# Towards Federated Learning using FaaS Fabric

Mohak Chadha
mohak.chadha@tum.de
Technische Universität München
Garching (near Munich), Germany

Anshul Jindal
anshul.jindal@tum.de
Technische Universität München
Garching (near Munich), Germany

Michael Gerndt
gerndt@in.tum.de
Technische Universität München
Garching (near Munich), Germany

Mohammed Alshayeb

Date: 12/8/2022

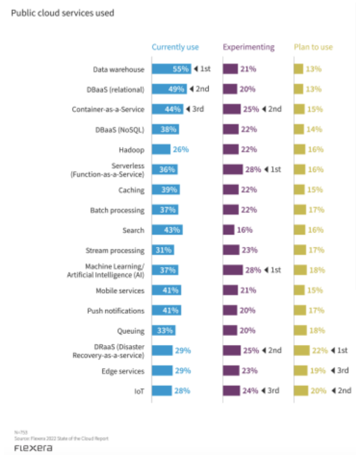
1

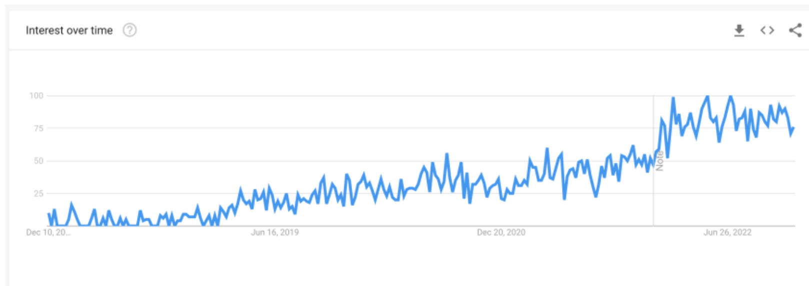
## Outlines

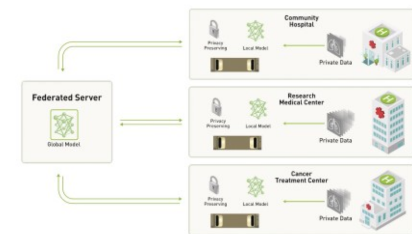
- Motivation
- Background
- Goals
- System Design
- Experimental Setup
- Results
- Conclusion and Future Work
- Critic

2

# Motivation



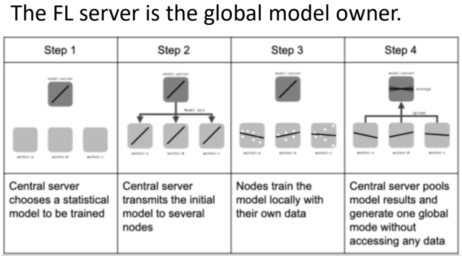

A centralized-server approach to federated learning.

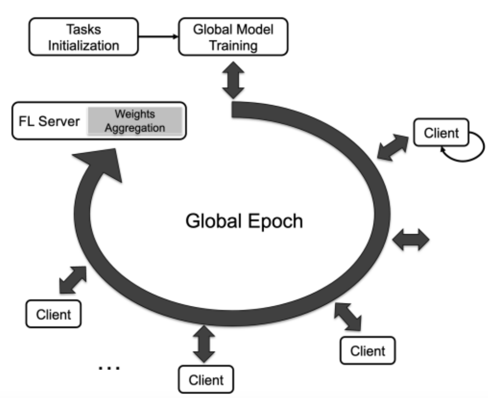
keeping the training data local providing
- privacy,
- security,
- and economic benefits.

3

# Background

- The objective of the standard FL problem is to learn a single ML or DNN model from decentralized data stored on multiple remote clients
- A key property of the FL problem is that the training data present on each client does not represent the population distribution
- Clients are data owners that participate in a particular round of the FL training process
- The FL server is the global model owner.




4

# Function-as-a-Service Fabric

- Combine resources from FaaS platforms deployed on heterogeneous devices to support invocation of each other's functions as *Function-as-a-Service fabric*.
- Authors utilize three FaaS platforms, i.e., OpenWhisk, OpenFaaS, and GCF, shown in Figure 1 as FaaS fabric.
- Authors provide a shared model for heterogeneous devices combining resource-constrained edge devices with the cloud to enable the efficient management of FL-clients.

Mohak Chadha et al.

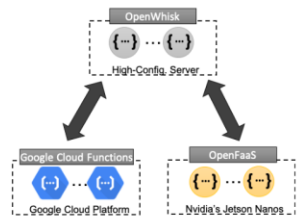



**Figure 1.** Combination of three FaaS platforms deployed on heterogeneous devices.

5

# Goals

- Extension of FaaS to multiple heterogeneous FaaS platforms.
- Enabling Federated Learning using Serverless Computing.
- Ease of use.

6

# System Design

- *FedKeeper*1 is a client-based python tool for propagating FL-client functions over FaaS fabric. It's main objective is to act as a manager or keeper of various client functions distributed over different FaaS platforms.
  - Facilitating the automatic creation, deletion, and invocation of FL-client functions for each FaaS platform.
  - *FedKeeper* keeps track of the functions running on each FaaS platform using activation IDs and automatically creates or invokes the functions which have stopped or failed.
- It consists of several sub components, i.e., *Client Register*, *Weights-Updater*, *Client-Invoker,* and the *FL-Server*.

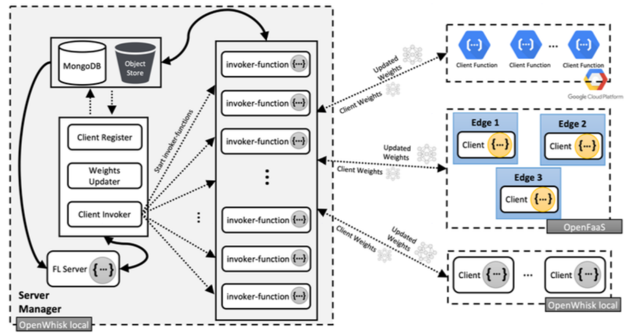



**Figure 2.** High-level architecture for Federated Learning over FaaS Fabric.

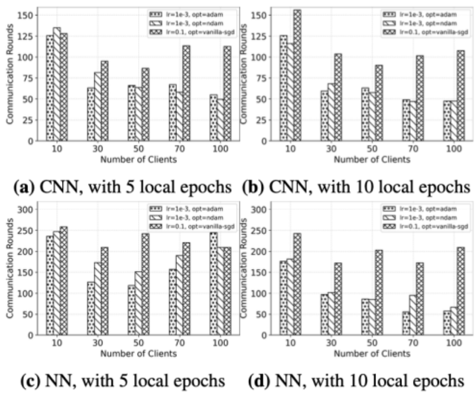


# Experimental Setup

- **OpenWhisk (OW)**
  - Deployed over a single node Kubernetes Cluster (On-premise)
  - Two sockets, Intel Cascade Lake-SP, 22 cores each
- **OpenFaaS (OF)**
  - Edge Cluster with 3 Nvidia Jetson Nano Devices (On-premise)
  - K3s (lightweight Kubernetes) as the container-orchestration system
- **Google Cloud Functions (GCF)**
- Each platform runs Tensorflow
- Evaluation on a Image Classification Task
- Two architectures:
  - 2-layer fully connected NN
  - CNN – convolutional neural net

| Configuration | OW | OF | GCF |
|---|---|---|---|
| Memory | 2 GiB | 2 GiB | 2 GiB |
| FL-Clients | 7 | 3 | 93 |

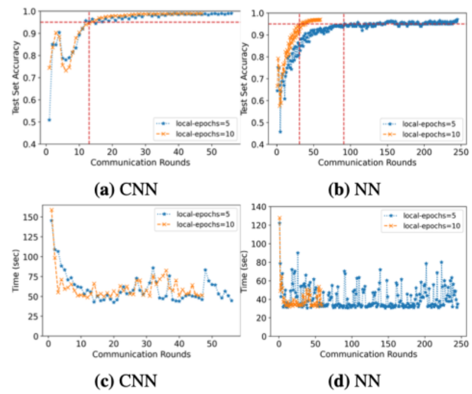
# Results




(a) CNN, with 5 local epochs (b) CNN, with 10 local epochs

(c) NN, with 5 local epochs (d) NN, with 10 local epochs

**Figure 3.** The number of communication rounds required for reaching 99% and 97% test set accuracy on the MNIST dataset for the two different architectures with a varying number of clients, different local computation, and optimizers.




(a) CNN (b) NN

(c) CNN (d) NN

**Figure 4.** The test set accuracy on the MNIST dataset and the average time across each communication round for the two network architectures for 100 clients, with different local computation and *adam* as the optimizer.

9

# Time Distribution

WoSC '20, December 7–11, 2020, Delft, Netherlands

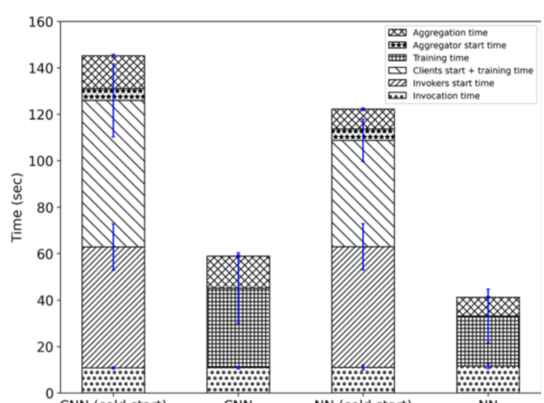



**Figure 5.** Time distribution for the two network architectures for 100 clients with 5 local epochs and *adam* as the optimizer.

10

# Conclusion and Future Work

- Federated learning can be performed on a FaaS-based environment consisting of heterogeneous devices.
- Manageability: FedKeeper offers easy creation, deletion, and invocation of FL-clients.
- Simplicity: Model training on individual clients is done using fine-grained FaaS-based functions.
- Scalability: FedKeeper offers the capability of running client functions remotely on Cloud FaaS platforms.
- Extend the FedKeeper to other FaaS platforms and add security related aspects in it. Furthermore, the paper explored techniques to optimize the performance of running client functions in parallel.

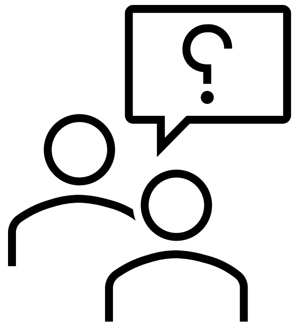
11

# Paper Critique

- Weakness:
  - Only used with Deep learning algorithm.
  - Only tested with one dataset.
- Improvement:
  - Send updated results to client when still in training to improve running time.

12

# Questions



13