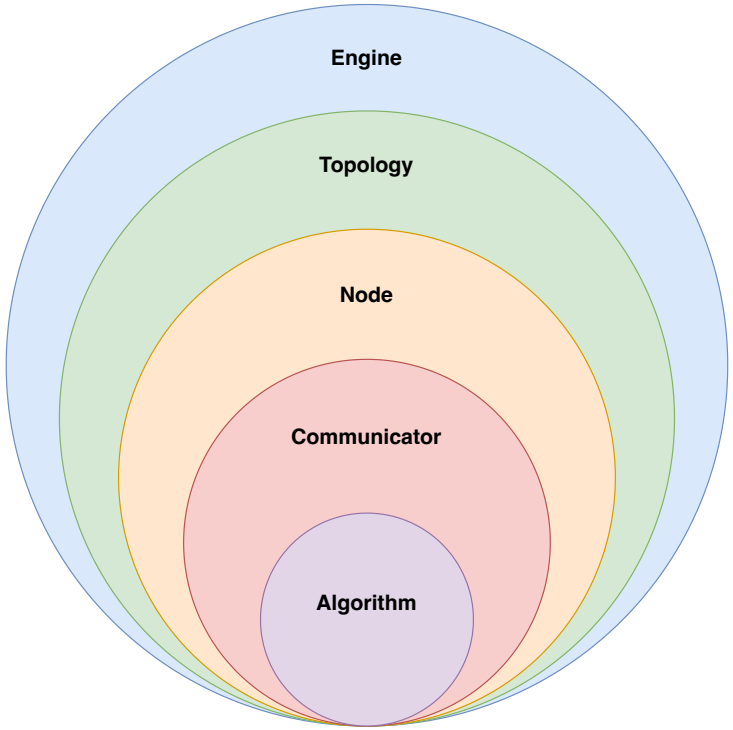
Use different FL algorithms (e.g., FedAvg, FedProx, etc.) by simply changing a line in the job's configuration file (line 16)

OmniFed's Software Stack

- Supported programming language: Python
- Core ML/AI backend: PyTorch
- Configuration management: Hydra
  - Configuration framework for managing complex experiment workflows.
    - Enables hierarchical config composition, command-line overrides, multi-run sweeps, and flexible job launching.
- Distributed execution: Ray
  - AI compute engine for orchestrating distributed workloads
  - Supports fine-grained parallelism across CPUs and GPUs
    - Easily scale from local machine to multi-node clusters.

OmniFed's Core Components

- **Engine**
  - Orchestrates rounds, resources and metrics
- **Topology**
  - Defines node graph and coordination pattern
- **Node**
  - Abstraction that owns model, data and communicator
- **Communicator**
  - Abstract primitives with a unified API for communication
- **Algorithm**
  - Defines client-local training logic through lifecycle hooks



Engine

Core orchestration engine for FL experiments.

Responsibilities:

- Launch & coordinate distributed experiments
- Manage node lifecycle and resource allocation
- Collect, analyze, and report metrics and statistics

Integration:

- Instantiated via YAML configuration file (Hydra)
- Coordinates with *Topology* definitions to spawn *Node* actors (Ray)
- Manages Node actors running *Algorithm* implementations

Topology

Defines the structure, coordination, and communication patterns for distributed nodes.

Responsibilities:

- Define node graph structure and relationships
- Support templates for centralized, decentralized, hierarchical
- Enabling custom topology definitions via graph-based representations

Integration:

- Configured via YAML topology definitions
- Used by *Engine* to spawn and organize *Node* actors
- Determines the pattern how *Nodes* communicate

Node

Refers to any participant in the federation.

Responsibilities:

- Manage local model state, data, and device resources
- Interface with *Communicators* for data exchange
- Execute local algorithm implementations

Integration:

- Instantiated as Ray actors by the *Engine*
- Configured via Hydra YAML files with optional per-node overrides
- Coordinates with *Algorithm* components for local compute logic

Communicator

Serves as data exchange utility for *Nodes*

Responsibilities

- Expose a unified API that abstracts the underlying protocol implementations
- Configurable communication backend selection *without* user code changes

Integration:

- Collective communicator using MPI, NCCL or Gloo backends (for HPC environments)
- RPC (remote procedure call) communicator using gRPC protocol (for geo-distributed computing)
- Message Queue/PubSub communicator using brokers (for internet-of-things (IoT) and middleware)

Algorithm

Provides FL logic via configurable lifecycle hooks

Responsibilities:

- Implements learning strategy and defines update, aggregation and post-aggregation training logic

Integration:

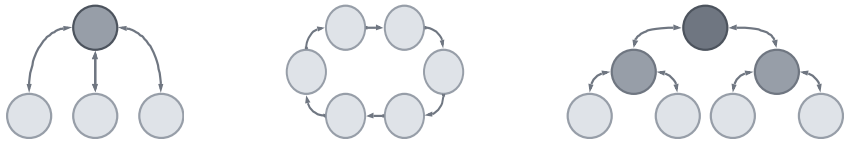
- Executed by *Nodes* over FL job lifecycle
- Flexible and extensible; currently supports 10+ SoTA FL algorithms: FedAvg, FedProx, Scaffold, FedMom, FedNova, FedBN, FedPer, Ditto, MOON, FedDyn, DiLoCo
- New algorithms can be implemented easily by overriding abstract methods from *BaseAlgorithm* class

OmniFed's Training Configuration

YAML configuration file defines job for experiment, simulation or deployment

- Defaults section
  - Provides default values which can be overridden
- Topology section
  - Comprises templates for common patterns like centralized, decentralized and hierarchical
- Algorithm section
  - Specifies training algorithm to use + hyperparameters

```yaml
defaults:
  - override topology: centralized
  - override algorithm: fedavg
  - override model: resnet18
  - override datamodule: cifar10

topology:
  _target_: src.omnifed.topology.CentralizedTopology
  num_clients: 8
  inner_comm:
    _target_: src.omnifed.communicator.GrpcCommunicator
    master_port: 50051
    master_addr: 127.0.0.1

algorithm:
  _target_: src.omnifed.algorithm.FedAvg
  lr: 0.01

global_rounds: 2
```



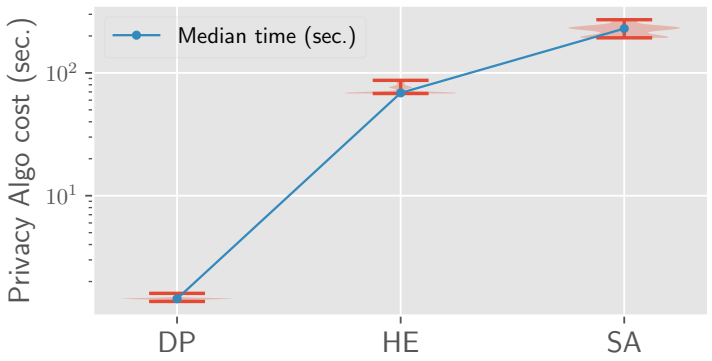
Integrating Security and Privacy

- Ensuring secure and privacy-preserving FL is paramount!
- Any job can be configured with privacy using:
  - Differential privacy (DP)
  - Homomorphic encryption (HE)
  - Secure multi-party aggregation (SA)

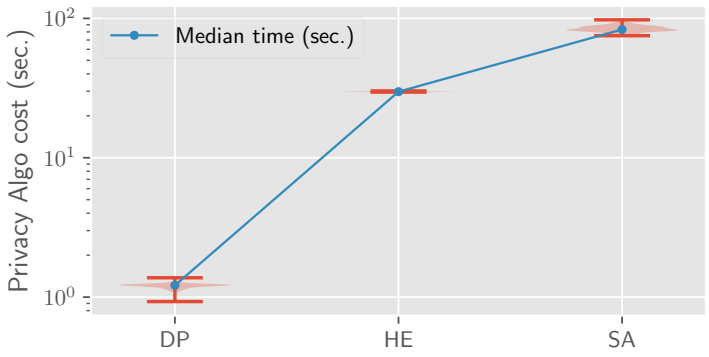
```
inner_comm:
    _target_: src.omnifed.communicator.TorchDistCommunicator
    port: 28670
    privacy:
        _target_: src.omnifed.privacy.DifferentialPrivacy
        \epsilon: 10.0
        \delta: 0.00001
```

```
inner_comm:
    _target_: src.omnifed.communicator.TorchDistCommunicator
    port: 28670
    compression:
        _target_: src.omnifed.privacy.HomomorphicEncryption
       polymod_degree: 2e8
```

ResNet18



VGG11



Model Accuracy with Differential Privacy

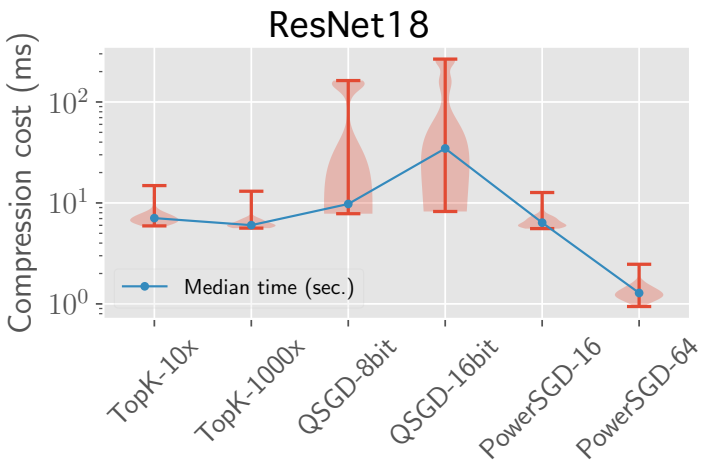
DNN	DP Acc. (%)	
	ϵ=1	ϵ=10
Res.	97.98	**98.06**
VGG	24.1	**28.6**

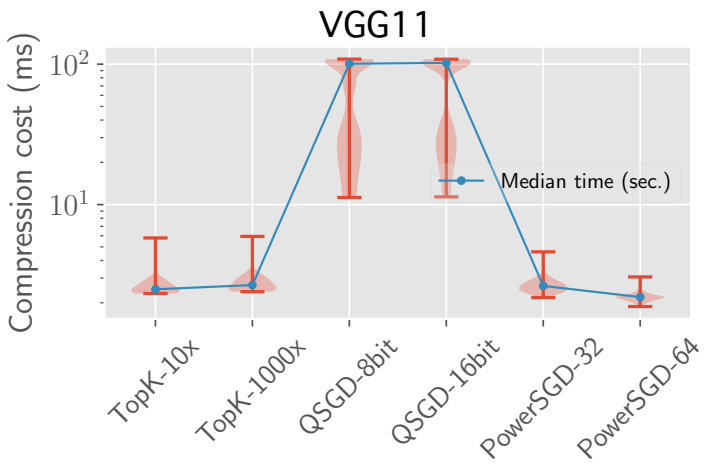
Mitigating Communication Cost with Compression

- Aggregation phase bottlenecks training under heterogeneous networks
- OmniFed supports model/gradient compression spanning:
  - Sparsification
  - Quantization
  - Low-rank approximation

```yaml
inner_comm:
    _target_: src.omnifed.communicator.TorchDistCommunicator
    port: 28670
    compression:
        _target_: src.omnifed.communicator.compression.TopK
        k: 1000x
```

```yaml
inner_comm:
    _target_: src.omnifed.communicator.TorchDistCommunicator
    port: 28670
    compression:
        _target_: src.omnifed.compression.PowerSGD
        rank: 16
```



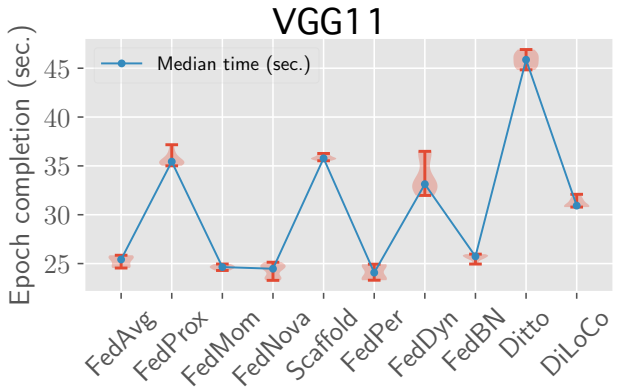


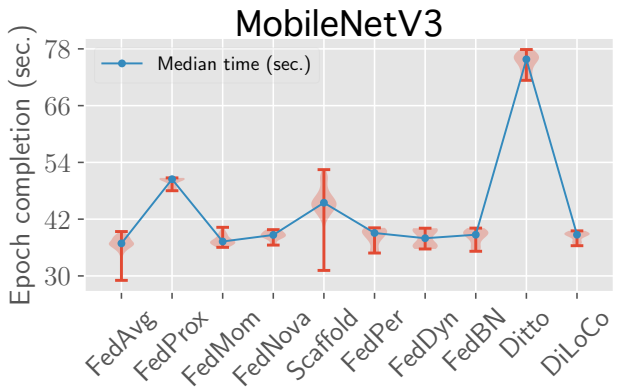
Convergence quality with compression

Compression	ResNet18	VGG11
Top$k$-10×	99.09%	84.6%
Top$k$-1000×	99.05%	78.05%
DGC-10×	99.18%	84.22%
DGC-1000×	98.5%	78.75%
QSGD 8-bit	**99.26%**	85.48%
QSGD 16-bit	99.12%	**85.52%**
PowerSGD r-64	99.07%	75.45%
PowerSGD r-32	90.31%	6.74%

Benchmarking FL Algorithms

- Users should quickly hit the ground running when testing different FL algorithms or trying different parameters with just one!





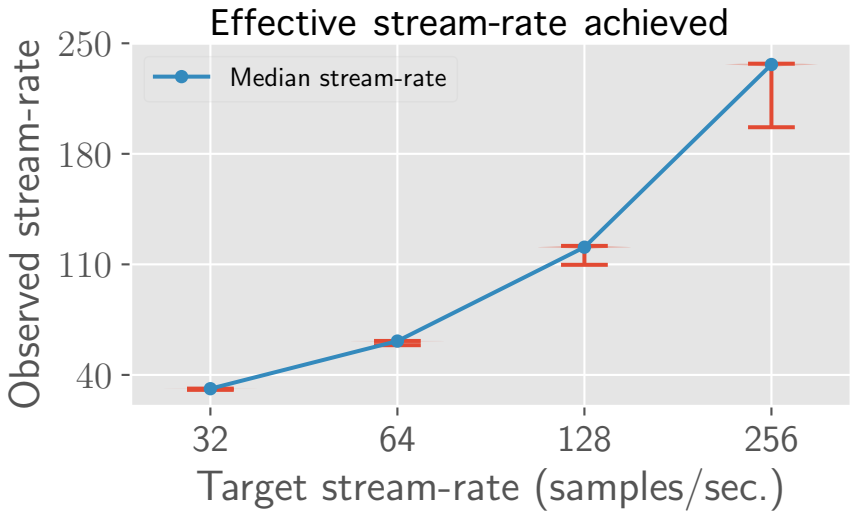

```yaml
defaults:
  - override topology: centralized
  - override algorithm: fedavg
  - override model: resnet18
  - override datamodule: cifar10

topology:
  _target_: src.omnifed.topology.CentralizedTopology
  num_clients: 8
  inner_comm:
    _target_: src.omnifed.communicator.GrpcCommunicator
    master_port: 50051
    master_addr: 127.0.0.1

algorithm:
  _target_: src.omnifed.algorithm.FedAvg
  lr: 0.01

global_rounds: 2
```

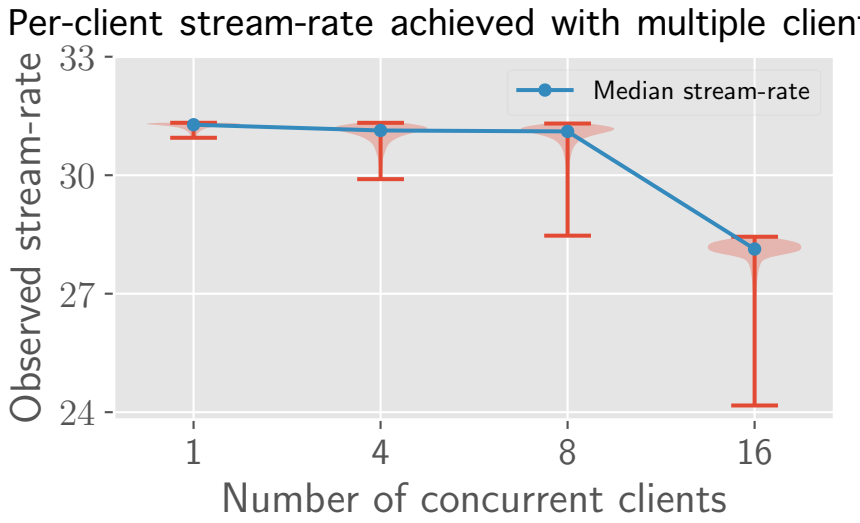
Convergence quality with different algorithms

Algorithm	ResNet18	VGG11	AlexNet	MobileNetV3
FedAvg	99.32%	**86.6%**	87.9%	81.35%
FedProx	99.26%	86.31%	87.98%	**82.96%**
FedMom	99.14%	66.39%	63.85%	48.98%
FedNova	91.18%	14.1%	58.1%	22.27%
Moon	**99.46%**	81.67%	87.28%	81.4%
FedPer	90.9%	26.93%	82.94%	14.59%
FedDyn	99.31%	86.18%	**88.78%**	79.15%
FedBN	99.33%	86%	88.7%	78.65%
Ditto	73.64%	5.5%	40%	9.84%
DiLoCo	84.88%	5.1%	45.17%	15.47%

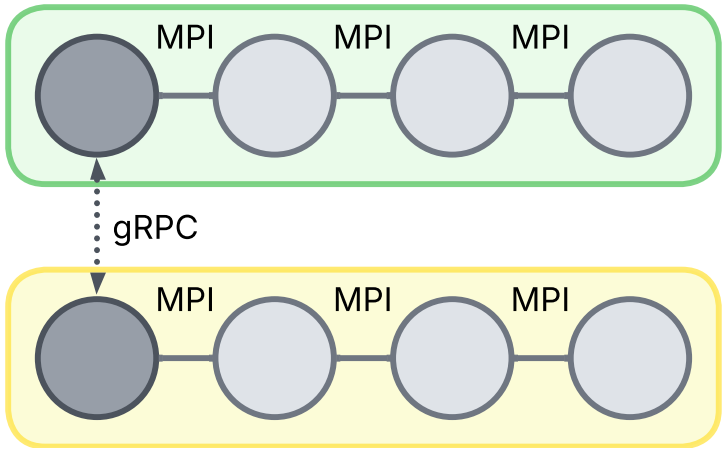
Real-time Learning over Streaming Data

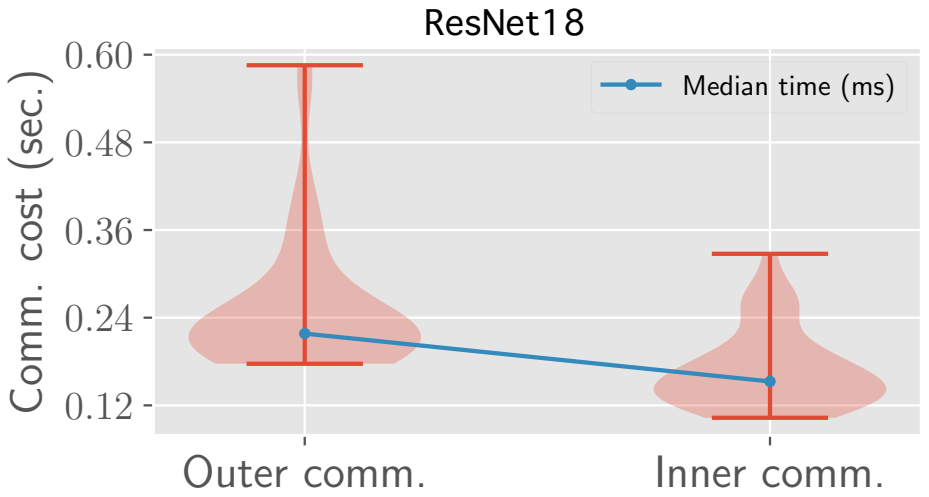
- High-volume streaming data presents opportunity for near real-time FL!
- OmniFed provides PyTorch dataloaders integrated with Kafka pub-sub broker to simulate real-time learning over streaming data





Custom Topologies with Multiple Communicators





- We share an instance of cross-facility FL analogous to a hub-and-spoke topology
- Clients/nodes are densely connected within a site, and sparsely connected across sites
- Hierarchical communication pattern with different protocols

Ongoing Work and Future Directions

- Exascale deployment training LLMs in a federated manner
- Cross-facility FL
- Adding trustworthiness/assurance and interpretability mechanisms as a core module of training phase
- Integrating fairness-aware mechanisms such as contribution evaluation and incentives-based FL
- Support for additional ML frameworks like TensorFlow, TorchTitan, ExecuTorch

- GitHub: https://github.com/at-aaims/OmniFed
- To discuss more, please reach out!

Thank you!