FL_PyTorch:

Optimization Research Simulator for Federated Learning

Konstantin Burlachenko, Samuel Horváth, Peter Richtárik

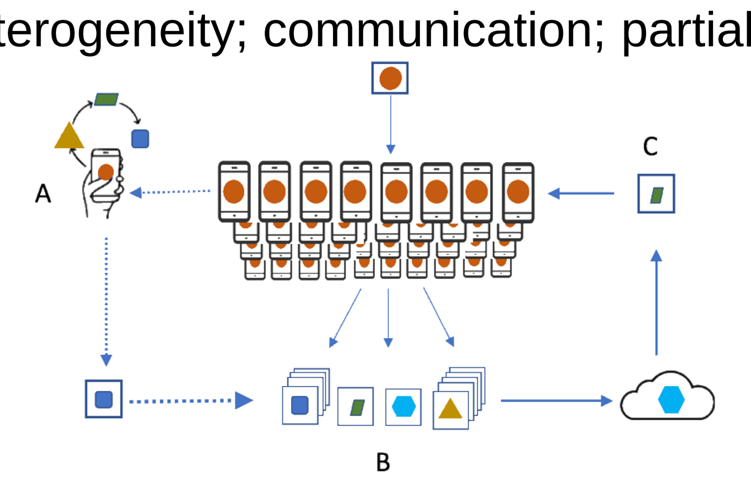
King Abdullah University of Science and Technology
VCC Visual Computing Center

ACM Conference: Distributed Machine Learning DistributedML 2021, December 7, 2021.

Abstract: Federated Learning (FL) has emerged as a promising technique for edge devices to collaboratively learn a shared machine learning model while keeping training data locally on the device. FL is difficult to implement, test and deploy in practice considering heterogeneity in common edge device settings, making it fundamentally hard for researchers to efficiently prototype their optimization algorithms. Our aim is to alleviate this problem by introducing FL_PyTorch: a suite of open-source software written in python that builds on top of one the most popular research Deep Learning (DL) framework PyTorch. We built FL_PyTorch as a research simulator for FL to enable fast development, prototyping and experimenting with new and existing FL optimization algorithms. We provide: State-of-the-arts Optimization Algorithms, standard Models and Datasets, Communication reduction mechanisms. Our goals were: (1) Low Entry Bar; (2) Extensibility; (3) Hardware Utilization; (4) Easy Debugging

Federated Learning (FL) setup

The goal of FL is to find a model deployable on each client while protecting the clients' data. Challenges: # devices; privacy; data heterogeneity; communication; partial participation.



$$\min_x F(x) \triangleq \mathbb{E}_{i\sim\mathcal{P}}[F_i(x)]$$
$$F_i(x) = \mathbb{E}_{\xi\sim\mathcal{D}_i}[f_i(x,\xi)]$$

Some Federated Learning Application

1. Google use it in the Gboard mobile keyboard for applications including next word prediction
2. Android Messages
3. Apple in iOS 13 for QuickType keyboard.
4. Apple uses FL for applications like the QuickType keyboard.
5. Apple uses FL for the vocal classifier for "Hey Siri". With FL Apple upgraded Siri to distinguish the person who owns the Phone from other people's voices.

Supported algorithms out of the box

State-of-the-arts Opt.Algorithms
- Distributed Compressed GD
- FedAVG
- SCAFFOLD
- FedProx
- DIANA
- MARINA

Communication reduction mechanisms
- Local updates
- Limiting the participating clients
- Compressors
  - Identical compressor
  - Lazy or Bernoulli compressor
  - Rand-K
  - Natural compressor
  - Standard dithering
  - Natural Dithering

Models
- ResNet(18,34,50,101,152), VGG(11,13,16,19)
- WideResNets (28_2, 28_4, 28_8)
- Controllable quadratics problems

Datasets:
- Standard FL datasets
- Synthetically generated

Generalized Fed Averaging

Input: Initial model $x^{(0)}$, ClientOpt, ServerOpt
Initialize server state H^0 = InitializeServerState()
for $t \in \{0,1,\ldots,T-1\}$ **do**
  Sample a subset $\mathcal{S}^{(t)}$ of available clients
  Generate state: $s^{(t)}$ = ClientState($H^{(t)}$)
  Broadcast $(x^{(t)}, s^{(t)})$ to workers
  for client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    Initialize local model $x_i^{(t,0)} = x^{(t)}$
    for $k = 0,\ldots,\tau_i - 1$ **do**
      Compute local stochastic gradient g_i = LocalGradient$(x_i^{(t,k)}, s_t)$
      Perform local update $x_i^{(t,k+1)}$ = ClientOpt$(x_i^{(t,k)}, g_i, k, t)$
    end
    Compute local model changes $\Delta_i^{(t)} = x_i^{(t,\tau_i)} - x_i^{(t,0)}$
    Create local state update: $U_i^{(t)}$ = LocalState$(x^{(t)}, x_i^{(t,\tau_i)})$
    Send $(\Delta_i^{(t)}, U_i^{(t)})$ to Master.
  end
  Obtain $(\Delta_i^{(t)}, U_i^{(t)}), \forall i \in \mathcal{S}^{(t)}$.
  Compute $G^{(t)}$ = ServerGradient$(\{\Delta_i^{(t)}, U_i^{(t)}\}_{i\in\mathcal{S}^{(t)}}, H^{(t)})$
  Update global model $x^{(t+1)}$ = ServerOpt$(x^{(t)}, G^{(t)}, \eta_s, t)$
  Update: $H^{(t+1)}$ = ServerGlobalState$(\{\Delta_i^{(t)}, U_i^{(t)}\}_{i\in\mathcal{S}^{(t)}}, H^{(t)})$
end

Generalized Fed Averaging Methods

InitializeServerState

This method returns a dictionary that initializes the server state. The method constructs and initializes the model.

ClientState

By our design client state is stateless. The client state is instantiated at the beginning of each round for each of the selected clients. If you need a client state you may initialize it from the server state.

LocalGradient

Obtain the algorithm specific local gradient estimator g_i .

ClientOpt

Local optimization step using the obtained direction g_i.

LocalState

Collect all the local states that are sent to the master jointly with the local update.

ServerGradient

The estimator of the global direction to be used for the global model update.

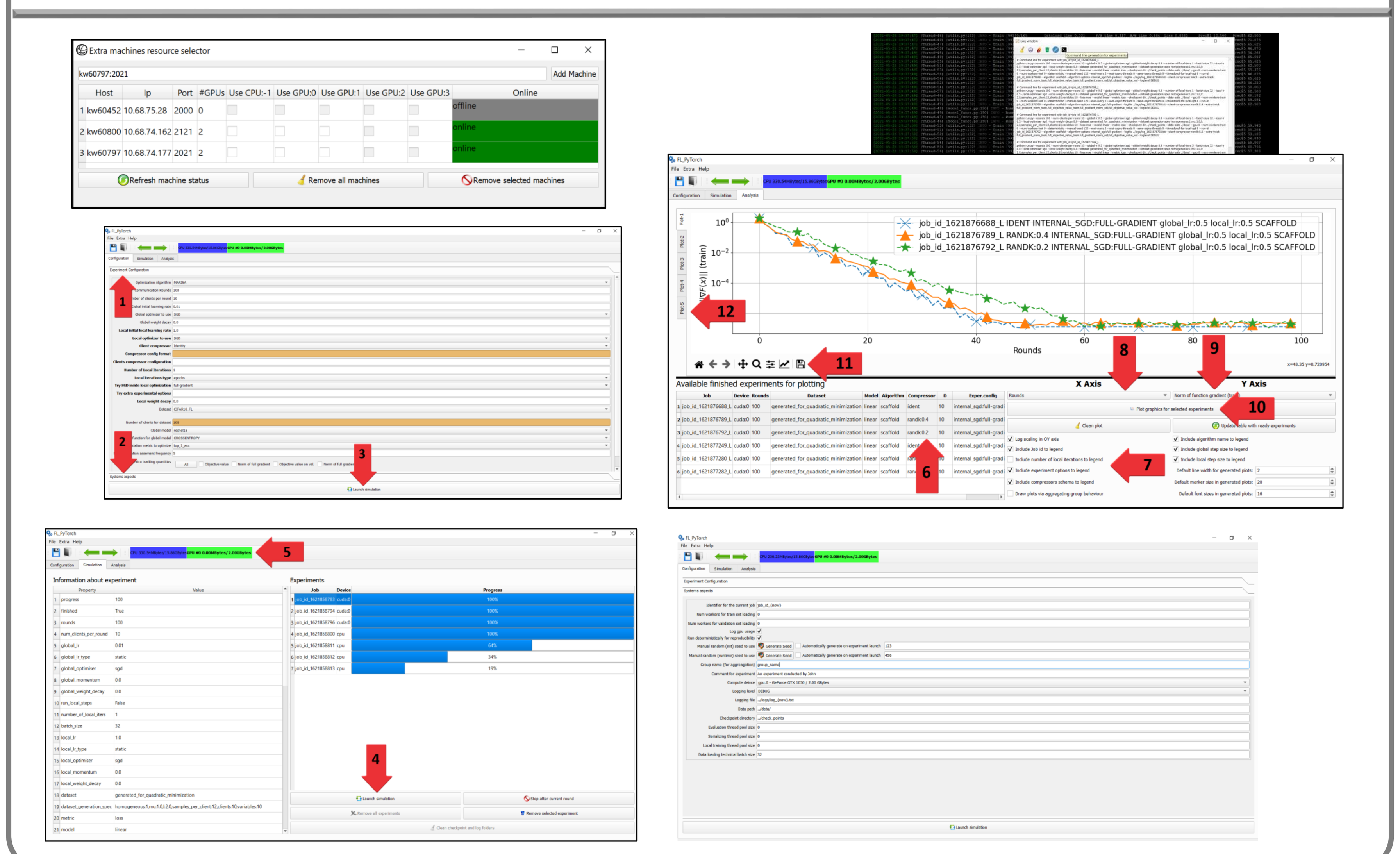
ServerOpt

Server optimization step using the obtained direction G^t.

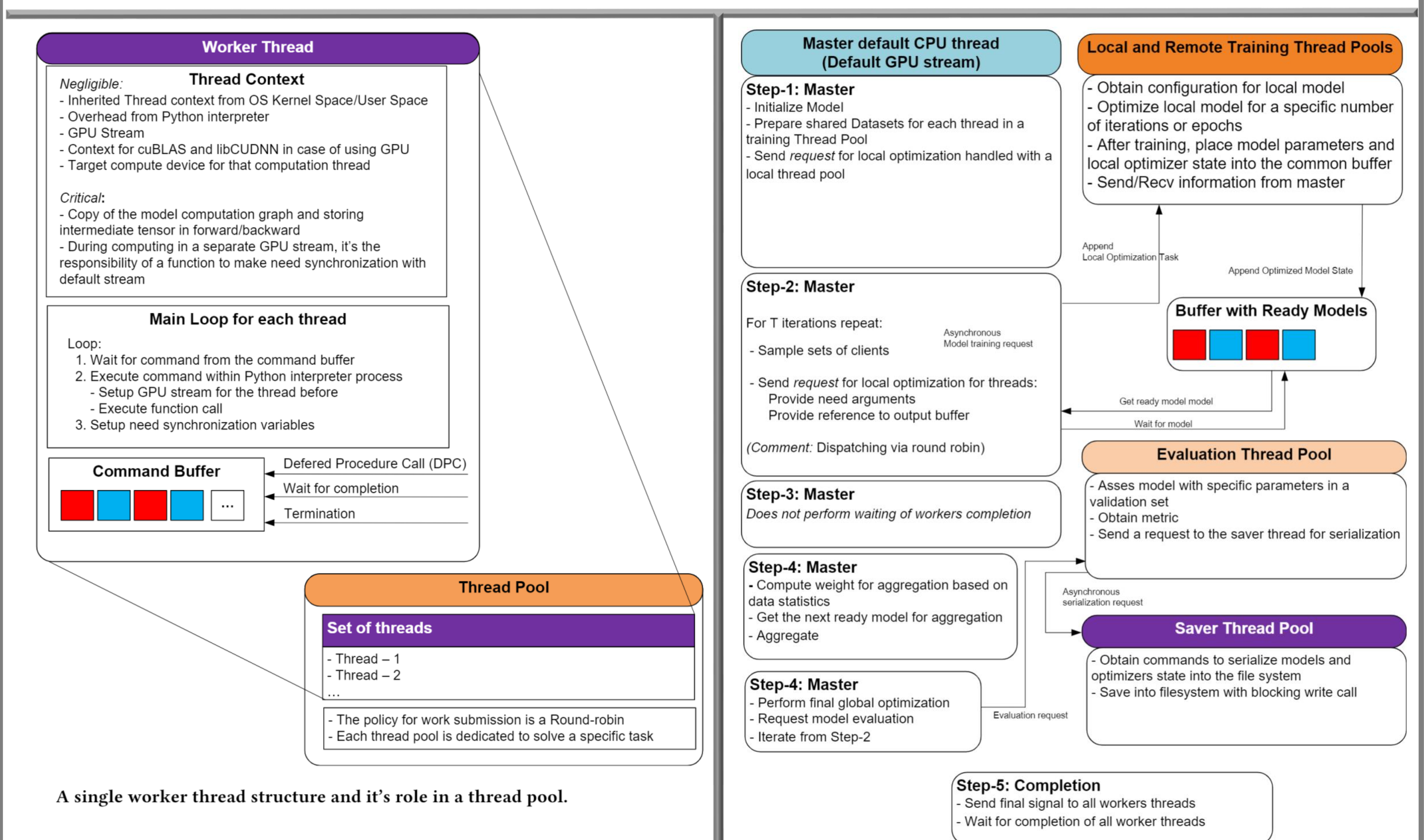
ServerGlobalState

The global server state update

User Interface



Inside Runtime



A single worker thread structure and it's role in a thread pool.

Communication between different threads during Algorithm 1 execution