Demo: FLaaS - Practical Federated Learning as a Service for Mobile Applications

Kleomenis Katevas, Diego Perino, Nicolas Kourtellis
Telefonica Research, Barcelona, Spain

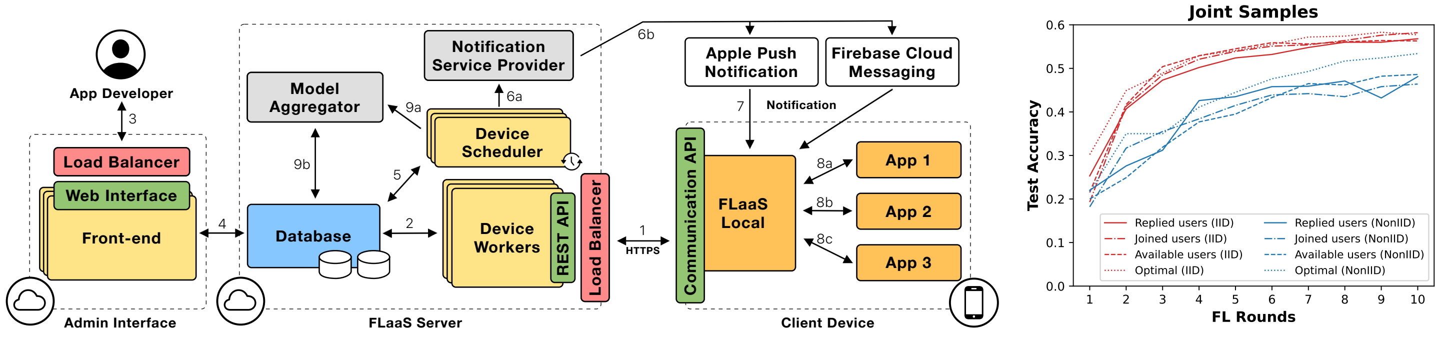

Figure 1: (Left) FLaaS architecture. A set of devices participate in FLaaS and periodically report their status (1) through the back-end's REST API, (2) to the DB for logging. App developers use the Admin Interface to (3) create a new FLaaS FL project, whose configuration is (4) stored in the DB. The Device Scheduler (5) detects the new FLaaS project and queries for device statuses from the DB, and (6a) sends an FL training request using its Notification Service provider to (6b) external services (e.g., APN or FCM). Each device's FLaaS Local (7) receives the training request and (8a, 8b, 8c) requests from collaborating third-party apps to send either their local samples for training, or to train their own models. It then performs the required on-device ML training (if it received samples) or model aggregation and averaging depending on ML training mode. When sufficient number of reported models is received by the Server (1), the Device Scheduler (9a, 9b) instructs the Model Aggregator to accumulate the received models, marks the FL round as complete, and continues with the next FL round until the project is complete (*i.e.*, stopping criteria are reached). (Right) Test accuracy achieved per FL round for different experiments on 100+ devices.

ACM Reference Format:
Kleomenis Katevas, Diego Perino, Nicolas Kourtellis. 2022. Demo: FLaaS - Practical Federated Learning as a Service for Mobile Applications. In *The 23rd International Workshop on Mobile Computing Systems and Applications (HotMobile '22), March 9–10, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3508396.3517074

1 FLAAS MOTIVATION

Federated Learning (FL) has emerged as a popular solution to distributedly train a model on user devices, improving privacy and system scalability. However, there are no practical systems to easily enable FL training on mobile apps, and especially in an "as-a-service" fashion. Thus, we implement and test FLaaS, our previously proposed end-to-end service [1]:
a) Easy to use by incorporating it in existing apps, and providing a simple user interface to configure the FL process.
b) Performs on-device training by building on TensorFlow Lite to support different FLaaS ML model training modes: i) individual model per app, ii) joint model across collaborating apps.
c) Enables secure & private FL training using i) secure communication channels and authentication between apps and FLaaS service, ii) secure on-device data storages.
d) Is production ready: as it was tested in the wild on 140 real devices to deal with challenges such as scalability and robustness using load balancers, cloud-based notification service, etc.

2 FLAAS DESIGN & IMPLEMENTATION

Fig. 1(left) outlines the FLaaS architecture and steps required to perform an FL training process within FLaaS. The back-end incorporates the roles of Admin Interface, Server and Notification Service. It is hosted in the Heroku Cloud platform (heroku.com) and uses *Standard-1X Dynos* (512MB RAM), Linux-based shared containers to enable easy deployment, scaling, load balancing and scheduling of FLaaS modules.
a) Admin Interface: Front-end interface for app developers used to bootstrap, configure, execute and monitor FL projects.
b) Server: Scalable, cloud-hosted service in charge of orchestrating / monitoring FL processes on behalf of app developers.
c) Notification Services: Mobile push notification services used for push comms to client devices for new FL projects and rounds execution (e.g., Apple Push Notification (APNs) and Firebase Cloud Messaging (FCM)), provided by pushwoosh.com service.
d) Client Devices: FLaaS currently supports Android-based devices, Android 8.0+, since iOS does not provide (yet) in-app comms features necessary for joint, cross-app training. Third-party apps can integrate the FLaaS-library with ~10 lines of code.

3 FLAAS DEMONSTRATION

This live demo includes the following steps:
Step 1: Creation of new FLaaS projects with all required configurations, using online front-end app developer interface.
Step 2: Deployment of project on several real mobile devices with 3 FLaaS-enabled mobile apps, to train FL models collaboratively.
Step 3: Monitoring of project's health and global FL model performance as in Fig. 1(right) on different experimental scenarios.

Acknowledgment:
This project received funding from EC under grant agreements No 830927 (Concordia), No 871370 (Pimcity) and No 871793 (Accordion).

References:
[1] Kourtellis, N., Katevas, K., and Perino, D. Flaas: Federated learning as a service. In ACM DistributedML (2020).