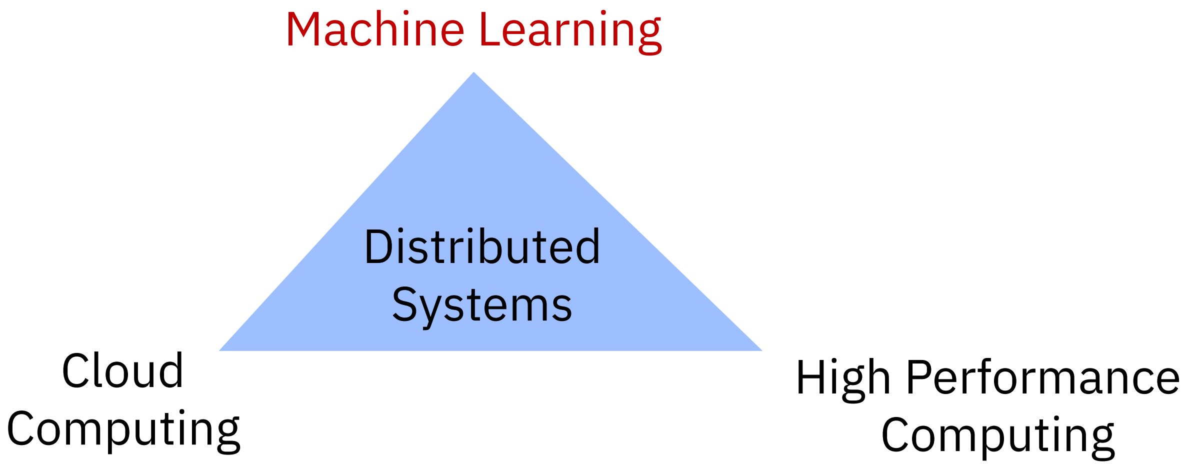
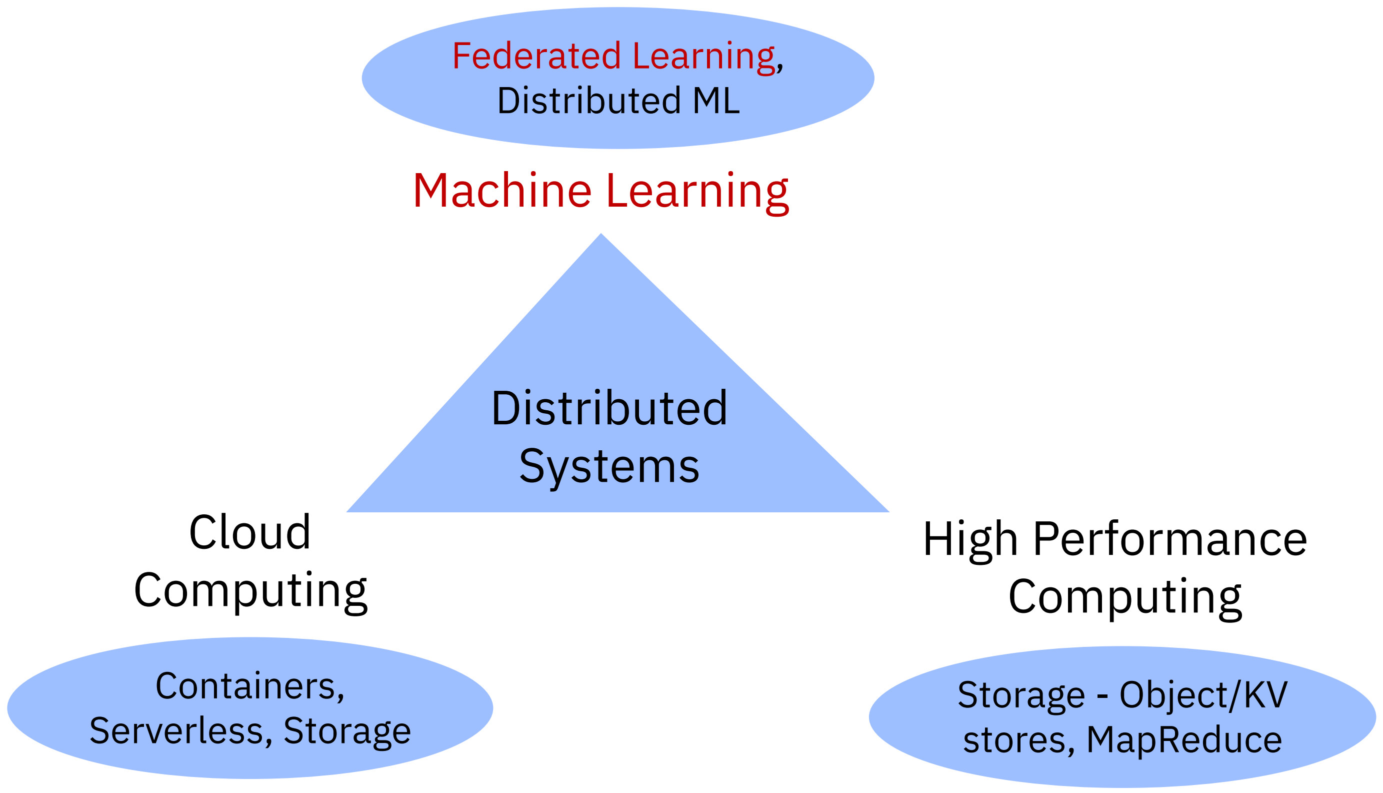
Towards Taming the Resource and Data Heterogeneity in Federated Learning

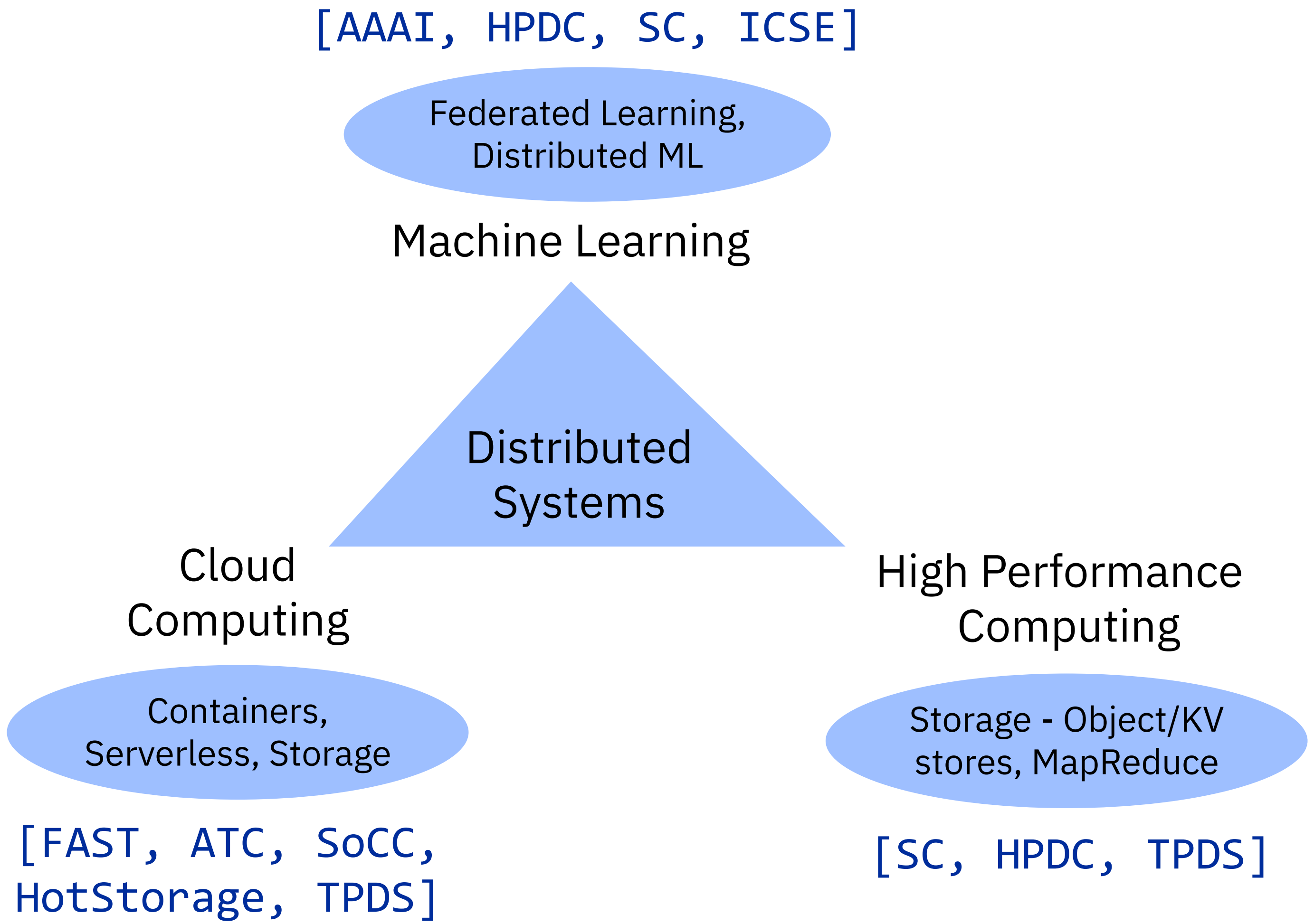
Ali Anwar

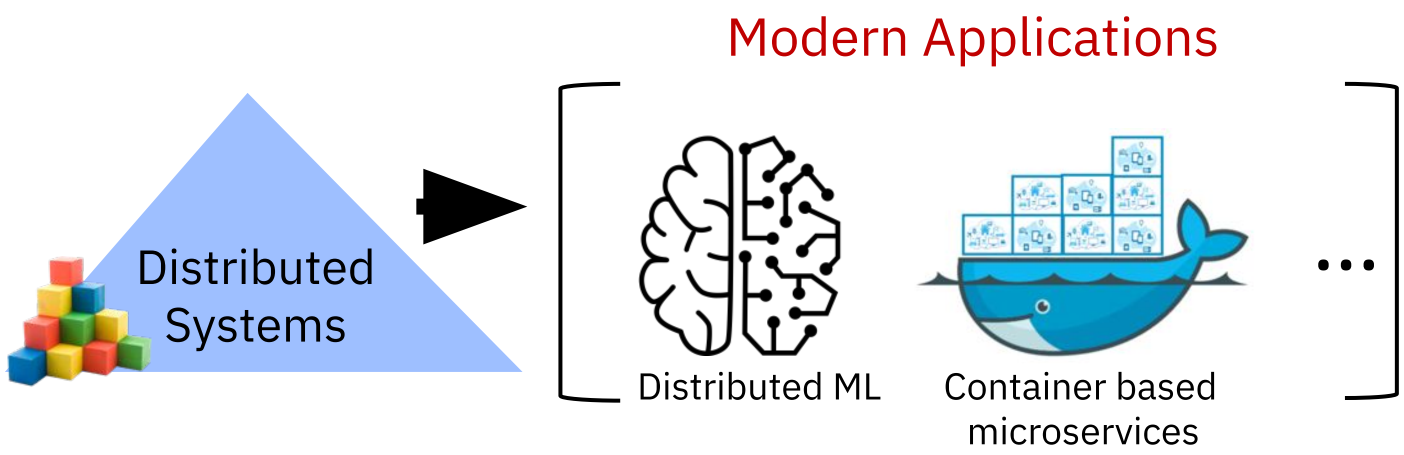
Assistant Professor
University of Minnesota

Department of Computer Science and Engineering

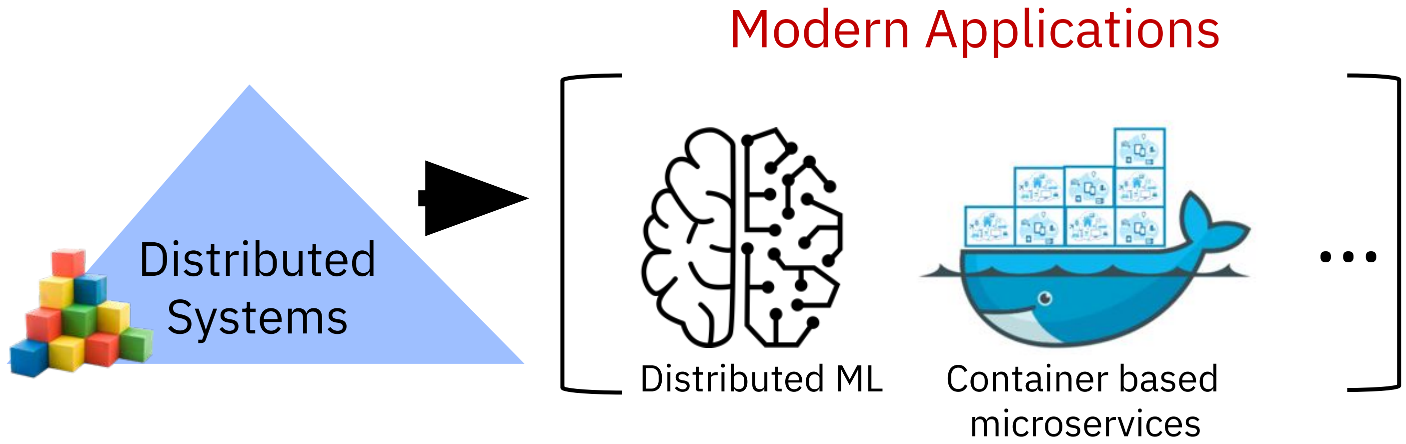

Machine Learning
Distributed Systems
Cloud Computing
High Performance Computing


Federated Learning,
Distributed ML
Machine Learning
Distributed Systems
Cloud Computing
Containers, Serverless, Storage
High Performance Computing
Storage - Object/KV stores, MapReduce

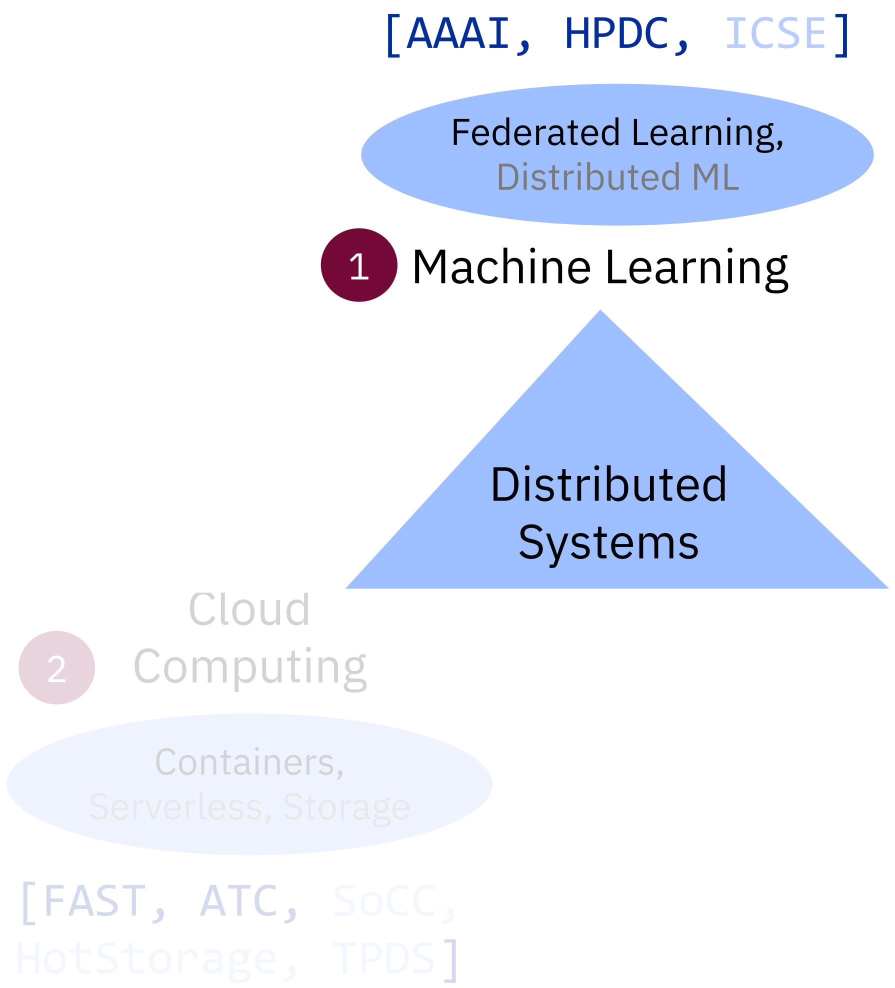

[AAAI, HPDC, SC, ICSE]
Federated Learning, Distributed ML
Machine Learning
Distributed Systems
Cloud Computing
Containers, Serverless, Storage
[FAST, ATC, SoCC, HotStorage, TPDS]
High Performance Computing
Storage - Object/KV stores, MapReduce
[SC, HPDC, TPDS]



- Distributed systems are building blocks of modern data applications
- With new applications underlying distributed system faces new challenges
- Analyzing the working of these applications from distributed systems perspective can help better understand these applications



Understanding the workload characteristics of these applications opens new opportunities to make informed design decision that *can improve* both the *application performance* and the *efficiency of underlying distributed system*


[AAAI, HPDC, ICSE]
Federated Learning,
Distributed ML
1
Machine Learning
Distributed Systems
Cloud Computing
2
Containers,
Serverless, Storage
[FAST, ATC, SoCC,
HotStorage, TPDS]

What is Federated Learning?

Federated learning allows us to collaboratively build ML models, no matter where data lives by *combining outputs from different parties*

- Multiple parties
- Train a machine learning model
- Collaboratively
- Without sharing training data

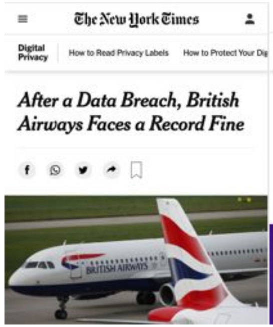
Why Federated Learning?

General Data Protection Regulation

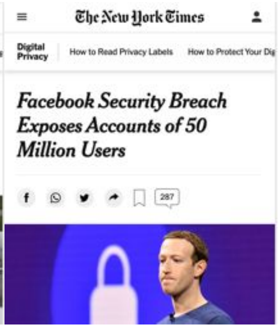
HIPAA
Health Insurance Portability & Accountability Act

CALIFORNIA CONSUMER PRIVACY ACT

Privacy concerns & Regulation


The New York Times
Digital Privacy | How to Read Privacy Labels | How to Protect Your Dig

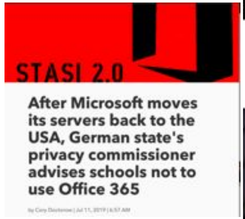
After a Data Breach, British Airways Faces a Record Fine


The New York Times
Digital Privacy | How to Read Privacy Labels | How to Protect Your Dig

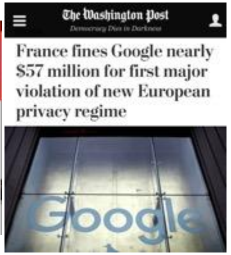
Facebook Security Breach Exposes Accounts of 50 Million Users

287

Liability


STASI 2.0

After Microsoft moves its servers back to the USA, German state's privacy commissioner advises schools not to use Office 365


The Washington Post
Democracy Dies in Darkness

France fines Google nearly $57 million for first major violation of new European privacy regime

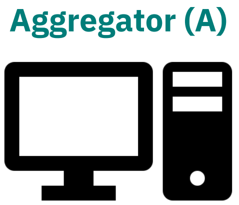
Data stored across Clouds or countries

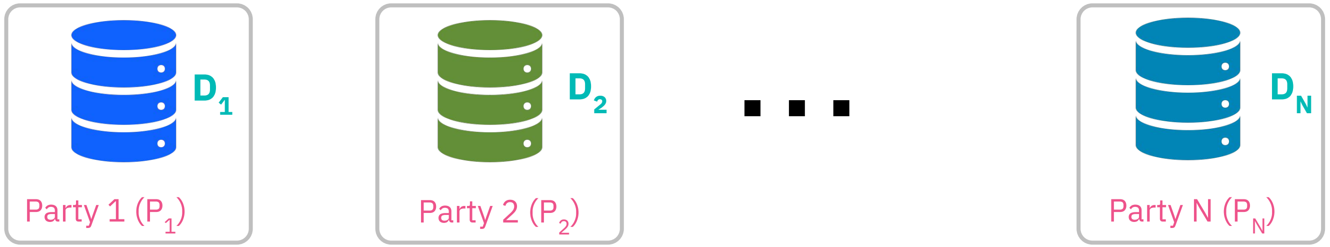
Trade Secrets

Source: IBM Federated Learning Framework https://ibmfl.mybluemix.net/introduction

Federated Learning

Architecture

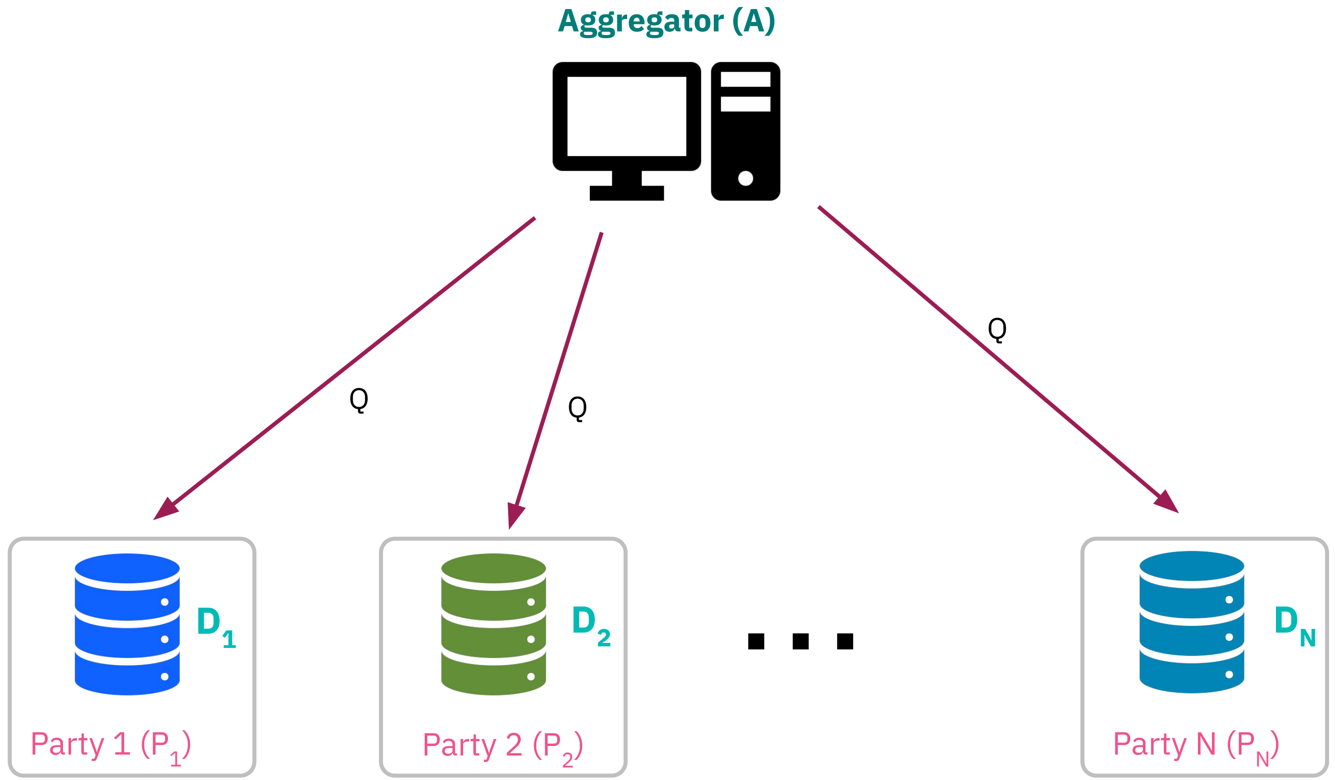




Federated Learning

Architecture

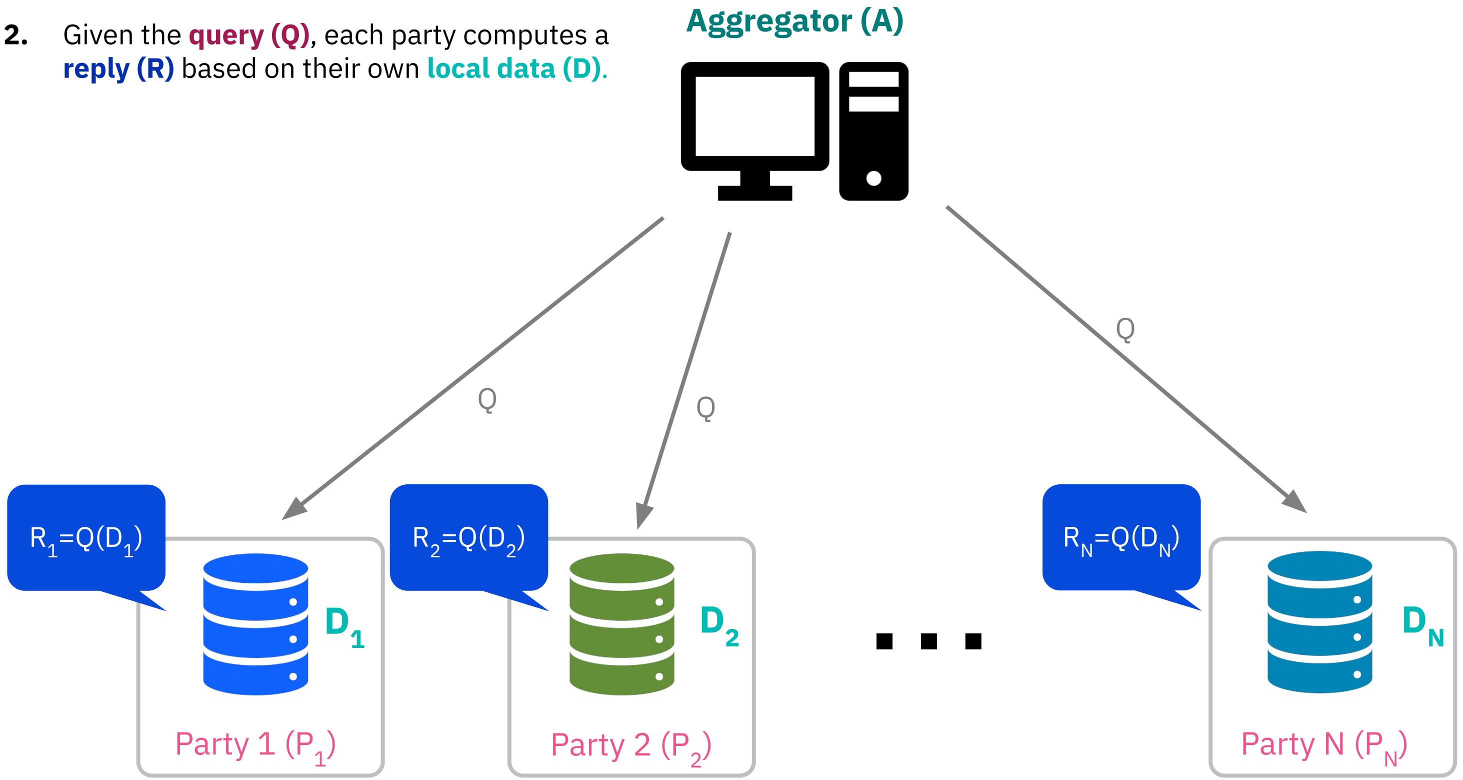
1. **Aggregator** queries **parties** along with information required for learning a model.



Federated Learning

Architecture

1. **Aggregator** queries **parties** along with information required for learning a model.
2. Given the **query (Q)**, each party computes a **reply (R)** based on their own **local data (D)**.



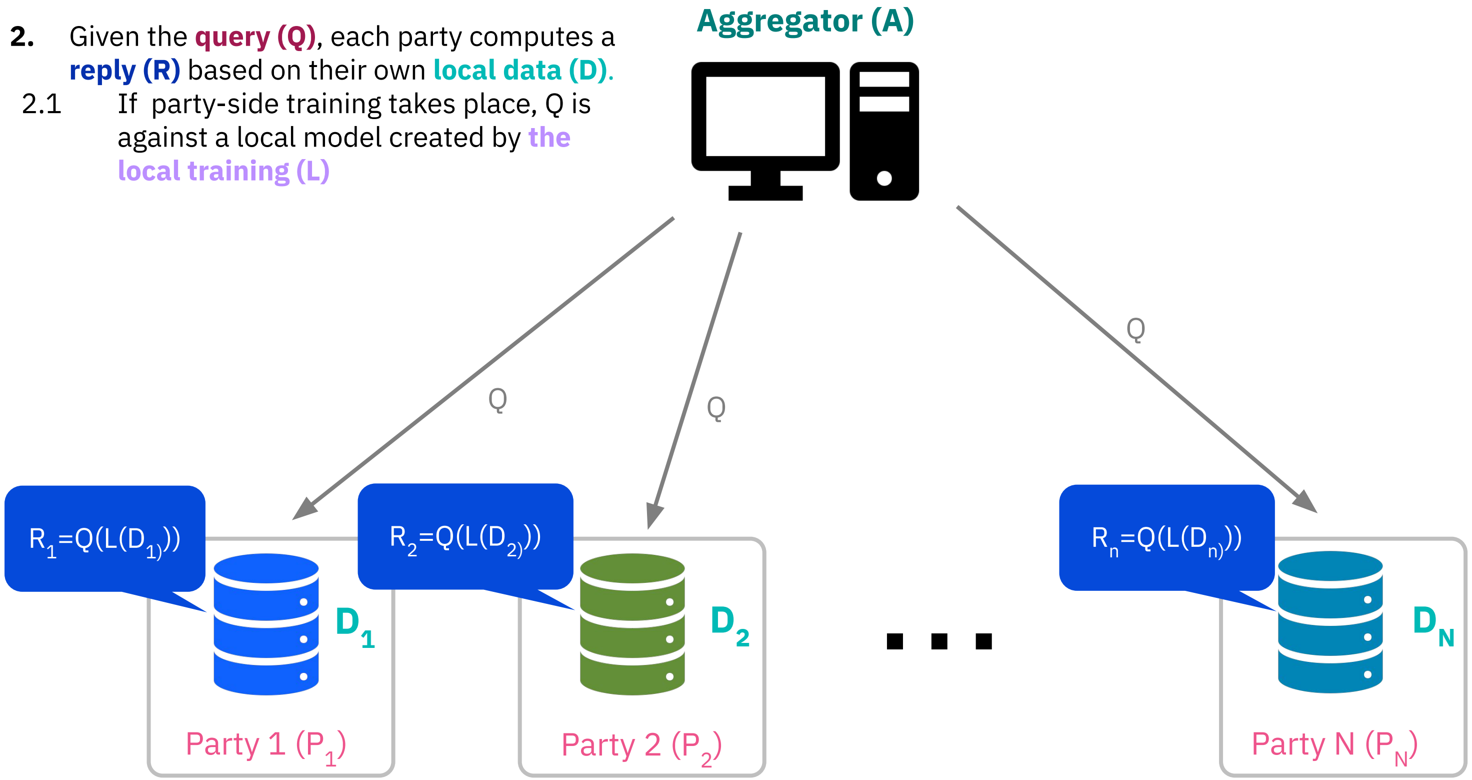
Federated Learning

Architecture

1. **Aggregator** queries **parties** along with information required for learning a model.

2. Given the **query (Q)**, each party computes a **reply (R)** based on their own **local data (D)**.

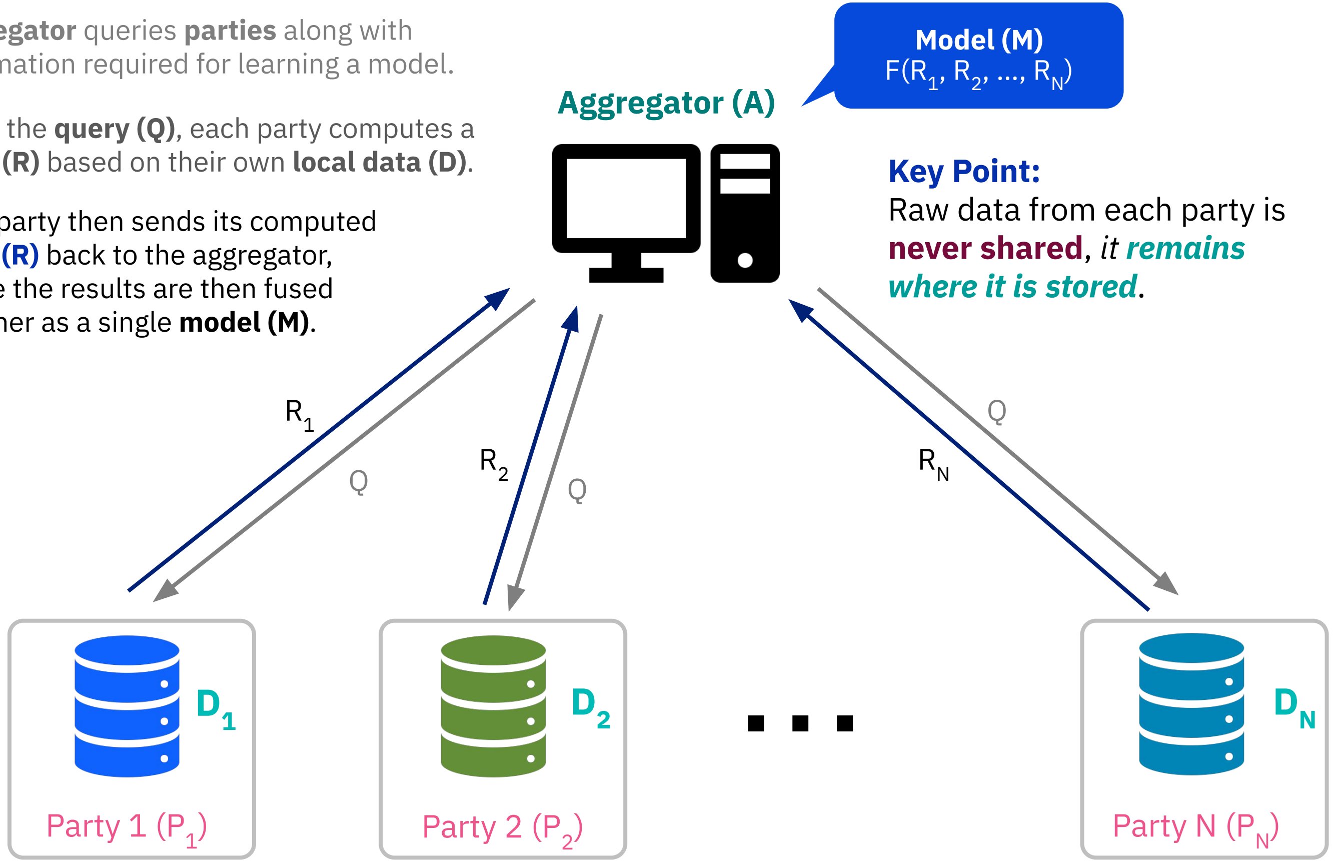
   2.1 If party-side training takes place, Q is against a local model created by **the local training (L)**



Federated Learning

Architecture

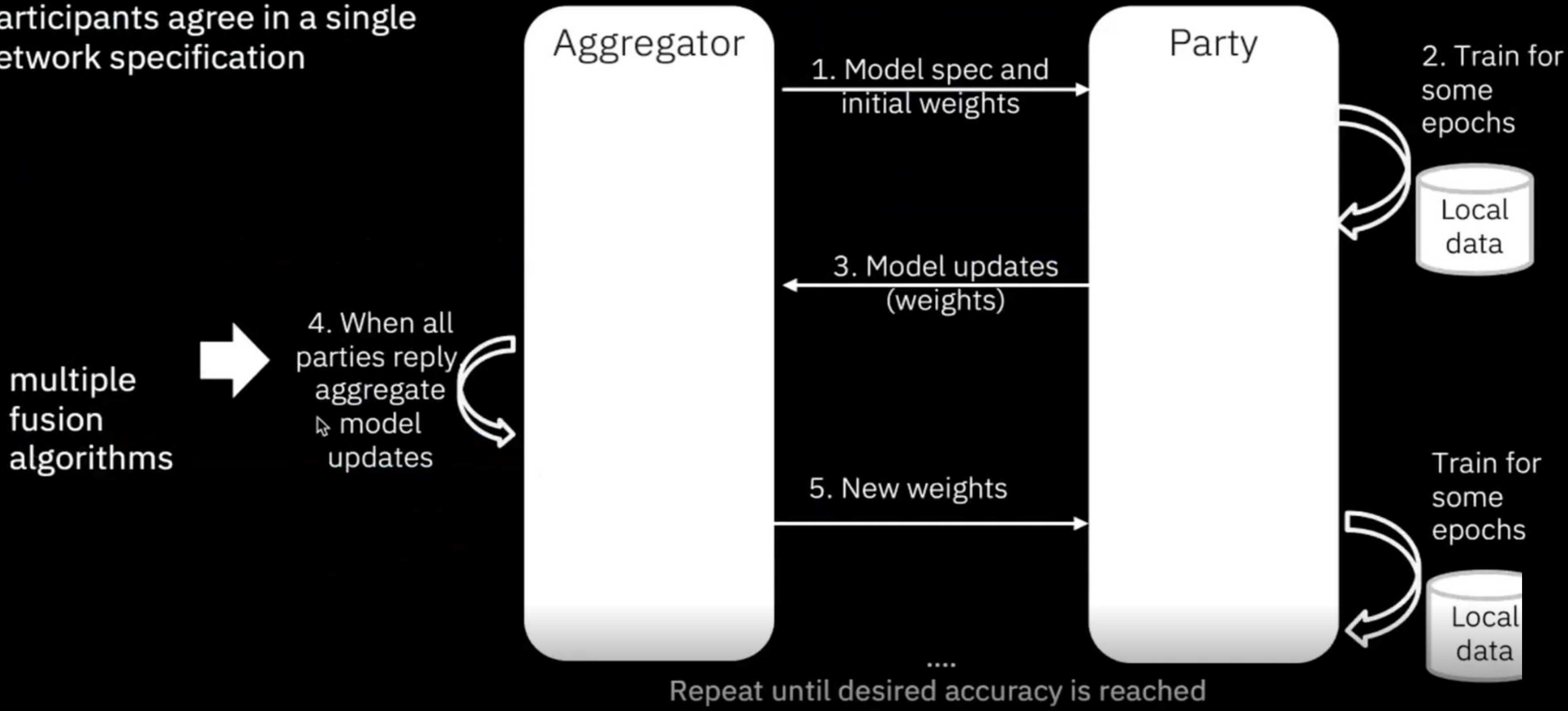
1. **Aggregator** queries **parties** along with information required for learning a model.
2. Given the **query (Q)**, each party computes a **reply (R)** based on their own **local data (D)**.
3. Each party then sends its computed **reply (R)** back to the aggregator, where the results are then fused together as a single **model (M)**.



Key Point:
Raw data from each party is **never shared**, *it* ***remains where it is stored***.

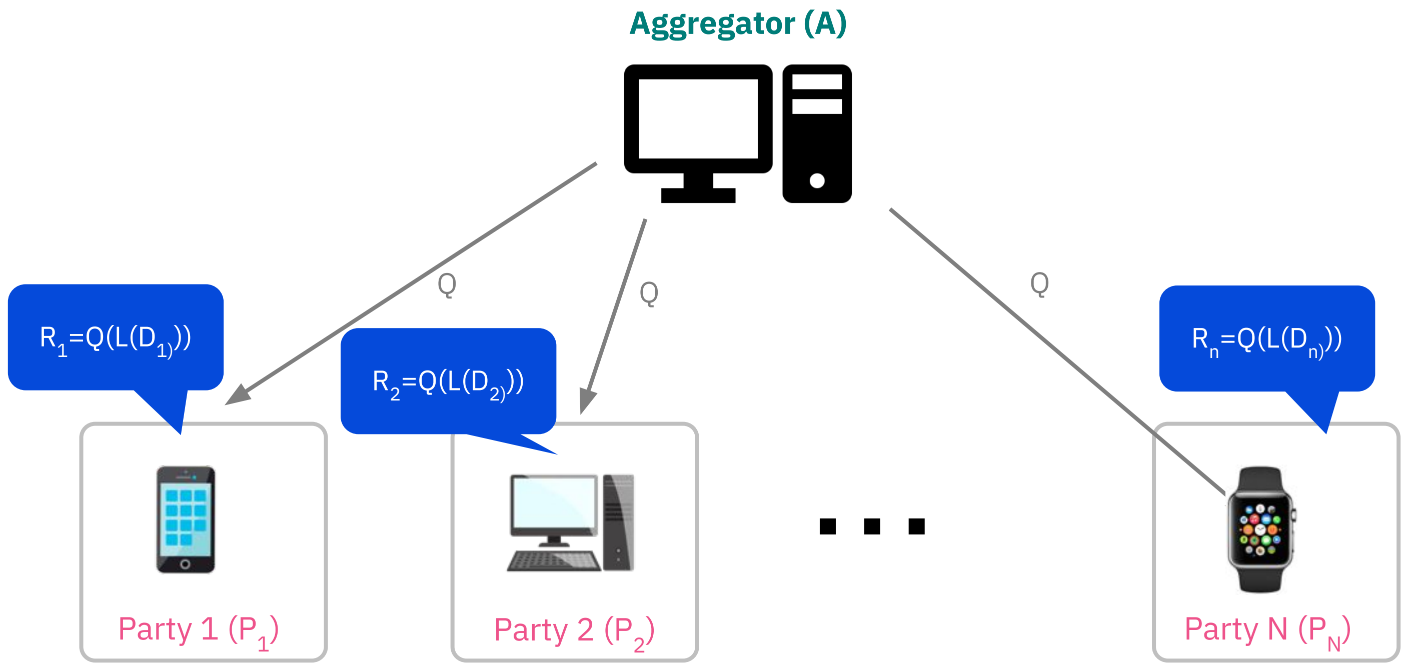
Neural networks in federated learning settings

Participants agree in a single network specification


Aggregator
Party
1. Model spec and initial weights
2. Train for some epochs
Local data
3. Model updates (weights)
multiple fusion algorithms
4. When all parties reply, aggregate model updates
5. New weights
Train for some epochs
Local data
....
Repeat until desired accuracy is reached

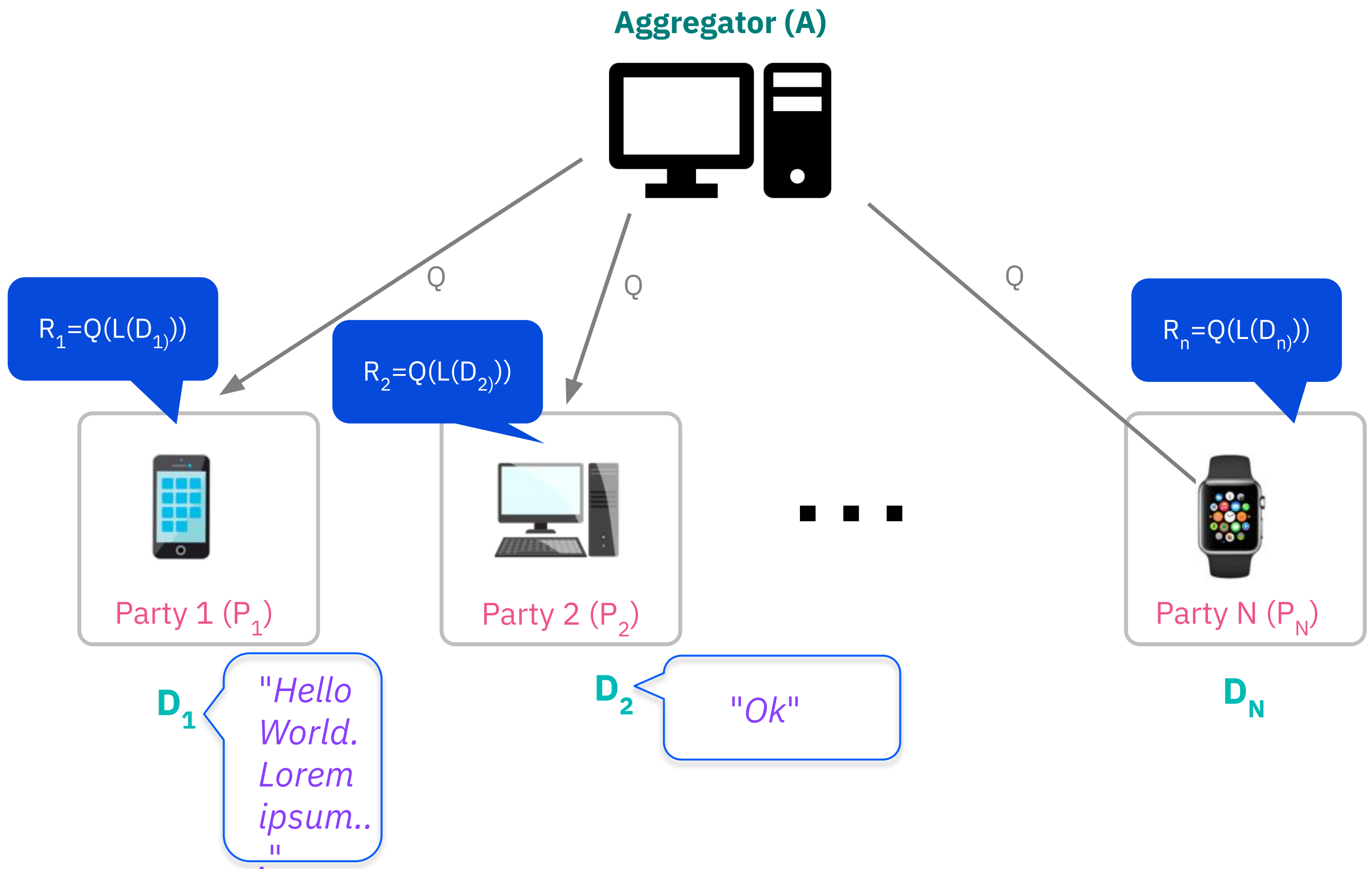
Challenges

Resource Heterogeneity



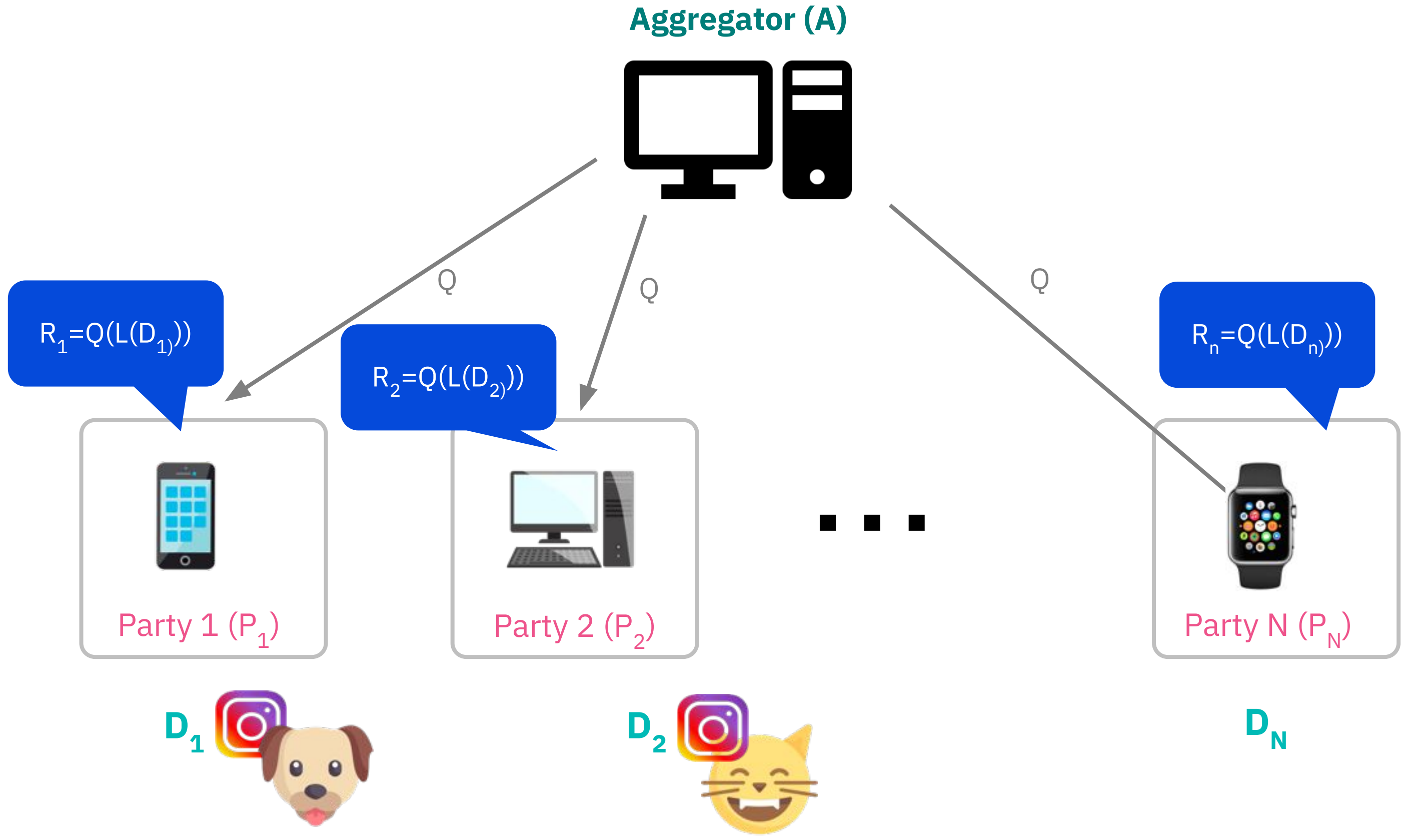
Challenges

Data Heterogeneity: Quantity



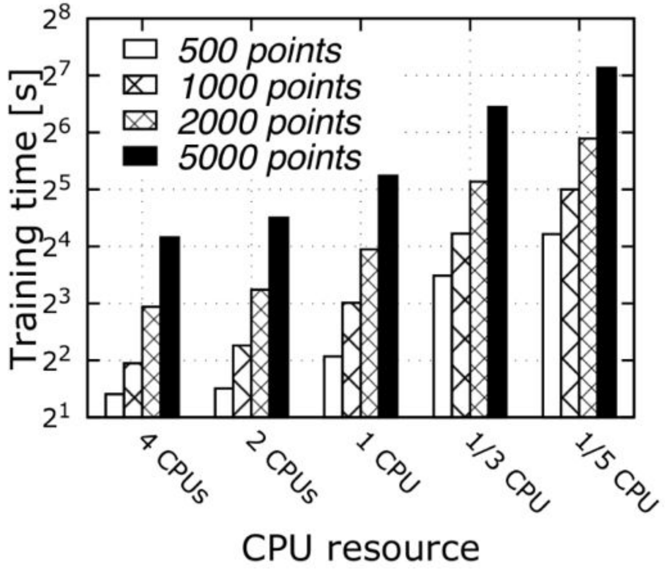
Challenges

Data Heterogeneity: Quality



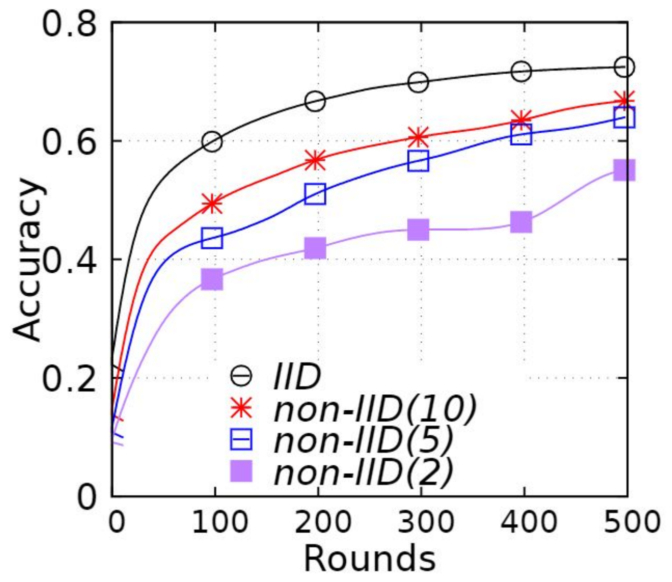
Resource + Data Quantity Heterogeneity

(Resource + Data Quantity) Heterogeneity impact training time

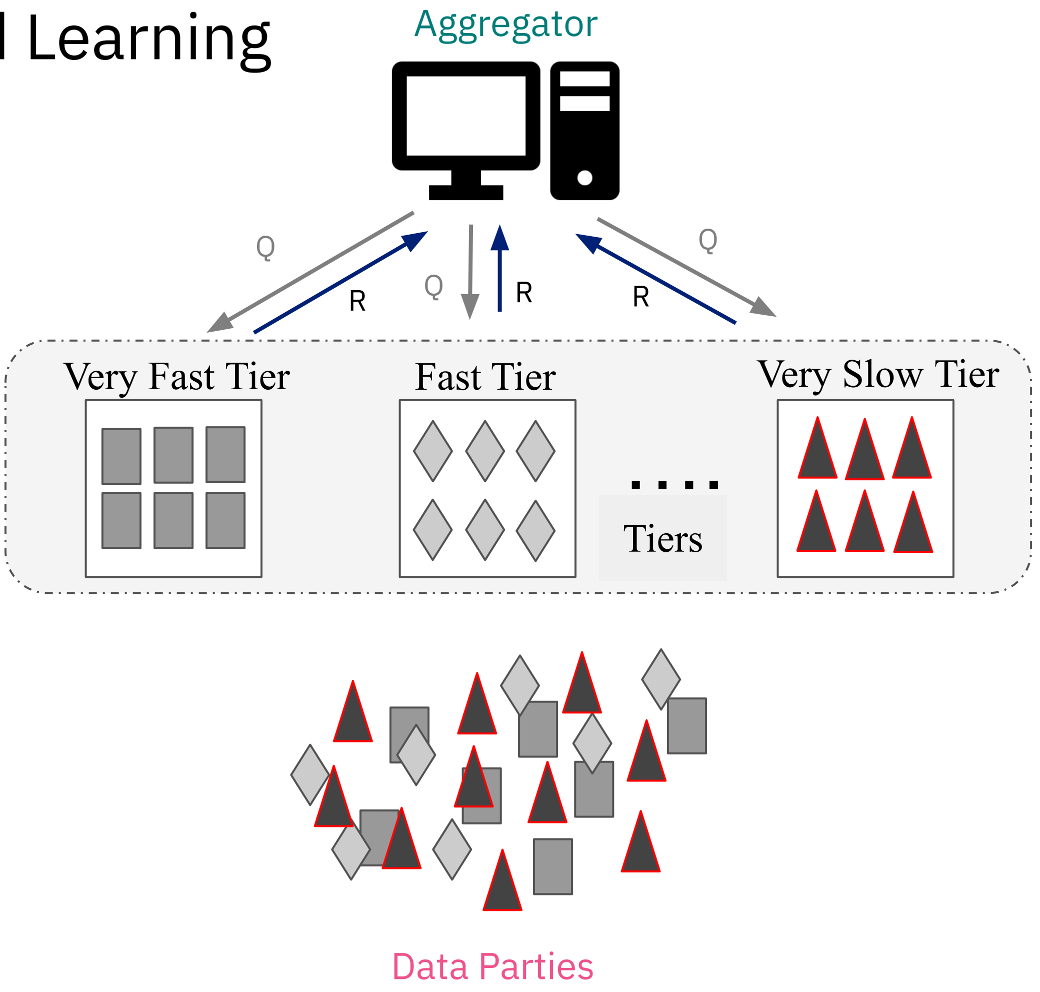


Data Quality Heterogeneity

Data Quality Heterogeneity impacts model performance



Tiered Federated Learning

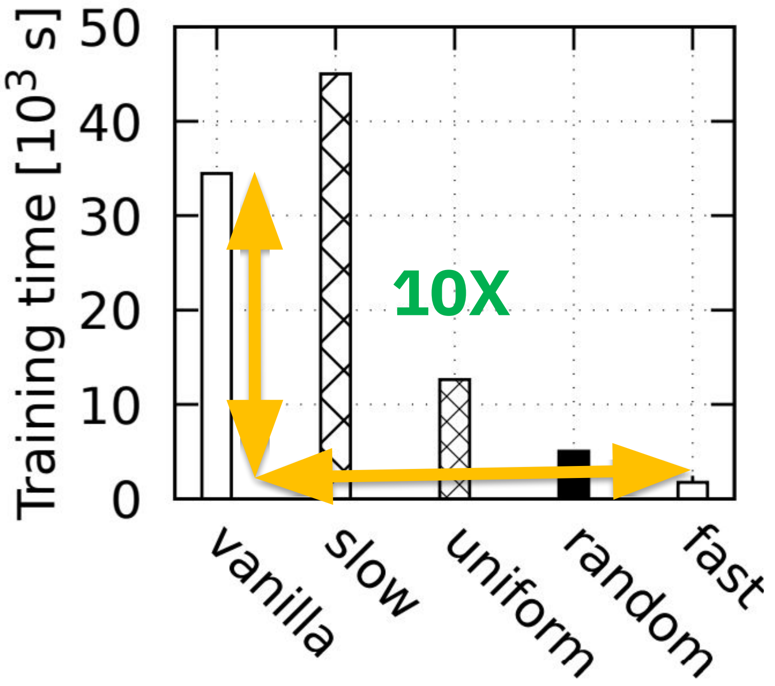


Setup details

- CIFAR10: Synthetic Federated Learning dataset
- FEMINIST: Practical Federated Learning benchmark

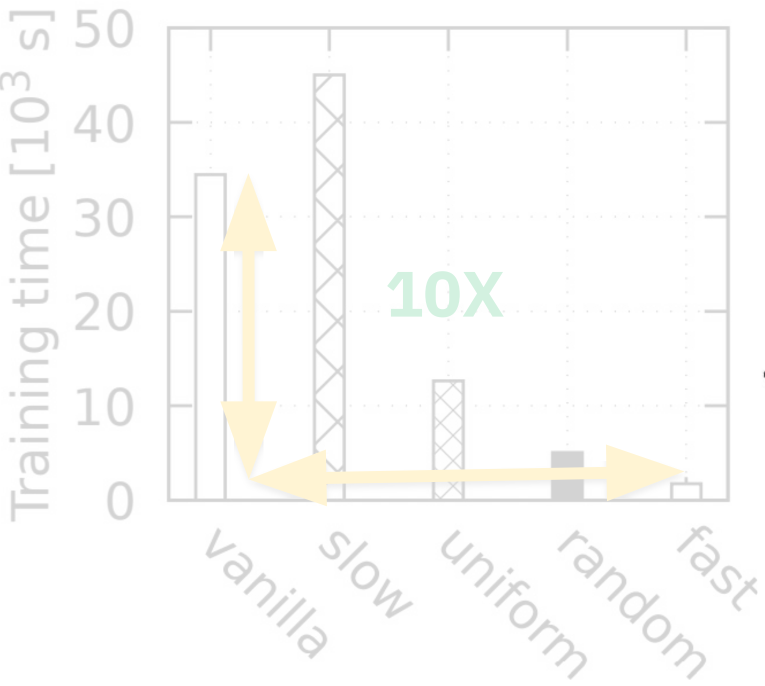
Datas et	Policy	Selection Probability				
		Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Cifar10/ FEMNIST	Vanilla	N/A	N/A	N/A	N/A	N/A
	Slow	0.0	0.0	0.0	0.0	1.0
	Uniform	0.2	0.2	0.2	0.2	0.2
	Random	0.7	0.1	0.1	0.05	0.05
	Fast	1.0	0.0	0.0	0.0	0.0

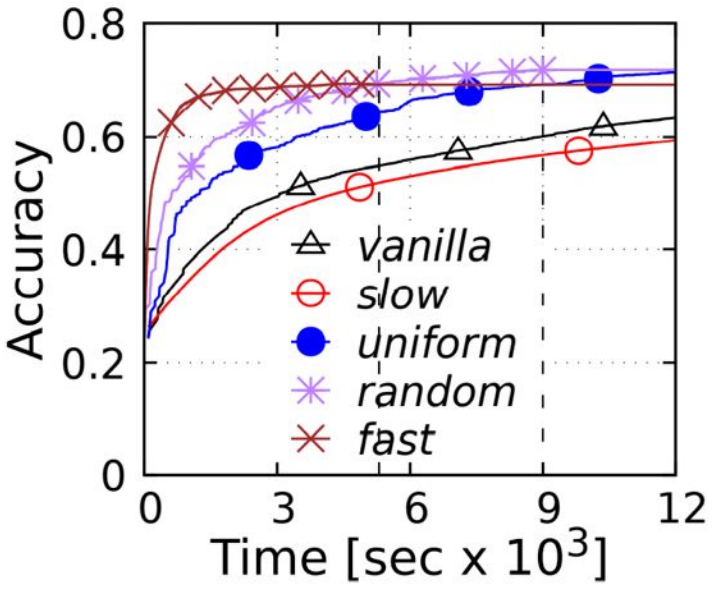
Resource Heterogeneity
Homogeneous Data (Quantity + Quality)



10X training speedup

Resource Heterogeneity
Homogeneous Data (Quantity + Quality)

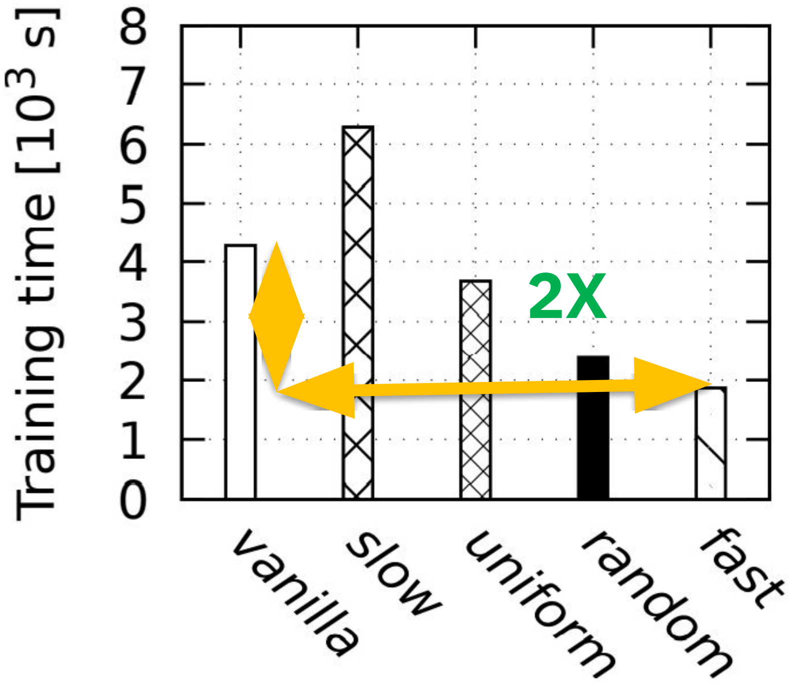




10X training speedup

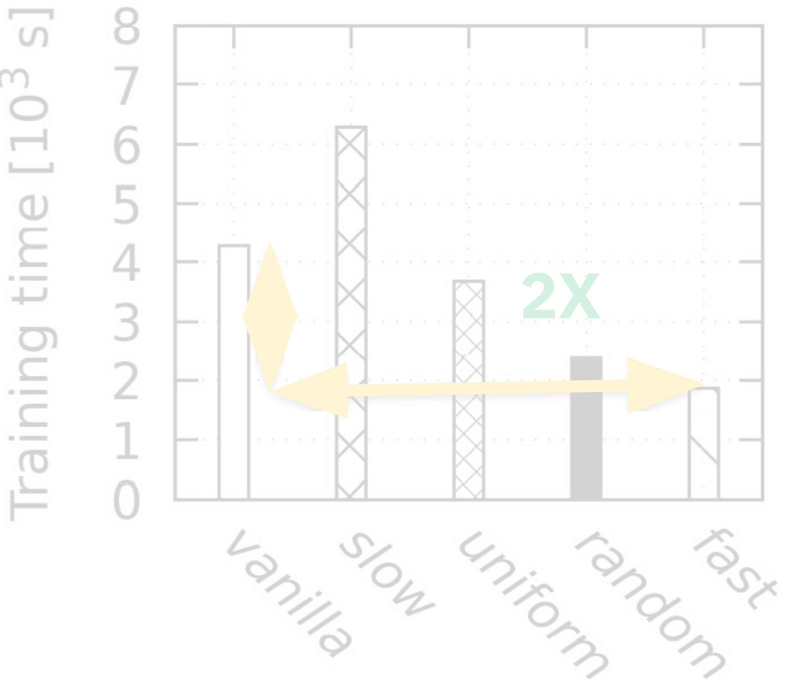
Higher accuracy

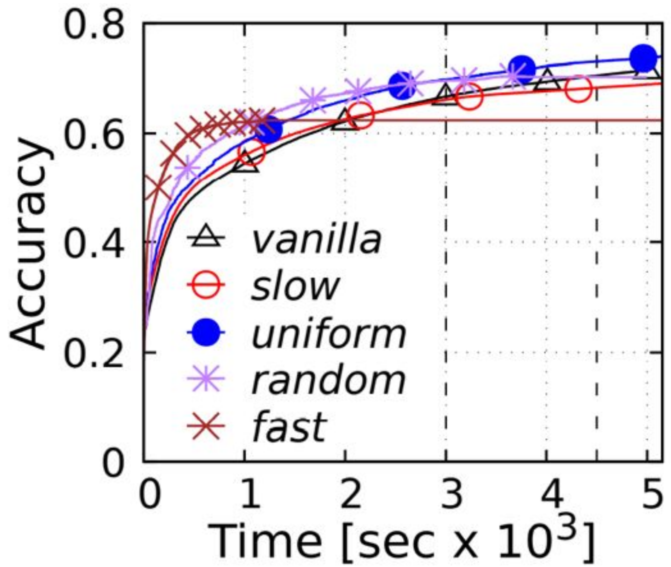
Data Quantity Heterogeneity
Homogeneous (Resource + Data Quality)



2X training speedup

Data Quantity Heterogeneity
Homogeneous (Resource + Data Quality)

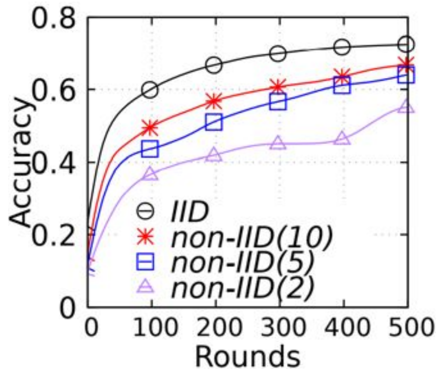




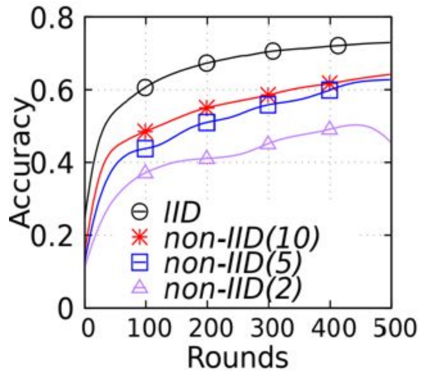
2X training speedup

Lower accuracy

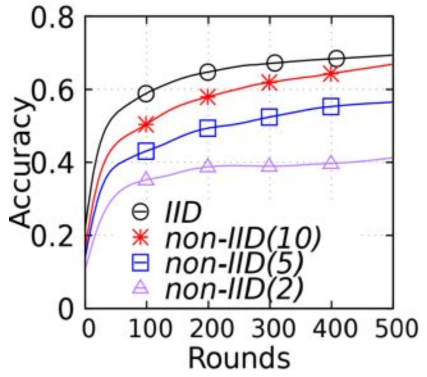
Data Quality Heterogeneity
Homogeneous (Resource + Data Quantity)



Vanilla



Uniform



Fast

non-IIDness leads to lower accuracy

Prioritizing makes it worse

Take away

- Model Performance $\longleftrightarrow$ Training Time
- Prioritizing some tiers over others causes biasness
- No single static selection policy achieves faster training with an efficient model

Important question

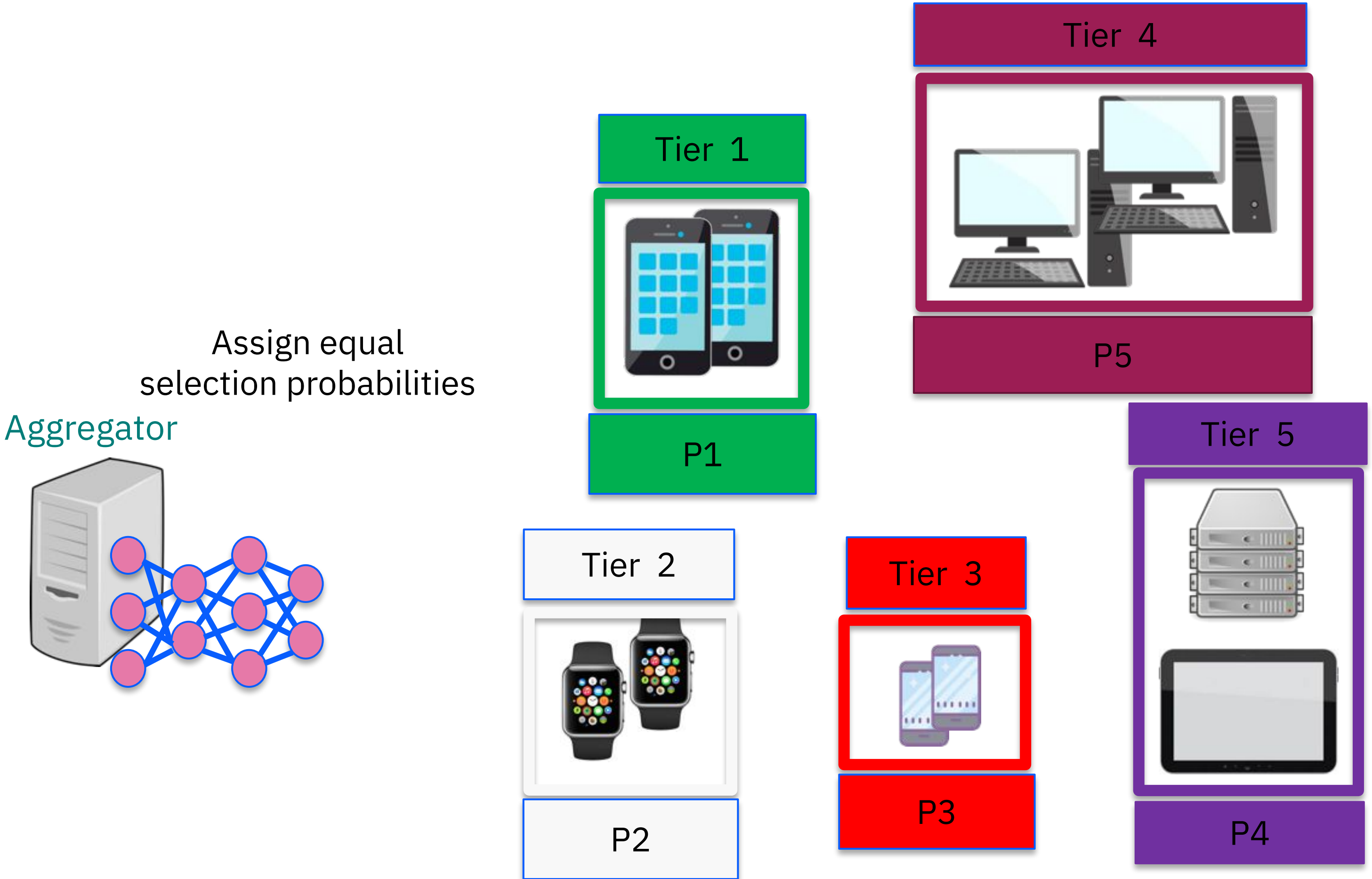
- Model Performance ⟷ Training Time
- Prioritizing some tiers over others causes biasness
- No single static selection policy achieves faster training with an efficient model

Can we achieve faster training with higher accuracy?

Adaptive tier selection

TiFL

Tiered Federated Learning

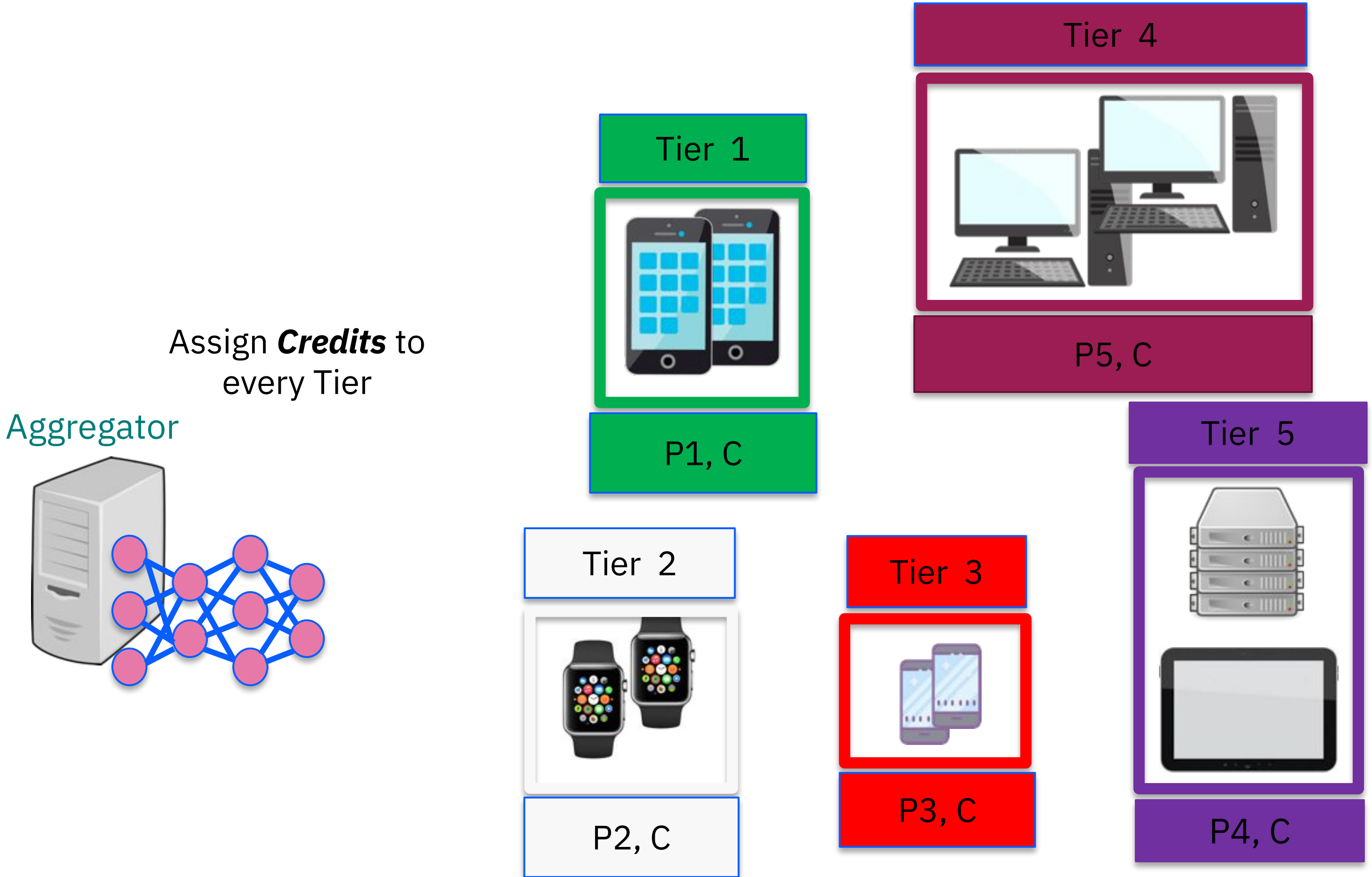


Adaptive tier selection

TiFL

Tiered Federated Learning

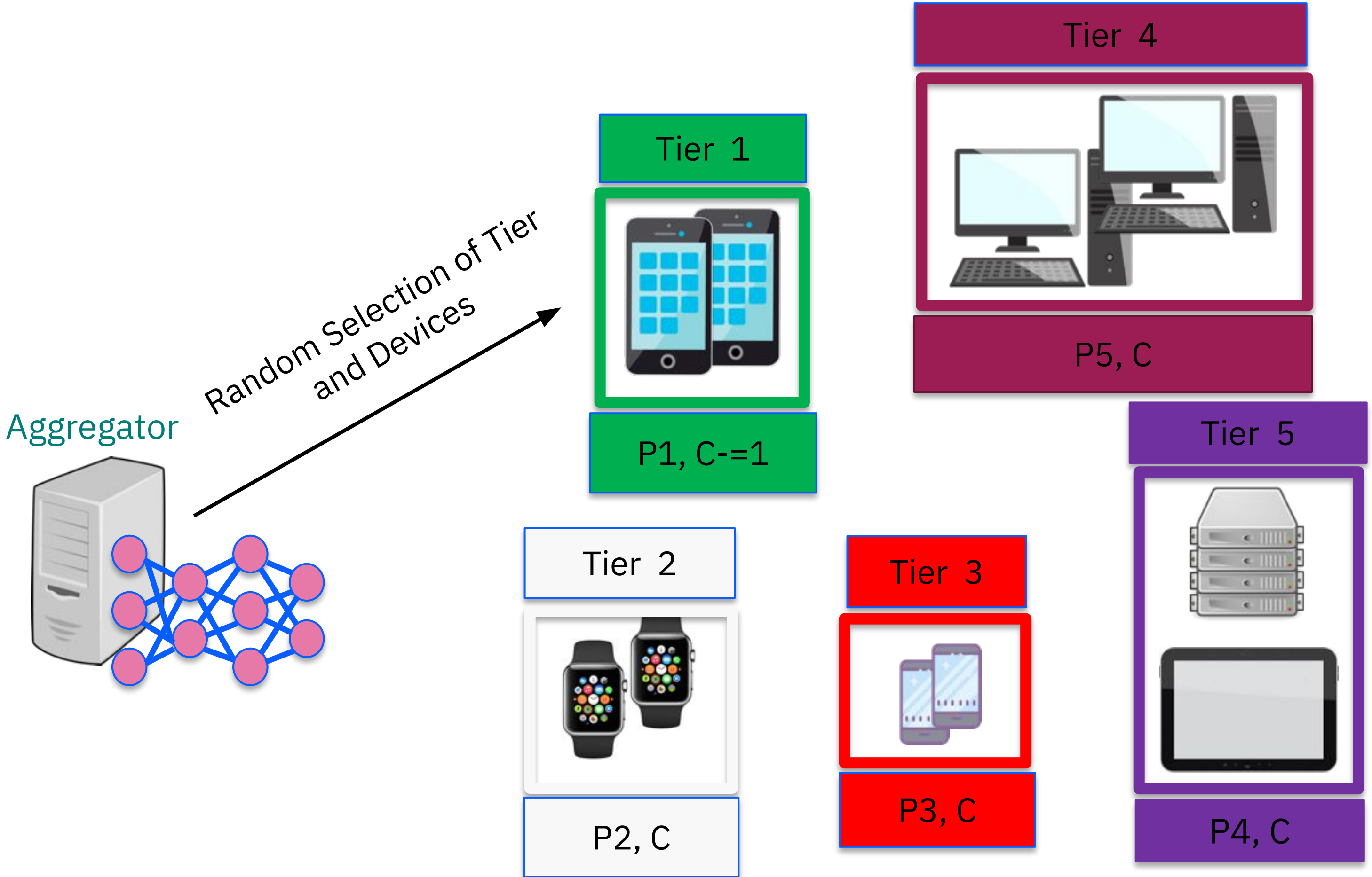
Assign ***Credits*** to every Tier



TiFL

Adaptive tier selection

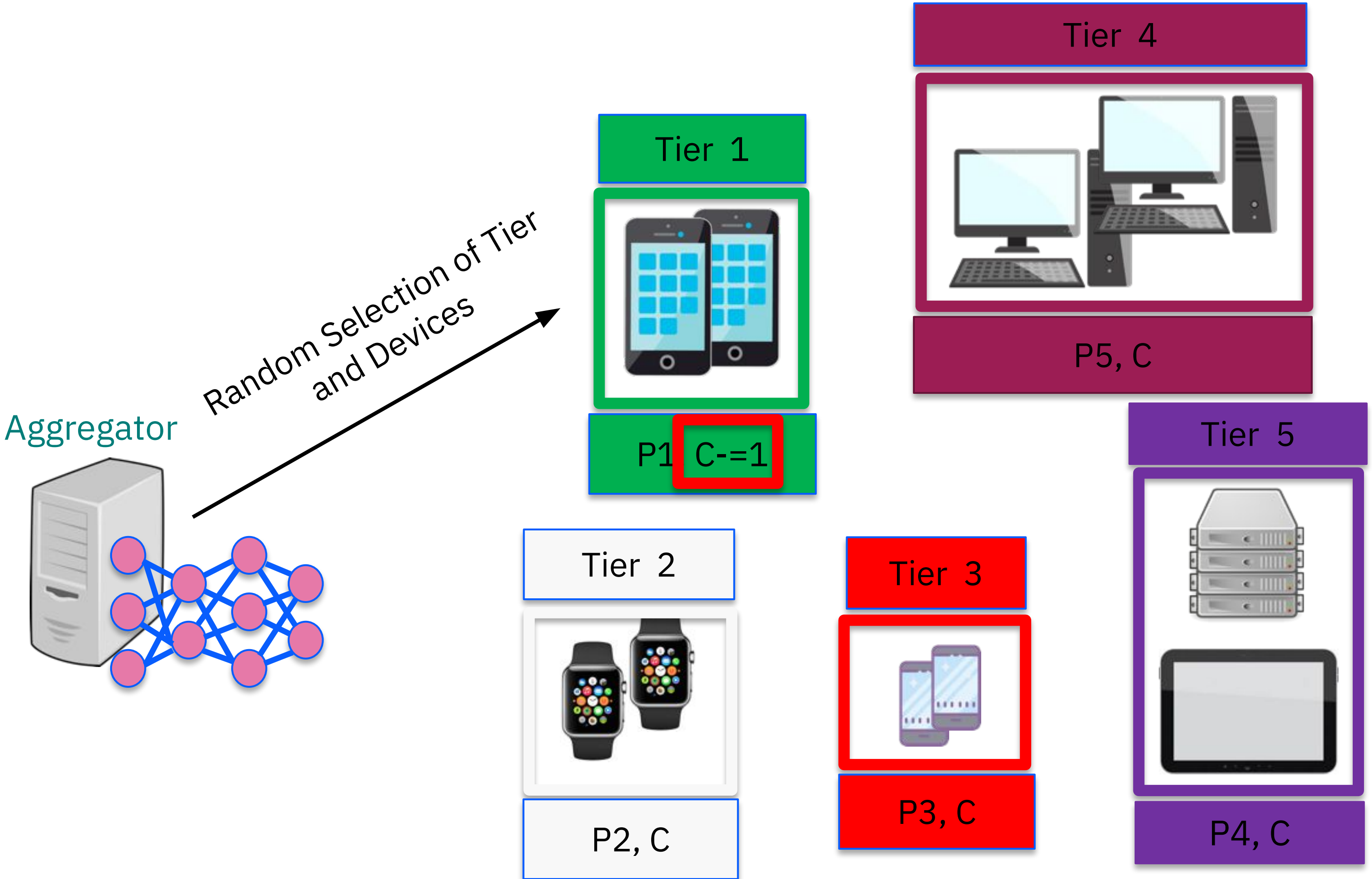
Tiered Federated Learning



Adaptive tier selection

TiFL

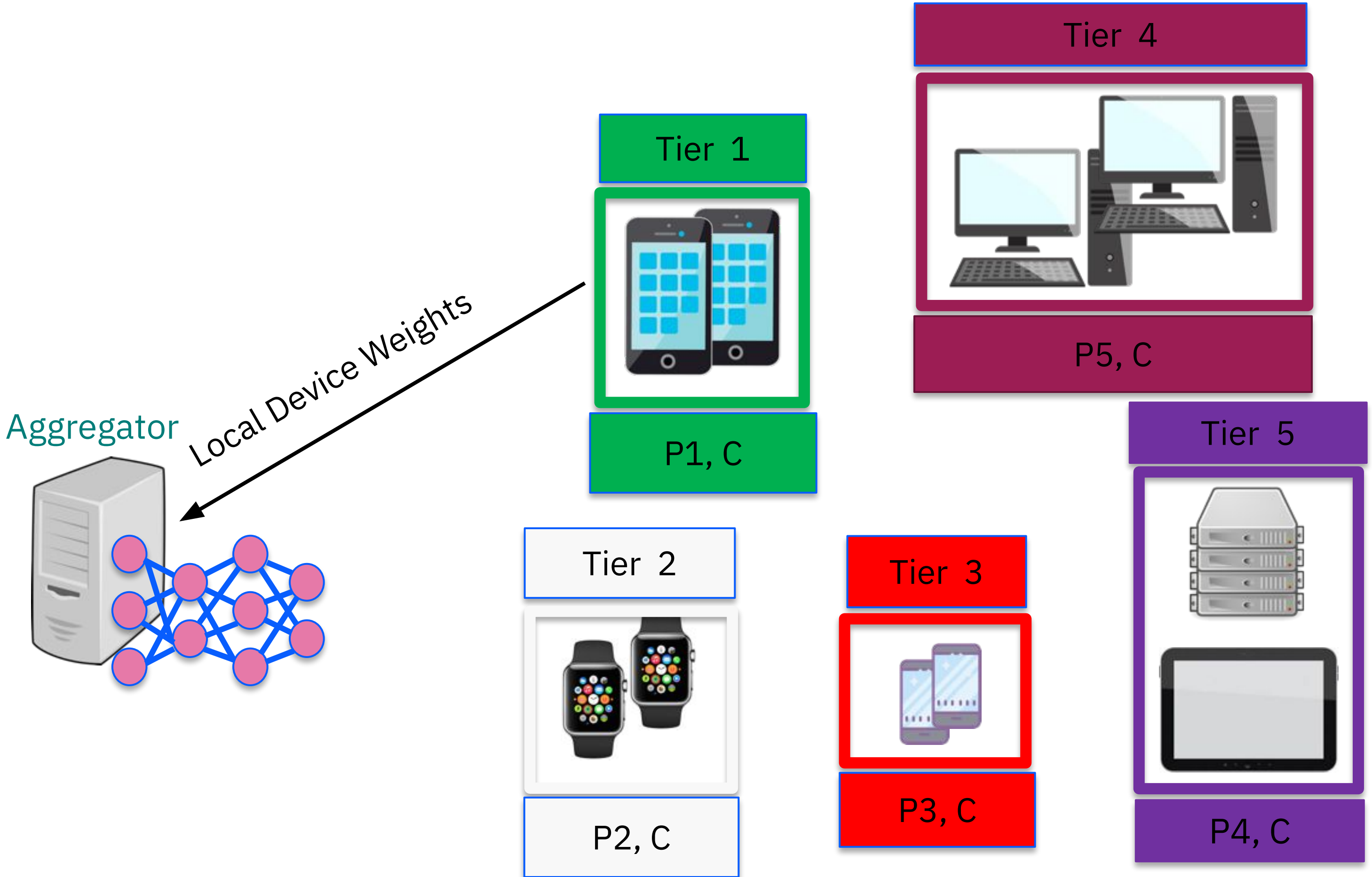
Tiered Federated Learning



Adaptive tier selection

TiFL

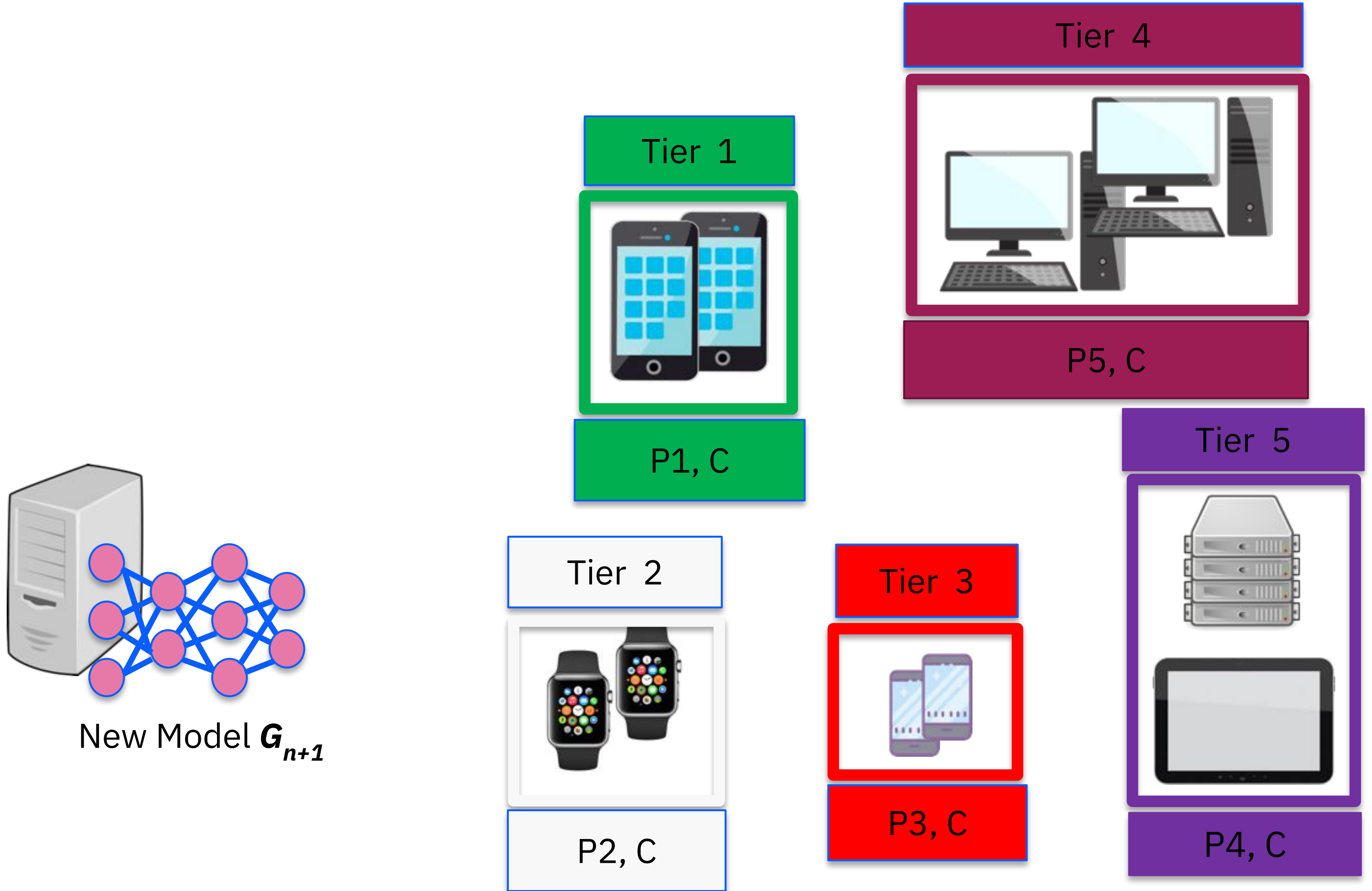
Tiered Federated Learning



Adaptive tier selection

TiFL

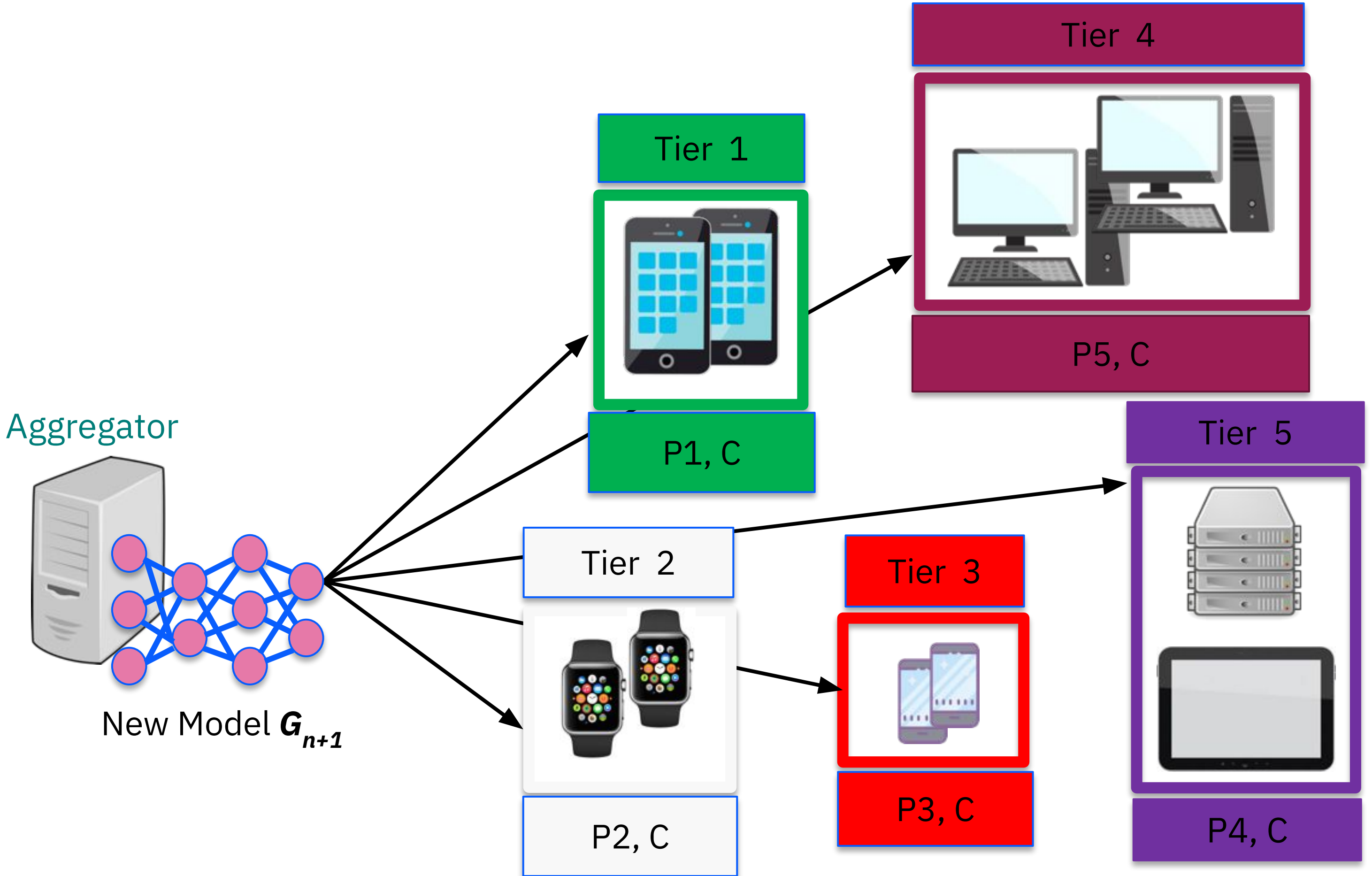
Tiered Federated Learning



TiFL

Tiered Federated Learning

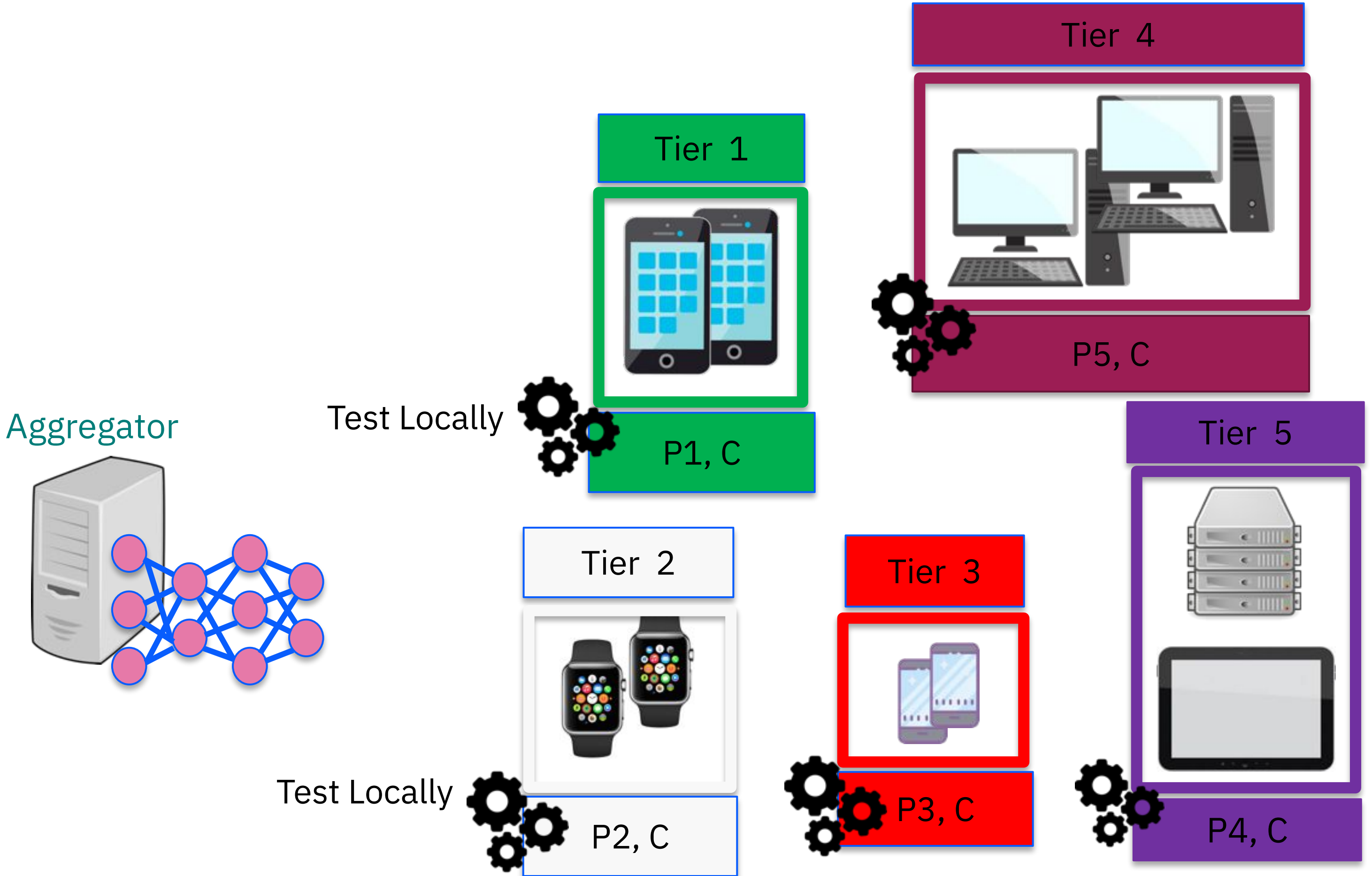
Adaptive tier selection



TiFL

Tiered Federated Learning

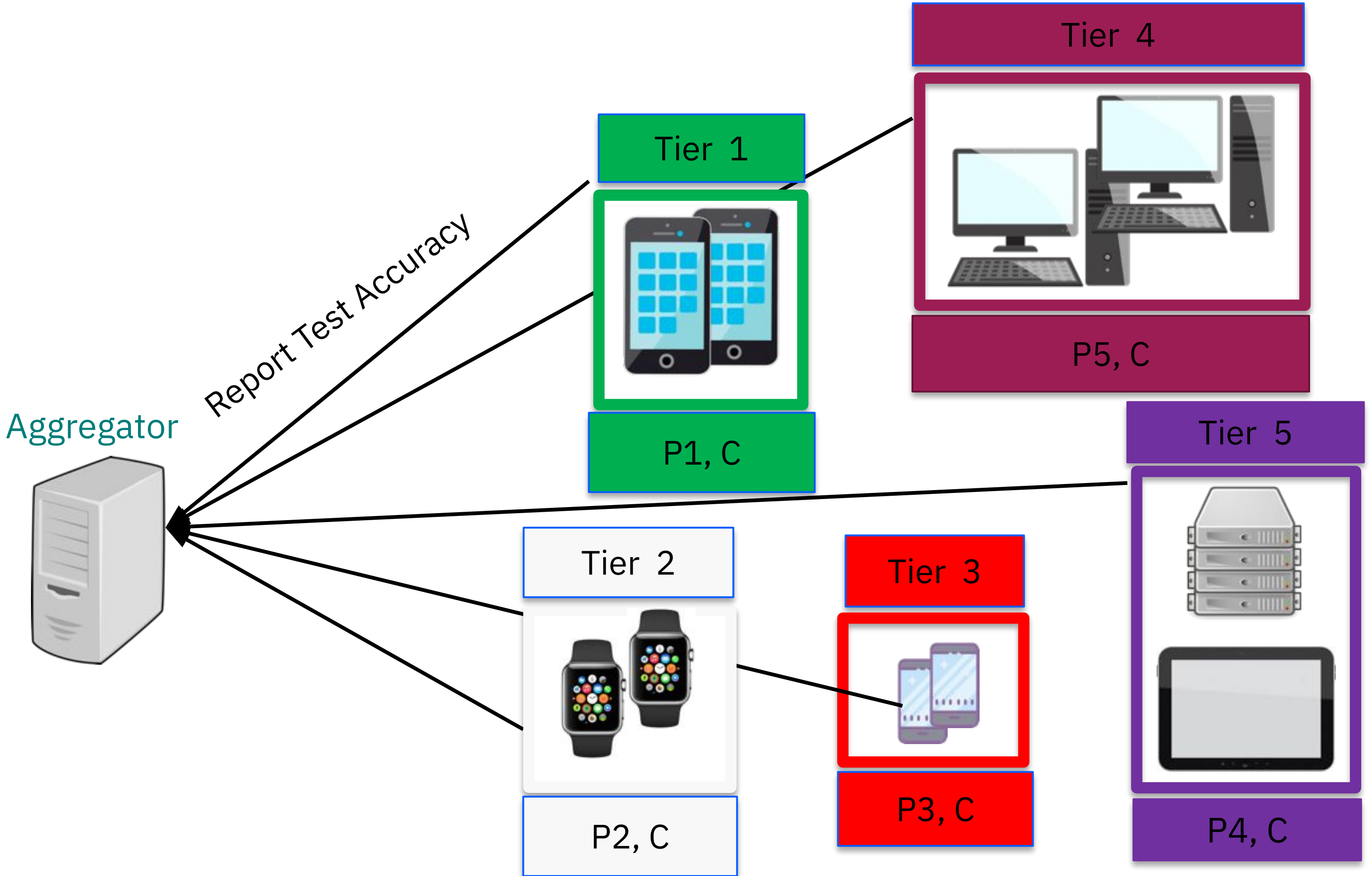
Adaptive tier selection



Adaptive tier selection

TiFL

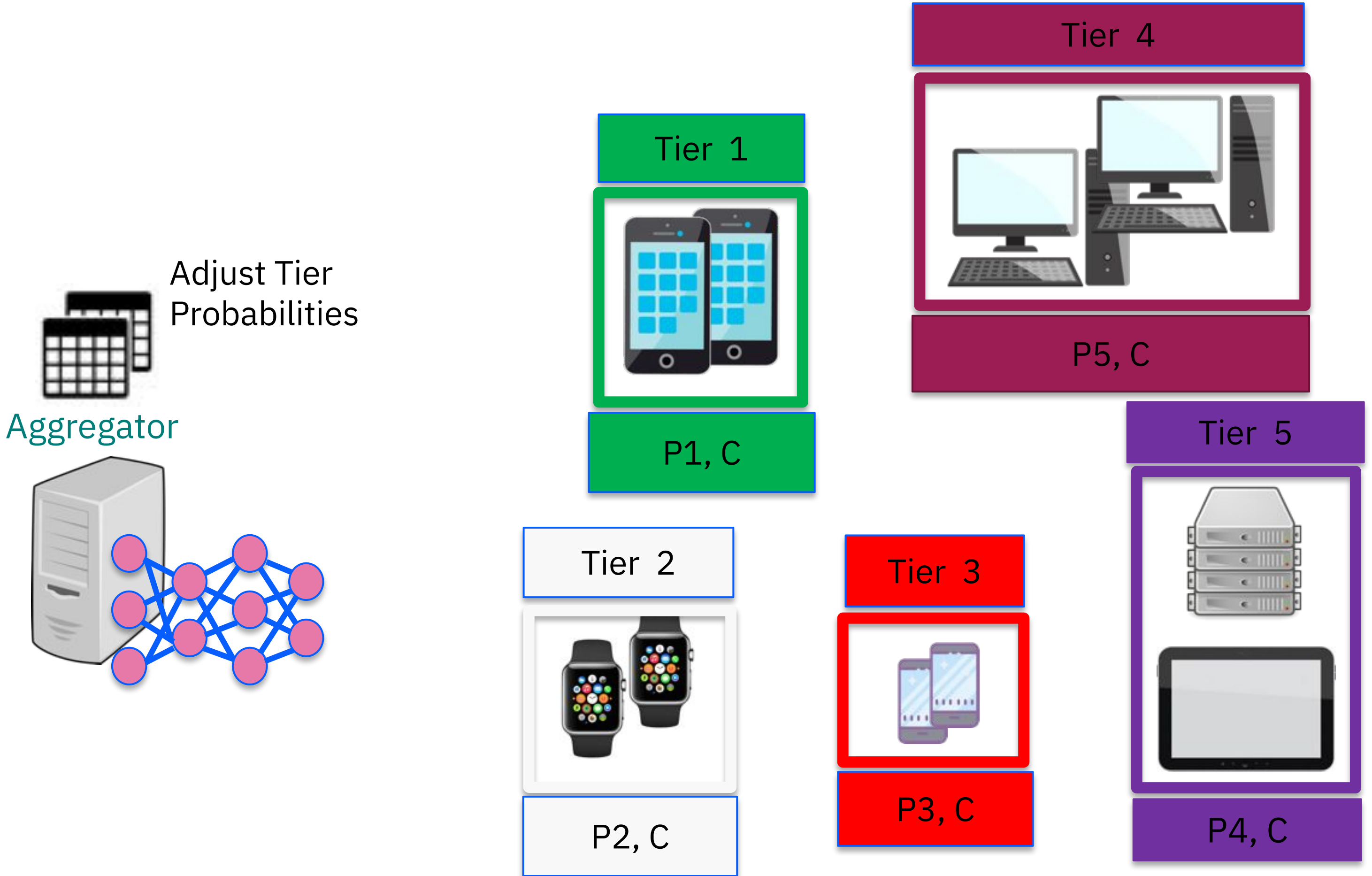
Tiered Federated Learning



Adaptive tier selection

TiFL

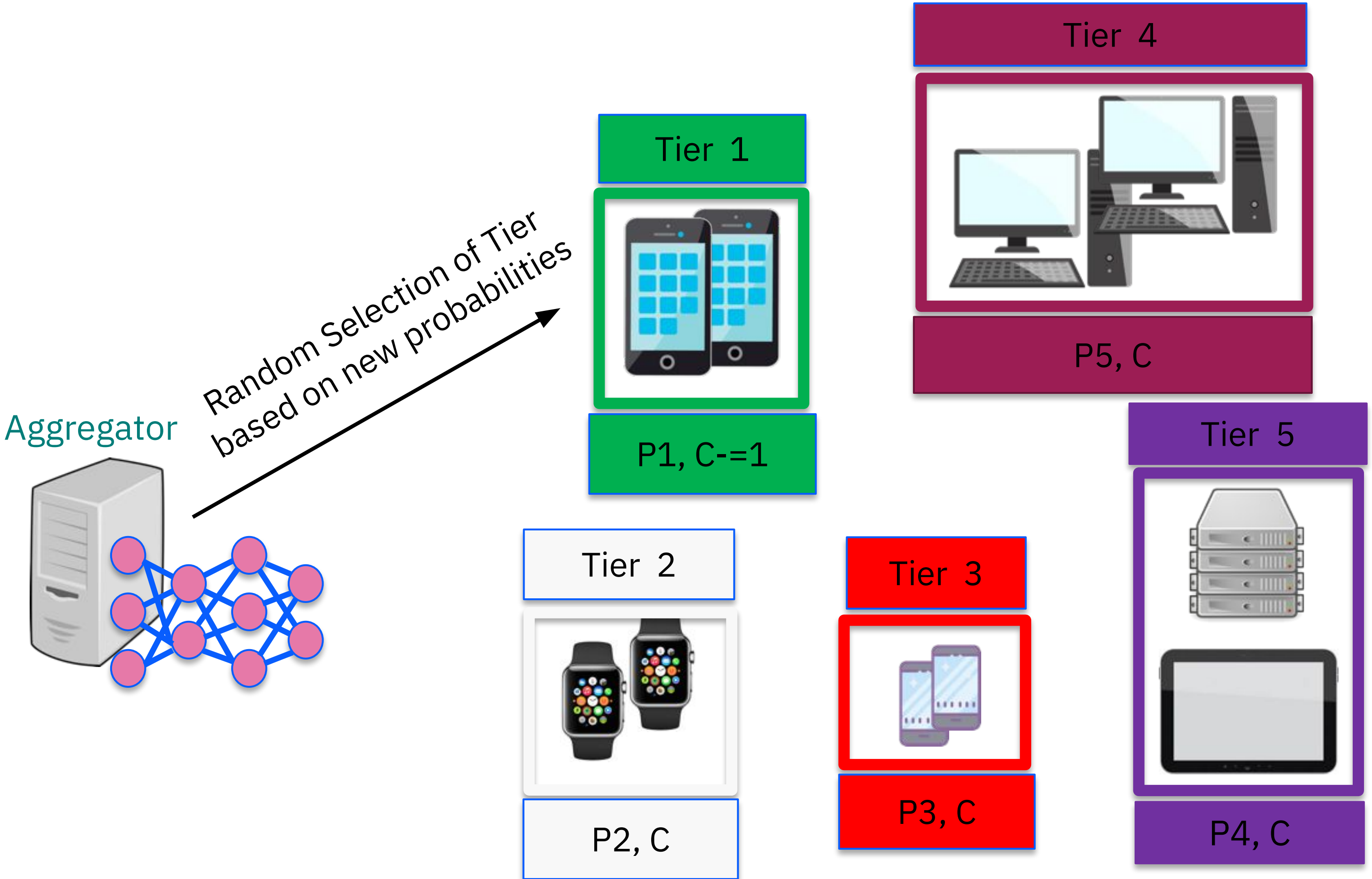
Tiered Federated Learning



Adaptive tier selection

TiFL

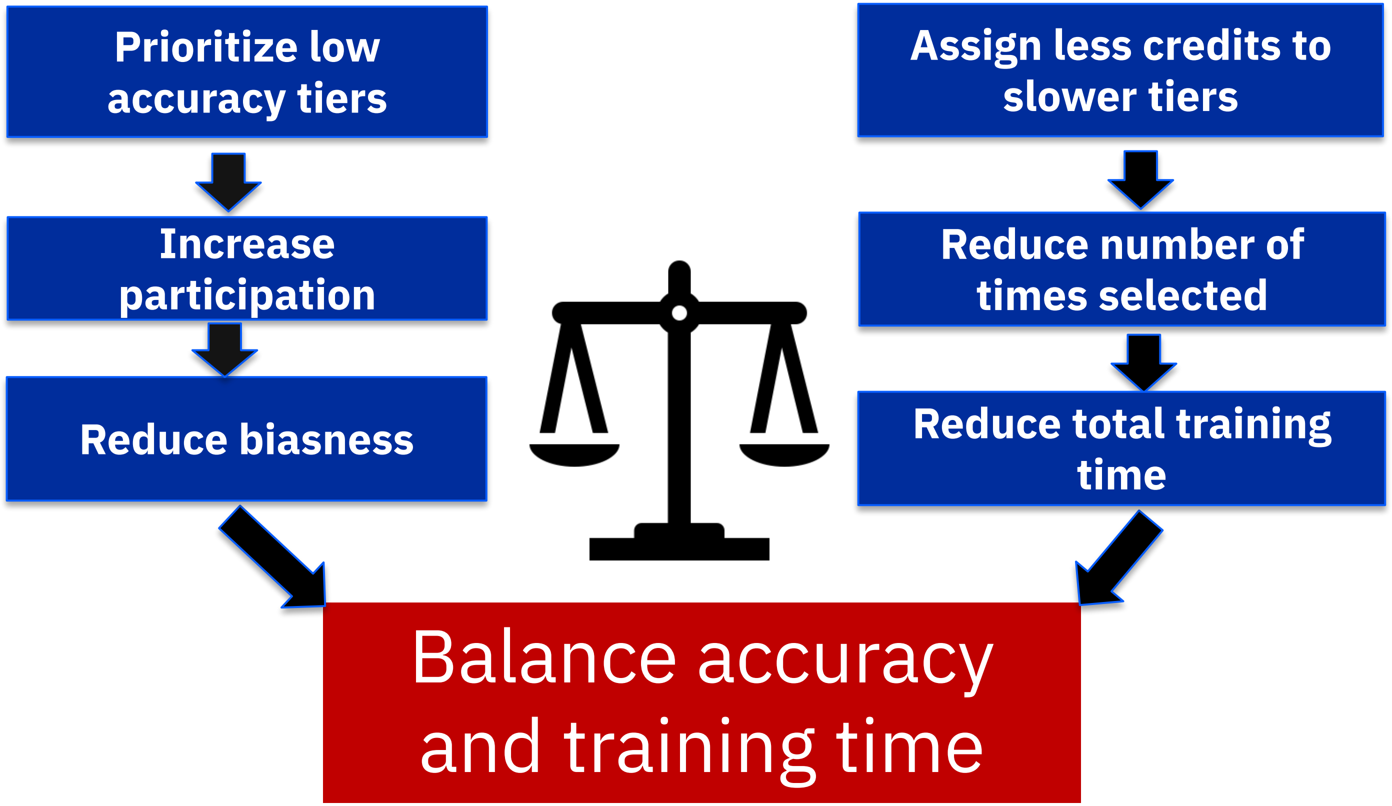
Tiered Federated Learning



Adaptive tier selection

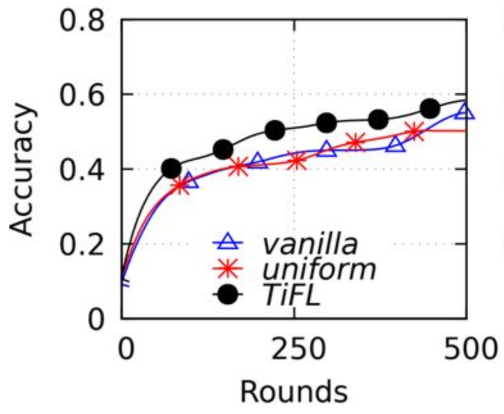
TiFL

Tiered Federated Learning

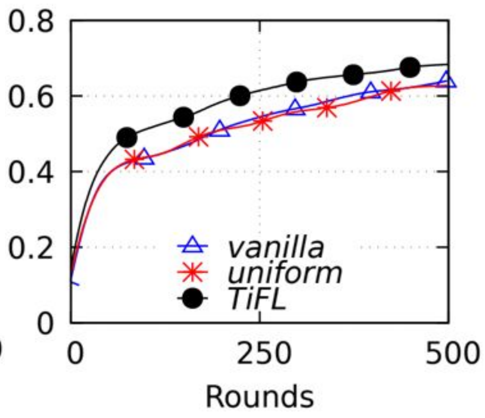


Prioritize low accuracy tiers
Increase participation
Reduce biasness
Assign less credits to slower tiers
Reduce number of times selected
Reduce total training time
Balance accuracy and training time

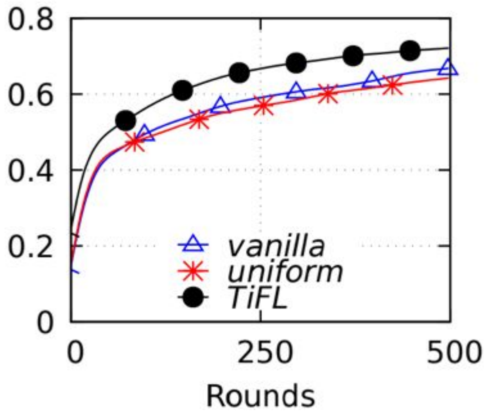
Data Quality Heterogeneity
Homogeneous (Resource + Data Quantity)



non-IID(2)



non-IID(5)

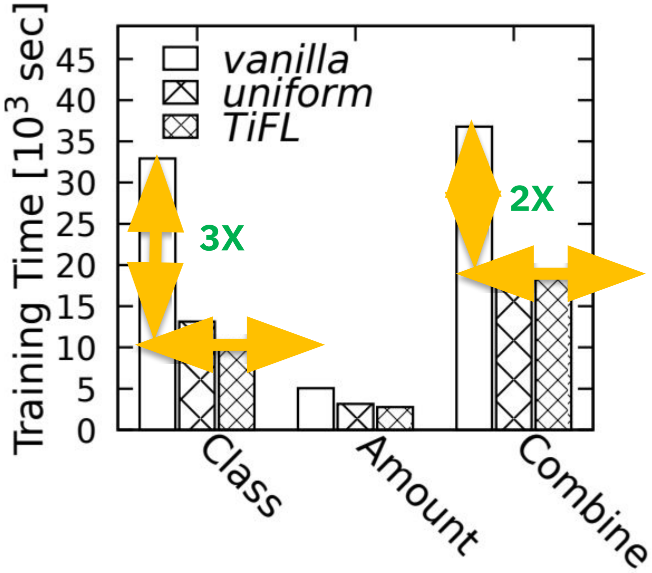


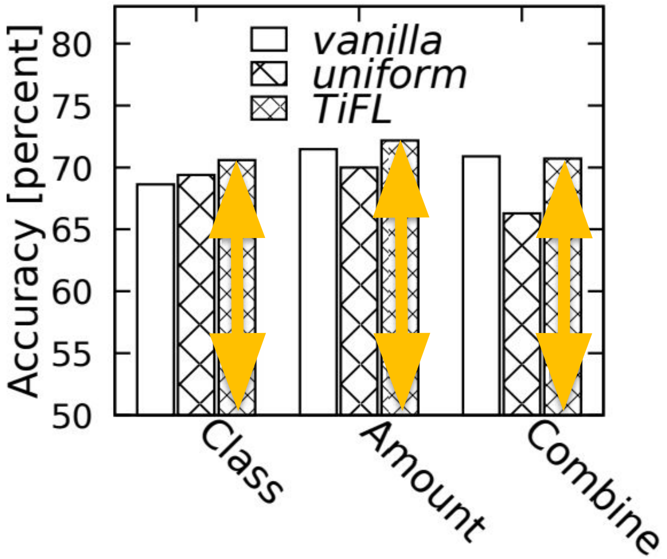
non-IID(10)

TiFL outperforms vanilla and uniform selection

TiFL Improves overall model performance

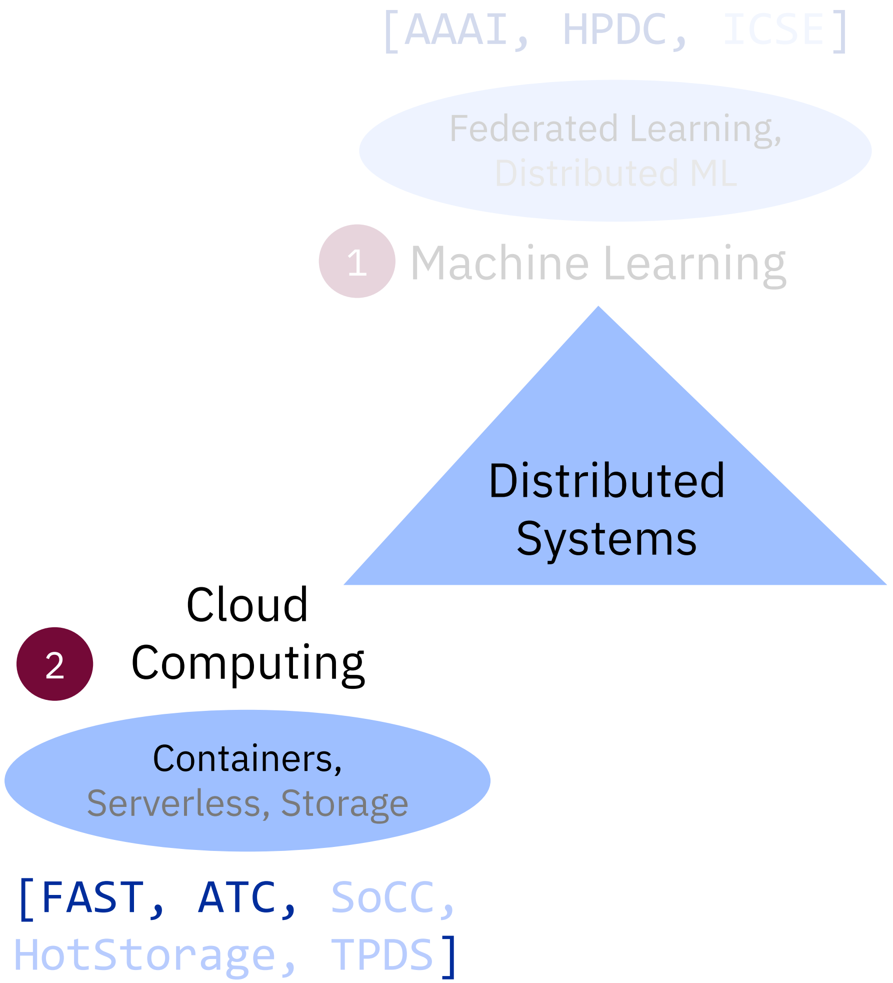
TiFL vs Static Selection
Heterogeneous (Resource + Data Quality + Data Quantity)



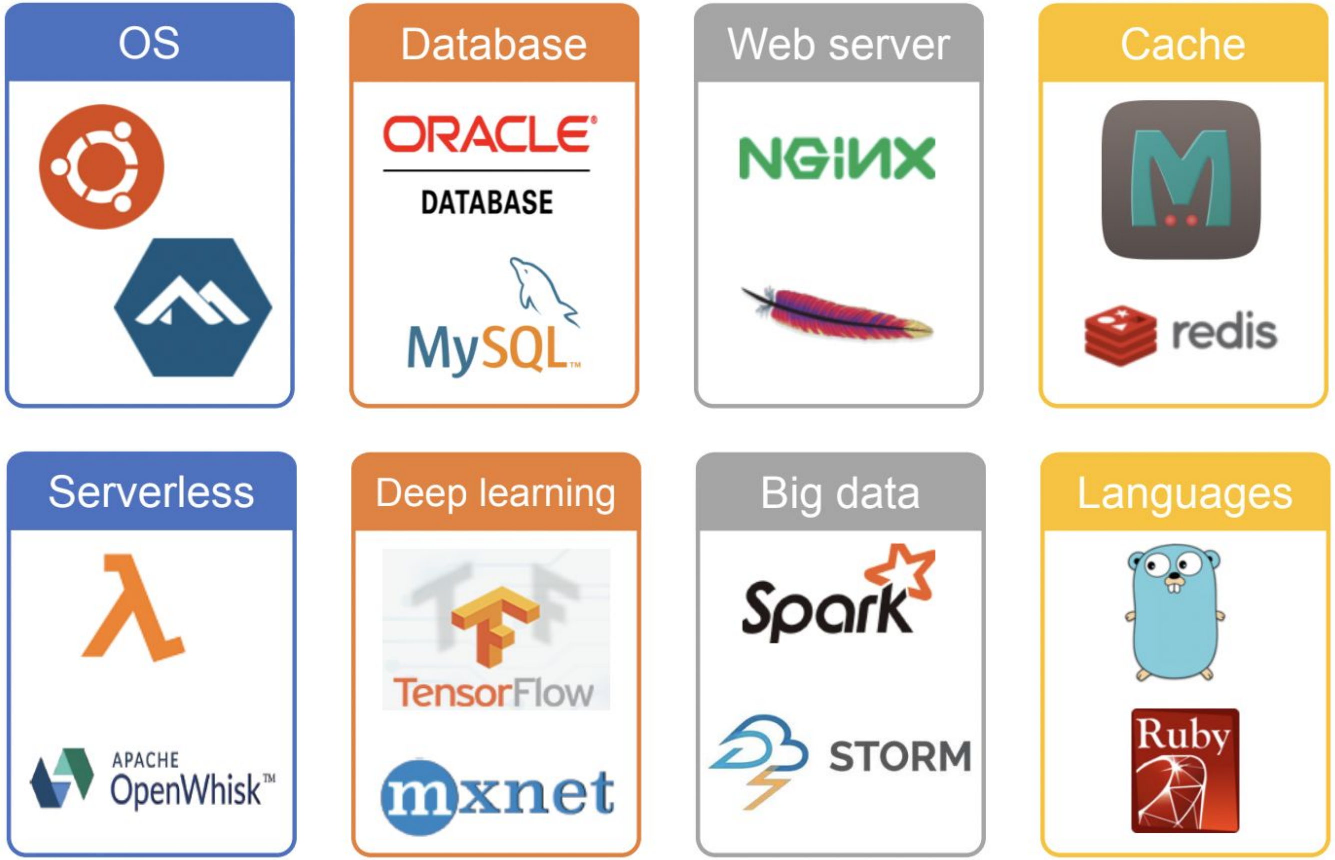


Achieves 3X to 2X training time speedup

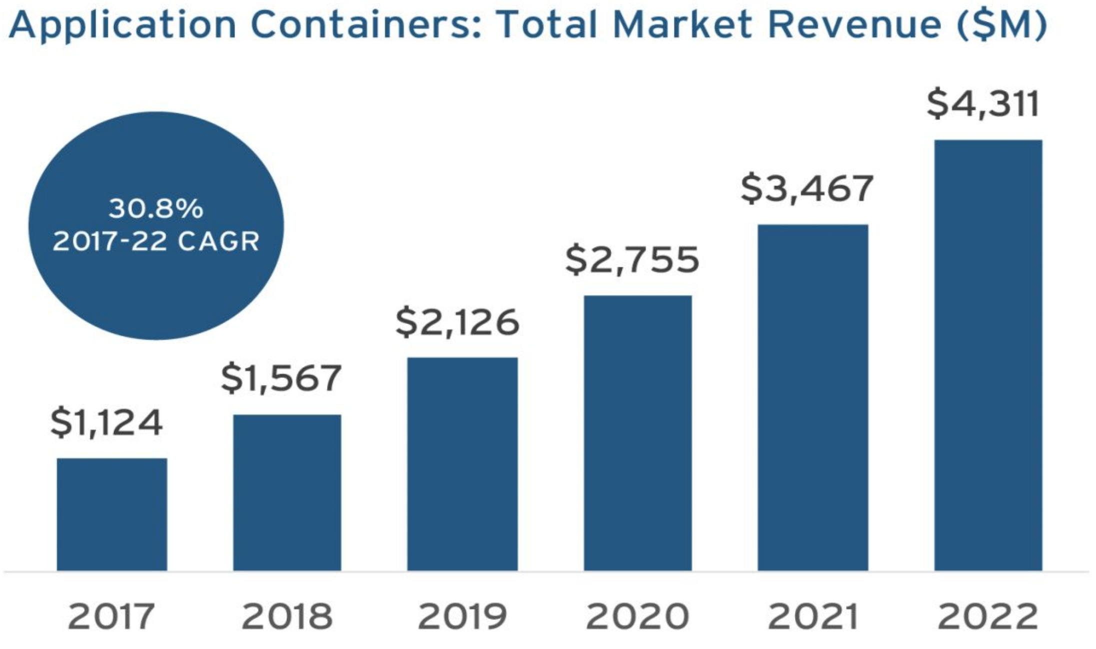
TiFL Achieves higher or on par accuracy in all cases


[AAAI, HPDC, ICSE]
Federated Learning,
Distributed ML
1
Machine Learning
Distributed
Systems
Cloud
Computing
2
Containers,
Serverless, Storage
[FAST, ATC, SoCC,
HotStorage, TPDS]

Containers are Ubiquitous



Application Containerization



Source: 451 Research's Market Monitor: Cloud-Enabling Technologies - Application Containers, November 2018

Container usage patterns remain a mystery

Docker is de-facto standard for datacenter container management

- How are Docker containers used and managed?
- How can we streamline Docker workflows?
- How do we facilitate Docker performance analysis?

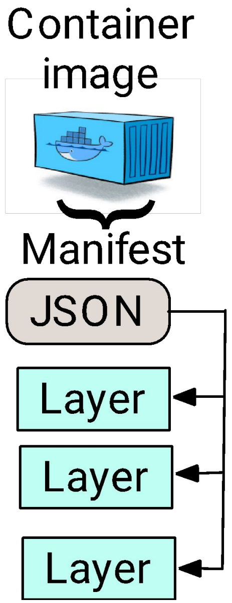
Our contribution: Characterization and optimization of Docker workflow

- Conduct a large-scale analysis of a real-world Docker workload from geo-distributed IBM container service
- Provide insights and develop heuristics to increase Docker performance
- Develop an open-source Docker workflow analysis tool*

* https://dssl.cs.vt.edu/drtp/

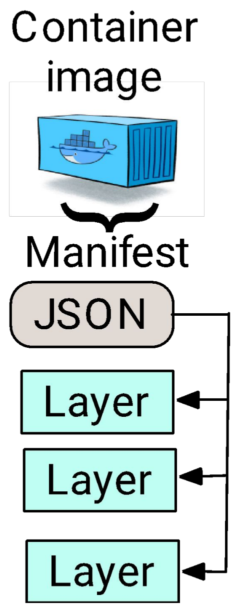
Background: Docker container image

- Container images are divided into **layers**.
- The metadata file is called **manifest.**



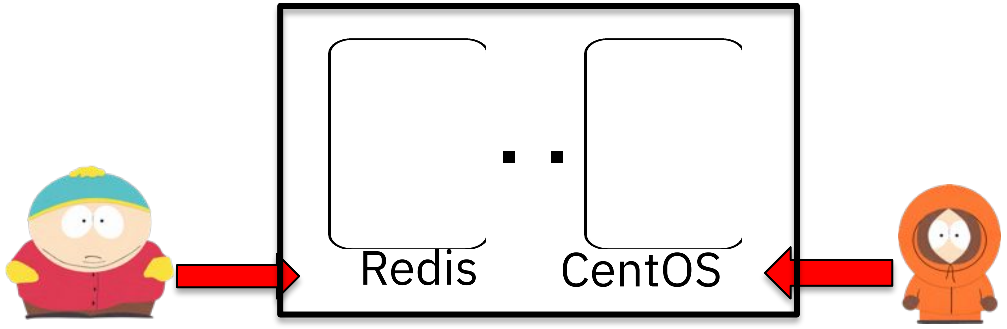
Background: Docker container image

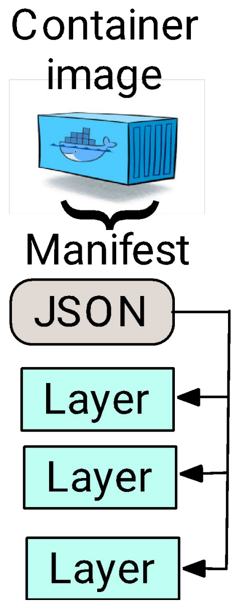
- Container images are divided into **layers.**
- The metadata file is called **manifest**.
- **Users** create **repositories** to store images.



Background: Docker container image

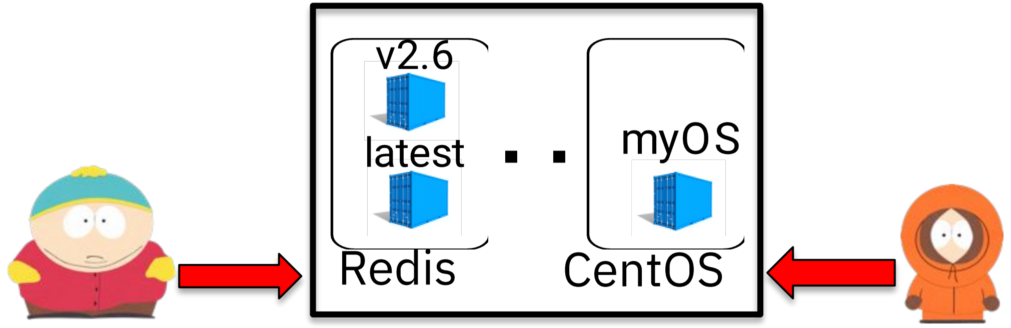
- Container images are divided into **layers.**
- The metadata file is called **manifest**.
- **Users** create **repositories** to store images.

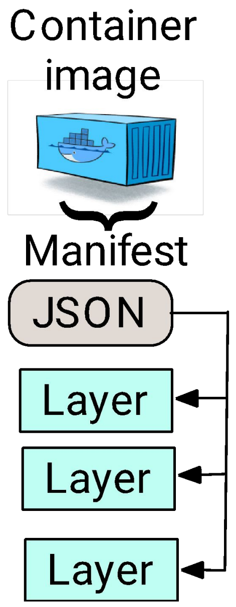




Background: Docker container image

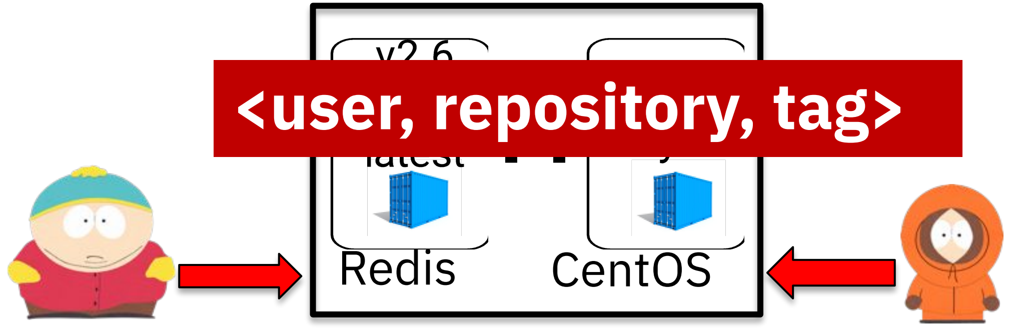
- Container images are divided into **layers.**
- The metadata file is called **manifest.**
- **Users** create **repositories** to store images.
- Images in a repository can have different **tags** (versions).



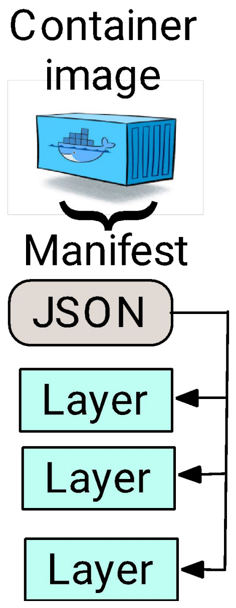


Background: Docker container image

- Container images are divided into **layers.**
- The metadata file is called **manifest**.
- **Users** create **repositories** to store images.
- Images in a repository can have different **tags** (versions).

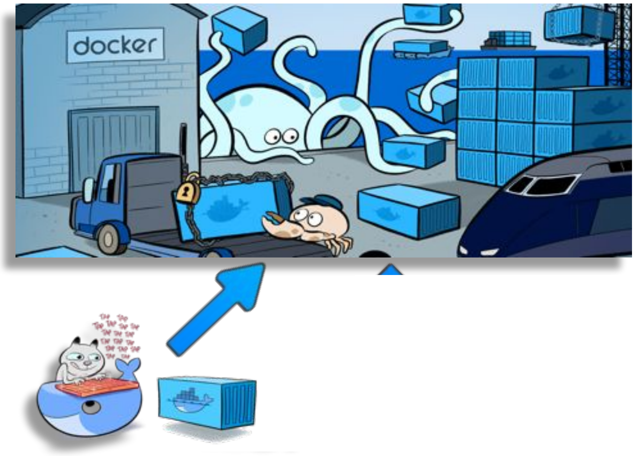


<user, repository, tag>



Background: Docker container registry

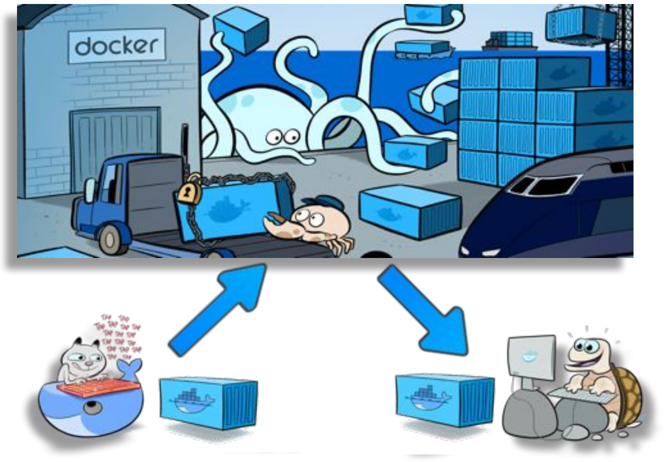
- Docker container images are stored online in **Docker registry**.
  - Push image:
    1. HEAD layers
    2. POST/PUT layer
    3. PUT manifest



docker push

Background: Docker container registry

- Docker container images are stored online in **Docker registry**.
  - Push image:
    1. HEAD layers
    2. POST/PUT layer
    3. PUT manifest
  - Pull image:
    1. GET manifest
    2. GET layers



docker push

docker pull

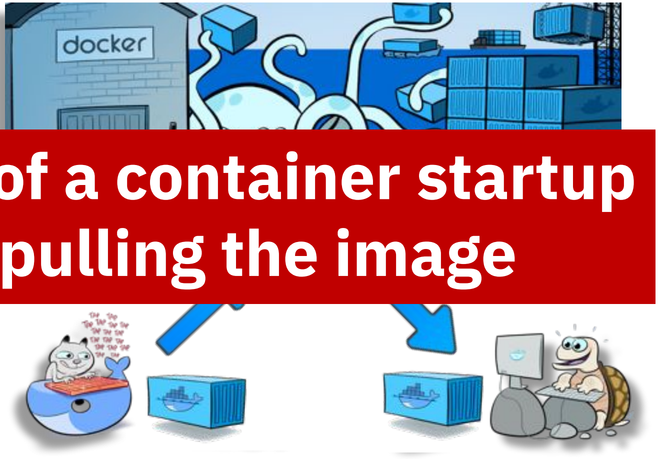
Background: Docker container registry

- Docker container images are stored online in **Docker registry**.

  - Push image:
    1. HEAD layers

Significant amount of a container startup time is spent in pulling the image

  - Pull image:
    1. GET manifest
    2. GET layers



docker push

docker pull

The IBM Cloud Docker registry traces

- Capture a diverse set of customers: individuals, small & medium businesses, government institutions
- Cover five geographical locations and seven availability zones
- Span 75 days and 38M requests that account for more than ~181TB of data transferred

IBM Docker registry service

- Five geographical locations constitute seven Availability Zones (AZ):

Production

1. Dallas (**dal**)
2. London (**lon**)
3. Frankfurt (**fra**)
4. Sydney (**syd**)

IBM Internal

5. Staging (**stg**)

Testing*

6. Prestaging (**prs**)
7. Development (**dev**)

*The registry setup is identical, except prs and dev are only half the size of the other Azs.

IBM Docker registry service

- Five geographical locations constitute seven Availability Zones (AZ):

Production

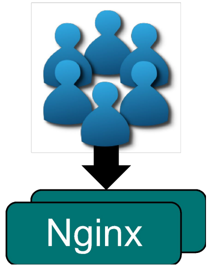
1. Dallas (**dal**)
2. London (**lon**)
3. Frankfurt (**fra**)
4. Sydney (**syd**)

IBM Internal

5. Staging (**stg**)

Testing*

6. Prestaging (**prs**)
7. Development (**dev**)



IBM Cloud Registry architecture

*The registry setup is identical, except prs and dev are only half the size of the other Azs.

IBM Docker registry service

- Five geographical locations constitute seven Availability Zones (AZ):

Production

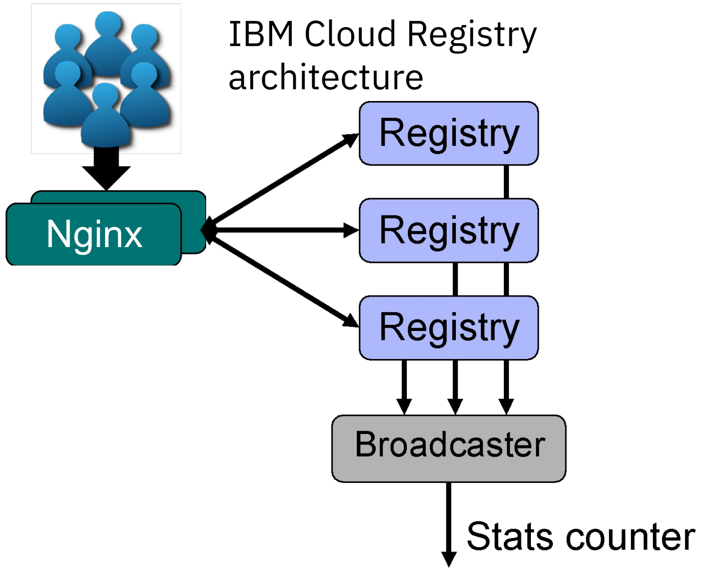
1. Dallas (**dal**)
2. London (**lon**)
3. Frankfurt (**fra**)
4. Sydney (**syd**)

IBM Internal

5. Staging (**stg**)

Testing*

6. Prestaging (**prs**)
7. Development (**dev**)



*The registry setup is identical, except prs and dev are only half the size of the other Azs.

IBM Docker registry service

- Five geographical locations constitute seven Availability Zones (AZ):

Production

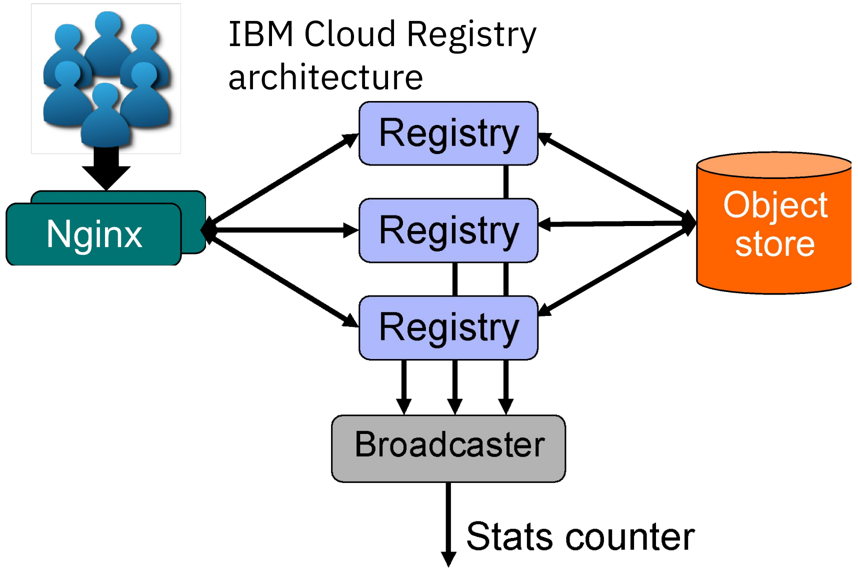
1. Dallas (**dal**)
2. London (**lon**)
3. Frankfurt (**fra**)
4. Sydney (**syd**)

IBM Internal

5. Staging (**stg**)

Testing*

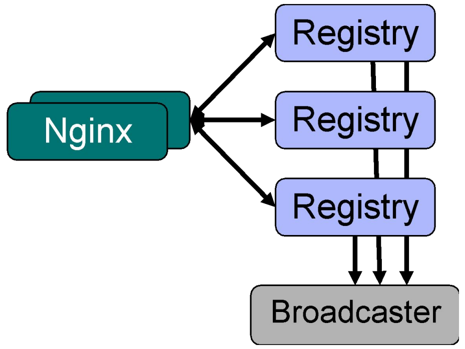
6. Prestaging (**prs**)
7. Development (**dev**)



*The registry setup is identical, except prs and dev are only half the size of the other Azs.

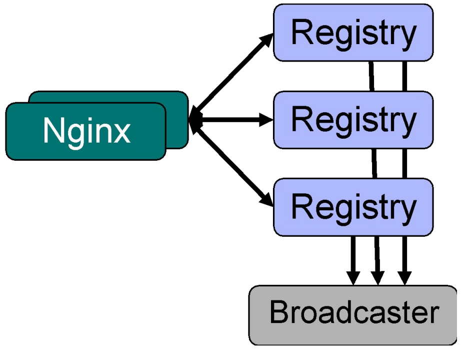
Tracing methodology

- Collected data from **Registry, Nginx, and Broadcaster**
- Studied requests: GET, PUT, HEAD, PATCH, POST



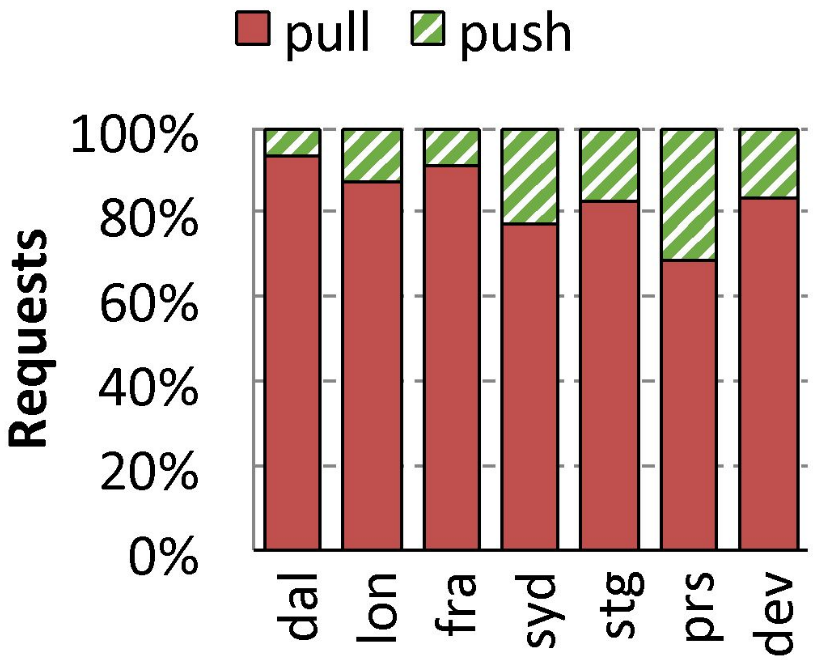
Tracing methodology

- Collected data from **Registry, Nginx, and Broadcaster**
- Studied requests: GET, PUT, HEAD, PATCH, POST
- Combined traces by matching the incoming HTTP request identifier across the components
- Removed redundant fields and anonymized the traces



Q1: What is the distribution of request types?

Analysis

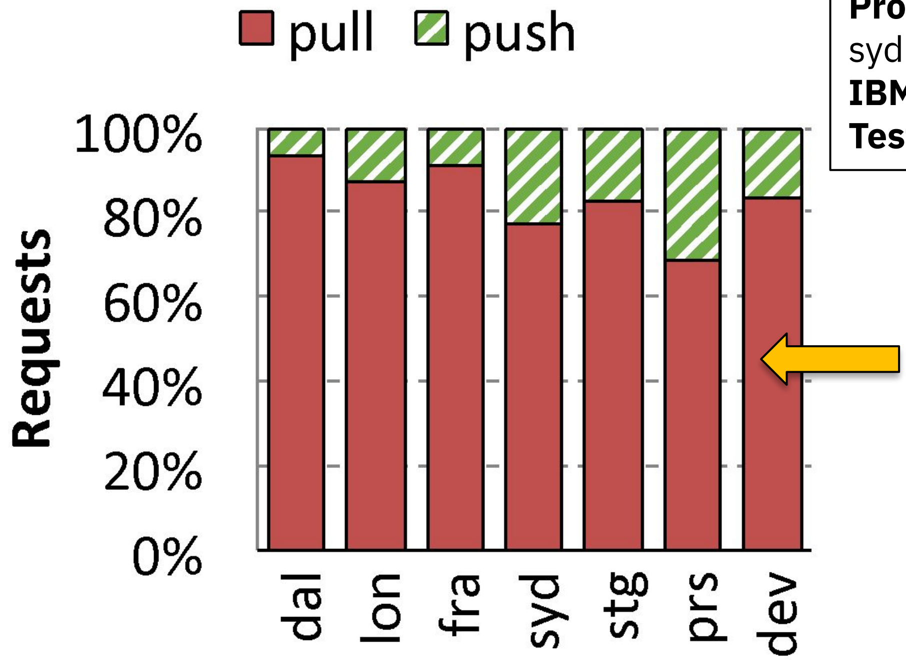


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q1: What is the distribution of request types?

80%–95% of requests are reads (pulls)

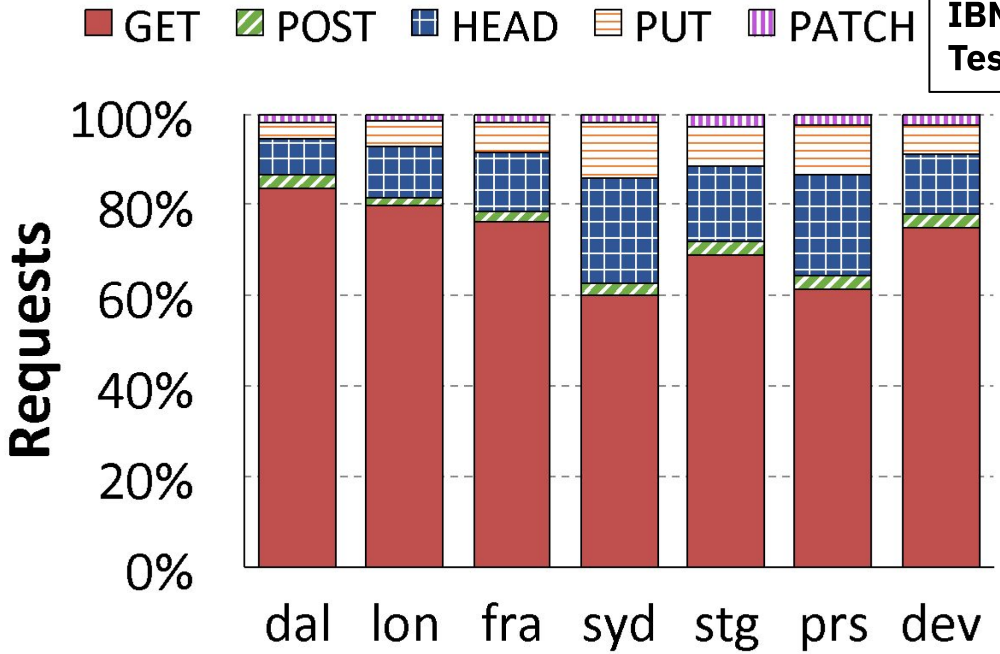
Analysis



Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q1: What is the distribution of request types?

Analysis



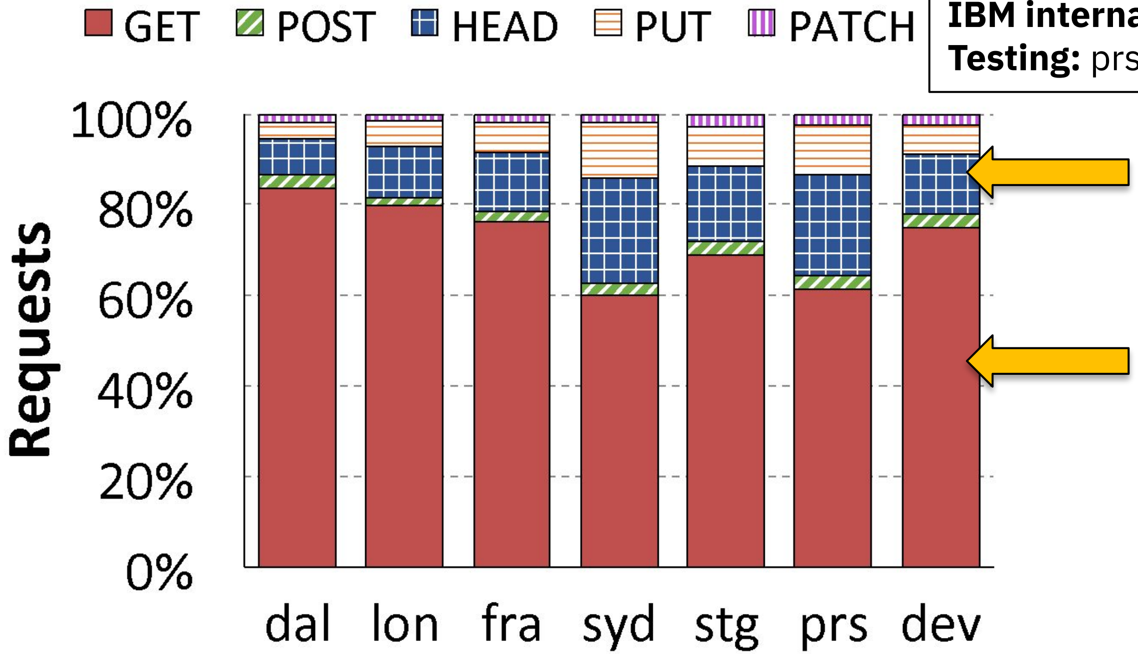
Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q1: What is the distribution of request types?

60% of the requests are GET and 10%–22% are HEAD

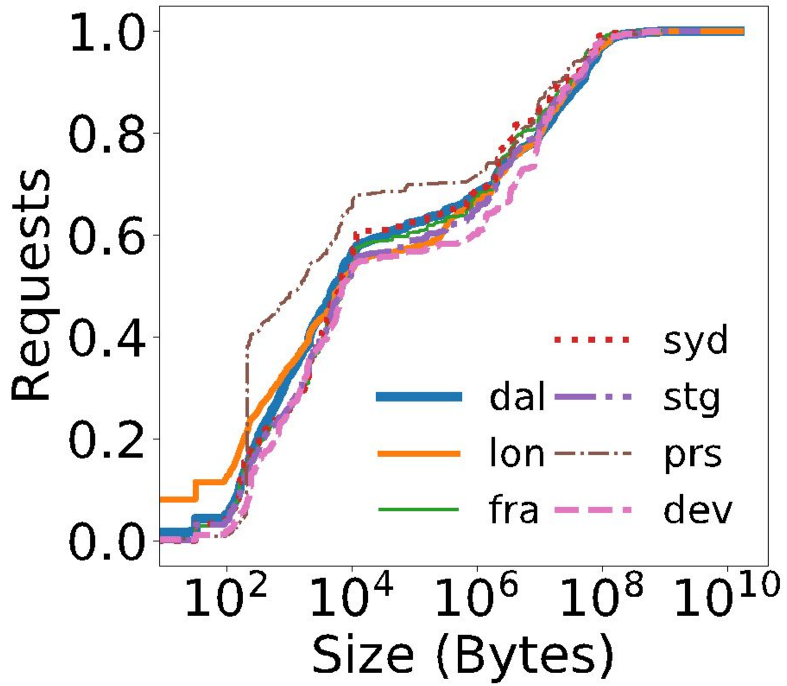
Analysis

Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev



Q2: What is the layer size distribution?

Analysis

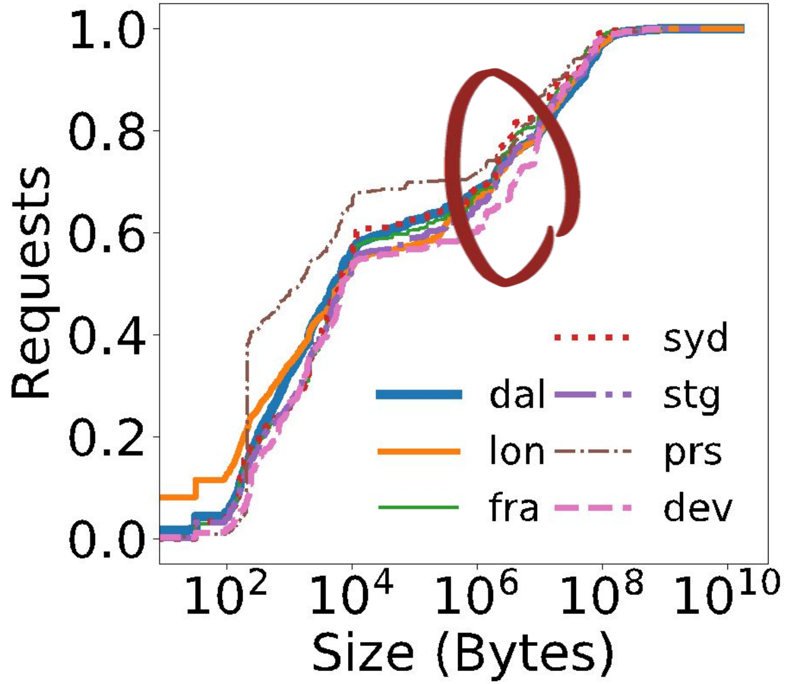


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q2: What is the layer size distribution?

Analysis

65% of the layers are smaller than 1 MB and around 80% are smaller than 10 MB

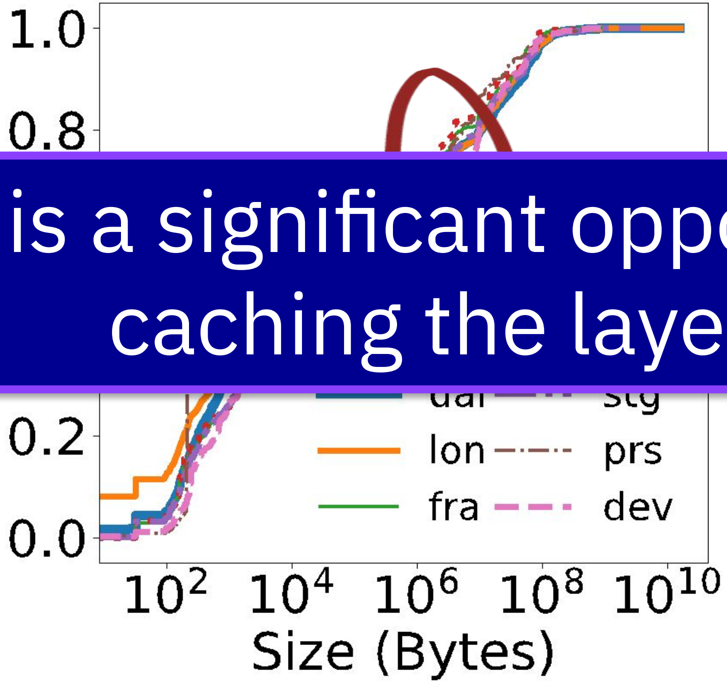


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q2: What is the layer size distribution?

Analysis

65% of the layers are smaller than 1 MB and around 80% are smaller than 10 MB

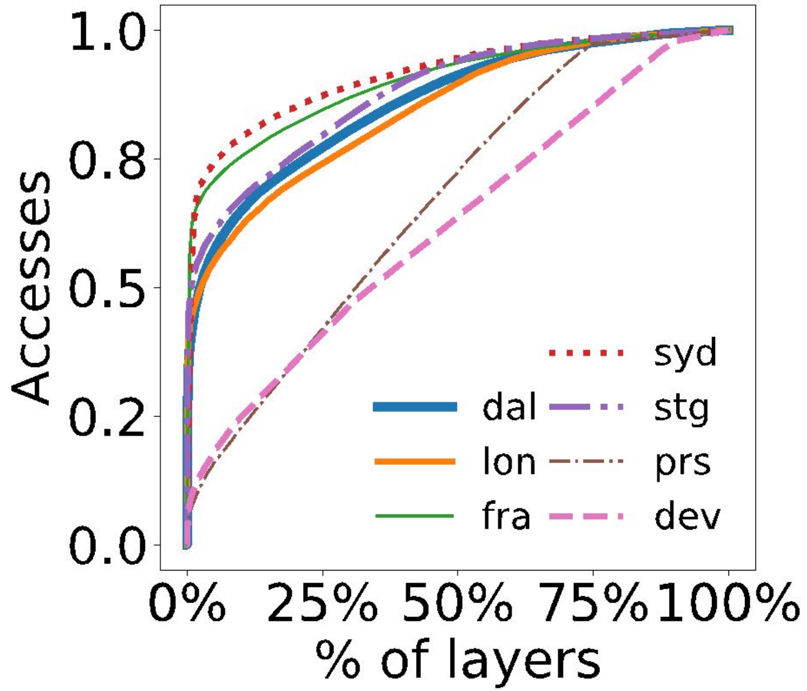


Production: dal, lon, fra, syd
IBM internal: stg

There is a significant opportunity for caching the layers

Q3: Is there spatial locality?

Analysis

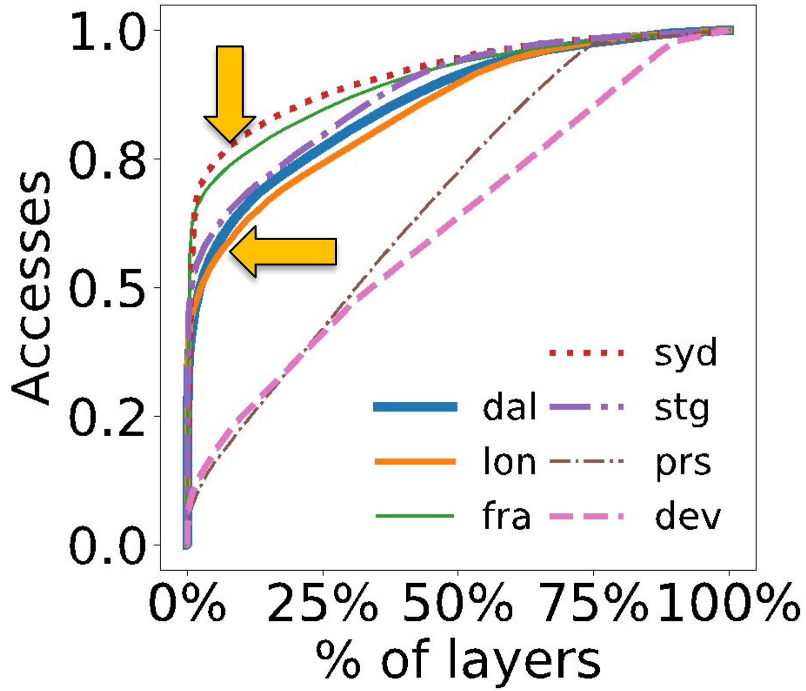


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q3: Is there spatial locality?

1% of most accessed layers account for 42% and 59% of all requests in dal and syd, respectively

Analysis

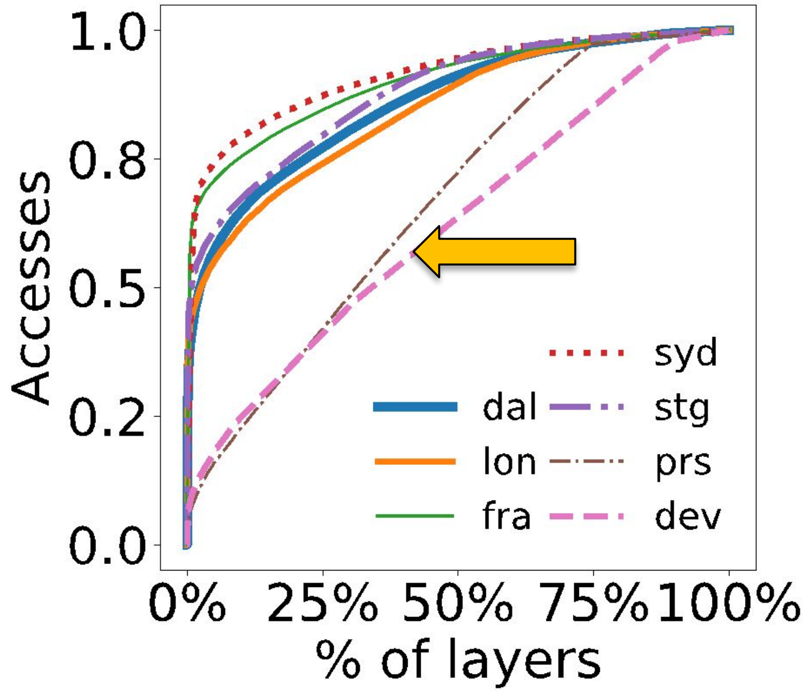


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q3: Is there spatial locality?

Analysis

1% of most accessed layers account for 42% and 59% of all requests in dal and syd, respectively

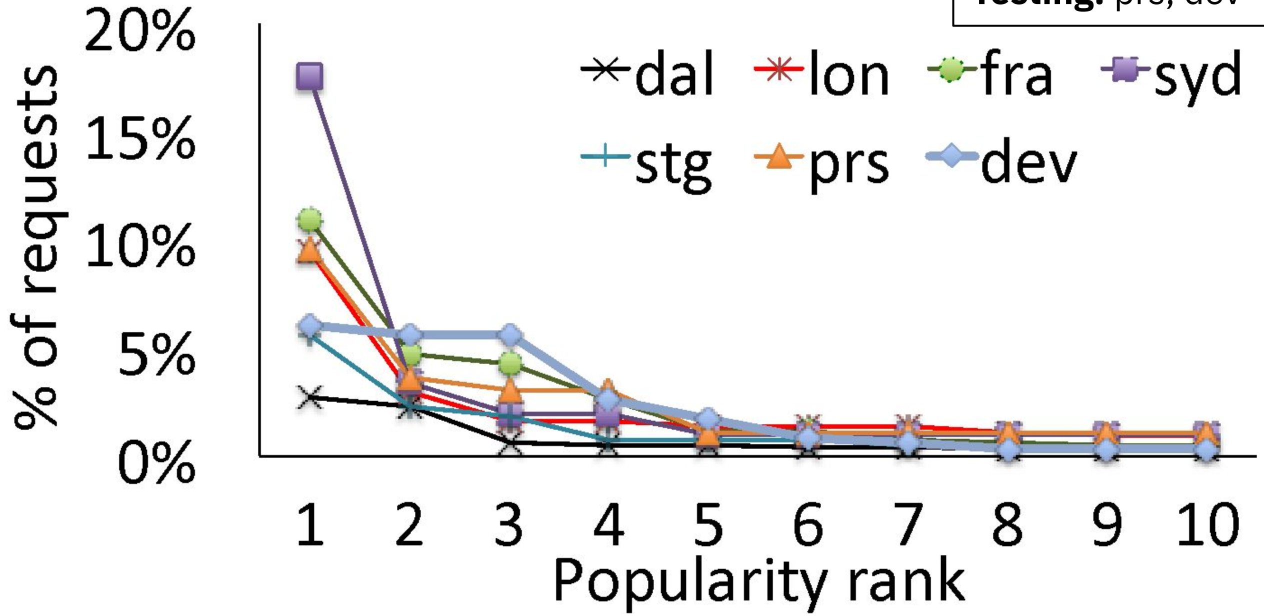


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q3: Is there spatial locality?

Analysis

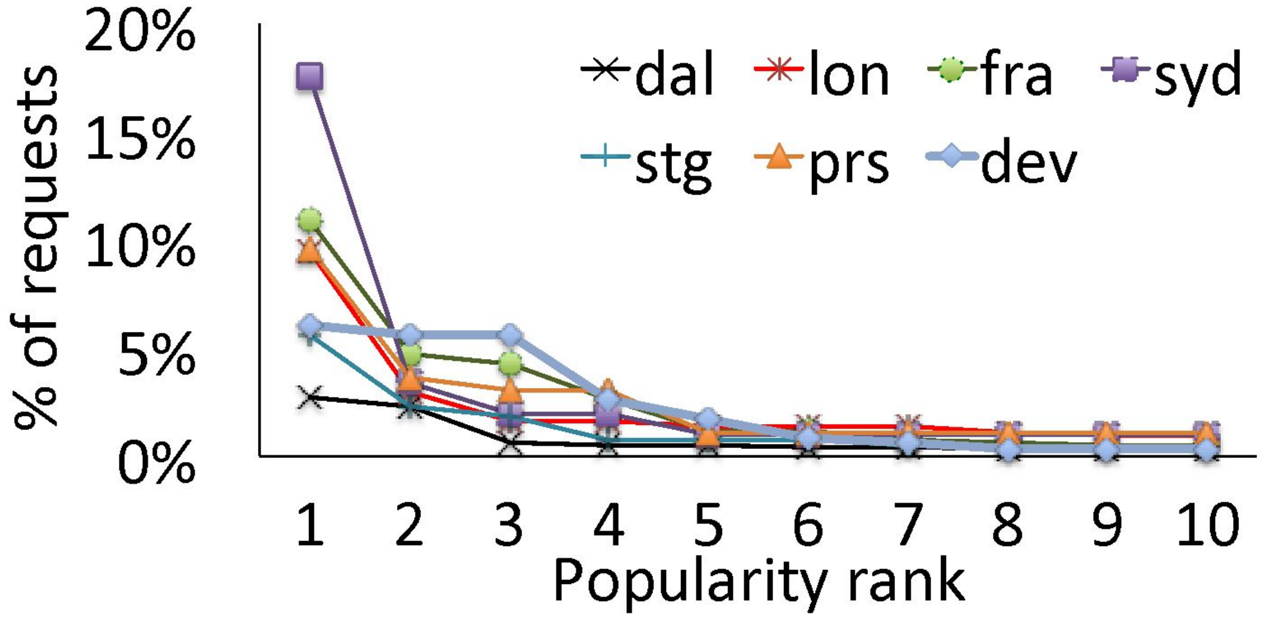
Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev



Q3: Is there spatial locality?

Analysis

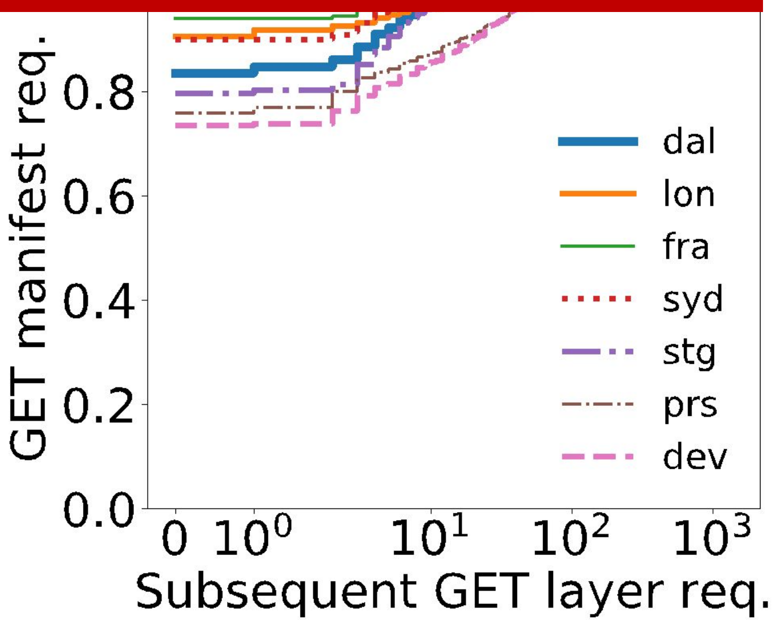
The popularity rate drops rapidly as we move from most popular to tenth most popular layer



Q4: Can future requests be predicted?

Analysis

GET manifest requests are not followed by any subsequent GET layer request

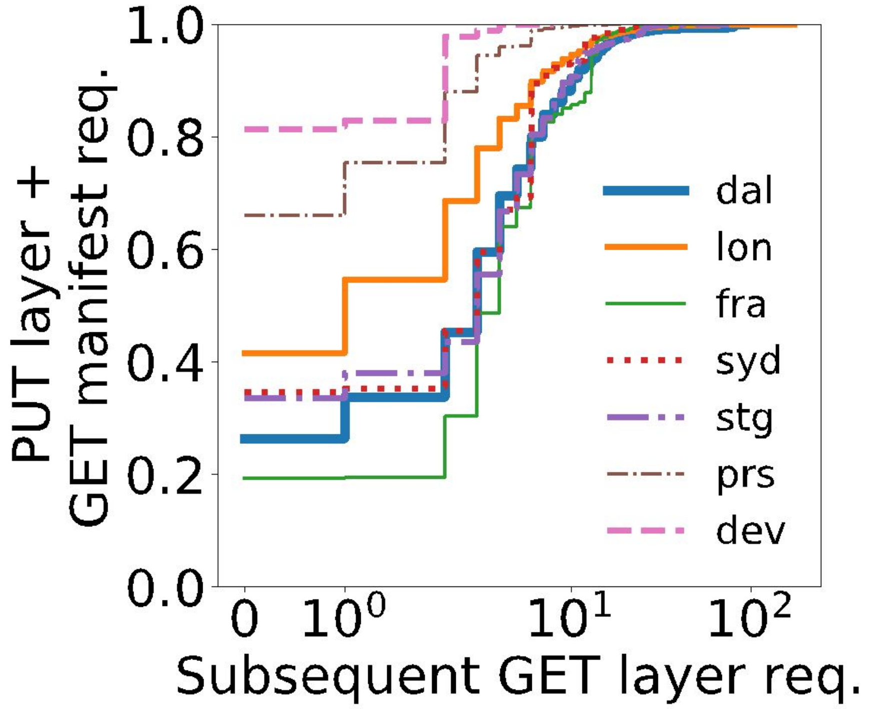


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q4: Can future requests be predicted?

Significant increase in subsequent GET layer requests within a session

Analysis

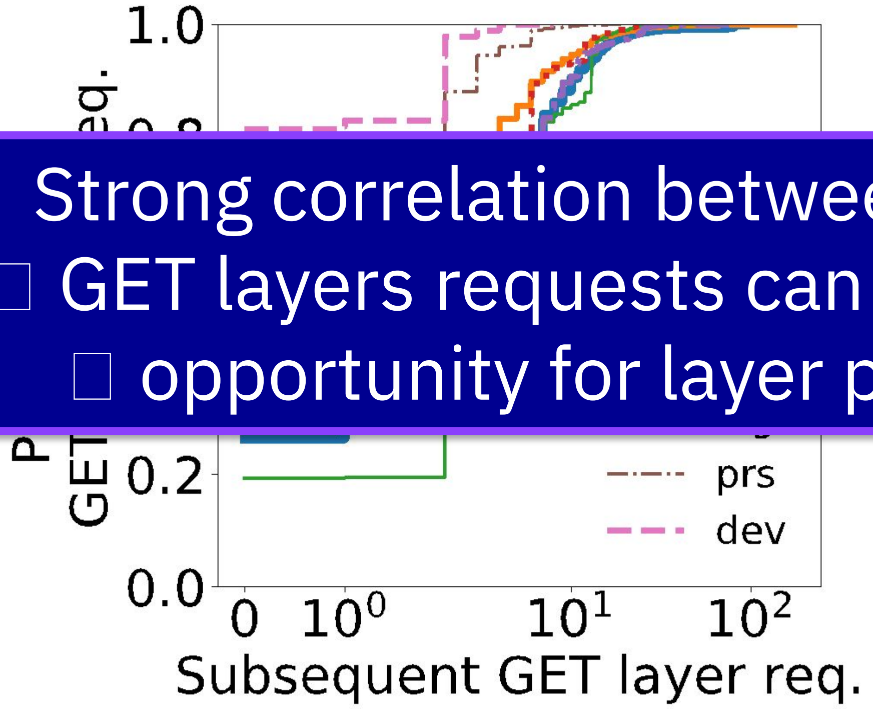


Production: dal, lon, fra, syd
IBM internal: stg
Testing: prs, dev

Q4: Can future requests be predicted?

Significant increase in subsequent GET layer requests within a session

Analysis



Production: dal, lon, fra, syd

Strong correlation between requests

 GET layers requests can be predicted

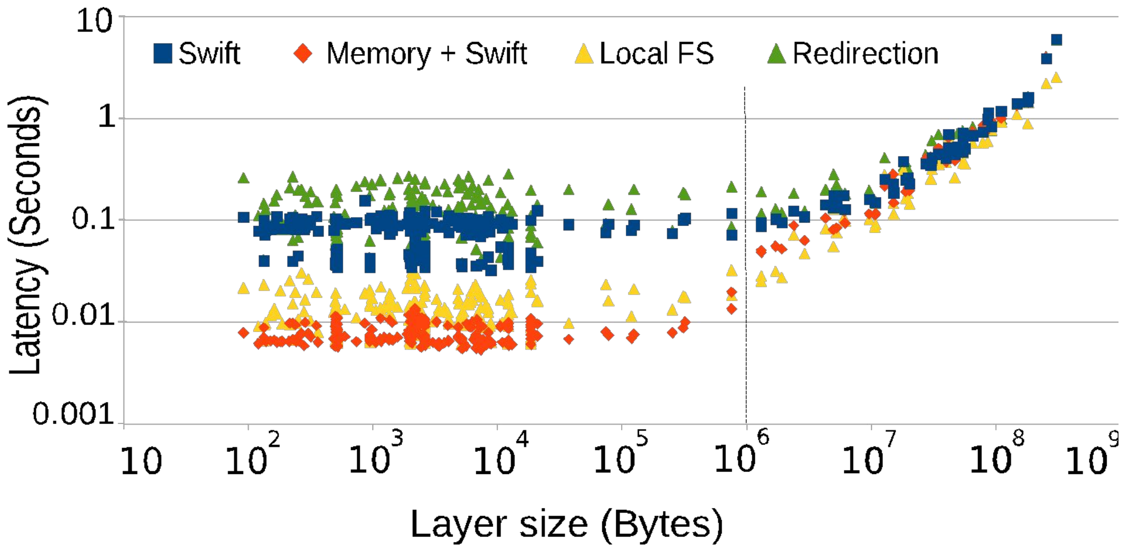
 opportunity for layer prefetching

Analysis

Effect of backend storage technologies

Experimental setup:

- Registry on 32 core machine with 64 GB RAM and 512 GB SSD
- Swift object store on 10 similar nodes
- Trace re-player on 6 additional nodes



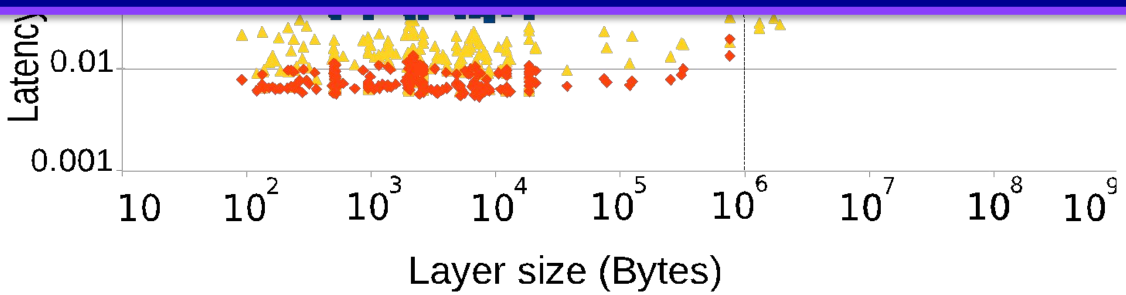
Analysis

Effect of backend storage technologies

Experimental setup:

- Registry on 32 core machine with 64 GB RAM and 512 GB SSD
- Swift object store on 10 similar nodes
- Trace re-player on 6 additional nodes

Fast backend storage/cache for the registry can significantly improve the overall performance



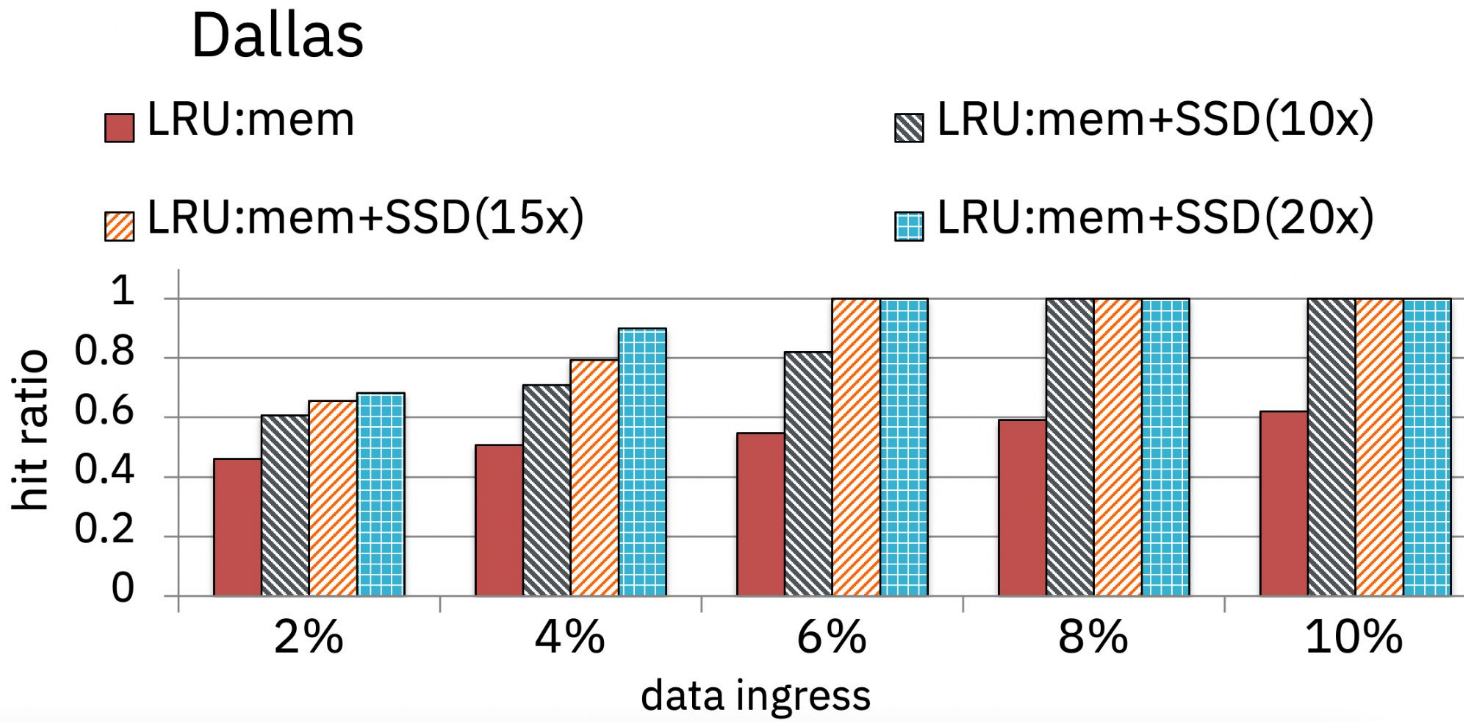
Effect of a two-level Main memory+SSD cache

Analysis

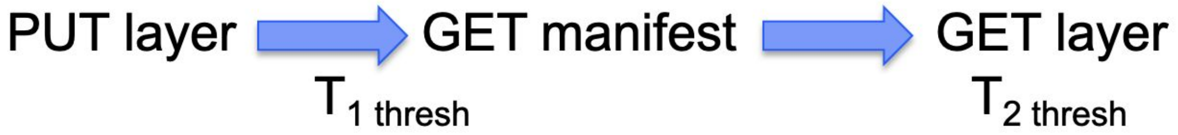
Experimental setup:

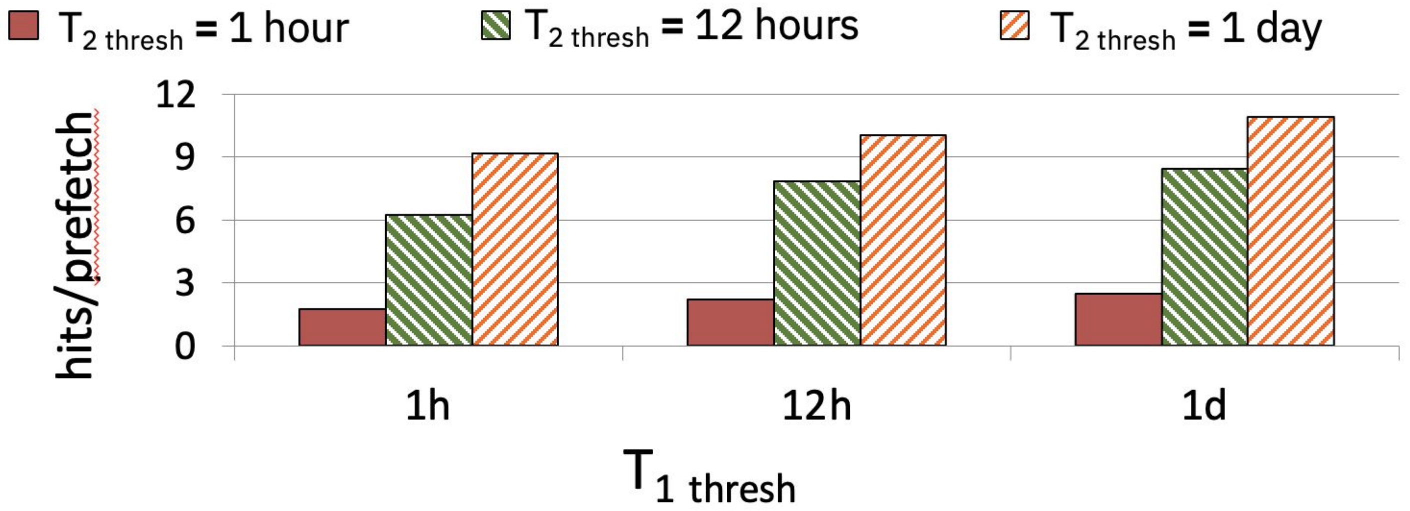
- Small layers (<100 MB) are stored in the main memory
- Replacement policy for both cache level is LRU
- Studied cache sizes:
  RAM: 2%, 4%, 6%, 8%, and 10% of the data ingress
  SSD: 10x, 15x, 20x the size of RAM cache
- Layers are content addressable
   cache invalidation is not a problem

Two-level cache: Main memory+SSD



Benefit of layer prefetching





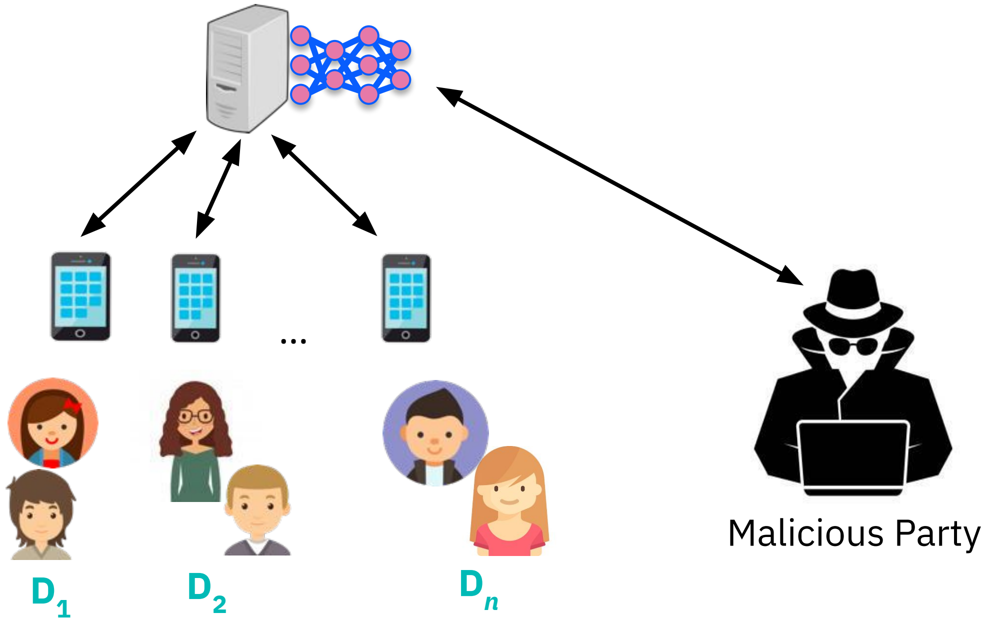
DockerHub analysis: DupHunter

- We did analysis on 167 TB of DockerHub data and found large number of redundant files in the dataset
- We developed DupHunter to overcome this issue
- DupHunter exploits the redundancy in container images and predictable user access patterns to achieve high space savings with low layer restore overhead
- DupHunter reduces storage space by up to 6.9x and can reduce the GET layer latency up to 2.8x compared to the state of the art

https://github.com/nnzhaocs/DupHunter

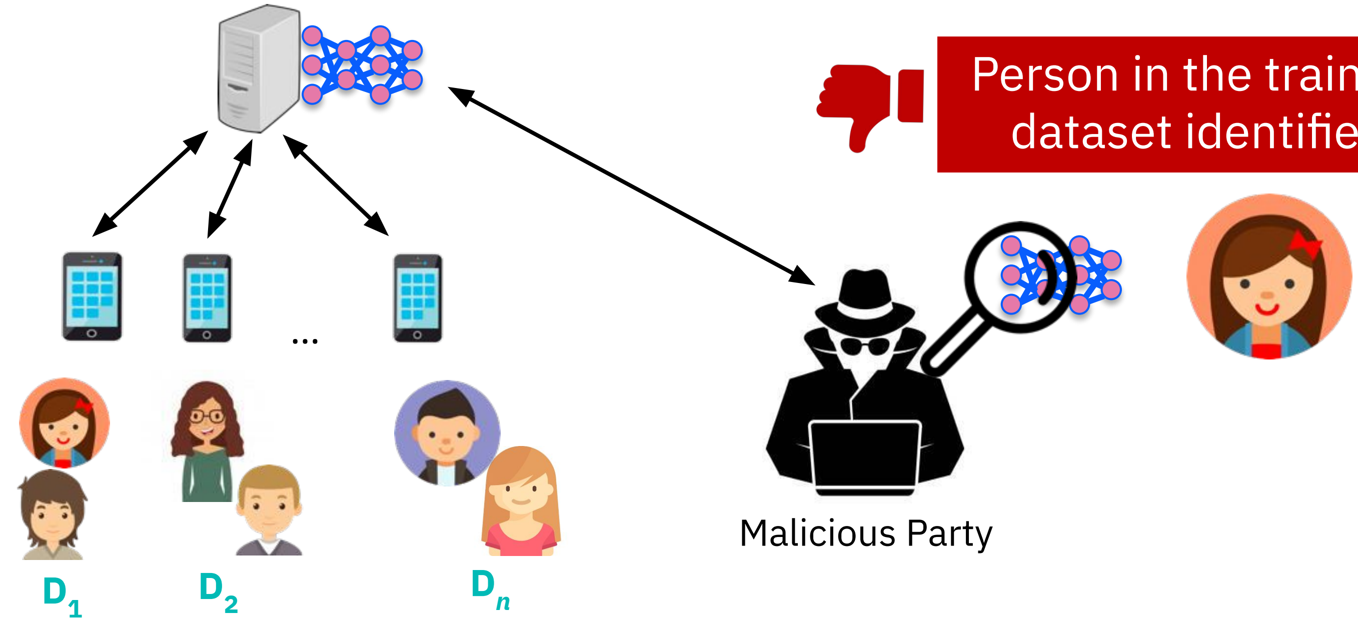
Accelerate Federated Learning by redesigning the system architecture

Current Research



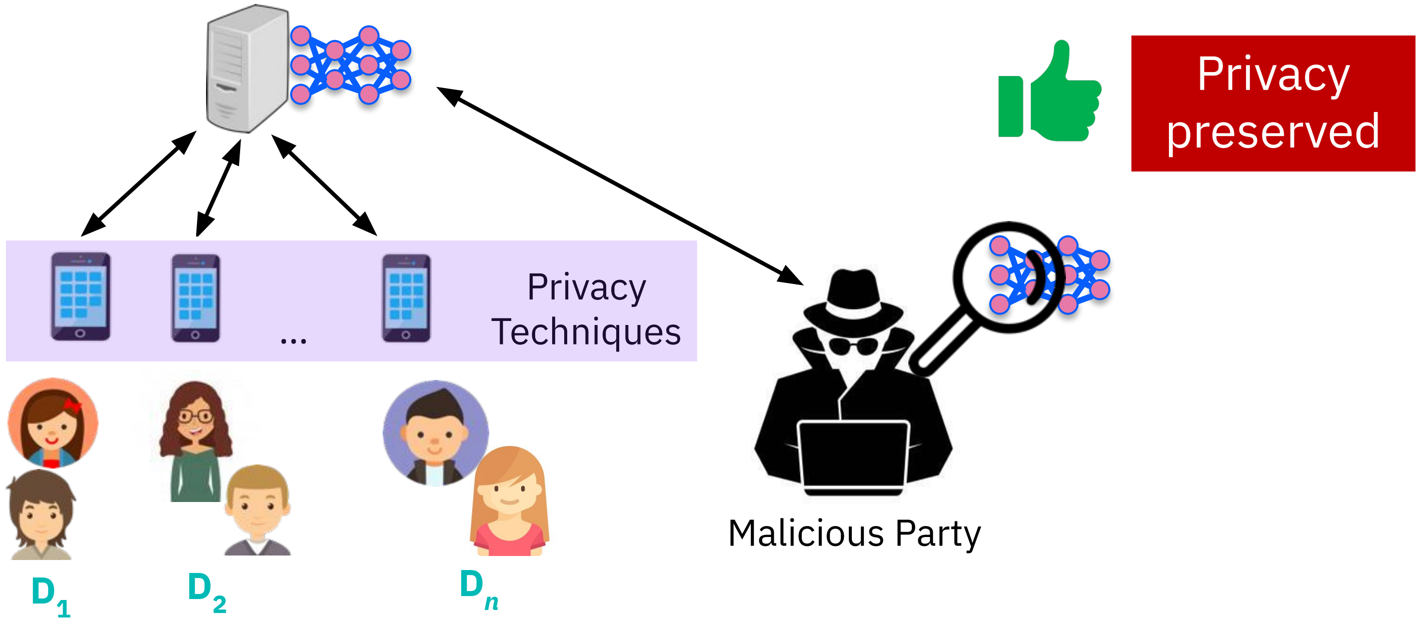
Accelerate Federated Learning by redesigning the system architecture

Current Research



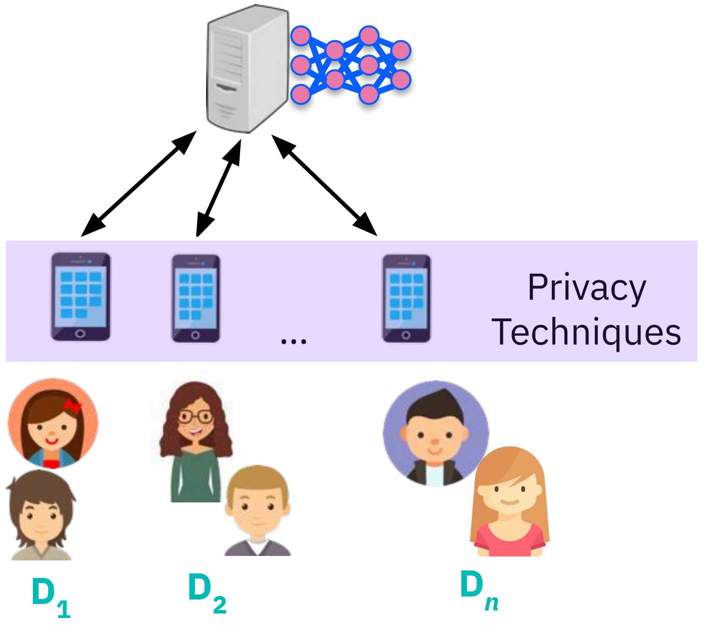
Accelerate Federated Learning by redesigning the system architecture

Current Research



Accelerate Federated Learning by redesigning the system architecture

Current Research



Privacy Techniques

- Differential Privacy
- Secure Multi Party Computation
- Homomorphic encryption
- Functional Encryption
- Hybrid Approaches

Accelerate Federated Learning by redesigning the system architecture

Current Research

$\rightarrow$

Overcome the overhead of the privacy preserving techniques

Accelerate Federated Learning by redesigning the system architecture

Current Research

Overcome the overhead of the privacy preserving techniques

→ Reducing the communication overhead in Vertical Federated Learning

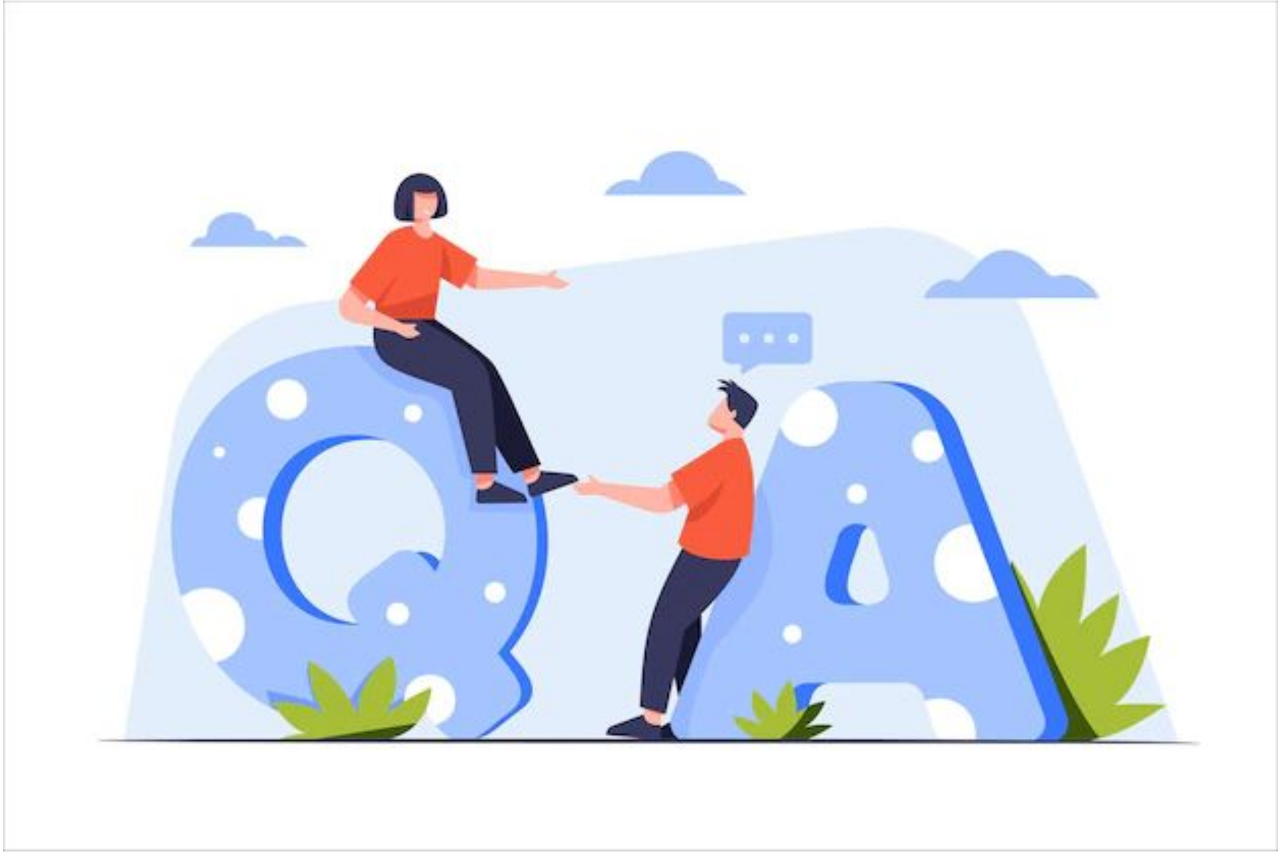
Accelerate Federated Learning by redesigning the system architecture

Current Research

Overcome the overhead of the privacy preserving techniques

Reducing the communication overhead in Vertical Federated Learning

Design serverless computing based Asynchronous Federated Learning



Thank you!

Ali Anwar
aanwar@umn.edu