THE UNIVERSITY OF CHICAGO · UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN · Argonne NATIONAL LABORATORY

FEDERATED LEARNING TUTORIAL: CONCEPTS, APPLICATIONS, CHALLENGES, AND FRAMEWORK

APPFL

ZILINGHAN LI

Machine Learning Engineer
Data Science and Learning Division,
Argonne National Laboratory
zilinghan.li@anl.gov

RAVI MADDURI

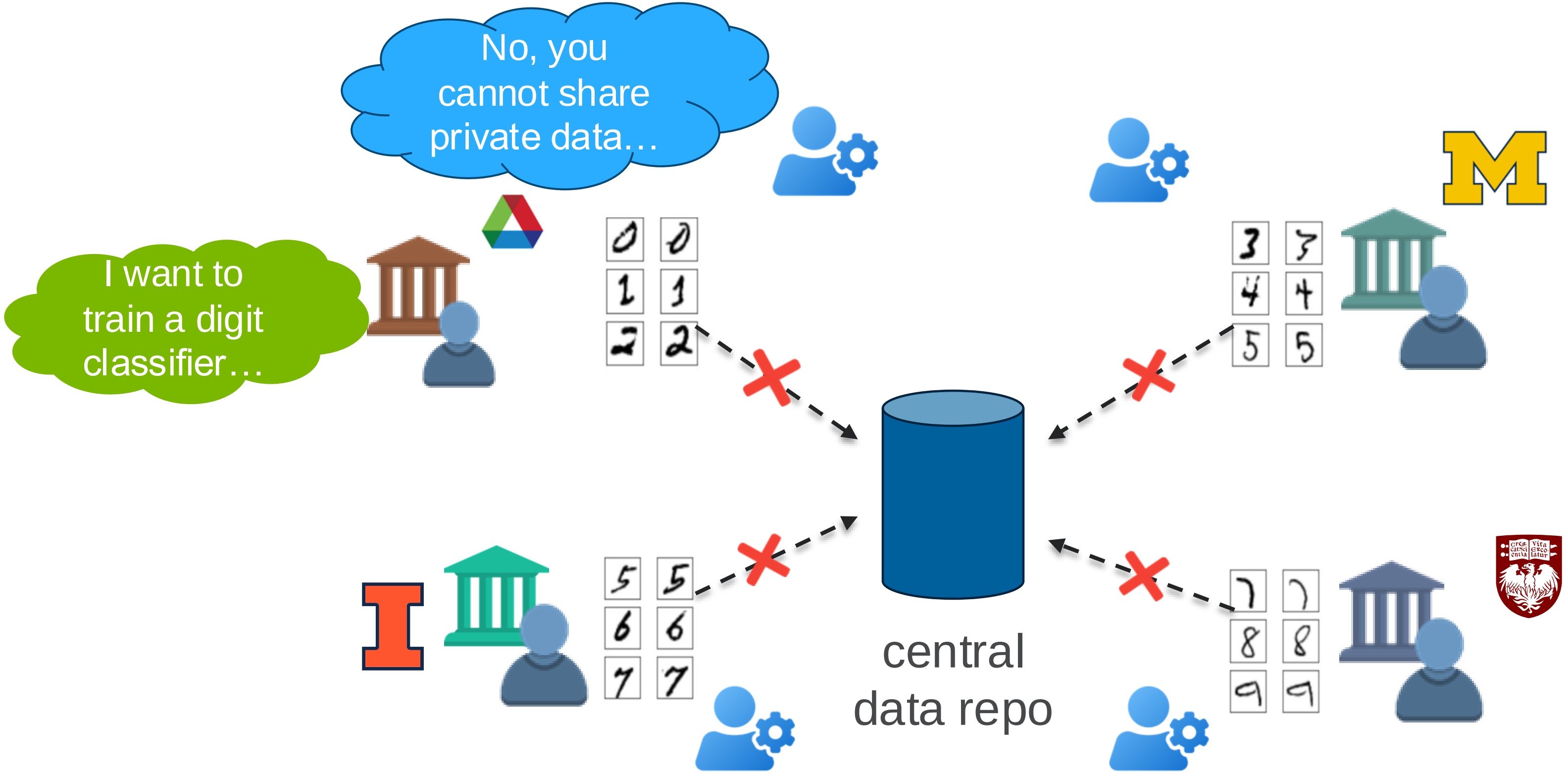
Senior Computer Scientist
Data Science and Learning Division,
Argonne National Laboratory
madduri@anl.gov

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

MICDE

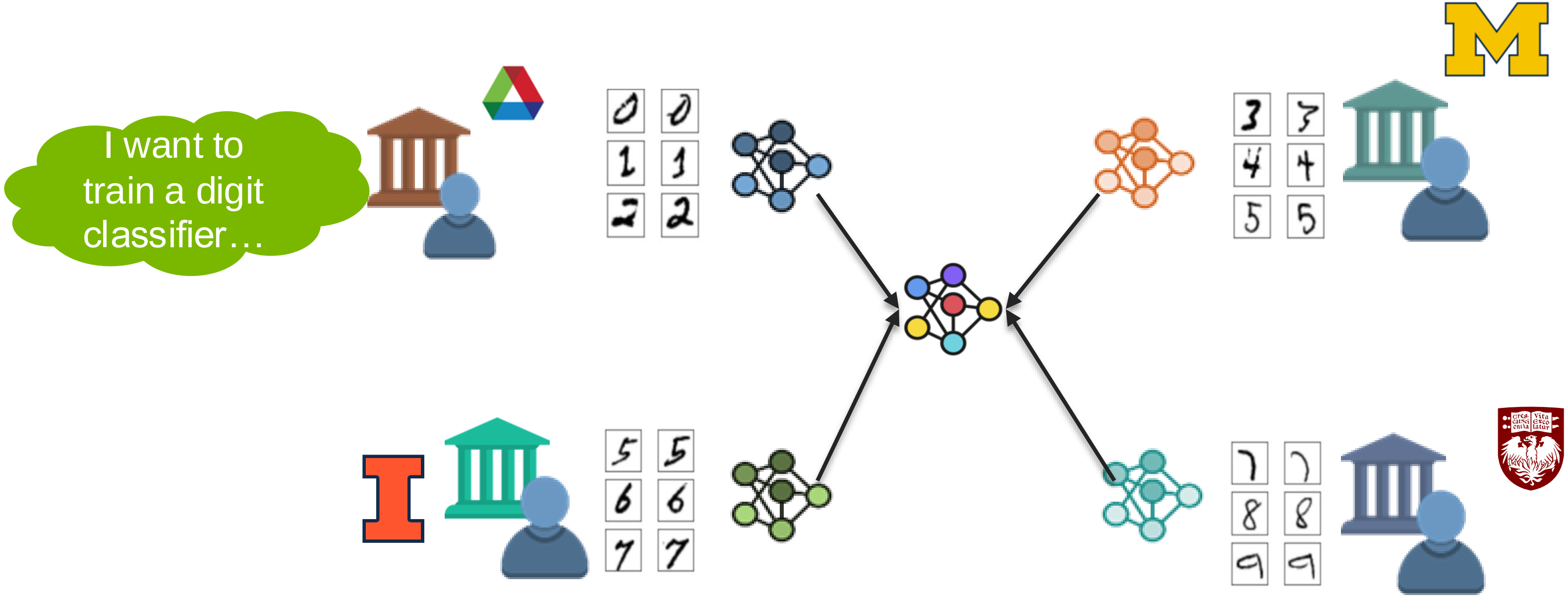
SciFM

MOTIVATION – AN “EXTREME” CASE


No, you cannot share private data…
I want to train a digit classifier…
central data repo

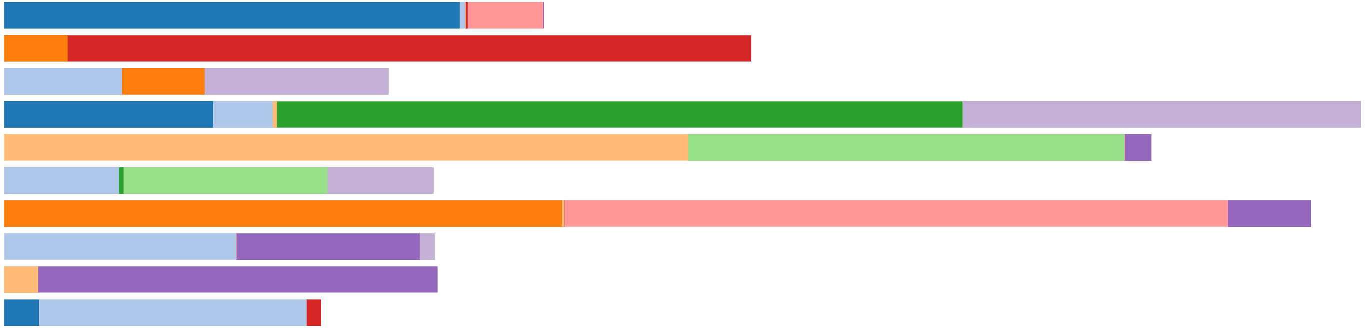
MOTIVATION – AN “EXTREME” CASE

Share the model trained on local data instead of sharing the data directly



These four clients create a loose federation, and we call it federated learning (FL).

MOTIVATION – AN “EXTREME” CASE



MNIST dataset partition: number of data samples for different labels in each client’s local datasets.

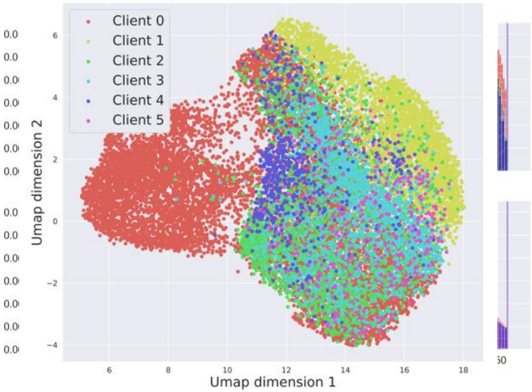
Training a CNN by sharing model parameters trained locally (i.e. using federated learning) can get **>95%** accuracy!

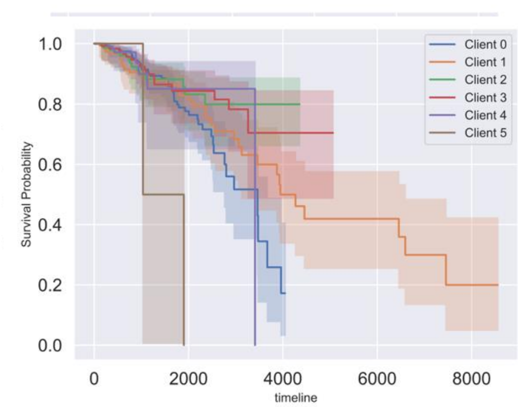
MOTIVATION – ANY MORE REAL REASONS?

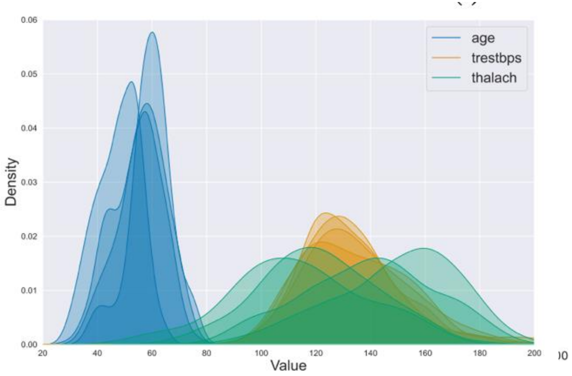
This example is trivial and does not sound like something will happen in real life…

So, again, why federated learning?

In real world, it is common that the data at different data silos have different distributions







Ogier du Terrail, Jean, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold et al. "Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings." *Advances in Neural Information Processing Systems* 35 (2022): 5315-5334.

MOTIVATION – ANY MORE REAL REASONS?

Privacy Concerns in Biomedical Data

In biomedical research, access to many types of data, such as identifiable electronic health records (HER), medical images, and electrocardiogram (ECG) readings, is strictly regulated by law like HIPAA in the United States and GDPR in the European Union. Data access committees and institutional review boards (IRB) manually manage access controls to ensure that research is ethical and that the privacy of research subjects is safeguarded.

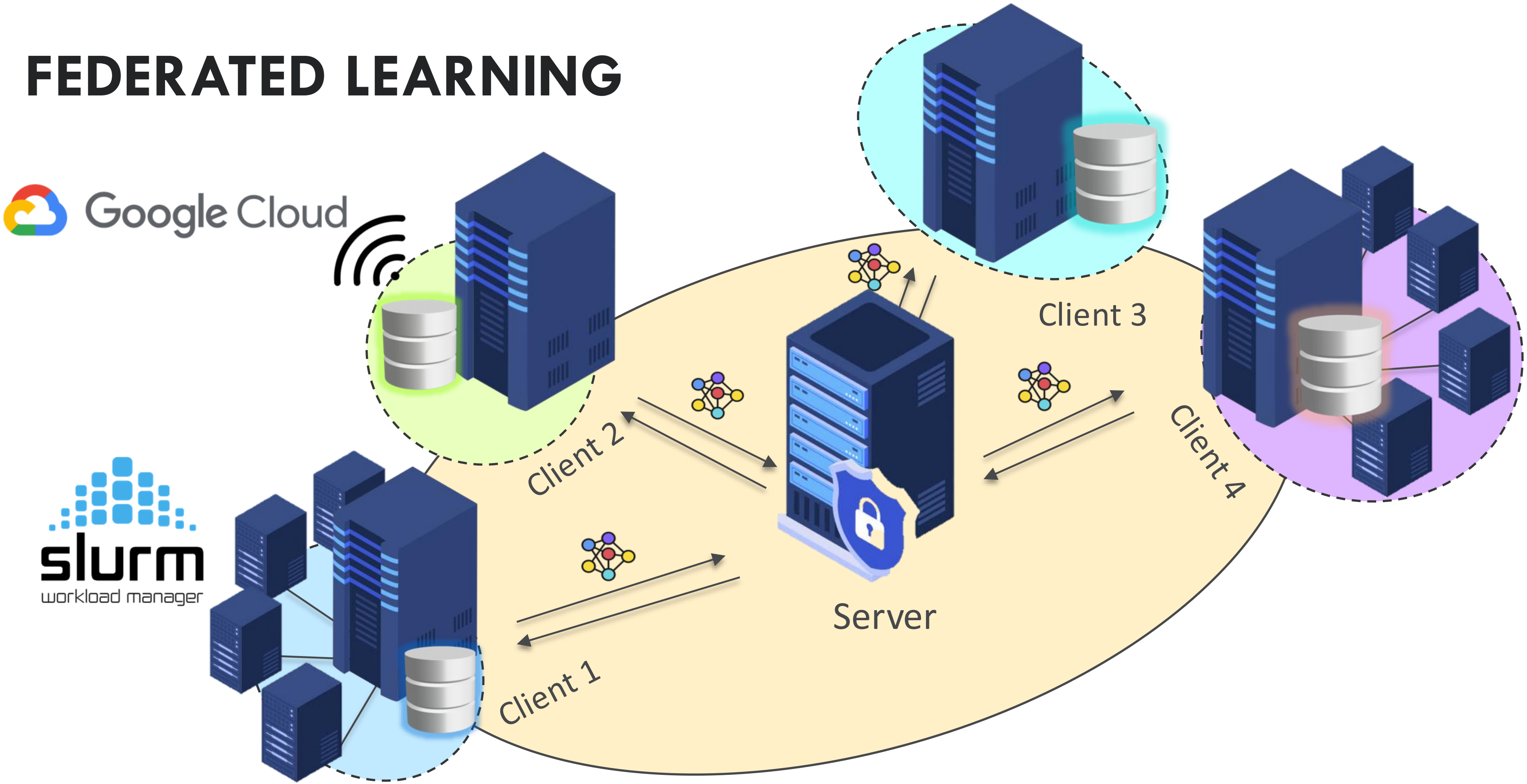
Historically, these necessary, but cumbersome, access restrictions have had the undesirable effect of siloing data at the host institution which in turn has stymied collaborative research.

MOTIVATION – ANY MORE REAL REASONS?

FL can also be used in training foundation models to leverage a large amount of private data located at distributed data silos.

- More than 80% of today’s data are private!
- In addition to the number of parameters in the foundation model, the amount of the training data is also one of the key factors to success.

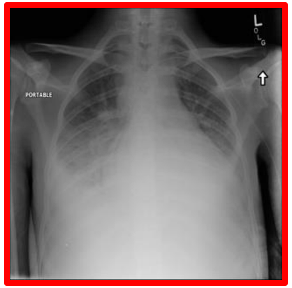
FEDERATED LEARNING

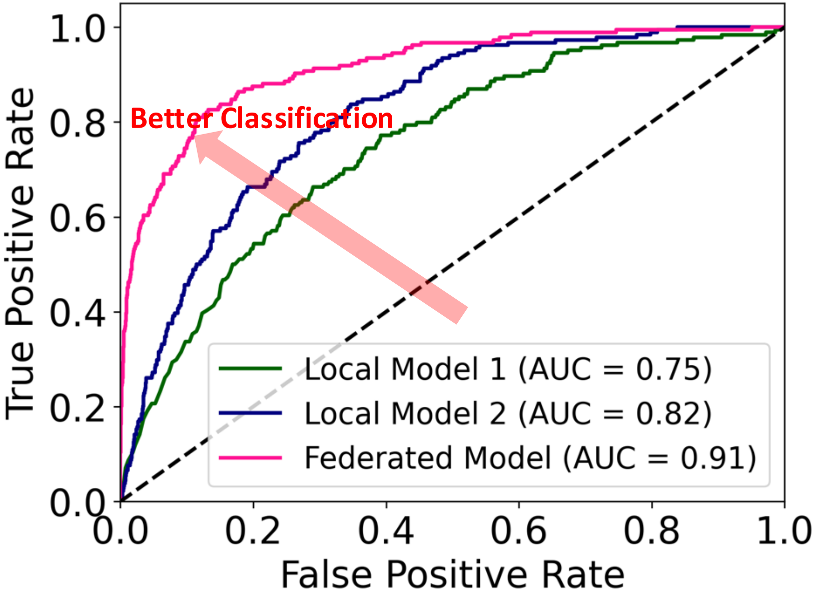


FEDERATED LEARNING APPLICATION

Detection of COVID-19 from Chest X-Rays

- **Datasets:**
  - ANL-COVID: the dataset is aggregated from multiple open-source datasets
  - Uchicago-COVID: private dataset collected by UChicago

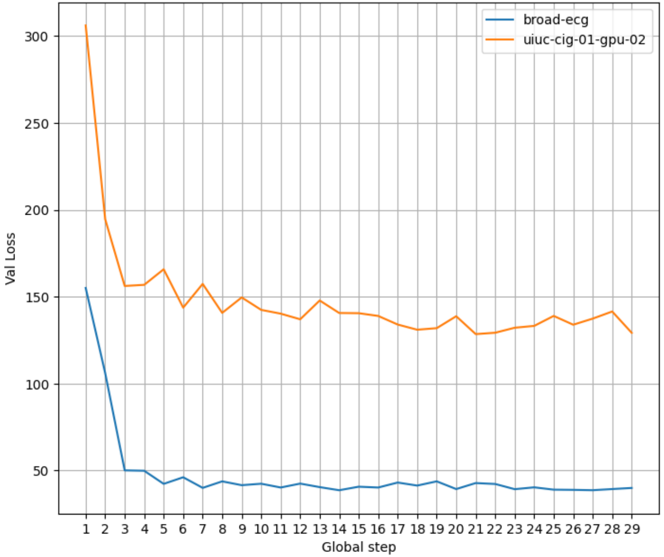



Better Classification
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Local Model 1 (AUC = 0.75)
Local Model 2 (AUC = 0.82)
Federated Model (AUC = 0.91)

Hoang, Trung-Hieu, Jordan Fuhrman, Ravi Madduri, Miao Li, Pranshu Chaturvedi, Zilinghan Li, Kibaek Kim et al. "Enabling end-to-end secure federated learning in biomedical research on heterogeneous computing environments with APPFLx." *arXiv preprint arXiv:2312.08701* (2023).

FEDERATED LEARNING APPLICATION

Biological Aging Prediction from ECG Signal

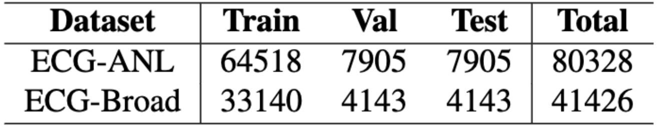


Best MSE on ECG-ANL = 125.00

Best MSE on ECG-Broad = 41.70

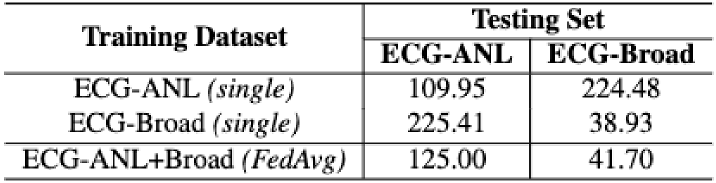
FL can learn a global model that performs relative well on both datasets

Table 1. Statistic of the datasets used in the biological aging prediction from ECG signal experiment.



Dataset	Train	Val	Test	Total
ECG-ANL	64518	7905	7905	80328
ECG-Broad	33140	4143	4143	41426

Table 2. Testing MSE error of the biological aging prediction from ECG signal models.

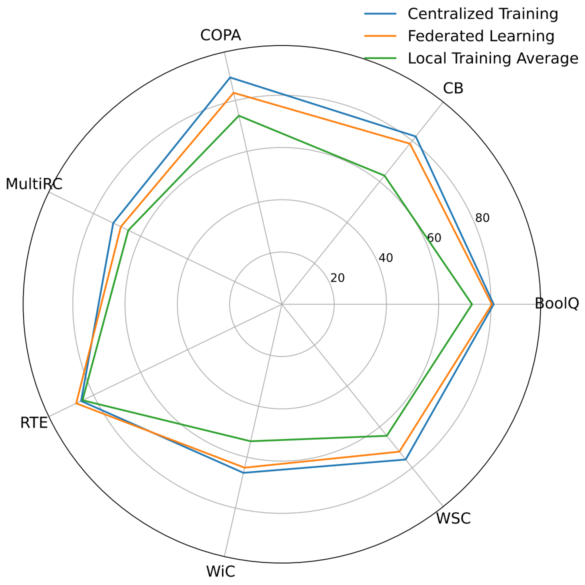


Training Dataset	Testing Set	
	ECG-ANL	ECG-Broad
ECG-ANL *(single)*	109.95	224.48
ECG-Broad *(single)*	225.41	38.93
ECG-ANL+Broad *(FedAvg)*	125.00	41.70

Hoang, Trung-Hieu, Jordan Fuhrman, Ravi Madduri, Miao Li, Pranshu Chaturvedi, Zilinghan Li, Kibaek Kim et al. "Enabling end-to-end secure federated learning in biomedical research on heterogeneous computing environments with APPFLx." *arXiv preprint arXiv:2312.08701* (2023).

FEDERATED LEARNING APPLICATION

Finetuning an LLM for Various Downstream Tasks

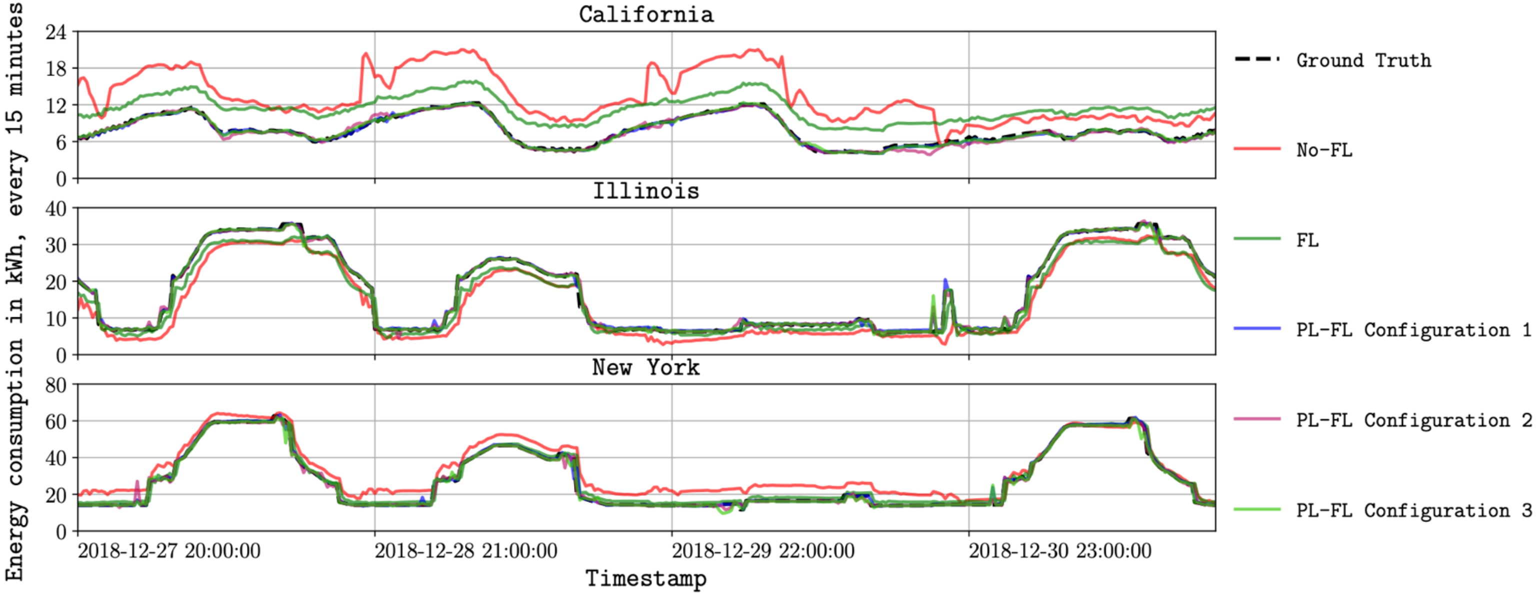


Compare FL with centralized training (collecting all data to a central location) and local training.

Li, Zilinghan, Shilan He, Pranshu Chaturvedi, Volodymyr Kindratenko, Eliu A. Huerta, Kibaek Kim, and Ravi Madduri. "Secure Federated Learning Across Heterogeneous Cloud and High-Performance Computing Resources-A Case Study on Federated Fine-tuning of LLaMA 2." *Computing in Science & Engineering* (2024).

FEDERATED LEARNING APPLICATION

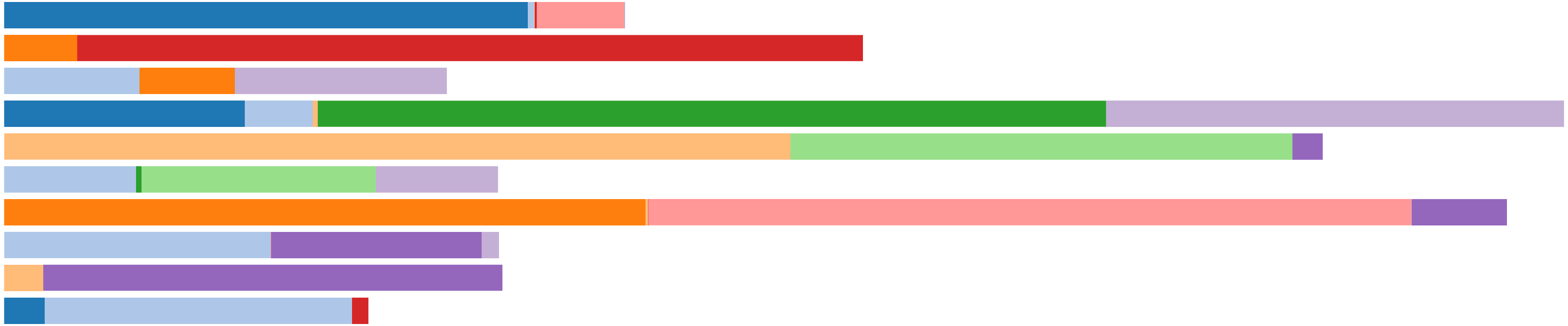
Forecasting Energy Consumption



Bose, Shourya, and Kibaek Kim. "Federated Short-Term Load Forecasting with Personalization Layers for Heterogeneous Clients." *arXiv preprint arXiv:2309.13194* (2023).

FEDERATED LEARNING CHALLENGE

Data Heterogeneity



Data heterogeneity: unbalanced and non-independent and identically distributed (non-IID) data.

Different clients have different local objectives, and drift away from each other.

FEDERATED LEARNING CHALLENGE

Data Heterogeneity

Solutions:

(1) Server-side momentum [1] or other optimizations [2] to avoid drastic changes in the global model

$$\omega \leftarrow \omega - \Delta\omega \text{ where}$$
$$\Delta\omega = \Sigma p_i \Delta\omega_j$$

Traditional FedAvg

$$v \leftarrow \beta v + (1 - \beta)\Delta\omega$$
$$\omega \leftarrow \omega - v$$
$$\text{and } \Delta\omega = \Sigma p_i \Delta\omega_j$$

FedAvg with momentum

[1] Hsu, Tzu-Ming Harry, Hang Qi, and Matthew Brown. "Measuring the effects of non-identical data distribution for federated visual classification." *arXiv preprint arXiv:1909.06335* (2019).

[2] Reddi, Sashank, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. "Adaptive federated optimization." *arXiv preprint arXiv:2003.00295* (2020).

FEDERATED LEARNING CHALLENGE

Data Heterogeneity

Solutions:

(2) Adding proximal term in client local training objectives to prevent local models from drifting far away from the global model.

$$\min_{w} h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 \, .$$

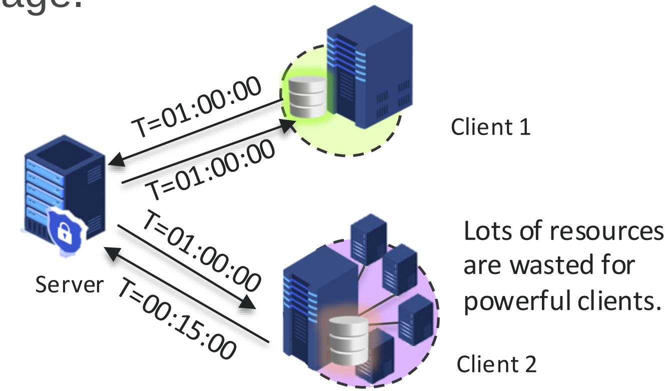
Li, Tian, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." *Proceedings of Machine learning and systems* 2 (2020): 429-450.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- As the computing capabilities of client machines could have large variance, clients may take significantly different amount of time to finish one local training round.
- Synchronous FL algorithms, where the server waits for all clients to send the local models back, suffer from resource wastage.

Heterogeneous client computing resources.



Lots of resources are wasted for powerful clients.

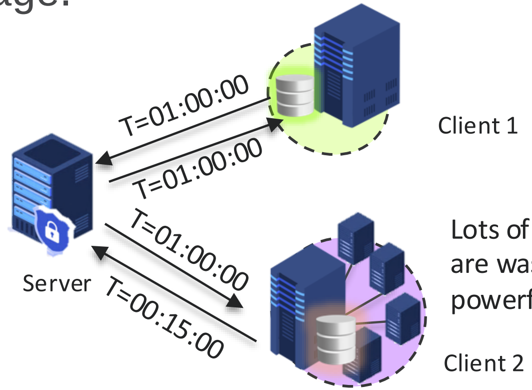
Resource wastage in synchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- As the computing capabilities of client machines could have large variance, clients may take significantly different amount of time to finish one local training round.
- Synchronous FL algorithms, where the server waits for all clients to send the local models back, suffer from resource wastage.

Heterogeneous client computing resources.



Lots of resources are wasted for powerful clients.

Resource wastage in synchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- Asynchronous FL – which updates global model immediately once receiving local model from each client – suffers from the stale (outdated) local models from slower clients, thereby causing the global model to drift away from slower clients.

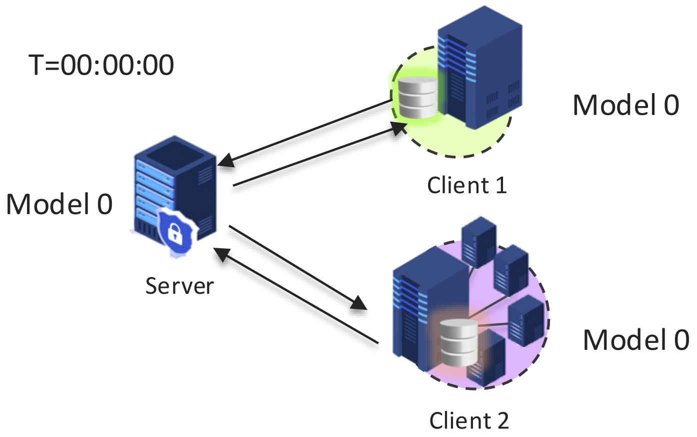


Figure 4: Staleness problem in asynchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- Asynchronous FL – which updates global model immediately once receiving local model from each client – suffers from the stale (outdated) local models from slower clients, thereby causing the global model to drift away from slower clients.

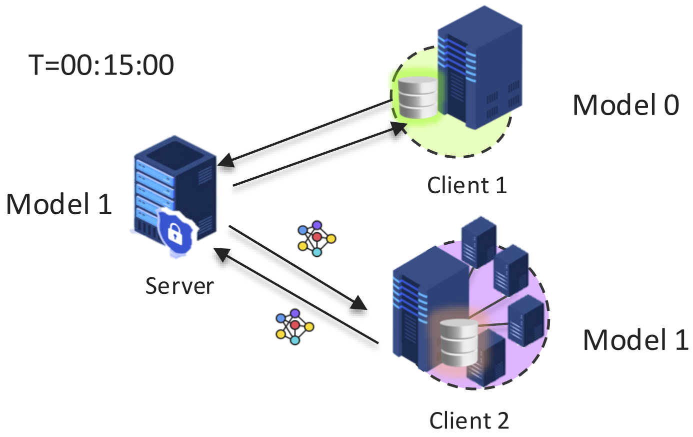


Figure 4: Staleness problem in asynchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- Asynchronous FL – which updates global model immediately once receiving local model from each client – suffers from the stale (outdated) local models from slower clients, thereby causing the global model to drift away from slower clients.

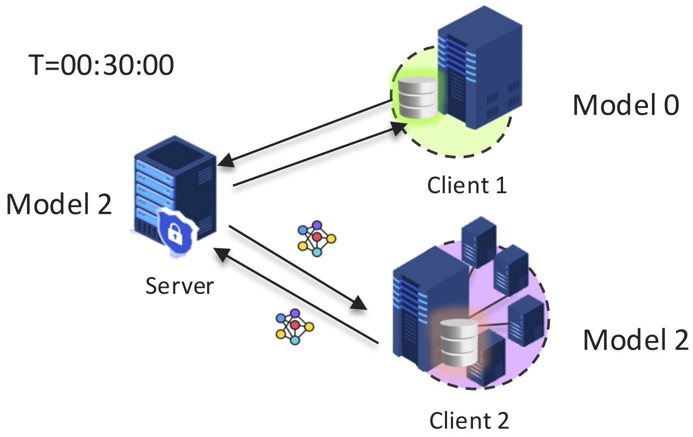


Figure 4: Staleness problem in asynchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- Asynchronous FL – which updates global model immediately once receiving local model from each client – suffers from the stale (outdated) local models from slower clients, thereby causing the global model to drift away from slower clients.

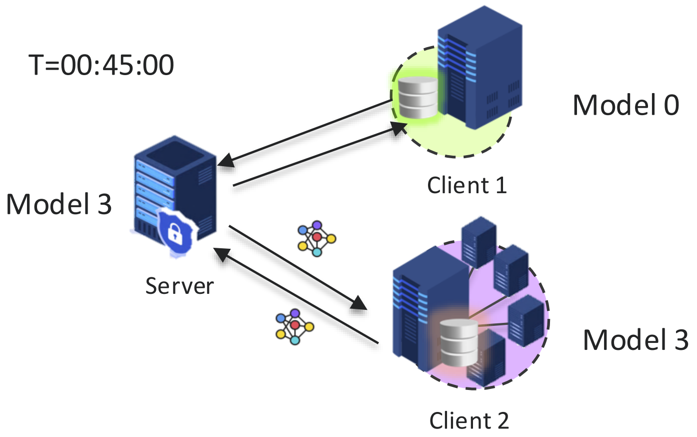


Figure 4: Staleness problem in asynchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- Asynchronous FL – which updates global model immediately once receiving local model from each client – suffers from the stale (outdated) local models from slower clients, thereby causing the global model to drift away from slower clients.

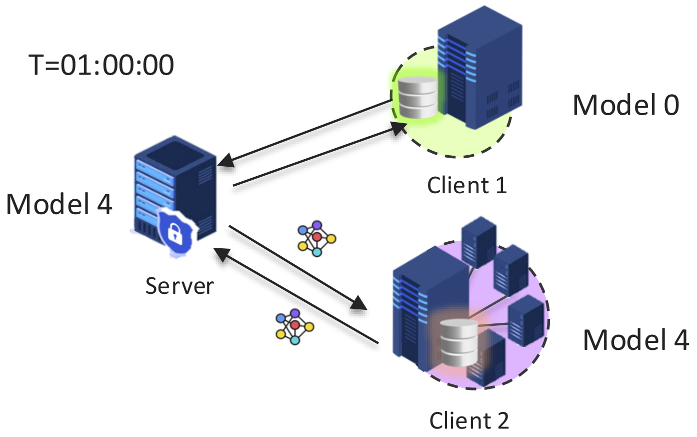
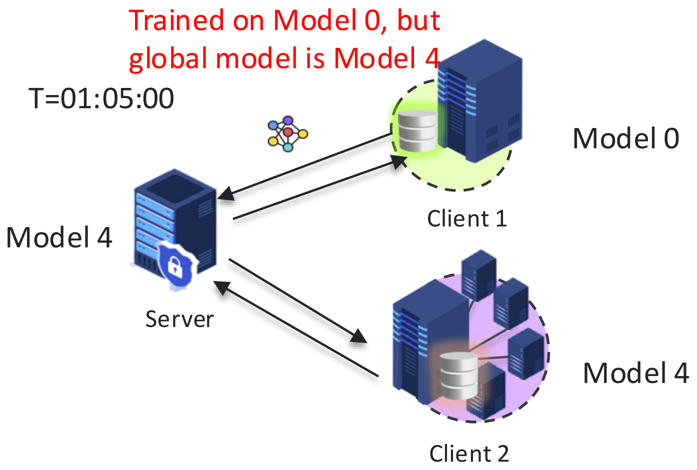


Figure 4: Staleness problem in asynchronous FL.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- Asynchronous FL – which updates global model immediately once receiving local model from each client – suffers from the stale (outdated) local models from slower clients, thereby causing the global model to drift away from slower clients.



Trained on Model 0, but global model is Model 4

Figure 4: Staleness problem in asynchronous FL.

Local model from slower clients are stale/oudated compared to the global model!

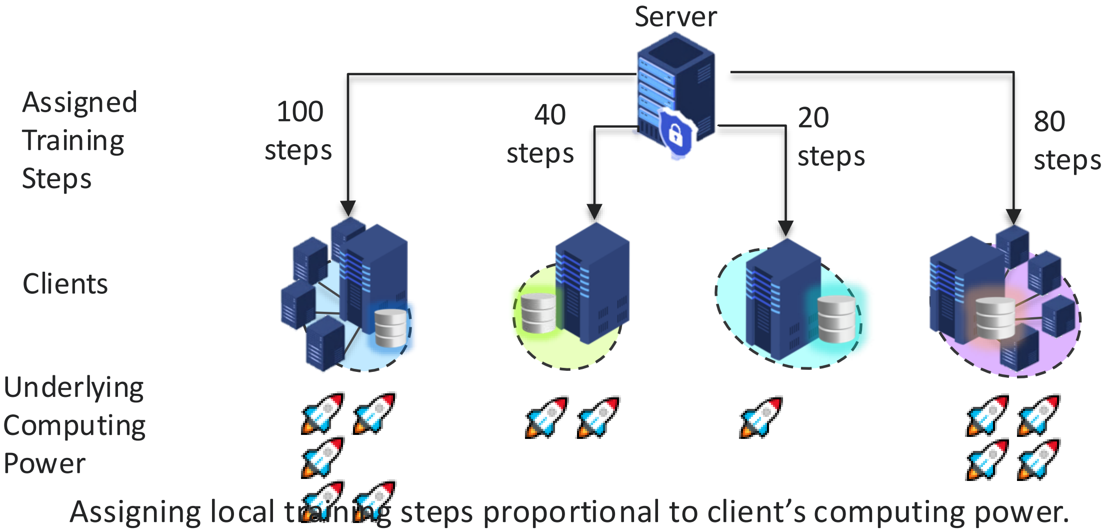
↓

(1) Either be detrimental to global model;
(2) or applying a small importance weight and causing client drift.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity

- “Synchronize” the arrival of clients’ locally trained models
  - by assigning different numbers of local training steps to them
  - according to the clients’ computing power



Server

Assigned Training Steps

100 steps

40 steps

20 steps

80 steps

Clients

Underlying Computing Power

Assigning local training steps proportional to client’s computing power.

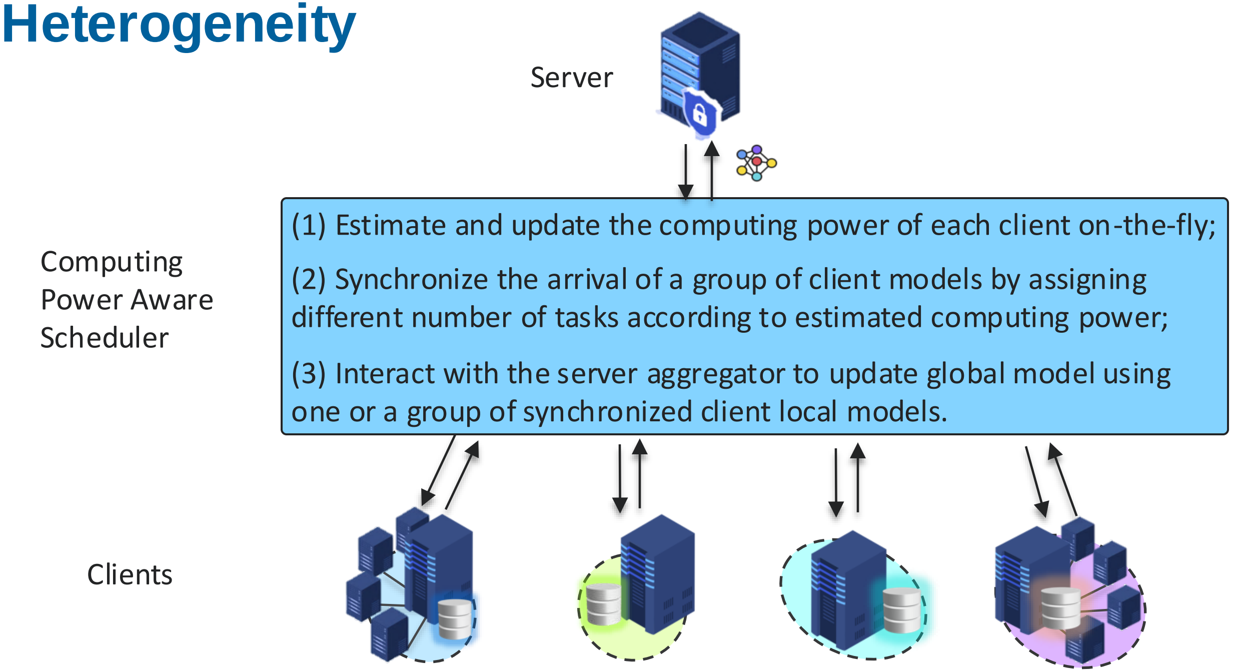
However, in practice

(1) The server does not know the clients’ computing power beforehand;

(2) And the computing power may change during the training.

FEDERATED LEARNING CHALLENGE

Device Heterogeneity



(1) Estimate and update the computing power of each client on-the-fly;

(2) Synchronize the arrival of a group of client models by assigning different number of tasks according to estimated computing power;

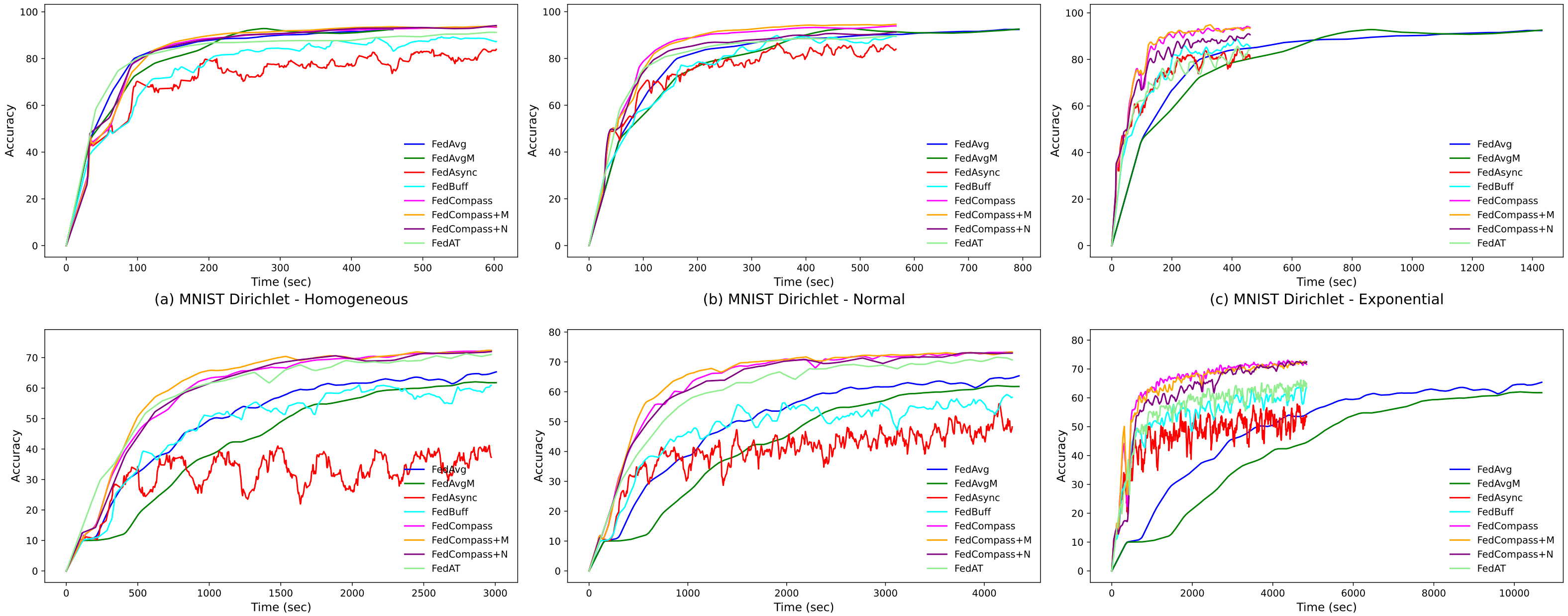
(3) Interact with the server aggregator to update global model using one or a group of synchronized client local models.

FedCompass - Federated learning with a computing power aware scheduler.

Li, Zilinghan, Pranshu Chaturvedi, Shilan He, Han Chen, Gagandeep Singh, Volodymyr Kindratenko, Eliu A. Huerta, Kibaek Kim, and Ravi Madduri. "FedCompass: efficient cross-silo federated learning on heterogeneous client devices using a computing power aware scheduler." *arXiv preprint arXiv:2309.14675* (2023).

FEDERATED LEARNING CHALLENGE

Device Heterogeneity



(a) MNIST Dirichlet - Homogeneous

(b) MNIST Dirichlet - Normal

(c) MNIST Dirichlet - Exponential

(d) CIFAR10 Class - Homogeneous

(e) CIFAR10 Class - Normal

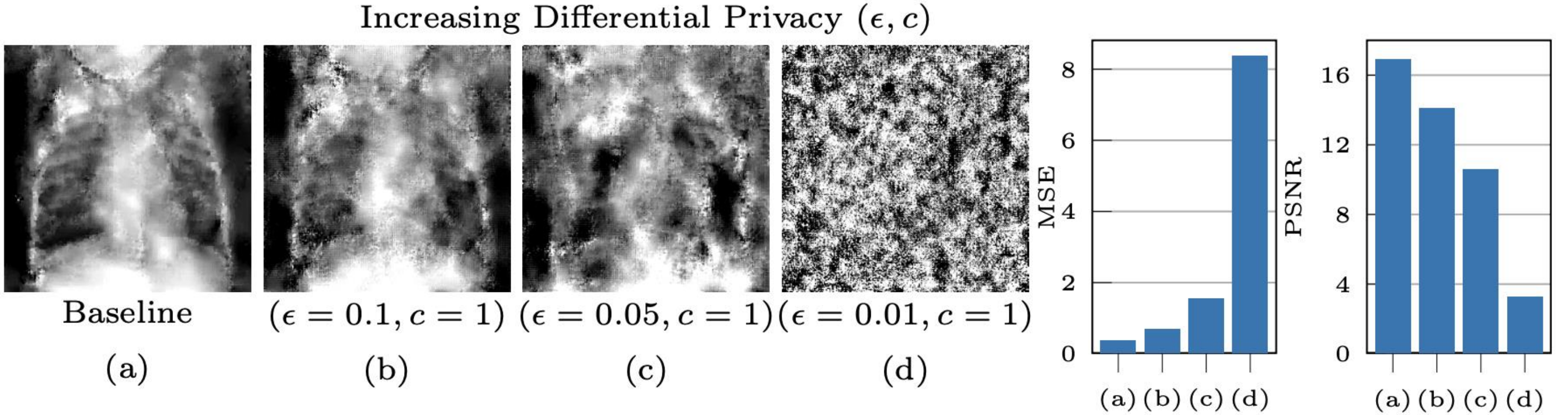
(f) CIFAR10 Class - Exponential

Change in validation accuracy for various FL strategies during the training.

FEDERATED LEARNING CHALLENGE

Security and Privacy

- Federated learning itself is not privacy preserving. The training data can be reversely constructed from model gradients.
- Differential privacy (DP), which adds some noise to model parameters, can significantly increase the difficulty of reconstruction.

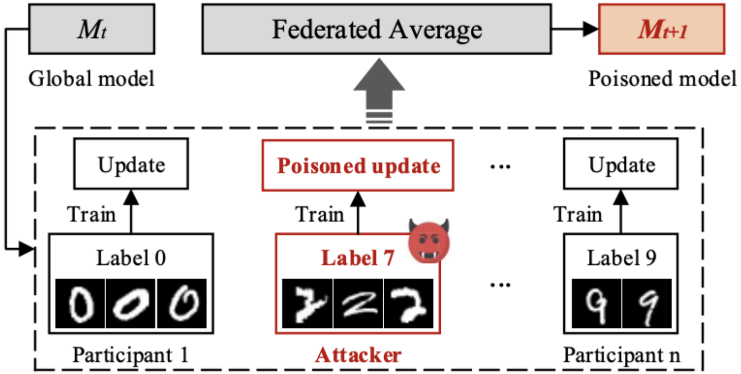


Hoang, Trung-Hieu, Jordan Fuhrman, Ravi Madduri, Miao Li, Pranshu Chaturvedi, Zilinghan Li, Kibaek Kim et al. "Enabling end-to-end secure federated learning in biomedical research on heterogeneous computing environments with APPFLx." *arXiv preprint arXiv:2312.08701* (2023).

FEDERATED LEARNING CHALLENGE

Security and Privacy

- Some clients may try to attack the FL training process by sending poisoned update for aggregation.
- Algorithmic solutions include using a small central validation set and decide whether to drop certain client updates.
- System level, it is important to build a secure and trusted federation.



Zhang, Jiale, Junjun Chen, Di Wu, Bing Chen, and Shui Yu. "Poisoning attack in federated learning using generative adversarial nets." In *2019 18th IEEE international conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*, pp. 374-380. IEEE, 2019.

FEDERATED LEARNING CHALLENGE

Cumbersome Setups

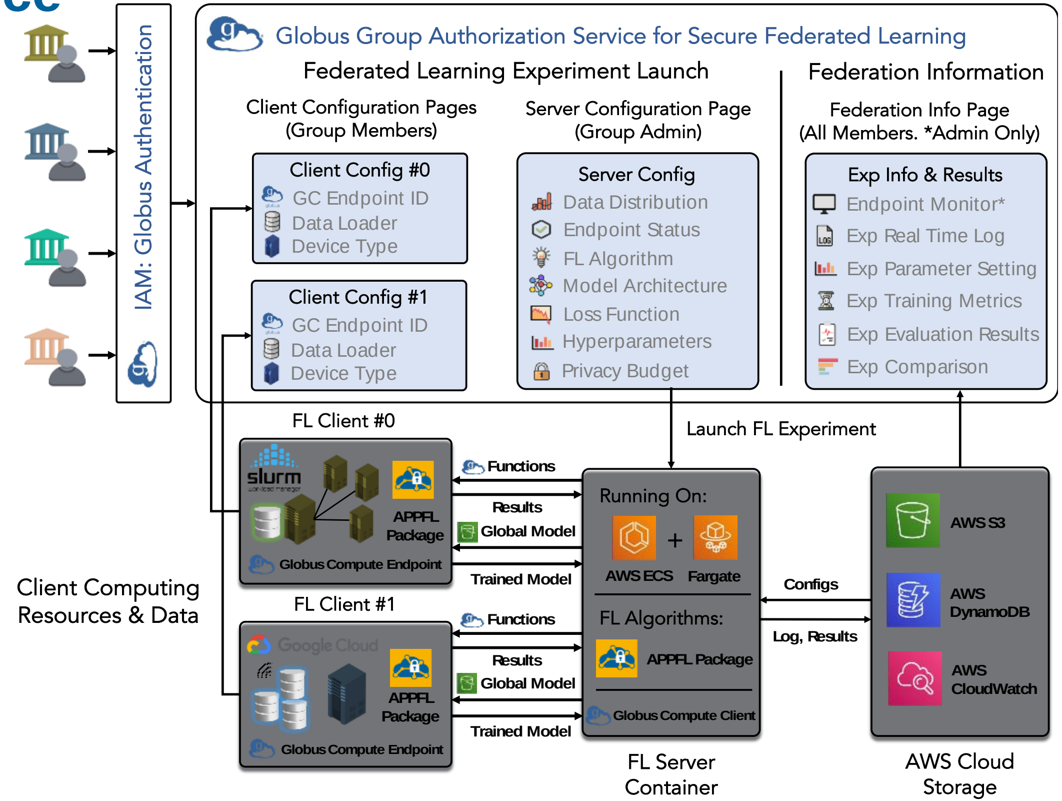
Due to the distributed nature of federated learning, setting up FL experiments can be tedious for domain experts, making the barriers to entry for leveraging FL in their work relatively high.

Our solution – Federated Learning **as a service**

FEDERATED LEARNING CHALLENGE

Federated Learning as a Service

- Login via Globus using institutional credentials
- Create a federation (FL group)
- Invite collaborators using institutional credentials
- Collaborators setup the globus compute endpoint
- Collaborators provide endpoint id and load data loader
- Configure and launch different FL experiments
- Monitor training in real-time, and obtain comprehensive reports
- Reason using data distribution visualization



IAM: Globus Authentication

Globus Group Authorization Service for Secure Federated Learning

Federated Learning Experiment Launch

Client Configuration Pages (Group Members)

Client Config #0
- GC Endpoint ID
- Data Loader
- Device Type

Client Config #1
- GC Endpoint ID
- Data Loader
- Device Type

Server Configuration Page (Group Admin)

Server Config
- Data Distribution
- Endpoint Status
- FL Algorithm
- Model Architecture
- Loss Function
- Hyperparameters
- Privacy Budget

Federation Information

Federation Info Page (All Members. *Admin Only)

Exp Info & Results
- Endpoint Monitor*
- Exp Real Time Log
- Exp Parameter Setting
- Exp Training Metrics
- Exp Evaluation Results
- Exp Comparison

Launch FL Experiment

Client Computing Resources & Data

FL Client #0: slurm workload manager, APPFL Package, Globus Compute Endpoint

FL Client #1: Google Cloud, APPFL Package, Globus Compute Endpoint

Functions
Results
Global Model
Trained Model

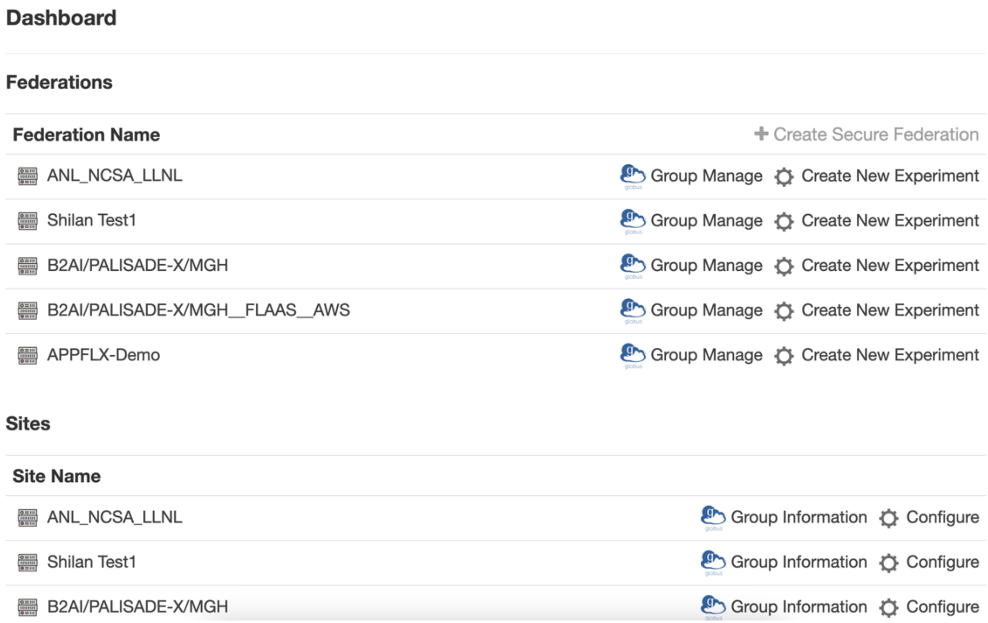
FL Server Container: Running On: AWS ECS + Fargate; FL Algorithms: APPFL Package; Globus Compute Client

Configs
Log, Results

AWS Cloud Storage: AWS S3, AWS DynamoDB, AWS CloudWatch

FEDERATED LEARNING CHALLENGE

Federated Learning as a Service



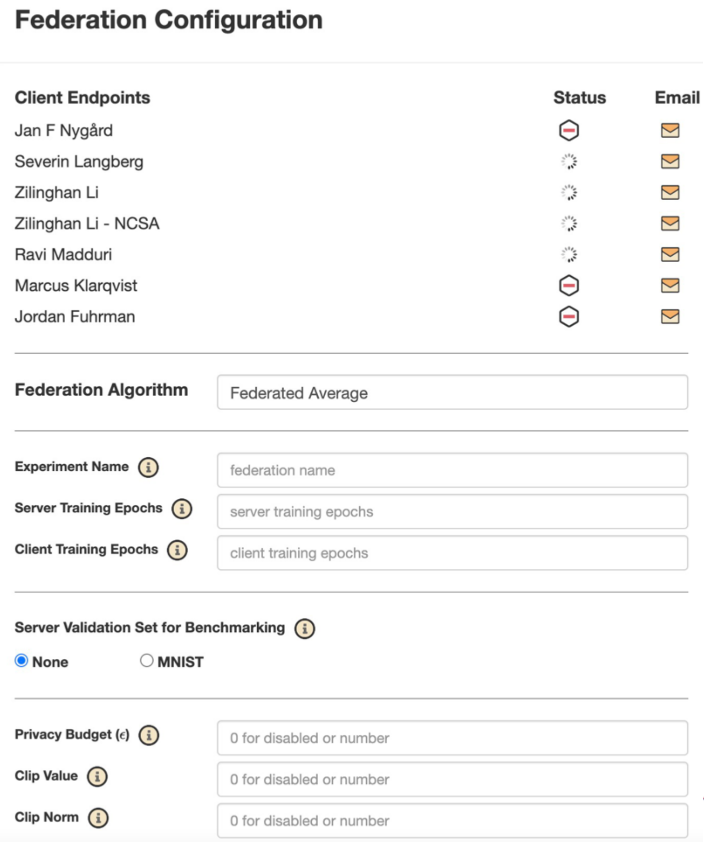
Dashboard

Federations

Federation Name	➕ Create Secure Federation
ANL_NCSA_LLNL	Group Manage ⚙ Create New Experiment
Shilan Test1	Group Manage ⚙ Create New Experiment
B2AI/PALISADE-X/MGH	Group Manage ⚙ Create New Experiment
B2AI/PALISADE-X/MGH__FLAAS__AWS	Group Manage ⚙ Create New Experiment
APPFLX-Demo	Group Manage ⚙ Create New Experiment

Sites

Site Name	
ANL_NCSA_LLNL	Group Information ⚙ Configure
Shilan Test1	Group Information ⚙ Configure
B2AI/PALISADE-X/MGH	Group Information ⚙ Configure



Federation Configuration

Client Endpoints	Status	Email
Jan F Nygård	⊖	✉
Severin Langberg	⁘	✉
Zilinghan Li	⁘	✉
Zilinghan Li - NCSA	⁘	✉
Ravi Madduri	⁘	✉
Marcus Klarqvist	⊖	✉
Jordan Fuhrman	⊖	✉

Federation Algorithm Federated Average

Experiment Name ⓘ federation name

Server Training Epochs ⓘ server training epochs

Client Training Epochs ⓘ client training epochs

Server Validation Set for Benchmarking ⓘ

● **None** ○ **MNIST**

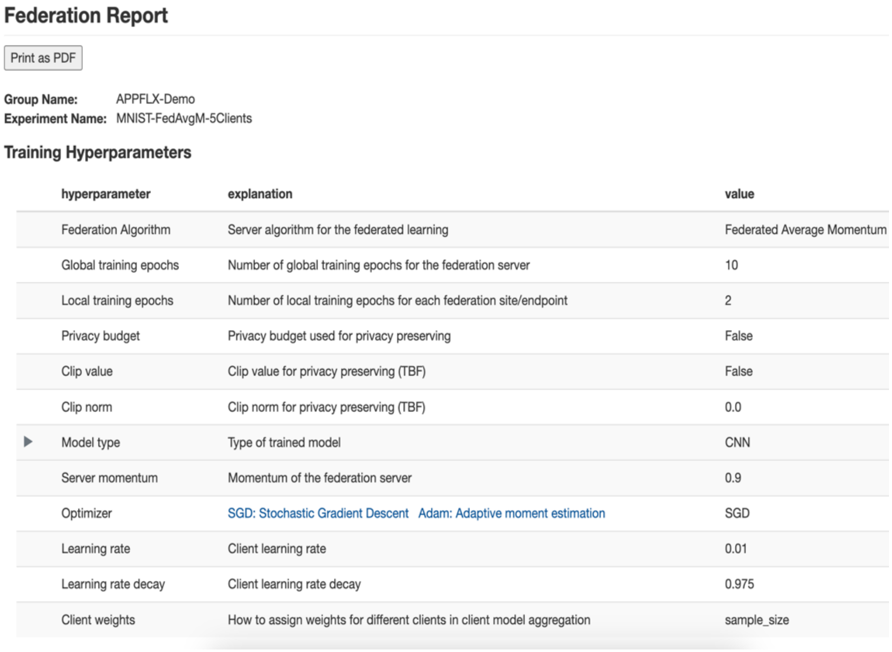
Privacy Budget (ε) ⓘ 0 for disabled or number

Clip Value ⓘ 0 for disabled or number

Clip Norm ⓘ 0 for disabled or number

FEDERATED LEARNING CHALLENGE

Federated Learning as a Service



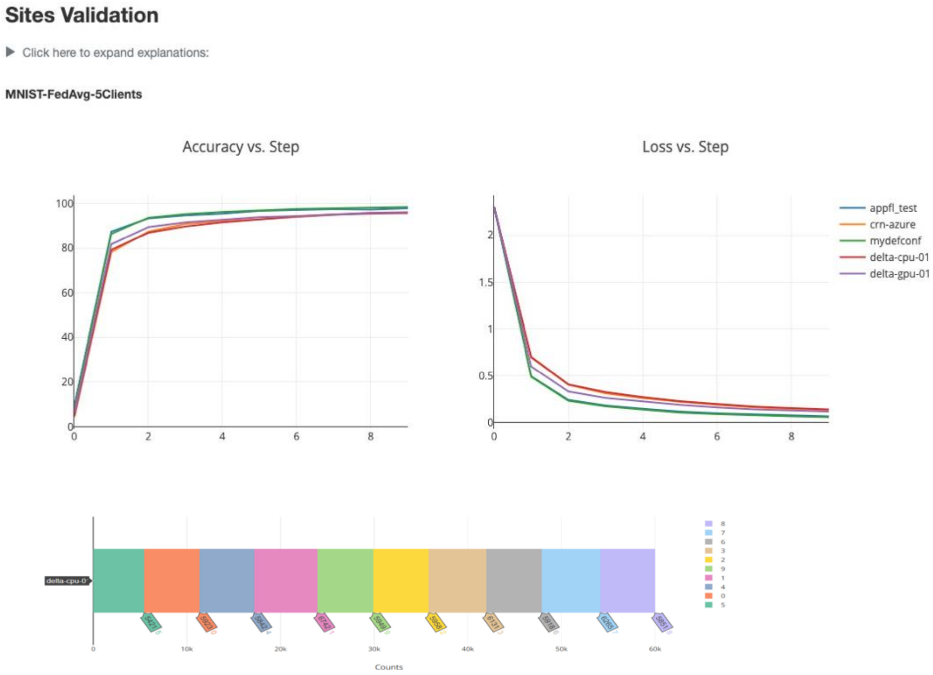
Federation Report

Print as PDF

Group Name: APPFLX-Demo
Experiment Name: MNIST-FedAvgM-5Clients

Training Hyperparameters

	hyperparameter	explanation	value
	Federation Algorithm	Server algorithm for the federated learning	Federated Average Momentum
	Global training epochs	Number of global training epochs for the federation server	10
	Local training epochs	Number of local training epochs for each federation site/endpoint	2
	Privacy budget	Privacy budget used for privacy preserving	False
	Clip value	Clip value for privacy preserving (TBF)	False
	Clip norm	Clip norm for privacy preserving (TBF)	0.0
▶	Model type	Type of trained model	CNN
	Server momentum	Momentum of the federation server	0.9
	Optimizer	SGD: Stochastic Gradient Descent Adam: Adaptive moment estimation	SGD
	Learning rate	Client learning rate	0.01
	Learning rate decay	Client learning rate decay	0.975
	Client weights	How to assign weights for different clients in client model aggregation	sample_size



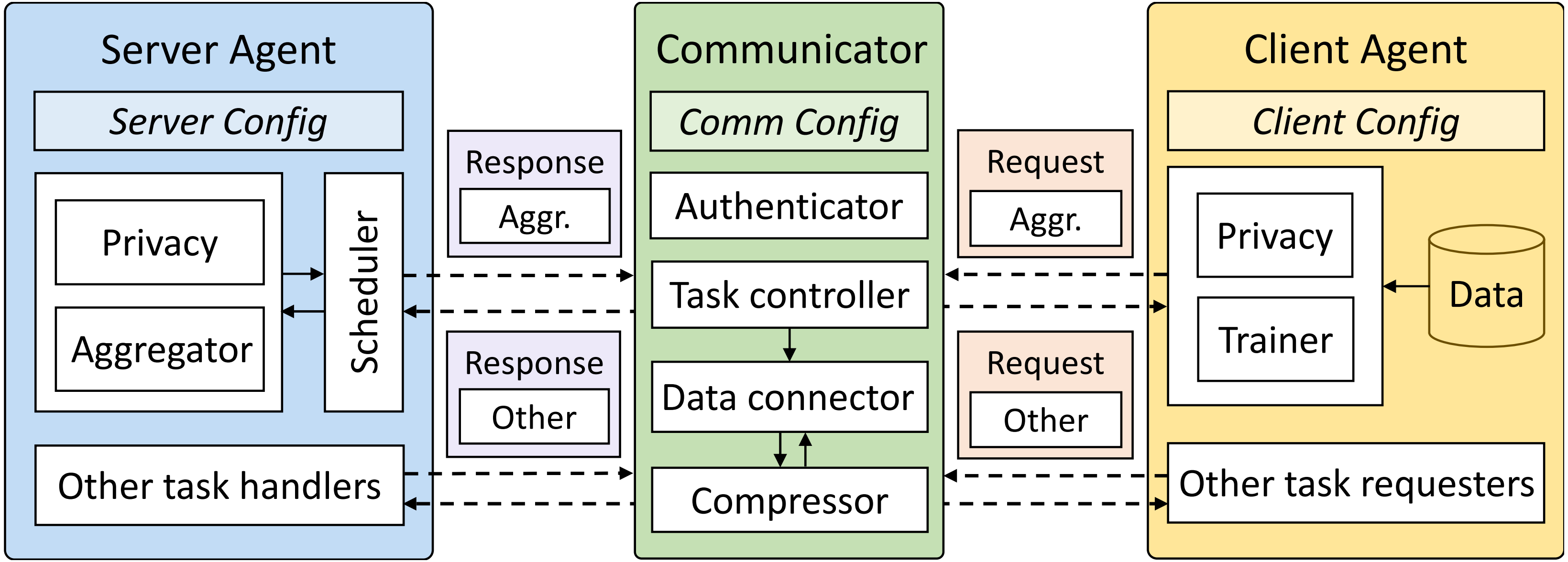
Sites Validation

▶ Click here to expand explanations:

MNIST-FedAvg-5Clients

FEDERATED LEARNING FRAMEWORK

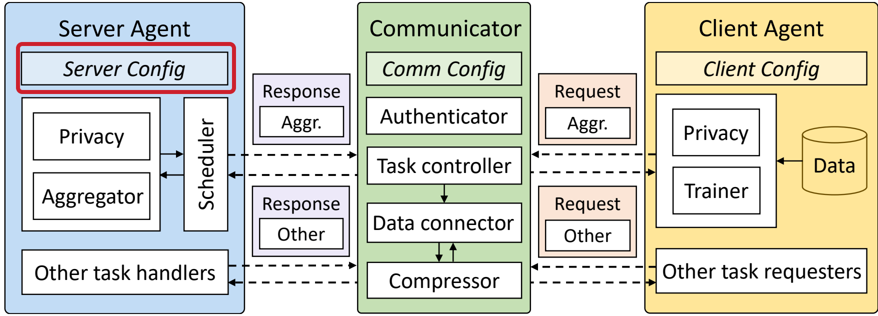
Advanced Privacy Preserving Federated Learning Framework



https://github.com/APPFL/appfl-scifm

APPFL

Server Config



- Server Config is a YAML file composed of two main parts:
  - client_configs: configurations that the user want to **share among all clients**, e.g., model arch, trainer type, learning rate, local steps, etc., which will be shared with all clients at the beginning of the FL experiments
  - server_configs: configurations for the server, e.g., scheduler and aggregator type, etc.

🍀 Having the server specify all shared client-side configurations makes the experiment coordination easier.

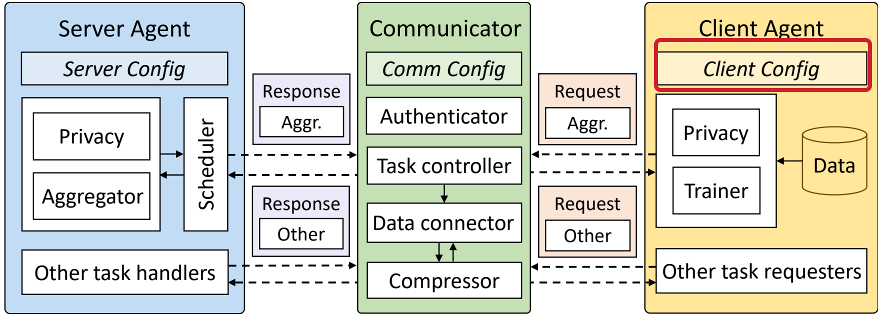
💡 This YAML formatted configuration also makes life easier to parse the configuration collected from the web service.

```
# Configurations shared among clients
client_configs:
  train_configs:
    trainer: "NaiveTrainer"
    mode: "step”
    ......
    # Loss function
        loss_fn_path: "./loss/celoss.py"
        loss_fn_name: "CELoss”
        ......
  model_configs:
        model_path: "./model/cnn.py"
        model_name: "CNN"
        model_kwargs:
          ......
  comm_configs:
    compressor_configs:
      ......
# Configurations for the server
server_configs:
  scheduler: "SyncScheduler"
  aggregator: "FedAvgAggregator”
  ......
```

APPFL

Client Config

- Client Config is another YAML file which contains the configurations **specific to** one certain client, e.g, dataloader to sensitive local data, device type, and logging settings.



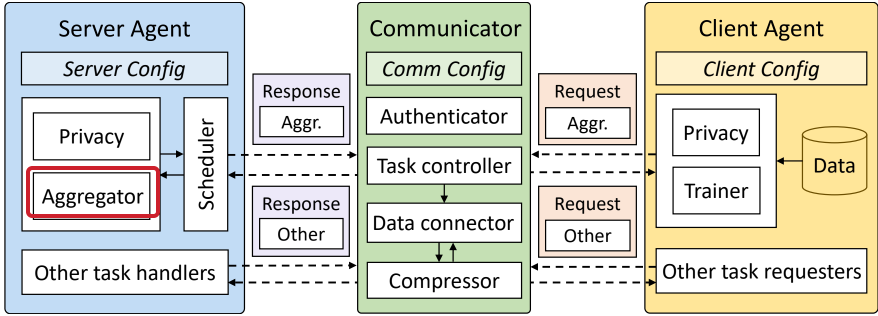
```
# Configurations for training
train_configs:
  device: "cpu"
  logging_id: "Client1"
  logging_output_dirname: "./output"
  logging_output_filename: "result"
# Configurations for dataloader
data_configs:
  dataset_path: "./dataset/mnist_dataset.py"
  dataset_name: "get_mnist"
  dataset_kwargs:
    ......
# Configurations for communicator
comm_configs:
  grpc_configs:
         server_uri: localhost:50051
         max_message_size: 1048576
         use_ssl: False
```

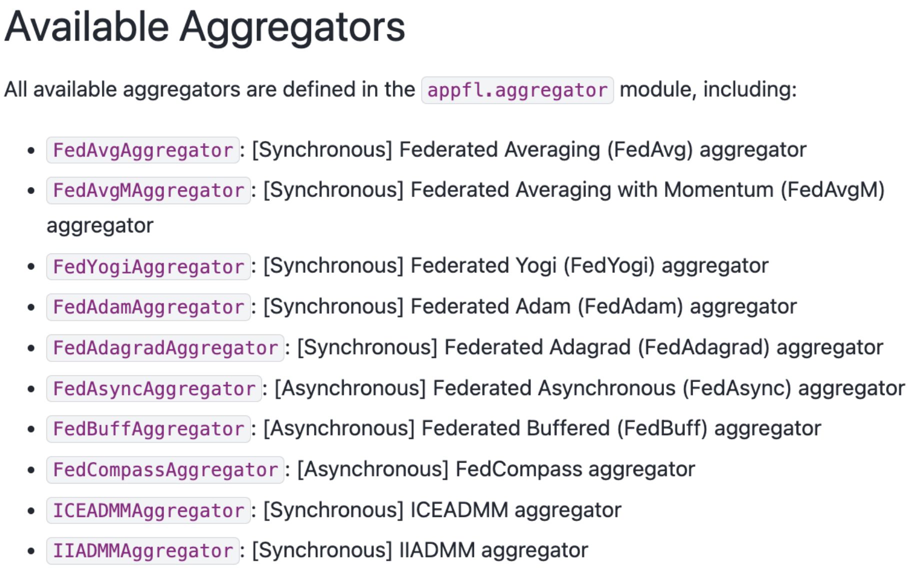
APPFL

Aggregator

The aggregator is used for aggregating one or more client local model(s) to update the global model. Depending on the synchroneity of the FL algorithm, the aggregator can take

- Only one client model to update the global model (for async FL)
- A list of client models to update the global model (for sync FL)
- One or a list of client models to update the global model (for async FL)





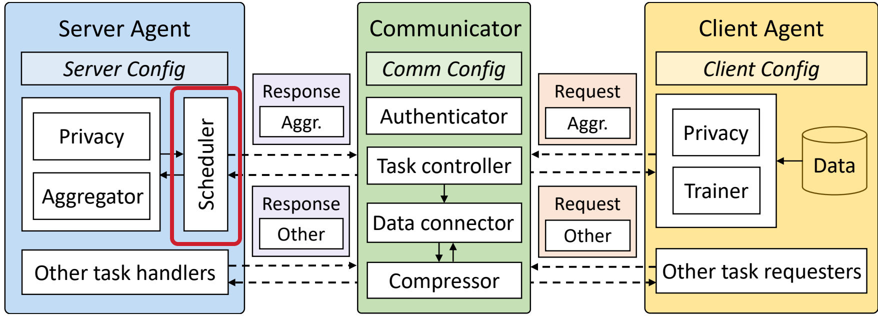
Available Aggregators

All available aggregators are defined in the `appfl.aggregator` module, including:

- `FedAvgAggregator`: [Synchronous] Federated Averaging (FedAvg) aggregator
- `FedAvgMAggregator`: [Synchronous] Federated Averaging with Momentum (FedAvgM) aggregator
- `FedYogiAggregator`: [Synchronous] Federated Yogi (FedYogi) aggregator
- `FedAdamAggregator`: [Synchronous] Federated Adam (FedAdam) aggregator
- `FedAdagradAggregator`: [Synchronous] Federated Adagrad (FedAdagrad) aggregator
- `FedAsyncAggregator`: [Asynchronous] Federated Asynchronous (FedAsync) aggregator
- `FedBuffAggregator`: [Asynchronous] Federated Buffered (FedBuff) aggregator
- `FedCompassAggregator`: [Asynchronous] FedCompass aggregator
- `ICEADMMAggregator`: [Synchronous] ICEADMM aggregator
- `IIADMMAggregator`: [Synchronous] IIADMM aggregator

APPFL

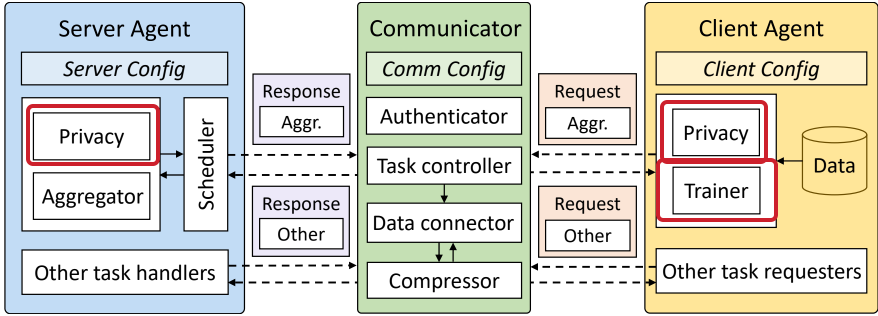
Scheduler



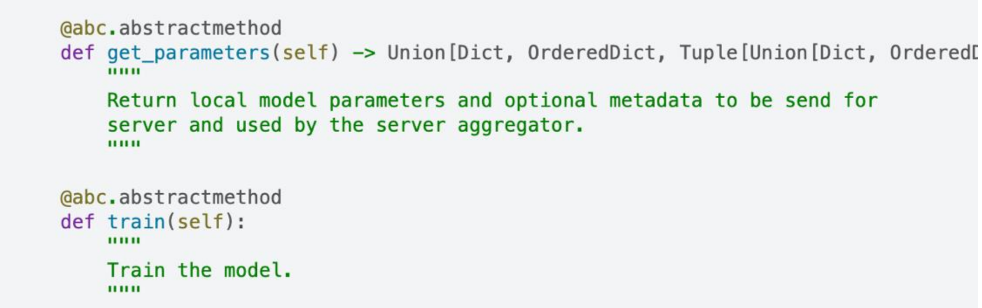
- APPFL scheduler is the **interface** between the **communicator** and the **aggregator**. Whenever the communicator receives the local model from a single client, it directly hands the local model to the scheduler, and the scheduler will decide when to pass the local model(s) to the aggregator for updating the global model. Currently, APPFL supports three scheduler:
  - Synchronous Scheduler (cache the models and pass all models after receiving all)
  - Naïve Asynchronous Scheduler (pass model once receiving it)
  - Compass Asynchronous Scheduler

APPFL

Trainer and Privacy



- The trainer trains the local model using the local dataloader. It needs to define two functions:
  - train(): train the local models
  - get_parameters(): return the parameters that are needed by the server-side aggregator (model state dict/model gradients/prime and dual states, etc.)



```
@abc.abstractmethod
def get_parameters(self) -> Union[Dict, OrderedDict, Tuple[Union[Dict, OrderedD
    """
    Return local model parameters and optional metadata to be send for
    server and used by the server aggregator.
    """

@abc.abstractmethod
def train(self):
    """
    Train the model.
    """
```

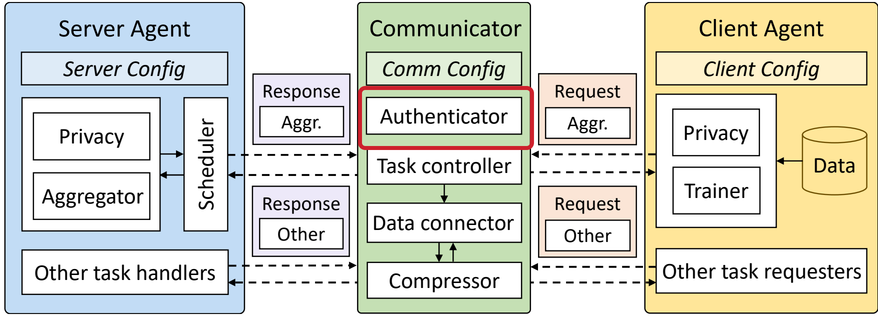
- The privacy module contains various privacy preservation algorithms to prevent training data reconstruction.

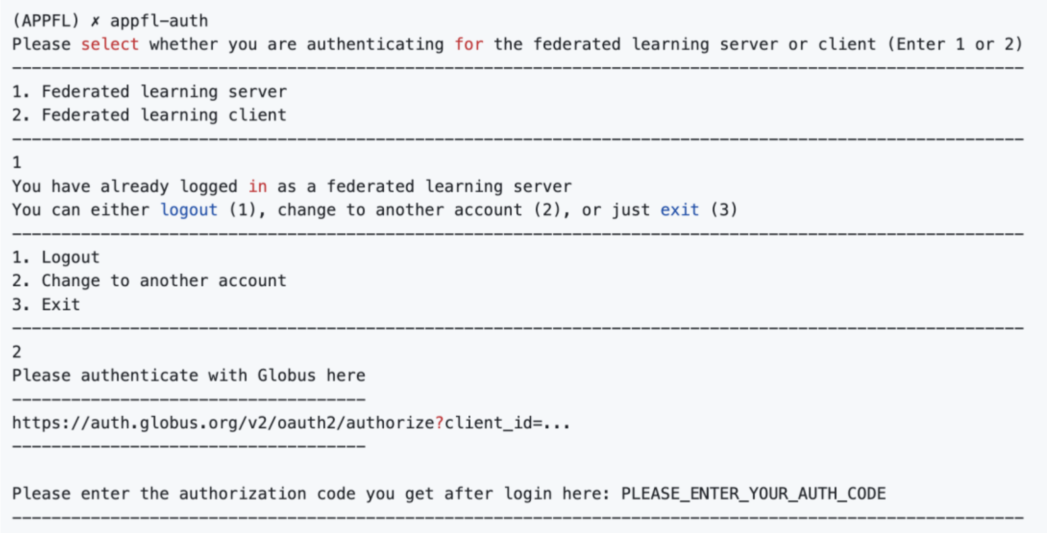
APPFL

Authenticator

globus

- The authenticator module helps the creation of a trusted federation to ensure the security of federated learning experiments.
- APPFL currently support Globus authenticator
- User can easily sign in using the APPFL CLI.
- Only clients within the same Globus group are authorized to join the FL experiments.





```
(APPFL) ✗ appfl-auth
Please select whether you are authenticating for the federated learning server or client (Enter 1 or 2)
----------------------------------------------------------------------------------------------------------
1. Federated learning server
2. Federated learning client
----------------------------------------------------------------------------------------------------------
1
You have already logged in as a federated learning server
You can either logout (1), change to another account (2), or just exit (3)
----------------------------------------------------------------------------------------------------------
1. Logout
2. Change to another account
3. Exit
----------------------------------------------------------------------------------------------------------
2
Please authenticate with Globus here
------------------------------------
https://auth.globus.org/v2/oauth2/authorize?client_id=...
------------------------------------

Please enter the authorization code you get after login here: PLEASE_ENTER_YOUR_AUTH_CODE
----------------------------------------------------------------------------------------------------------
```

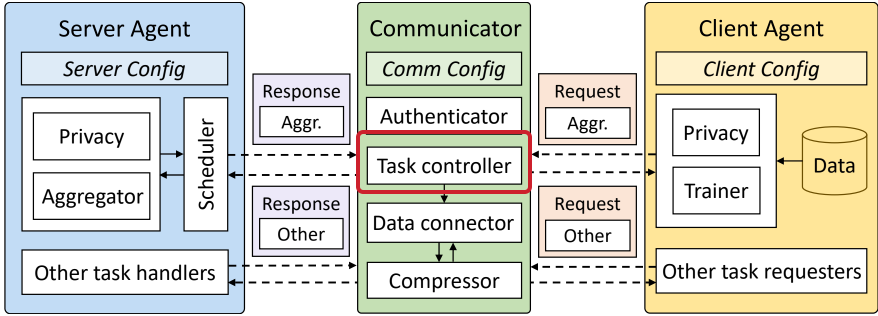
Tuecke, Steven, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. "Globus Auth: A research identity and access management platform." In *2016 IEEE 12th International Conference on e-Science (e-Science)*, pp. 203-212. IEEE, 2016.

APPFL

Task Controller

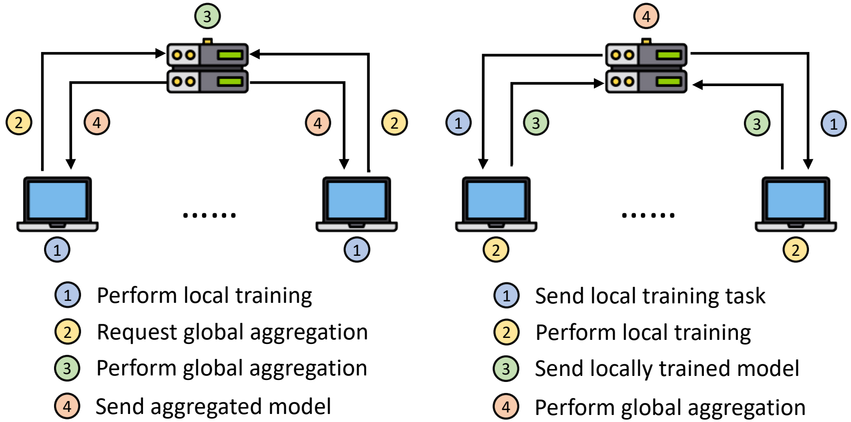
Two types of communications of task control signals:

- Client driven
- Server driven



APPFL supports:

- gRPC for client driven communication
- Globus Compute for server driven communication



1 Perform local training
2 Request global aggregation
3 Perform global aggregation
4 Send aggregated model

1 Send local training task
2 Perform local training
3 Send locally trained model
4 Perform global aggregation

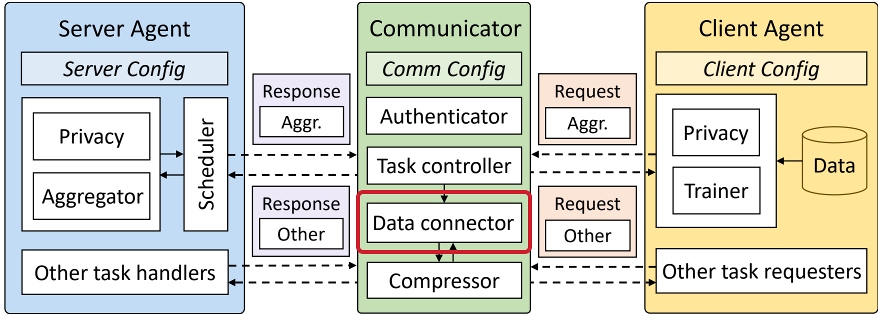
(a) Client-driven communication

(b) Server-driven communication

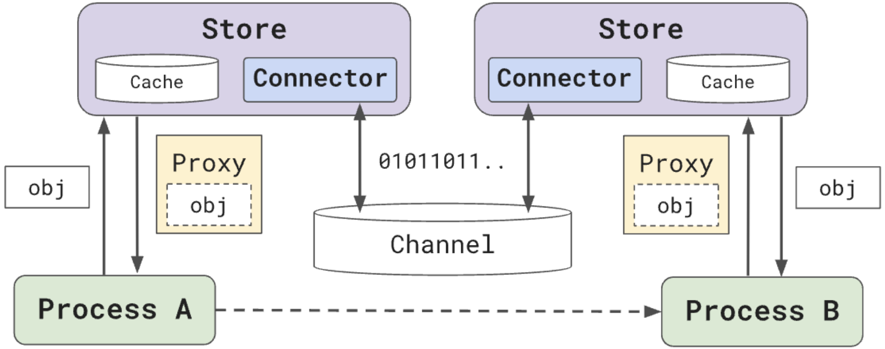
Chard, Ryan, Yadu Babuji, Zhuozhao Li, Tyler Skluzacek, Anna Woodard, Ben Blaiszik, Ian Foster, and Kyle Chard. "Funcx: A federated function serving fabric for science." In *Proceedings of the 29th International symposium on high-performance parallel and distributed computing*, pp. 65-76. 2020.

APPFL

Data Connector ProxyStore



APPFL integrates with ProxyStore to enable users to transfer large model parameters using various data connectors (e.g., Globus Transfer, Relay Server, AWS S3, etc.)

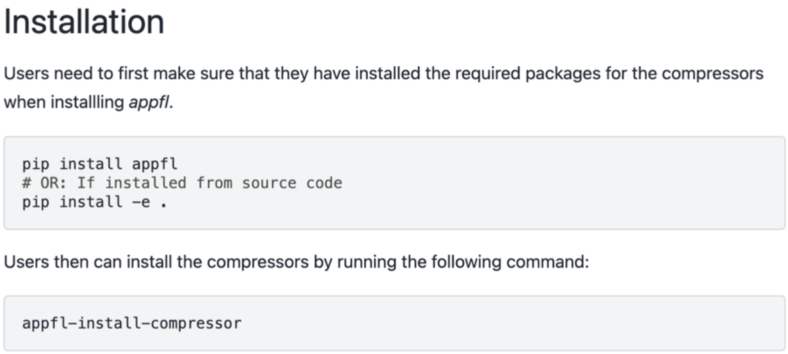


Pauloski, J. Gregory, Valerie Hayot-Sasson, Logan Ward, Nathaniel Hudson, Charlie Sabino, Matt Baughman, Kyle Chard, and Ian Foster. "Accelerating communications in federated applications with transparent object proxies." In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-15. 2023.

APPFL

Compressor

- Compressor is seamlessly integrated in APPFL to do lossy compression for the transferred model parameters.
- Easy installation
- Compressor can be used in a standalone manner as well.



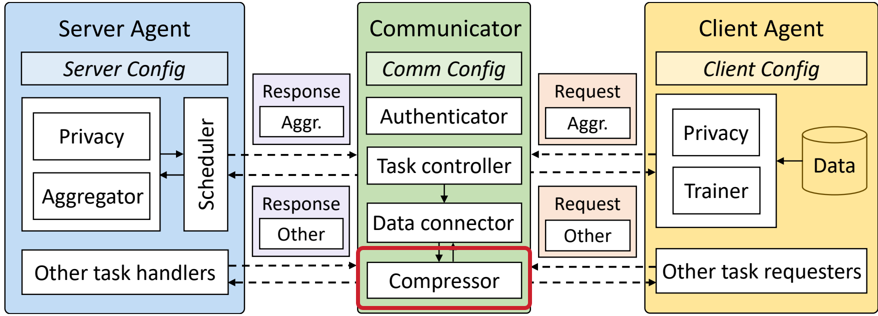
Installation

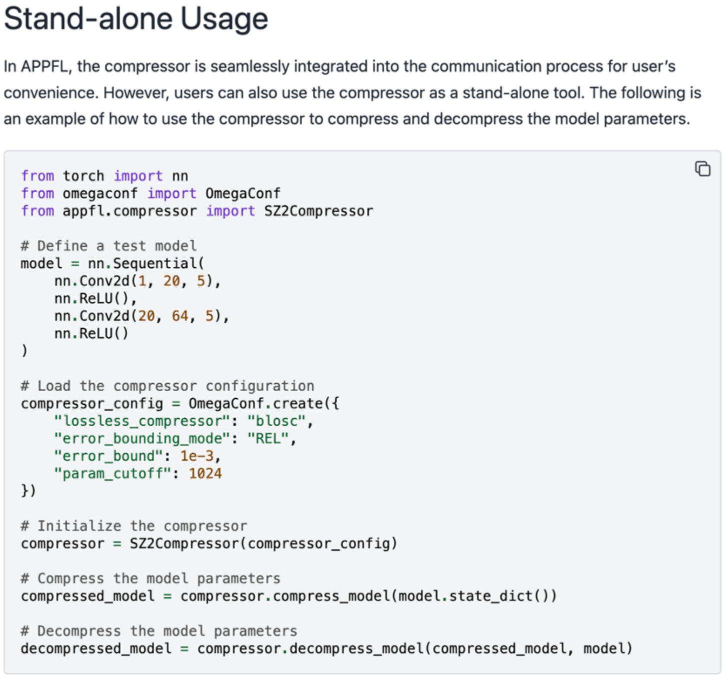
Users need to first make sure that they have installed the required packages for the compressors when installling *appfl*.

```
pip install appfl
# OR: If installed from source code
pip install -e .
```

Users then can install the compressors by running the following command:

```
appfl-install-compressor
```





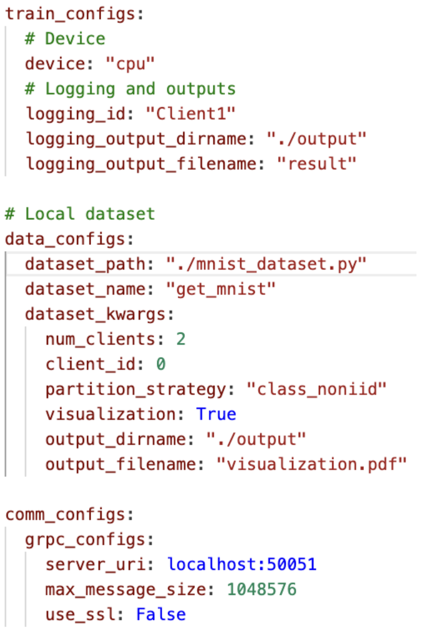
Stand-alone Usage

In APPFL, the compressor is seamlessly integrated into the communication process for user's convenience. However, users can also use the compressor as a stand-alone tool. The following is an example of how to use the compressor to compress and decompress the model parameters.

```
from torch import nn
from omegaconf import OmegaConf
from appfl.compressor import SZ2Compressor

# Define a test model
model = nn.Sequential(
    nn.Conv2d(1, 20, 5),
    nn.ReLU(),
    nn.Conv2d(20, 64, 5),
    nn.ReLU()
)

# Load the compressor configuration
compressor_config = OmegaConf.create({
    "lossless_compressor": "blosc",
    "error_bounding_mode": "REL",
    "error_bound": 1e-3,
    "param_cutoff": 1024
})

# Initialize the compressor
compressor = SZ2Compressor(compressor_config)

# Compress the model parameters
compressed_model = compressor.compress_model(model.state_dict())

# Decompress the model parameters
decompressed_model = compressor.decompress_model(compressed_model, model)
```

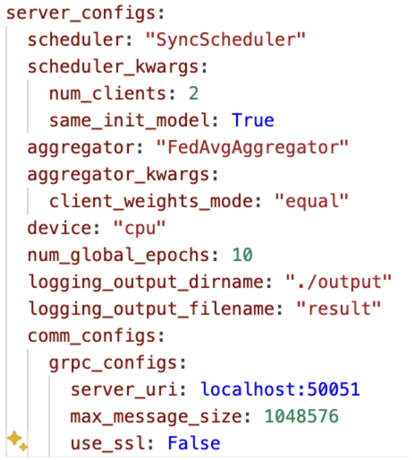
DEMO

https://github.com/APPFL/appfl-scifm



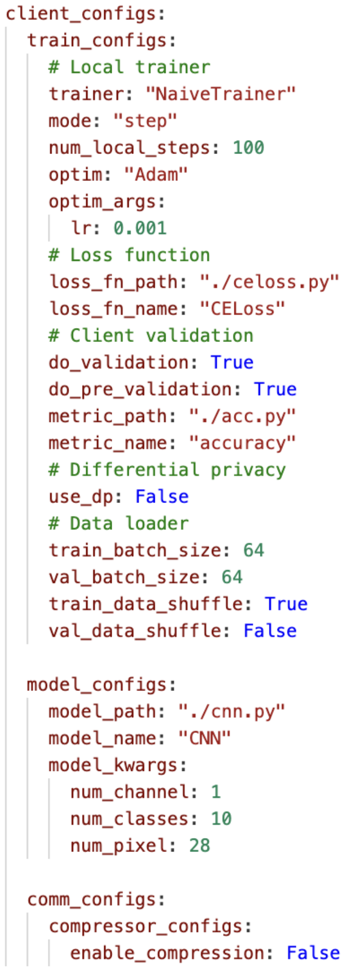
```yaml
train_configs:
  # Device
  device: "cpu"
  # Logging and outputs
  logging_id: "Client1"
  logging_output_dirname: "./output"
  logging_output_filename: "result"

# Local dataset
data_configs:
  dataset_path: "./mnist_dataset.py"
  dataset_name: "get_mnist"
  dataset_kwargs:
    num_clients: 2
    client_id: 0
    partition_strategy: "class_noniid"
    visualization: True
    output_dirname: "./output"
    output_filename: "visualization.pdf"

comm_configs:
  grpc_configs:
    server_uri: localhost:50051
    max_message_size: 1048576
    use_ssl: False
```

client_config.yaml



```yaml
server_configs:
  scheduler: "SyncScheduler"
  scheduler_kwargs:
    num_clients: 2
    same_init_model: True
  aggregator: "FedAvgAggregator"
  aggregator_kwargs:
    client_weights_mode: "equal"
  device: "cpu"
  num_global_epochs: 10
  logging_output_dirname: "./output"
  logging_output_filename: "result"
  comm_configs:
    grpc_configs:
      server_uri: localhost:50051
      max_message_size: 1048576
      use_ssl: False
```

server_config.yaml

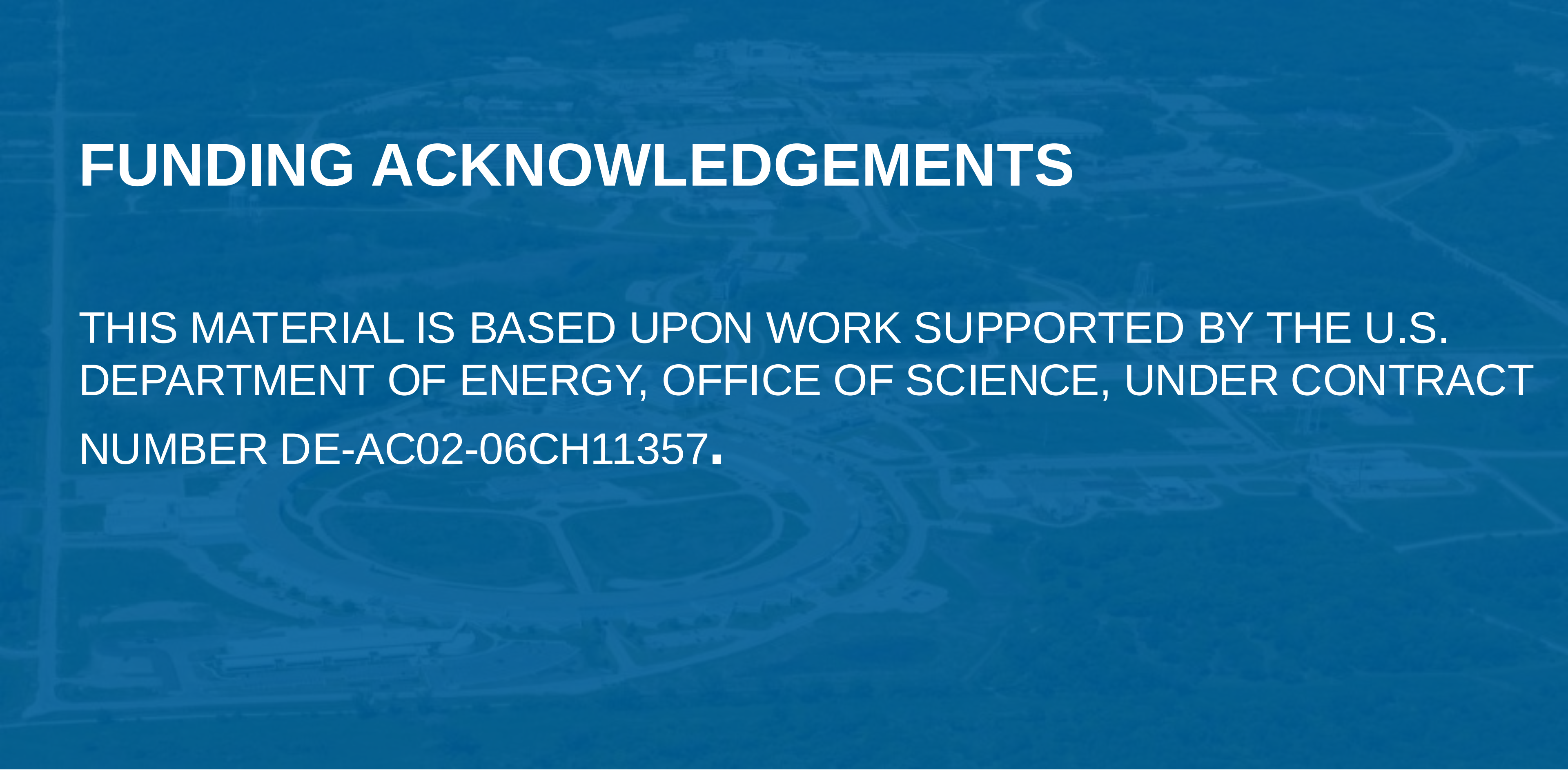


```yaml
client_configs:
  train_configs:
    # Local trainer
    trainer: "NaiveTrainer"
    mode: "step"
    num_local_steps: 100
    optim: "Adam"
    optim_args:
      lr: 0.001
    # Loss function
    loss_fn_path: "./celoss.py"
    loss_fn_name: "CELoss"
    # Client validation
    do_validation: True
    do_pre_validation: True
    metric_path: "./acc.py"
    metric_name: "accuracy"
    # Differential privacy
    use_dp: False
    # Data loader
    train_batch_size: 64
    val_batch_size: 64
    train_data_shuffle: True
    val_data_shuffle: False

  model_configs:
    model_path: "./cnn.py"
    model_name: "CNN"
    model_kwargs:
      num_channel: 1
      num_classes: 10
      num_pixel: 28

  comm_configs:
    compressor_configs:
      enable_compression: False
```

DEMO

https://github.com/APPFL/appfl-scifm

```
git clone https://github.com/APPFL/appfl-scifm.git
cd appfl-scifm
pip install -e ".[examples]"
cd demo
chmod +x run.sh
./run.sh
```



FUNDING ACKNOWLEDGEMENTS

THIS MATERIAL IS BASED UPON WORK SUPPORTED BY THE U.S. DEPARTMENT OF ENERGY, OFFICE OF SCIENCE, UNDER CONTRACT NUMBER DE-AC02-06CH11357.

Q&A

https://github.com/APPFL/APPFL/tree/thoang/funcx