Pervasive AI

Tutorial of the 37th AAAI Conference on Artificial Intelligence
February 7, 2023 – Washington DC, USA

Davide Bacciu, Antonio Carta, Patrizio Dazzi, Claudio Gallicchio
University of Pisa, Italy

Solutions and Infrastructures for distributed and federated learning

Davide Bacciu, Antonio Carta, **Patrizio Dazzi**, Claudio Gallicchio
University of Pisa, Italy

Outline

- Need for going beyond single machine learning
- Distributed learning
- Federated learning
- Beyond federated learning

Need for going beyond single machine for learning

One Machine
~~The World~~ is not enough

- ✓ Learning using single machines could be limiting
  - ✓ Computational limitations
    - ✓ a single machine may not have **enough computational power** to train large models in a reasonable amount of time
  - ✓ Memory limitations
    - ✓ large models require **more memory** than a single machine can provide
  - ✓ Scalability challenges
    - ✓ a single machine may not be able to handle the increase **in data size and complexity** when training large models

Computational Limitations

A single machine does not have enough computational power to train large models in a reasonable amount of time

Insufficient computational power can lead to: longer training times, decreased accuracy

Any way to improve computational capabilities?

To some extent...

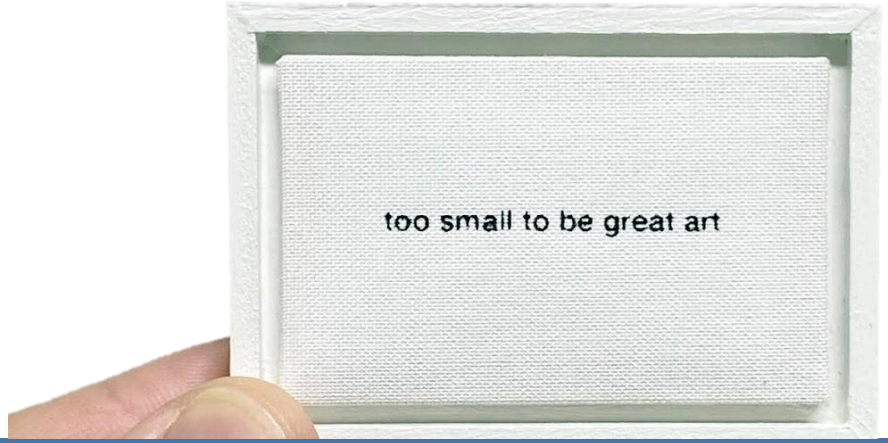
Multicores

GPUs

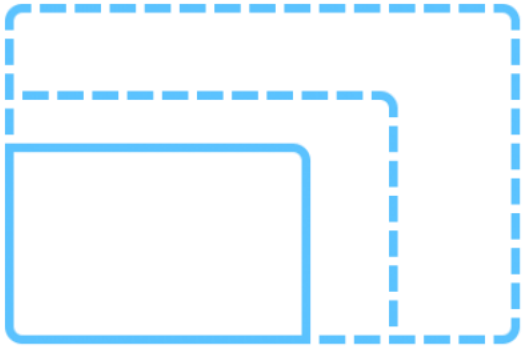
FPGAs

Memory Limitations

The **greatest** challenges when training large models using a single machine



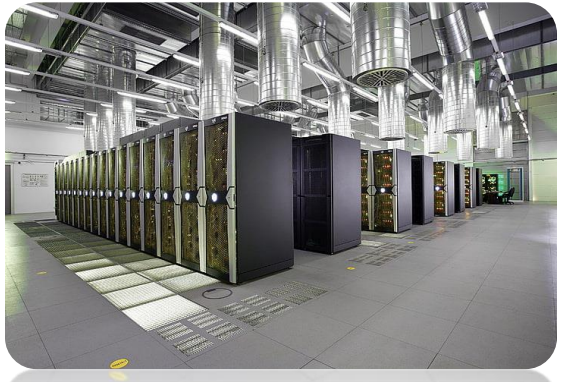
Scalability Challenges



Scalability refers to the **ability** of a **system** to perform well under an **increased or expanding workload**.

A system that scales well will be able to **maintain or increase its level of performance** even as it is tested by **larger and larger** operational **demands**.

a single machine may not be able to handle the **increase** in data size and **complexity** when training large models



What to do when a machine is not enough ?

“When a machine is not enough, should accelerate with parallel or distributed computing ?”

Parallel vs. Distributed

Parallel Computing	Distributed Computing
Many operations are performed simultaneously	System components are located at different locations

Parallel Computing	Distributed Computing
Many operations are performed simultaneously	System components are located at different locations
Single computer is required	Uses multiple computers

Parallel Computing	Distributed Computing
Many operations are performed simultaneously	System components are located at different locations
Single computer is required	Uses multiple computers
Multiple processors perform multiple operations	Multiple computers perform multiple operations

Parallel Computing	Distributed Computing
Many operations are performed simultaneously	System components are located at different locations
Single computer is required	Uses multiple computers
Multiple processors perform multiple operations	Multiple computers perform multiple operations
It may have shared or distributed memory	It have only distributed memory

Parallel Computing	Distributed Computing
Many operations are performed simultaneously	System components are located at different locations
Single computer is required	Uses multiple computers
Multiple processors perform multiple operations	Multiple computers perform multiple operations
It may have shared or distributed memory	It have only distributed memory
Processors communicate with each other through bus	Computer communicate with each other through message passing.

Parallel Computing	Distributed Computing
Many operations are performed simultaneously	System components are located at different locations
Single computer is required	Uses multiple computers
Multiple processors perform multiple operations	Multiple computers perform multiple operations
It may have shared or distributed memory	It have only distributed memory
Processors communicate with each other through bus	Computer communicate with each other through message passing.
Improves the system performance	Improves system scalability, fault tolerance and resource sharing capabilities

...but the main issue with “not-distributed” parallel machines is on...

SCALABILITY when problem **complexity increases** and **data grows**

This sounds familiar, right?

“When the going gets tough, the tough get going.”

one machine is not enough!

Distributed Learning

What is distributed learning ?

Learning performed using a Distributed System!

What is a distributed system?

Various definitions have been given

- none of them completely **satisfactory**
- none of them in **agreement** with any of the others

"A distributed system is a **collection of autonomous** computing elements that **appears to its users as a single coherent system**."

[Distributed Systems 3, Tanenbaum & Van Steen]

This definition refers to two key features

- A distributed system is a collection of computing elements, each being able to **behave independently** of the other
- End users (humans or software) believe they are **dealing with a single system**

This means that one way or another the autonomous nodes need to **collaborate**.

How collaboration happens?

nodes can act independently from each other

- nodes need to achieve **common goals** realized by **exchanging messages** with each other
- nodes **react to messages** leading to **further communication** through **message passing**

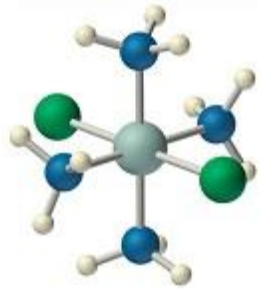
End users should **not even notice** that processes, data, and control are **dispersed across a computer network**

Single coherent system

Coherent if it behaves according to the expectations of its users in a **single coherent system**

The **collection of nodes** as a whole operates the same, no matter where, when, and how **interaction takes place**

This so-called **distribution transparency** is an important design goal of distributed systems.



How distributed learning is performed?

Namely, how do machines collaborate to speed-up the computation?

Let’s have a look at two main strategies for training!

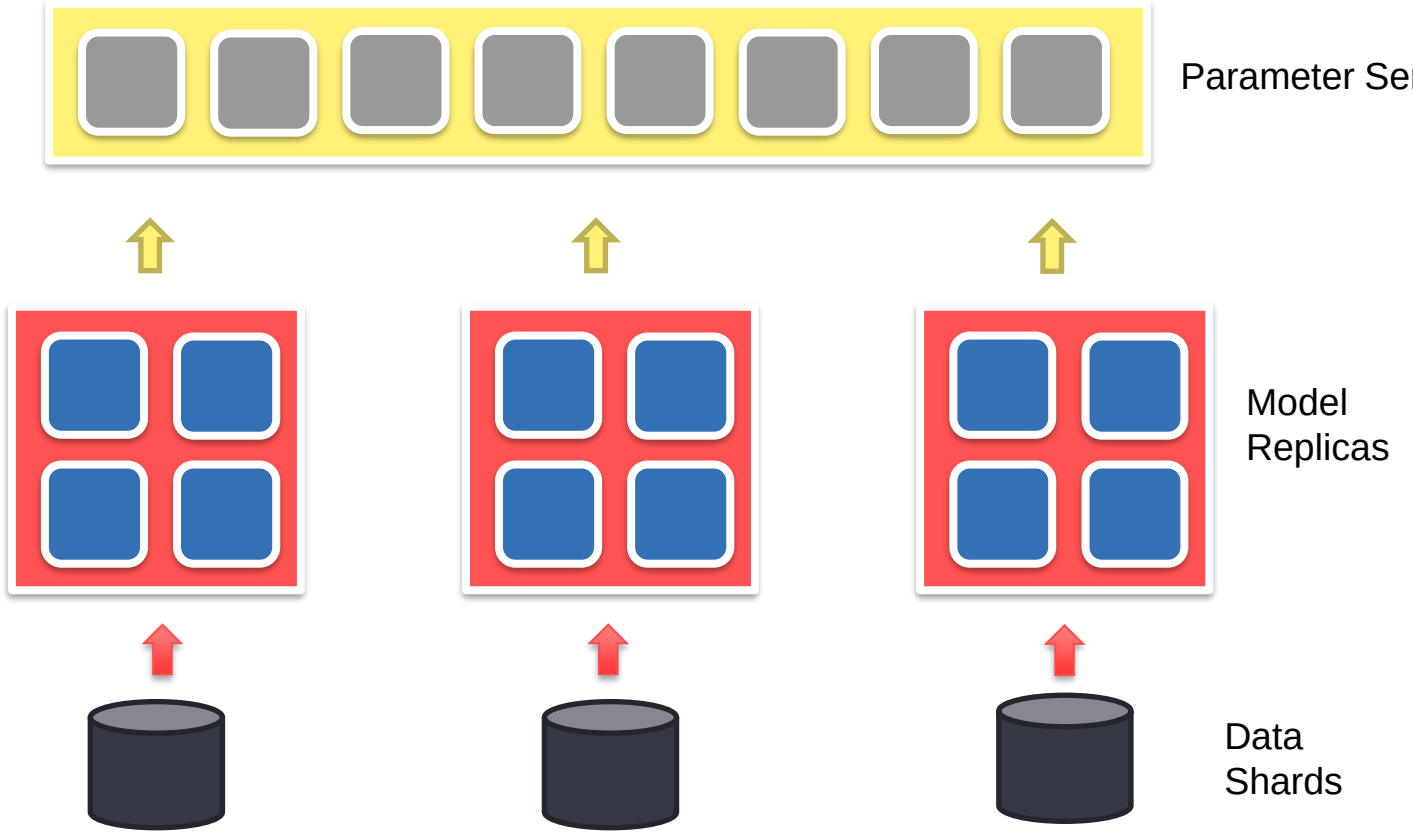
Two flavours of Distributed Training: Data Parallelism vs Model Parallelism

Data Parallelism vs. Model Parallelism

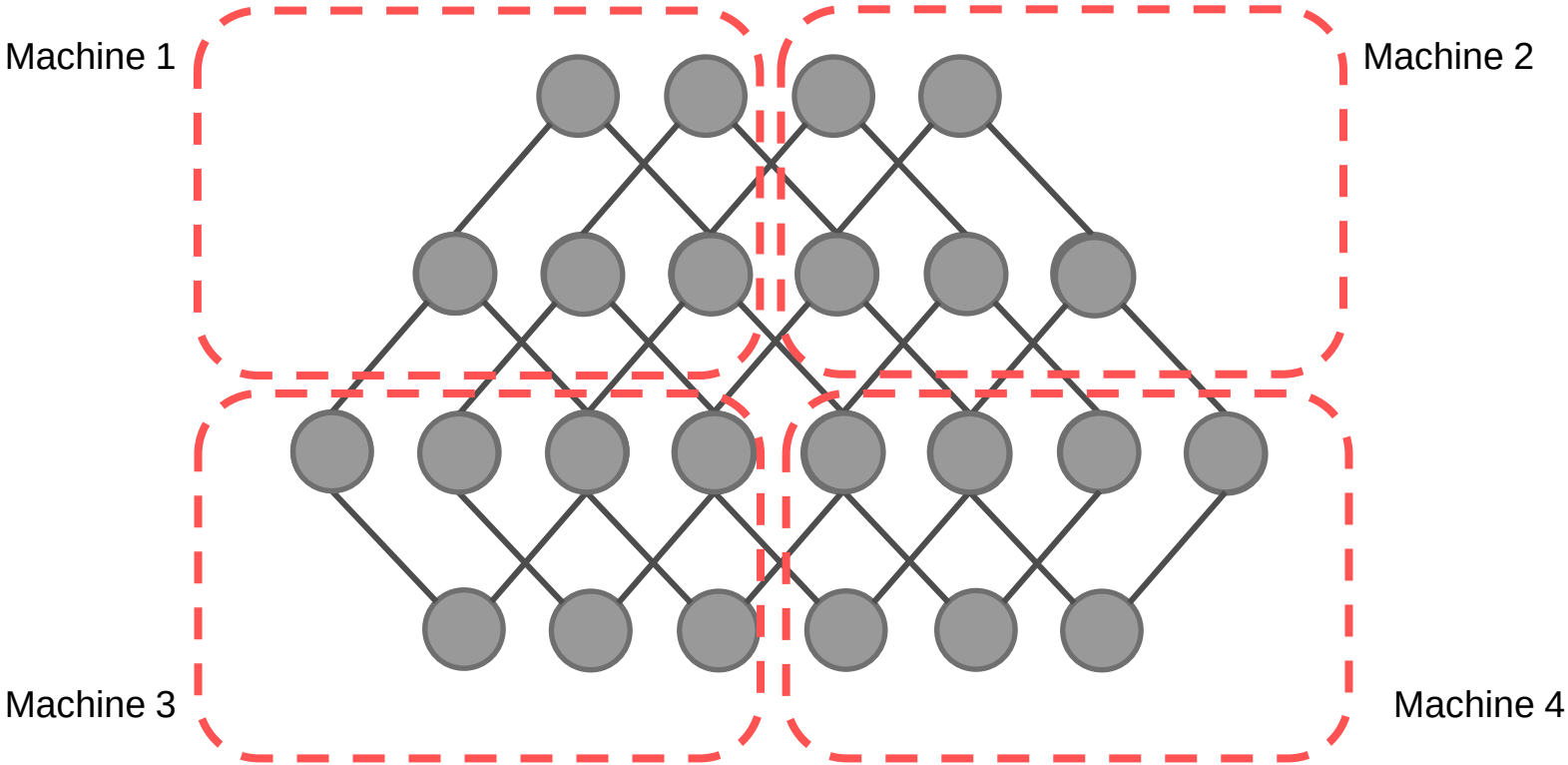
data is **scattered** throughout a **set of machines** that perform the **training loops** in all of them either **synchronously** or **asynchronously**

When a model is so big that it **doesn't fit in the memory of a single device (heard about GPT-3 ?)**, it is possible to divide it into different parts, distribute them across **multiple machines** and train each one of them **independently** using the same data

Data Parallelism



Model Parallelism



Communication and Synchronization Models

Fundamental to understand how the **nodes synchronise** with each other

Relevant as different workers **may work at different speeds** and hence the **partial gradients** may not be available from all the workers at the same time

During the gradient update phase

Using stale gradients then convergence may be slower

Waiting for all workers to finish, then it may not be very efficient.

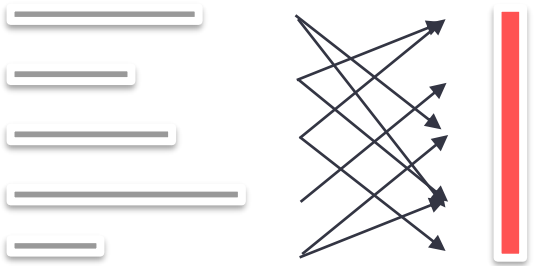
Three models of communication arise as a result of the tradeoff between speed and convergence.

Bulk Synchronous Parallel - BSP

BSP is, at the most basic level, a **two-step** process performed **iteratively** and **synchronously**:

1) perform **task computation on local data**
2) communicate **the results**, and then repeat the two steps.

Thus the BSP model is composed of the **workers**, the **communication between them** and a **barrier**



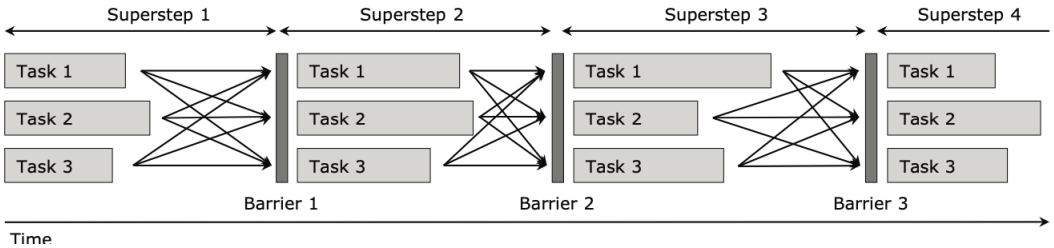
The **barrier** marks the end of a **super step** or an **iteration**

Bulk Synchronous Parallel - BSP

In the Distributed ML case each **worker** works on its own **gradient** and the **barrier** ensures that the **parameter server** updates the weights only when it receives the gradient from **all the workers**

The BSP model trades off speed for convergence.

synchronization of the parallel tasks occur at the super step barriers, depicted below



Asynchronous Parallel - ASP

With ASP, all workers send their gradients to the server, but **no synchronisation is implemented**

Workers **do not wait for other workers to complete**; hence, the parameter server may have **stale gradients** from a few workers.

This causes **errors in the gradient calculation** and hence **delays the convergence**. Also, each worker may obtain **a different version** of the weight from the parameter server.

Consequently, **ASP has the least training time but** yields a **lower accuracy** and is **not stable** in terms of model **convergence**

SYNCHRONOUS

VERSUS

ASYNCHRONOUS

SSP

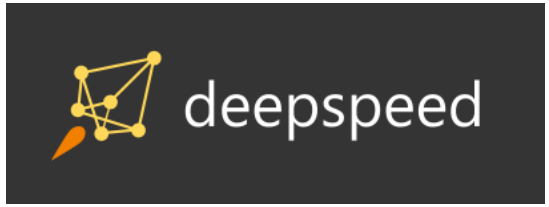
SSP **combines ASP and BSP** and uses a policy to switch between ASP and BSP during training dynamically

The idea is that **the difference in the iteration number** for the fastest and the slowest worker **should not exceed** a user-defined number

There is **no waiting time**, but the **fastest workers may have to wait** for the slowest worker to catch up

The model **convergence guarantee** is high but **decreases as the staleness increases**

Existing Frameworks



What is still an issue

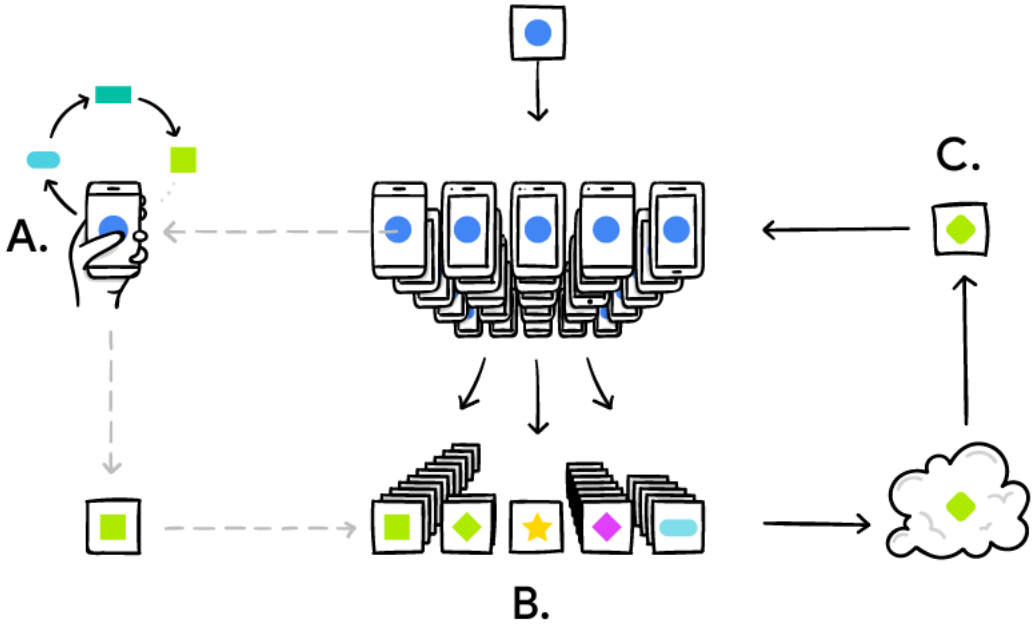
Distributed learning is a way to address the limitation of a single machine

However, Distributed learning, in practice, still needs data to be collected on a cluster or cloud

When users generating data are in order of millions (or even greater) this implies:

- scalability concerns
- privacy concerns

Do we have an answer for these issues?



Let’s play with Federated Learning!

Federated Learning

Intro e Motivation

Federated learning **trains a model across multiple decentralised networked devices** holding **local data** samples **without exchanging** them

Federated learning enables **multiple actors** to build a robust machine learning model **without sharing data**, thus addressing critical issues such as:

- data privacy
- data security
- data access rights
- access to heterogeneous data

Federated vs. Distributed (data parallel) Learning

On the assumptions made on the properties of the **local datasets**:

distributed (data parallel) learning originally aims at **parallelizing computations**

federated learning aims at **training on heterogeneous datasets**

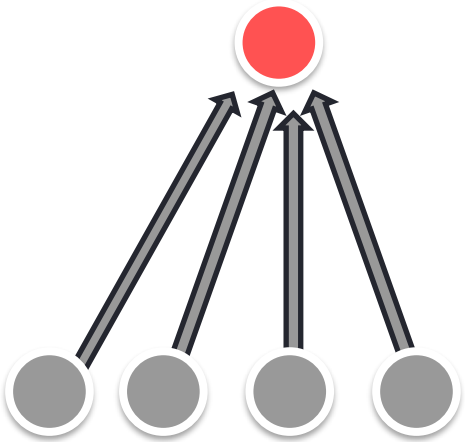
Distributed (data parallel) learning aims at training a single model on multiple servers

- a common assumption is that the local datasets are **independent and identically distributed**
- roughly have the **same size**

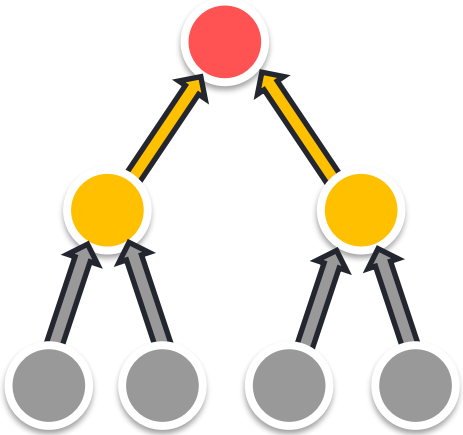
With **Federated learning** the datasets are typically heterogeneous and their sizes vary

Actors involved in federated learning may be unreliable as they are subject to more failures or drop

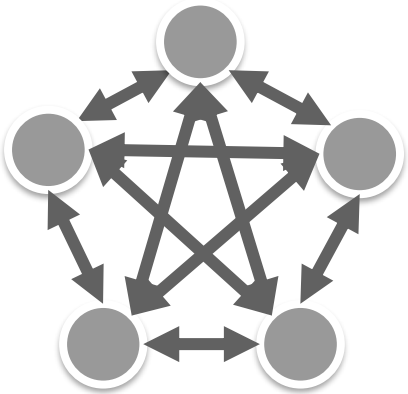
Aggregation architectures



centralised



hierarchical



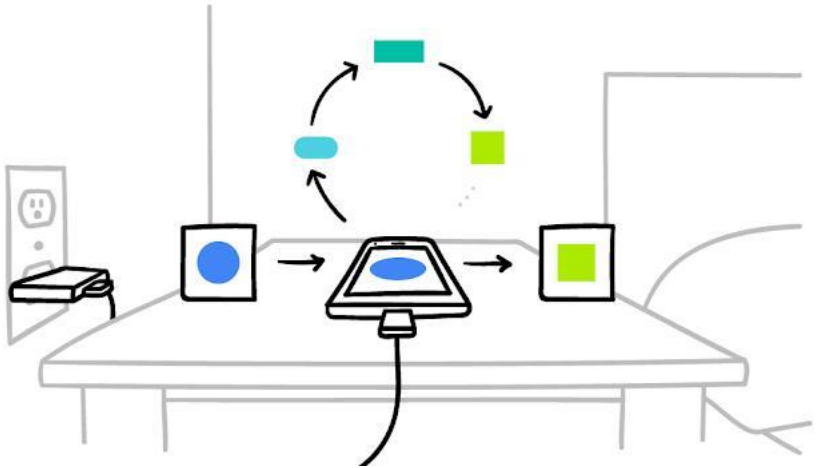
fully decentralised

First-class entities

Actor-centric design and development model

Topology definition

Synchronisation policies



Existing Frameworks

FATE

Substra Foundation

Industry-oriented
Scarce prototyping tools

PySyft

Mostly for federated analytics

intel OpenFL

Flower

Only client-server approaches
Lack of flexibility for new methods

Bound to TF logic
Challenging to implement new algorithms

FedRay

An R&D-Oriented Framework for easy, end-to-end experimentation in Federated Learning

Rapid prototyping and **evaluation** of FL algorithms via:
- Well-known off-the-shelf algorithms
- API for implementing new algorithms

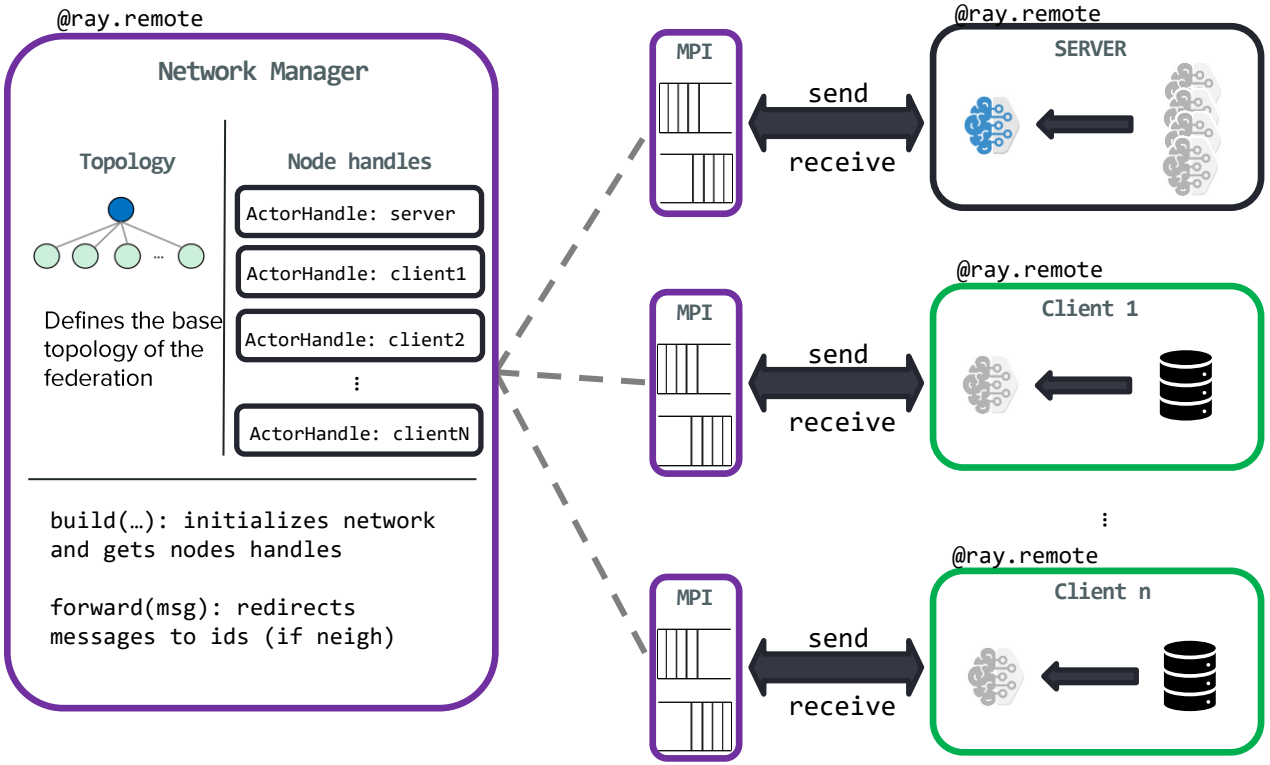
Works with any federation scheme: **Client-Server, Hierarchical, Decentralized**

Both **synchronous** and **asynchronous**

Based on Ray **seamless multiprocessing on any Ray Cluster**

Completely Pythonic API easy implementation and execution

FEDRAY



- Network Manager:
  - Takes care of the network topology (can be dynamic)
  - Keeps references of all the active nodes
  - Forwards messages to participants (only the hex code of the objects in the Ray Object Store)
- Node:
  - Implements the local logic of the federated process
  - Can be either *internal* or *external*
  - Communicates with others via send and receive

FEDRAY

Key use cases

Autnomous driving cars: high number of agents, need to quickly respond to real world situations. Federated learning as a solution for limiting volume of data transfer and accelerating learning

Industry 4.0: privacy of sensitive data for manufacturing companies is of key importance. Federated learning algorithms can be applied to these problems as they do not disclose any sensitive data

e-health: the ability to train machine learning models at scale across **multiple medical institutions** without moving the data is a critical technology

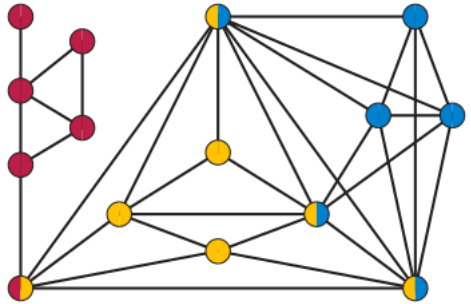
Beyond federated learning

Bringing Decentralised Federated Learning to the next level

Decentralised Federated Learning performances depend on the topology of the network

[H. Kavalionak et al. "Impact of Network Topology on the Convergence of Decentralized Federated Learning Systems," 2021 IEEE Symposium on Computers and Communications, Athens, Greece, 2021]

Adopting “vanilla” approaches for decentralised data exchange (e.g., all-to-all) could lead to inefficient communications



Gossip Learning

Gossip Learning is a method for learning models from **fully distributed** data **without central control**

[István Hegedűs et al. Decentralized learning works: An empirical comparison of gossip learning and federated learning, Journal of Parallel and Distributed Computing, Vol. 148, 2021]

Each node in the network initialises a **local model $\mathbf{w_k}$** (and its age t_k)

The model is then **periodically sent** to another node in the network **without any synchronisation**

A so-called **sampling service** supports the node selection

Gossip Learning

Upon receiving a model w_r, the node **merges** it with the **local model** and updates it using the **local data set D_k**

Merging is achieved **by averaging the model parameters**

In the simplest case, the received model merely **overwrites** the local model

This mechanism results in the models **taking random walks** in the network and being updated when visiting a node

Gossip Learning

Algorithm 1 Gossip Learning

1: $(t_k, w_k, b_k) \leftarrow (\mathbf{0}, \mathbf{0}, 0)$
2: **loop**
3: wait(Δ_g)
4: $p \leftarrow$ selectPeer()
5: send sample(t_k, w_k, b_k) to p
6: **end loop**
7:
8: **procedure** ONRECEIVEMODEL(t_r, w_r, b_r)
9: $(t_k, w_k, b_k) \leftarrow$ merge$((t_k, w_k, b_k), (t_r, w_r, b_r))$
10: $(t_k, w_k, b_k) \leftarrow$ update$((t_k, w_k, b_k), D_k)$
11: **end procedure**

Conclusion

Conclusion

When data and models became too big, their training on a single machine is unfeasible

Using more machines is possible, adopting different strategies for distributed learning

When privacy is a concern, federated learning is a possibility

Federated Learning is a vivid research area, with also some nice proposals for going beyond it