Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds

Kevin Hsieh, Aaron Harlap, Gregory R. Ganger, Nandita Vijaykumar, Phillip B. Gibbons, Dimitris Konomis, Onur Mutlu, ETH Zurich and CMU (NSDI 2017)

Presentors: Yin Lin, Jinyang Li, Jie Liu

Motivation

1. Centralized data is infeasible, no ML systems is designed to run **across data centers.** Develop a <u>geo-distributed ML system</u>.

2. Training within data center v.s. Training across data centers

   1. Data is centrally distributed
   2. Training happens within the LAN

   1. Data is geo-distributed
   2. Training over WAN is slow (1.8-53.8X slower than LAN)

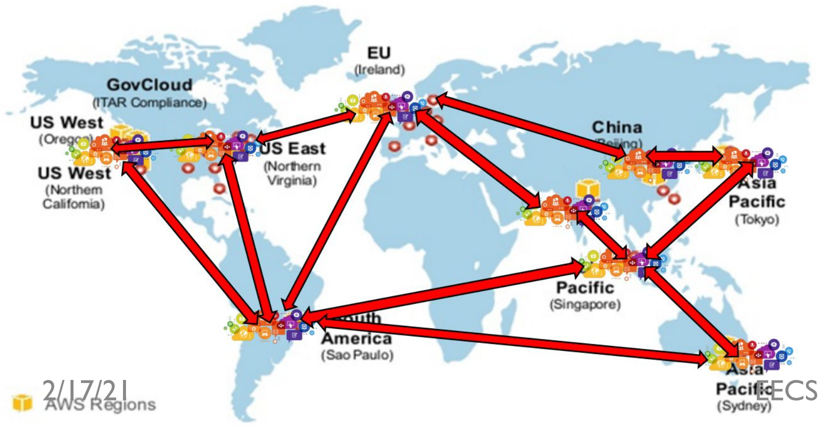


Image reference: https://www.usenix.org/sites/default/files/conference/protected-files/nsdi17_slides_hsieh.pdf

Main Contributions

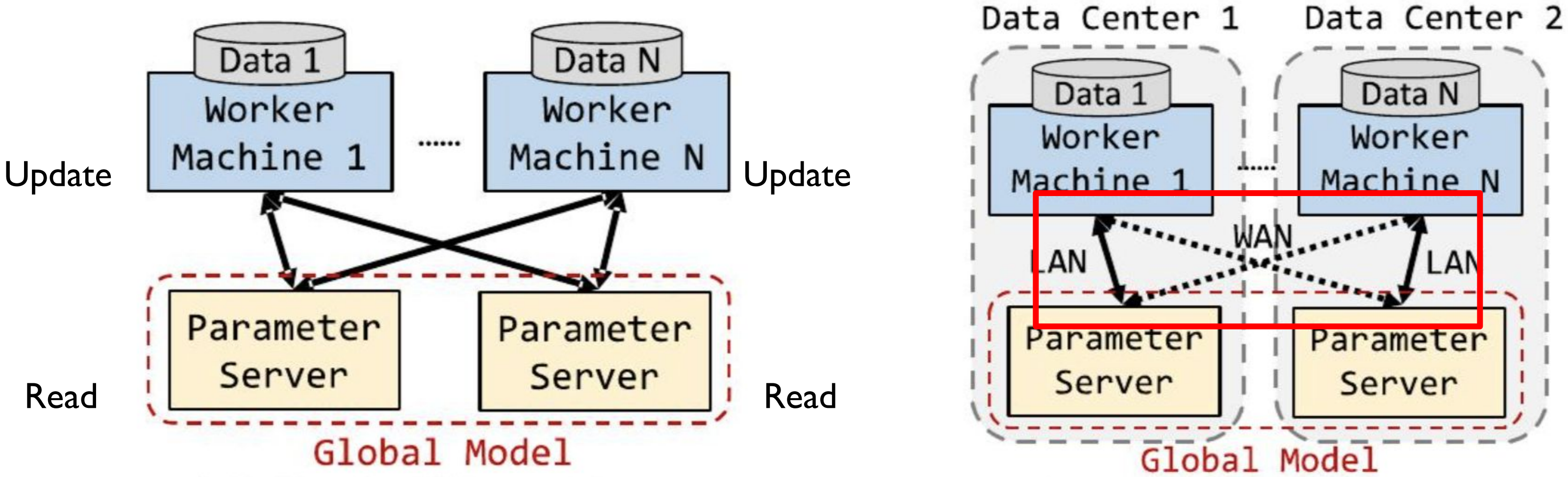
1. Minitize the communication over wide-area networks (WAN) training. Achieved a **1.8-53.5x** speed up over two state-of-the-are distributed algorithms on WANs.

2. Retain the **accuracy** and **correctness** of machine learning algorithms.

3. **Does not require changes** to the ML algorithms.

Background

Parameter Server Architecture has been widely adopted in many ML systems



(a) *Basic PS architecture* (b) *Simple PS on WANs*

***Synchronization** is critical to the accuracy and correctness of ML algorithms.*

WAN bandwidth is very scarce resource

WAN bandwidth is 15X smaller than LAN bandwidth on average and can be up to 60X smaller

The extra cost of WAN communication could be up to 38X greater than LAN.

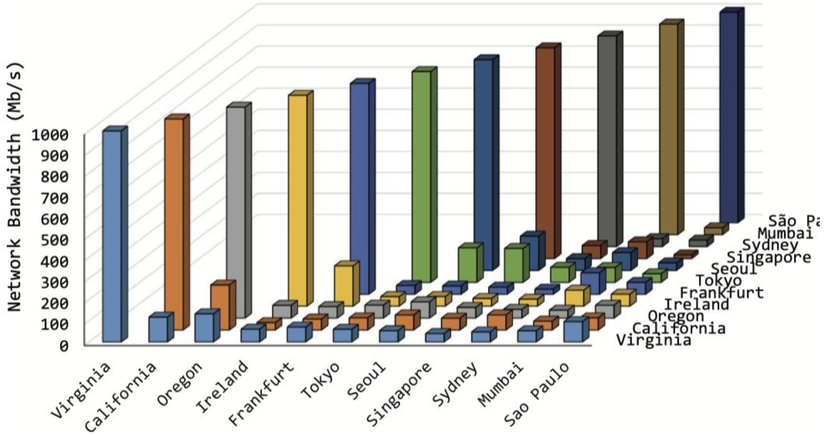


Figure 2: Measured network bandwidth between Amaz EC2 sites in 11 different regions

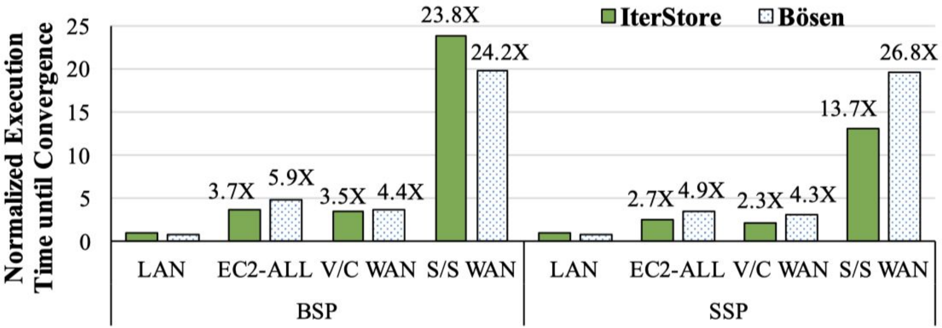


Figure 3: Normalized execution time until ML algorithm convergence when deploying two state-of-the-art distributed ML systems on a LAN and WANs

Gaia System Overview

Communicate over WANs only significant updates

Key idea: **Decouple the synchronization model** within the data center from the synchronization model between data centers. (Approximate Synchronous Parallel)

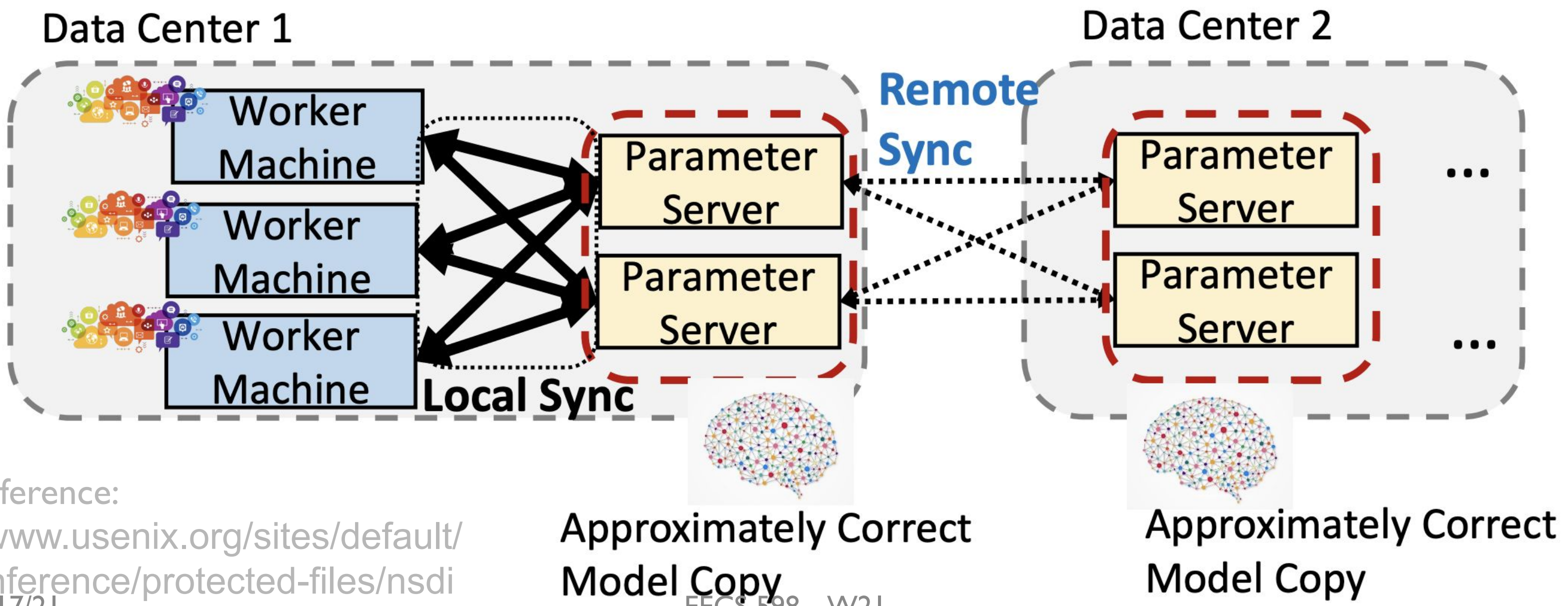


Image reference: https://www.usenix.org/sites/default/files/conference/protected-files/nsdi17_slides_hsieh.pdf

Approximate Synchronous Parallel

The significance filter

To filter updates based on their significance

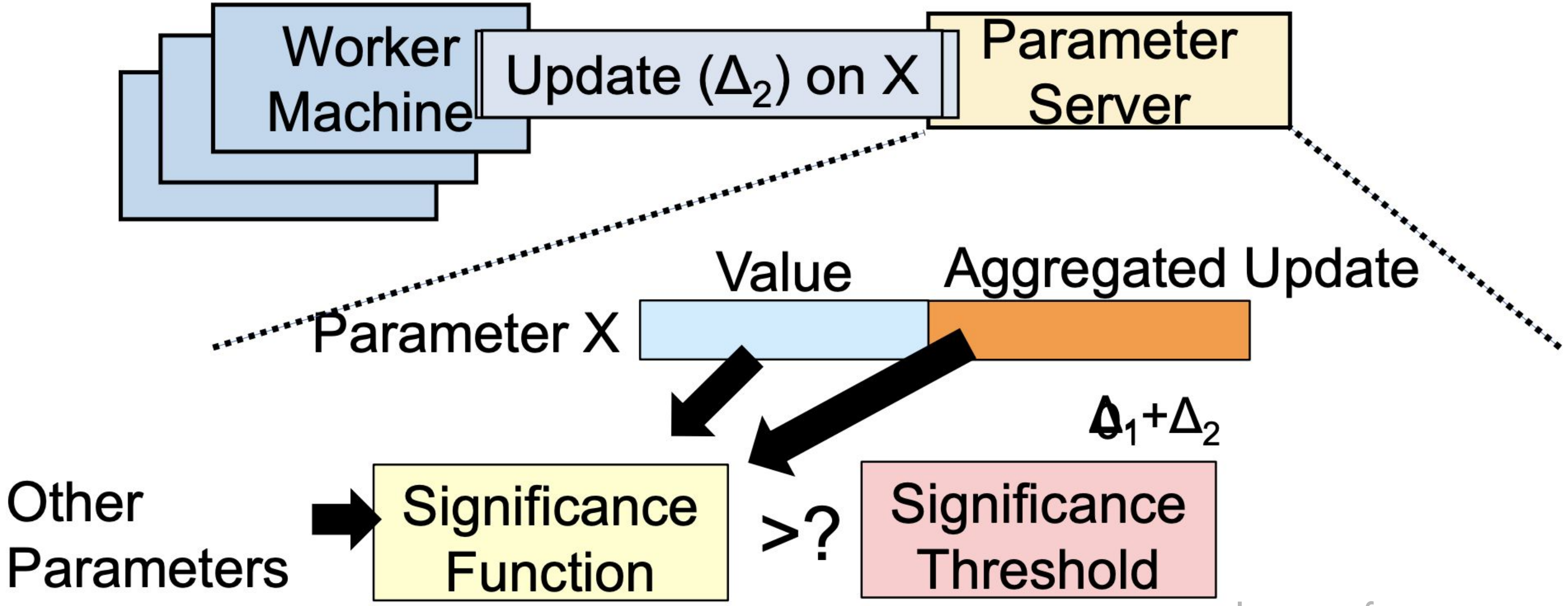
ASP selective barrier

To ensure significant updates are read in time

Mirror clock

Safe guard for pathological cases.

The Significance Filter



$$\left\|\frac{Agg.Update}{Value}\right\|$$

$$10\% \quad \frac{1\%}{\sqrt{T}}$$

Image reference: https://www.usenix.org/sites/default/files/conference/protected-files/nsdi17_slides_hsieh.pdf

Approximate Synchronous Parallel

The significance filter

To filter updates based on their significance

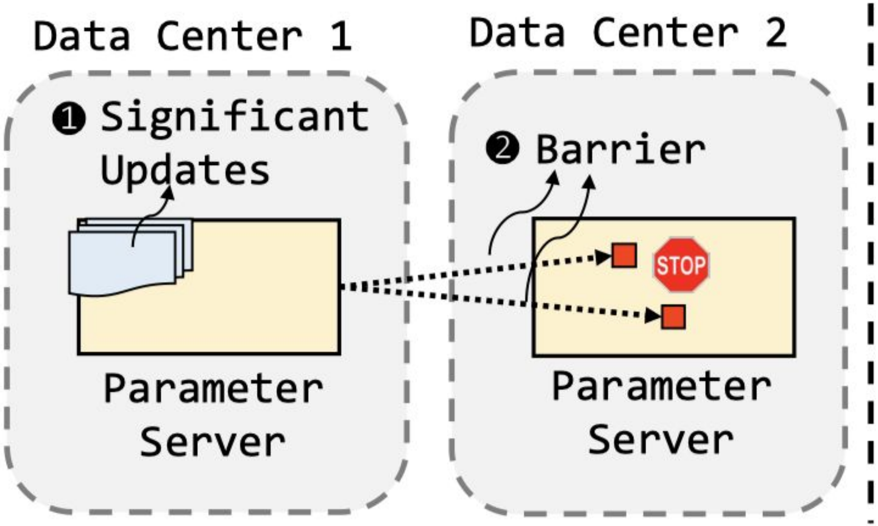
ASP selective barrier

To ensure significant updates are read in time

Mirror clock

Safe guard for pathological cases.

ASP Selective Barrier



(a) *ASP selective barrier*

The significant updates could arrive too late to the other data centers.

To solve this problem, each parameter server would first sent a selective barrier to the other data center and the other data center would use the barrier to blocks the late updates.

Approximate Synchronous Parallel

The significance filter

To filter updates based on their significance

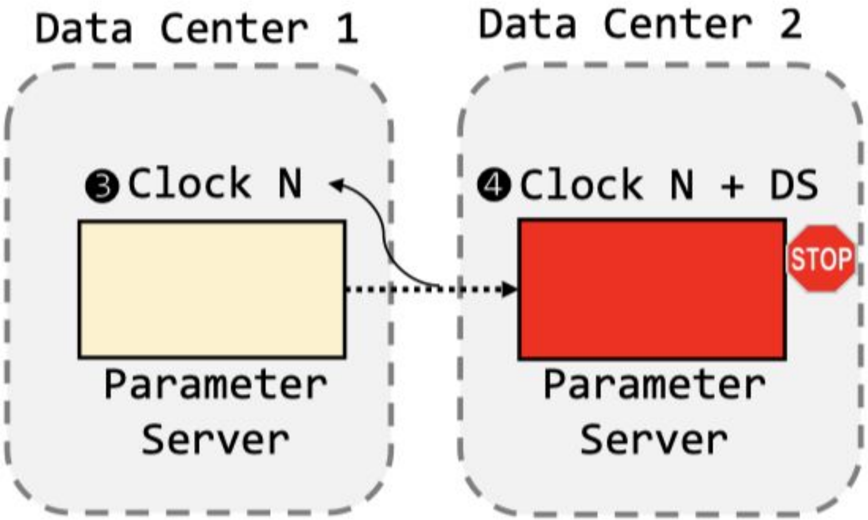
ASP selective barrier

To ensure significant updates are read in time

Mirror clock

To ensure worker machines are aware of the significant updates in time

Mirror Clock



(b) *Mirror clock*

Guarantee that the worker machines are aware of the significant updates in time, irrespective of the WAN latency

Experiment Setup

Baseline:

IterStore (Cui et al., SoCC'14) and GeePS (Cui et al., EuroSys'16) on WAN

ML Applications:

- Matrix Factorization with the Netflix dataset
- Topic Modeling with the Nytimes dataset
- Image Classification with the ILSVRC12 dataset

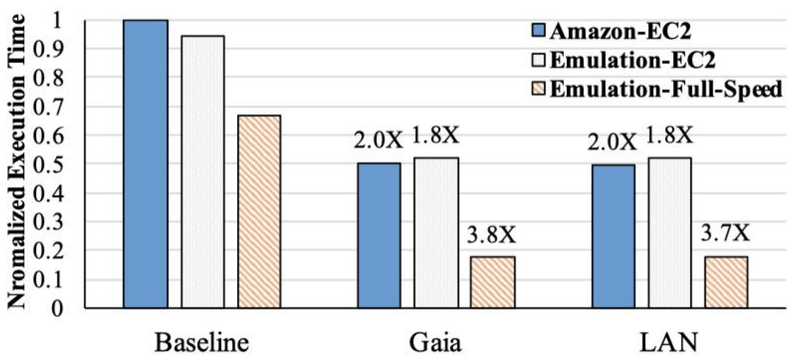
Metrics:

Execution time until algorithm convergence

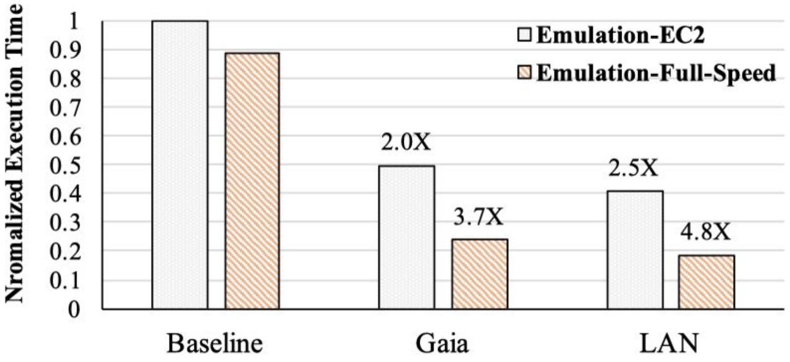
Monetary cost of algorithm convergence

Platforms: Amazon-EC2, Emulation-EC2, Emulation-Full-Speed

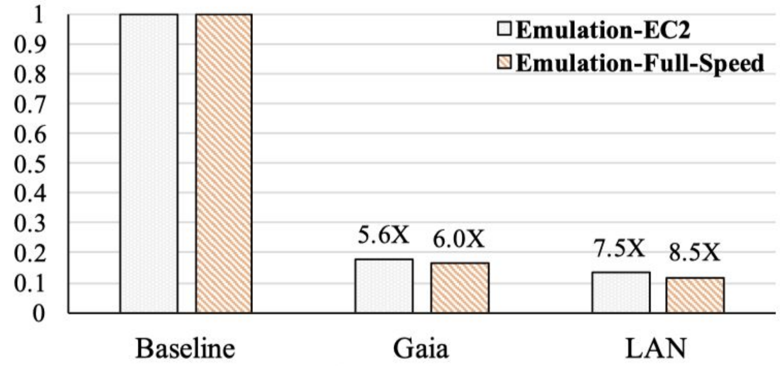
Result (1) convergence time improvement



(a) *Matrix Factorization (MF)*



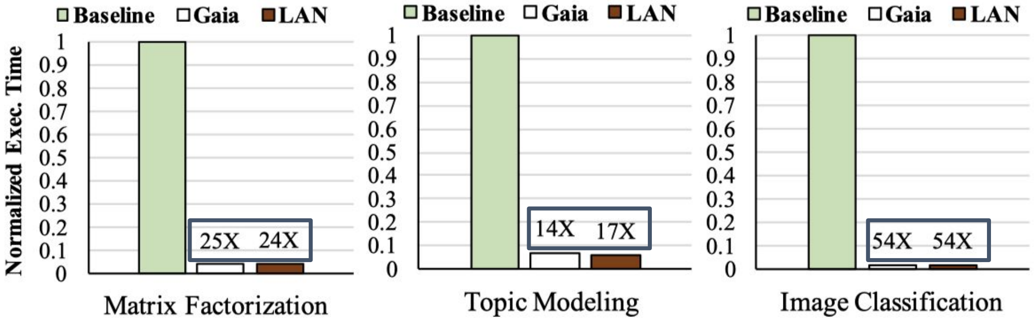
(b) *Topic Modeling (TM)*



(c) *Image Classification (IC)*

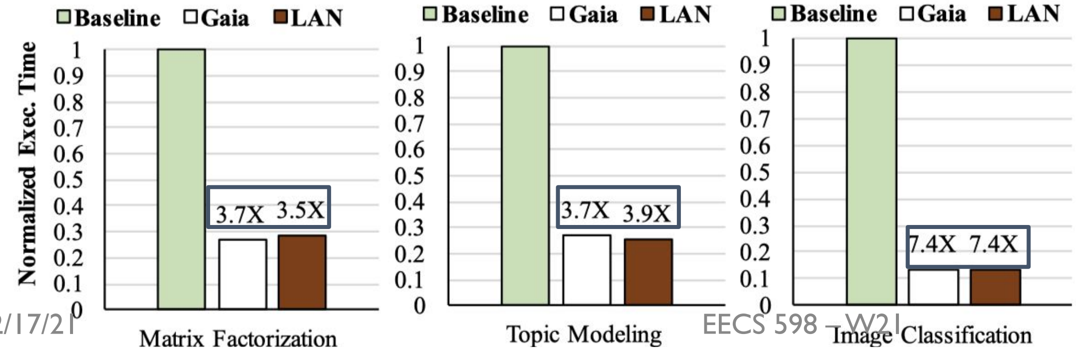
Result (2) Convergence time with WAN bandwidth

Virginia <-> California (nearby)



Gaia achieves 3.7-53.5X speedup over Baseline and is at most 1.23 X of LAN speeds

Singapore <-> Sao Paulo (far apart)



Result (3) EC2 Monetary Cost

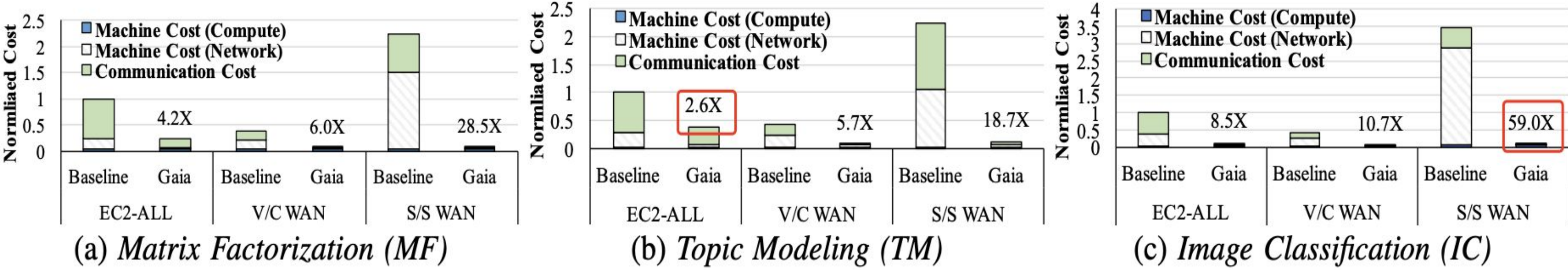


Figure 11: Normalized monetary cost of Gaia vs. Baseline

Gaia is 2.6-59.0X cheaper than the Baseline.

Summary and discussion

Key problem: How to perform ML on geo-distributed data?

Centralized data is not feasible and communication through WAN would be expensive.

Gaia: Decouple the synchronization model within the data center from that across the data centers

A new synchronization model: Approximate Synchronous Parallel (ASP)

Still achieve accuracy and correctness of ML algorithms.

Synchronization models comparison

	Models	Key idea	Convergence guarantee
Among workers	BSP bulk synchronous parallel	Sync all updates after each worker finishes All see up-to-date data before next iteration	Yes
	SSP stale synchronous parallel	Fastest worker can be ahead of slowest Fast workers may use stale model	Yes
	TAP total asynchronous parallel	No sync between workers. Workers send/receive updates as many as possible	No
Among data centers	**ASP** approximately synchronous parallel	Share aggregated updates when it is significant Use BSP/SSP within a data center	Yes

Gaia limitation: significance of updates

- Gaia: compare its *absolute value* (magnitude) with a threshold
- Only considers *speed* of training, not *optimization direction*
- Unable to tell if local updates align with the *collaborative optimization trend*
- CMFL: checks if an update aligns with the global tendency
- CMFL saves much more communication rounds compared to Gaia

[3] L. Wang, W. Wang, and B. Li, "Cmfl: Mitigating communication overhead for federated learning."

Piazza questions?

- Gaia paper: Whether it was applied in a real-world scenario. Does anyone know whether Amazon, Google etc are running something similar? Or they just train in one datacenter?

- Answer: we haven't found real-world application

Towards Federated Learning at Scale: System Design

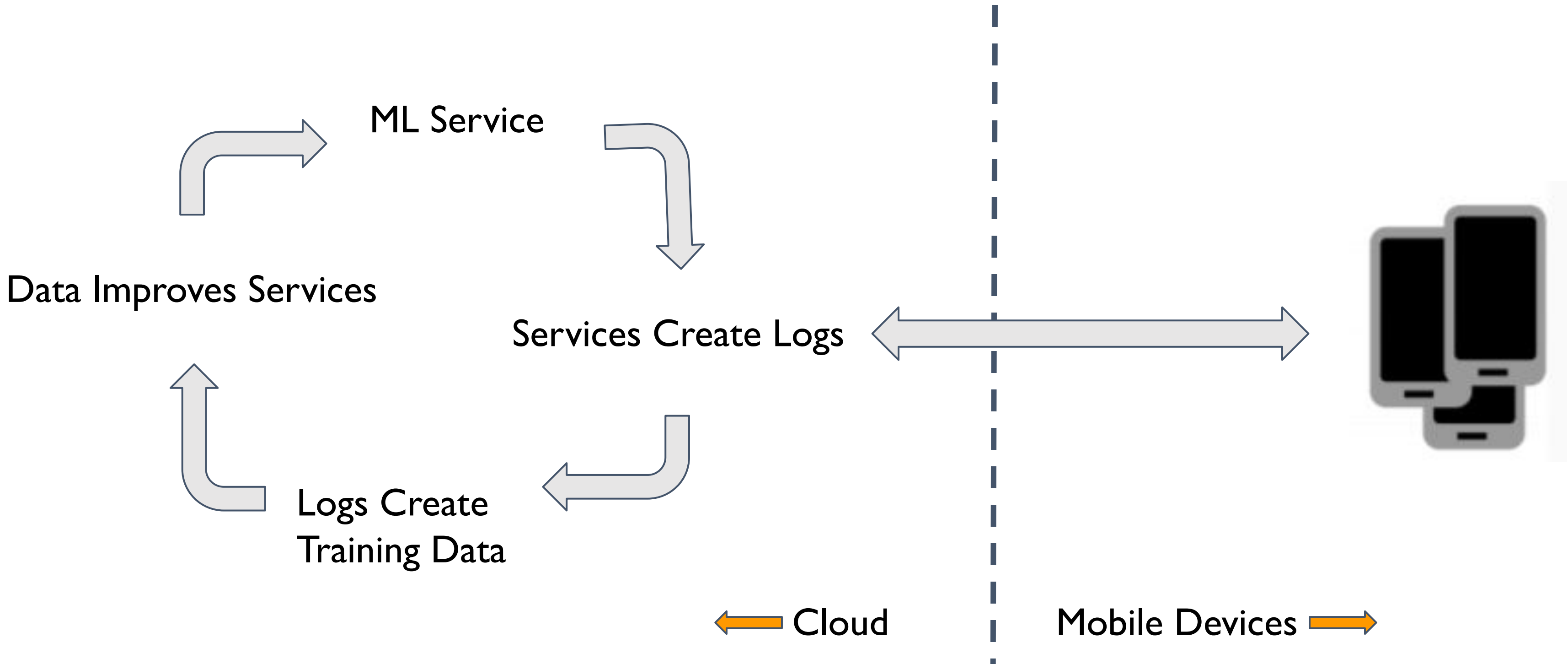
Keith Bonawitz, Hubert Eichner, et al., SysML 2019

Presentors: Yin Lin, Jinyang Li, Jie Liu

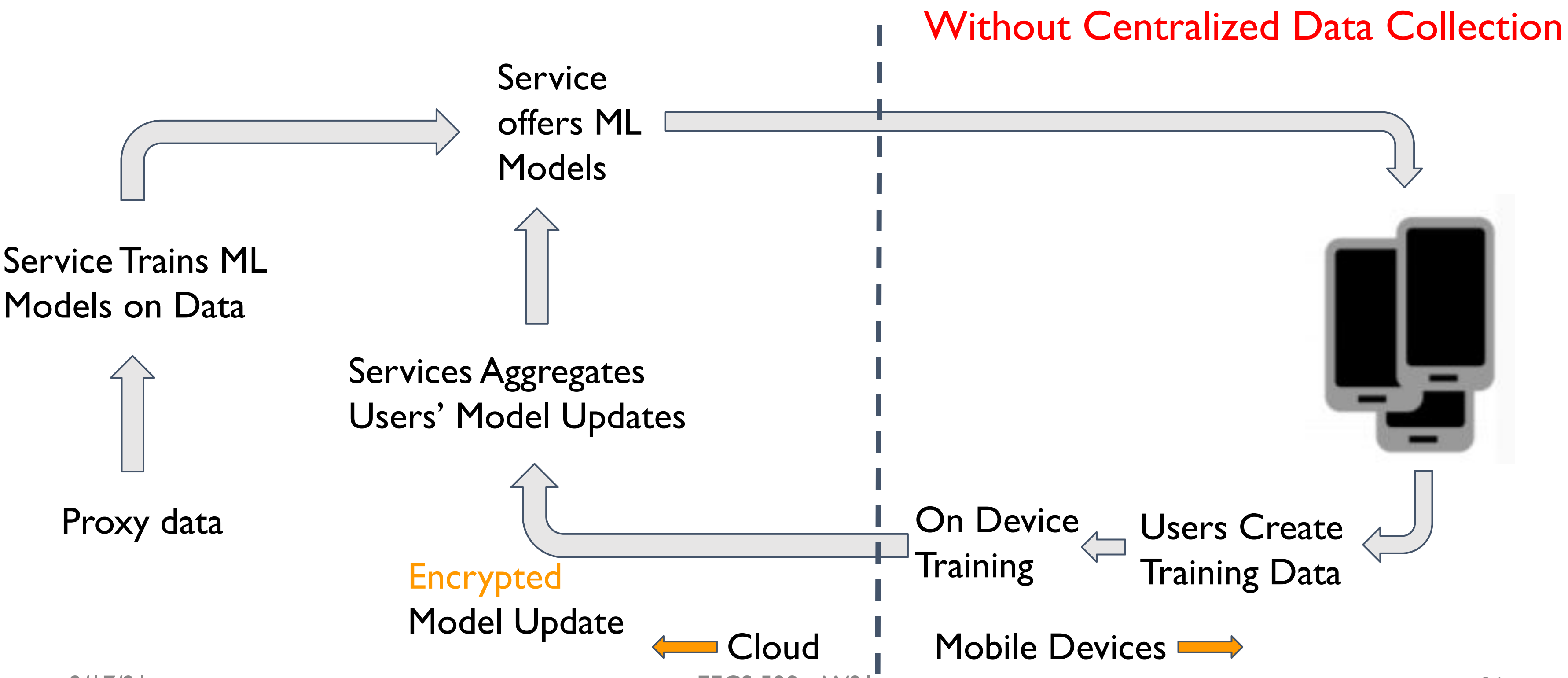
Page 19-21, 23, 26-28, 30, 34, 37 of the slides are borrowed from Wolfgang Grieskamp's slides at SysML 2019
"https://www.youtube.com/watch?v=_vAGhIS5Y_s"

What is Federated Learning?

Traditional Approach - Bring data to code



Federated Learning - Bring code to data

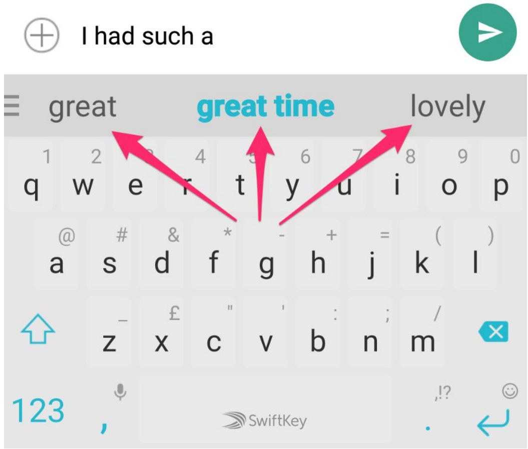


Motivation

- On-device data is
  - privacy-sensitive
  - undesirable or infeasible to transmit to servers
- A large number of mobile phones are available for training and inference

Applications:

- On-device item ranking
- Content suggestions for on-device keyboards
- Next word prediction



Federated Learning is Hard

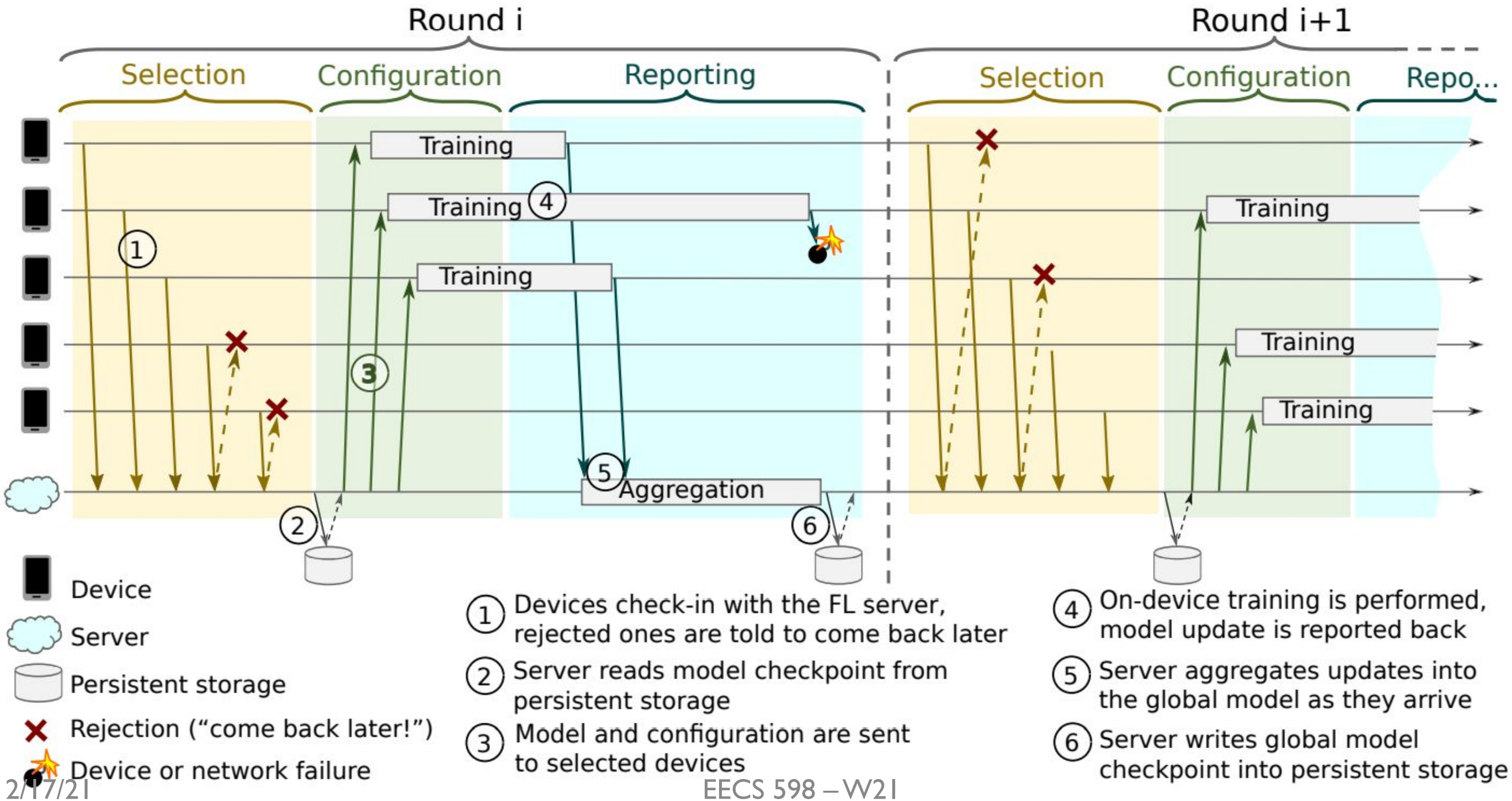
For example:

- Health of user devices must not be compromised
- User devices can drop out any moment and at high rate
- No direct access to devices that fail for diagnosis
- ...

Overview of the System

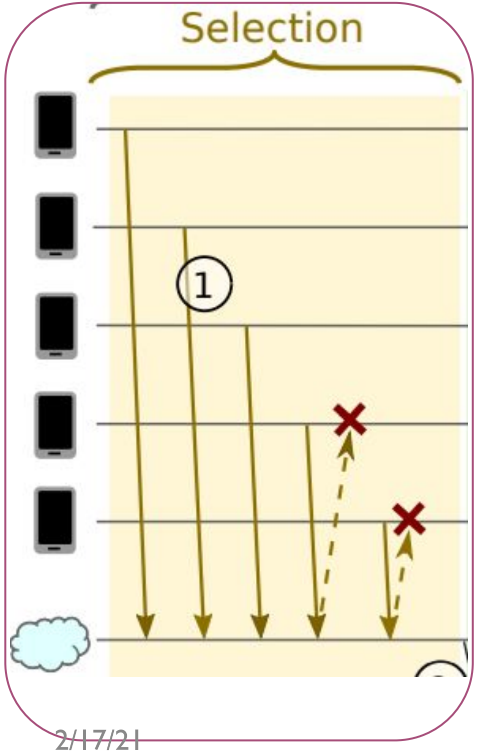
- FL Server: global model
- FL population: a set of devices that periodically compute updates
- FL population push updates to FL server & FL server aggregates updates
- Synchronous training

Protocol



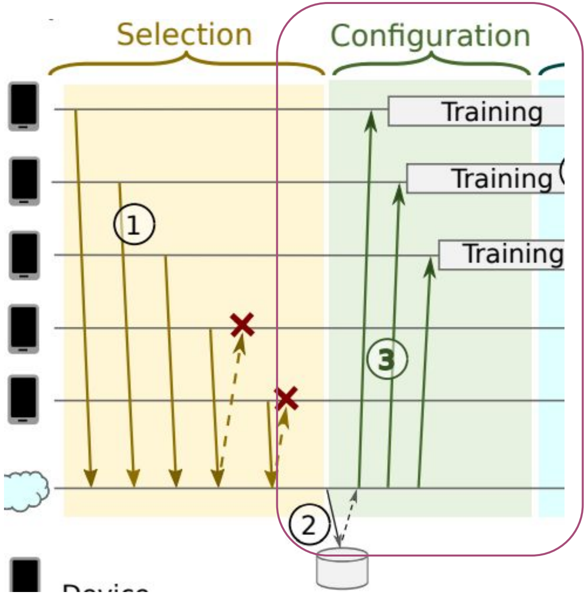
Round i
Round i+1
Selection
Configuration
Reporting
Selection
Configuration
Repo...
Training
Training
Training
Training
Training
Training
Aggregation
1
2
3
4
5
6
Device
Server
Persistent storage
Rejection ("come back later!")
Device or network failure
1 Devices check-in with the FL server, rejected ones are told to come back later
2 Server reads model checkpoint from persistent storage
3 Model and configuration are sent to selected devices
4 On-device training is performed, model update is reported back
5 Server aggregates updates into the global model as they arrive
6 Server writes global model checkpoint into persistent storage

Protocol - Selection Phase



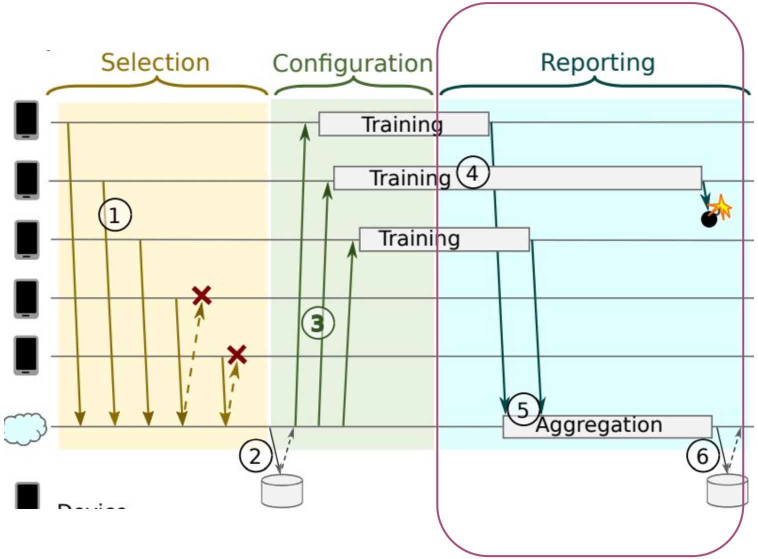
- Devices check in to server to announce availability for training, when device state allows
- Server makes a random selection of participating devices -- a few hundred out of thousands will be selected

Protocol - Configuration Phase



- Server reads model checkpoint from persistent storage
- Server sends the FL plan and an FL checkpoint with the global model to each of the devices.

Protocol - Reporting Phase

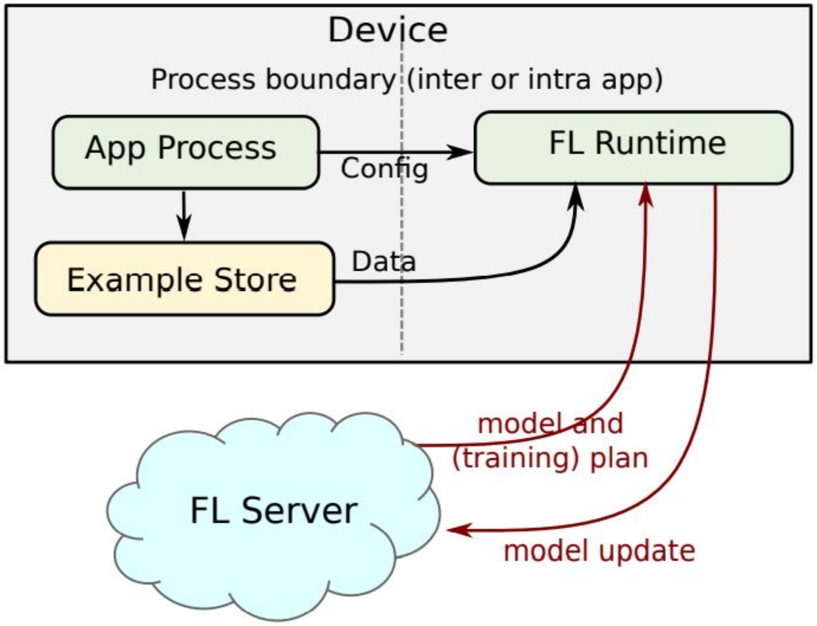


- When a device is ready, it reports an encrypted model update back to the server
- Server aggregates updates as they arrive
- Server closes the round and updates its global model if enough devices report in time
  - Otherwise the round is abandoned

Protocol - Peer Steering

- ❖ For small FL populations: Ensure that a sufficient number of devices connect to the server simultaneously
  - ➢ important both for the rate of task progress and for the security properties of the Secure Aggregation protocol
- ❖ For large FL populations: Randomize device check-in times
  - ➢ avoiding the "thundering herd" problem

Device - Architecture



- ❖ Applications on devices make their data available to the FL runtime as an example store
- ❖ FL runtime, when provided a task by the FL server, accesses an appropriate example store to perform training or evaluation

Device - Control Flow

- Programmatic Configuration
    - App configures the FL runtime by providing an FL population name and registering its example stores.
- Job invocation
    - FL runtime contacts FL server to announce that it is ready.
- Task execution
    - FL runtime receives the FL plan and computes plan-determined model updates.
- Reporting
    - FL runtime sends updates to FL server.

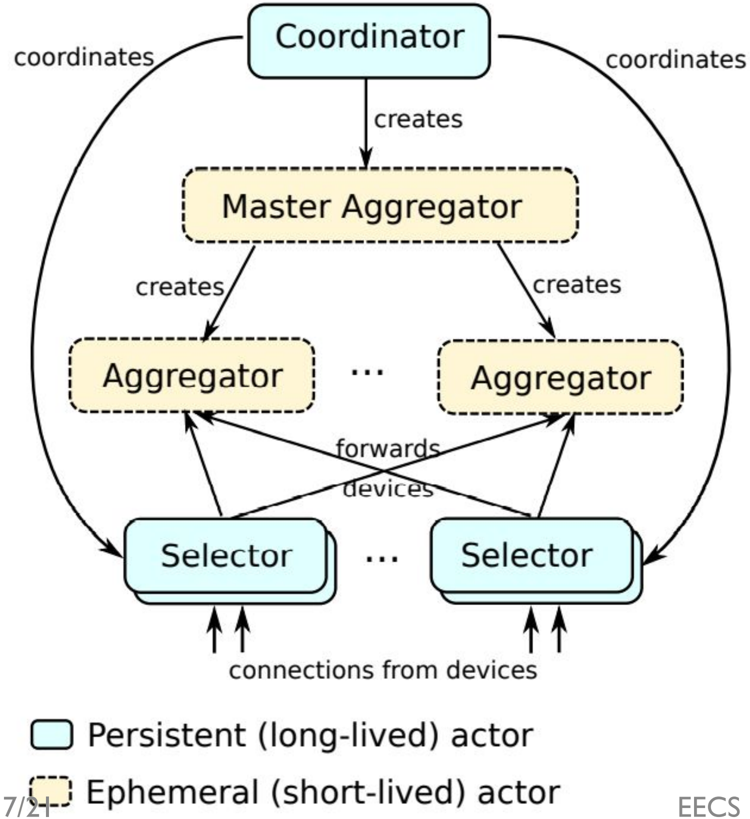
Device - More Details

- Multi-Tenancy
  - Allow for coordination between multiple training activities.
  - Avoid the device being overloaded by many simultaneous training sessions at once.
- Attestation
  - Protect against data poisoning via compromised devices by using Android’s remote attestation mechanism.

Server - Actor Model

- FL server is designed around the Actor Programming Model
  - Each actor handles a stream of messages/events strictly sequentially
  - An actor can make local decisions, send messages to other actors, or create more actors dynamically
  - Ephemeral instances of actors enables dynamic resource management and load-balancing decisions.

Server - Architecture

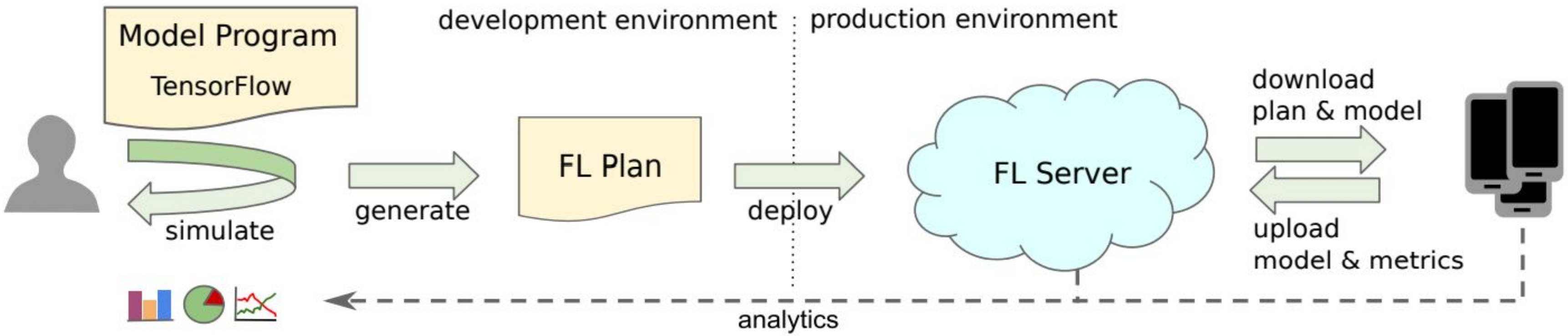


- Coordinator manages a training population.
- Selectors are responsible for accepting and forwarding device connections.
- Aggregators are spawn when a training round is initiated and process device reports.

Secure Aggregation

- Secure Aggregation
  - Use encryption to make individual devices' updates uninspectable by a server.
  - Only reveal the sum after a sufficient number of updates have been received.
- Costs grow quadratically with the number of users
  - Limit the maximum size of a Secure Aggregation to hundreds of user.
  - Solution: Run an instance of Secure Aggregation on each Aggregator actor and Master Aggregator then further aggregates the intermediate results without Secure Aggregation.

Workflow



Model Program
TensorFlow
development environment
production environment
download
plan & model
simulate
generate
FL Plan
deploy
FL Server
upload
model & metrics
analytics

Results in Production

- Can handle a cumulative FL population size of approximately 10 M daily active devices, spanning several different applications
- Up to 10k devices are participating simultaneously
- Federated learning is is roughly 7× slower than in comparable data center training of the same model
- 6% ~ 10% of devices drop out due to computation errors, network failures, or changes in eligibility
- Server typically selects 130% of the target number of devices to initially participate in order to compensate for device drop out

Discussion - Federated Learning

Advantages

- Protected privacy and ownership of data
- Preserved locality of data
- Enables edge devices to collaborate

Disadvantages

- Potential bias
- Slow convergence time
- Blind device scheduling & Sampling
- ...

FL - bias

- Problem:
  - Devices only train when on unmetered network and charging
  - Limit deployment to certain phones (Android)
- Current solution:
  - Models are not used to do user-visible predictions
  - Evaluated the trained model using multiple application-specific metrics.
  - Bias can be detected if it leads to a bad model
- So far it isn't an issue in practice

FL - convergence time

- FL has a slower convergence time than ML on centralized data
- Current FL only uses 100s of devices in parallel, with more available
  - Need better algorithms to use more parallelism
- Dynamically adjust time windows -- between select devices to train and wait for reporting

FL - device scheduling

- Currently: a simple worker queue
- Blind to which apps the user uses frequently.
- Result: repeatedly train on old data, ignore newer data
- Optimizations expected

FL - device sampling

- FL randomly selects devices
- Strategies to select participating devices at each round.
- [1]: sample based on system resources, aim at letting server aggregate more devices updates within a time window
- [2]: prefer devices with higher-quality data by incentive mechanisms

[1] T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *International Conference on Communications*, 2019.
[2] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim. Incentive design for efficient federated learning in mobile networks: A contract theory approach. *arXiv preprint arXiv:1905.07479*, 2019.

FL VS Parameter Server

- Scheduling
  - PS: the central node has the highest authority
  - FL: the working nodes has the freedom to participate
    - adding layers of complexity when it comes to scheduling an optimal learning environment

- Data storage:
  - PS: data center setting, shared storage; worker machines fetch data.
  - FL: data and computation are done locally; can be heterogeneous.

- Fault tolerance (dropping out issue):
  - PS: store copies of parameters, relocate buckets from failed machines to others
  - FL: nothing for clients dropping out; but selects 130% of target number of devices

Piazza questions?

- TFF paper: How they do the backprop. In case of DNN, they only do the forward prop on device, or both?

- Answer: Both. Server sends the global model to each of the devices. Devices have both data and model ready. On-device training is performed and then model update is reported back to server.

Reference

- [1] T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *International Conference on Communications*, 2019.
- [2] J. Kang, Z. Xiong, D. Niyato, H. Yu, Y.-C. Liang, and D. I. Kim. Incentive design for efficient federated learning in mobile networks: A contract theory approach. *arXiv preprint arXiv:1905.07479*, 2019.
- [3] L. Wang, W. Wang, and B. Li, "Cmfl: Mitigating communication overhead for federated learning."
- [4] Lim, Wei Yang Bryan, et al. "Federated learning in mobile edge networks: A comprehensive survey." *IEEE Communications Surveys & Tutorials* 22.3 (2020): 2031-2063.