NSF: CCRI: New: A Scalable Hardware and Software Environment Enabling Secure Multi-party Learning

Aidong Zhang[1], Heng Huang[2], Tianhao Wang[1], Ashish Venkat[1]
[1]Computer Science, University of Virginia, [2]Computer Science, University of Maryland College Park

Multi-Party Computing

Multi-party computing is an emerging computing paradigm to train a learning model collaboratively by multiple workers. The distributed learning and federated learning (FL) are special cases of multi-party computing. Multi-party computing solves the following objective function.

$$\min_{x\in\mathbb{R}^d} f(x) \triangleq \frac{1}{N}\sum_{i=1}^{N} f_i(x), \quad \text{where} \quad f_i(x) = \mathbb{E}_{\xi\sim\mathcal{D}_i} f_i(x,\xi).$$

Our research proposes novel FL algorithms with rigorous theoretical foundations, which could function effectively in the presence of real-world complexities, *e.g.* client drift due to high degree of statistical/system heterogeneity, more complex nested multi-level objectives, and stringent privacy requirements.

I. Proxy Workload Generation

Multi-party computing providers customize their hardware architectures to accommodate specific workloads. Due to the private nature of such workloads, providers rely on synthetic benchmarks to guide hardware design.

Existing proxy benchmark generators not only don't enable fine-grained trade-off decisions with respect to privacy and performance, but often fail to scale with emerging workloads and accelerator-rich platforms.

Our research proposes ProxyVM, a scalable, retargetable compiler system that generates synthetic workloads with great performance predictability [a]. This research is expected to benefit key stakeholders in the ML supply chain by streamlining the hardware design process and minimizing vendor clearing expenses.

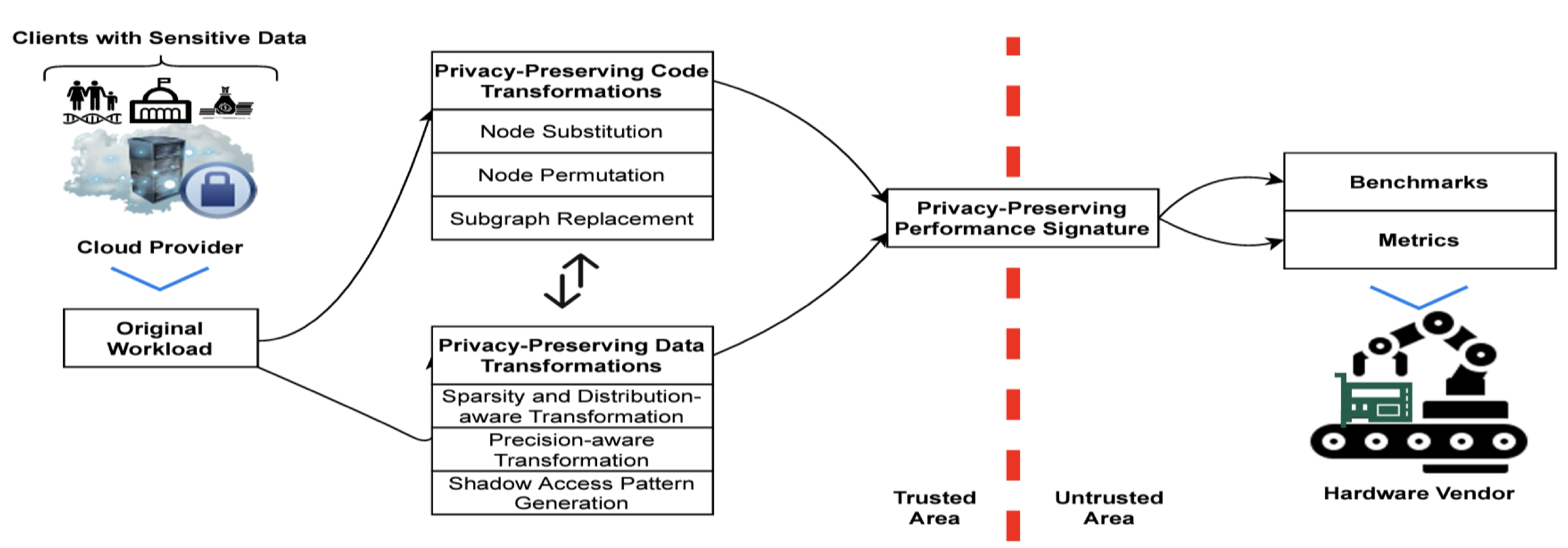

Figure 1: Overview of ProxyVM

[a]ProxyVM: A Scalable and Retargetable Compiler Framework for Privacy-Aware Proxy Workload Generation, SRC TECHCON '22

II. Federated Minimax Optimization

Minimax optimization is critical in many machine learning (ML) applications, such as adversarial training, reinforcement learning, and AUROC maximization. We consider the following federated minimax optimization problem [a]:

$$\min_{x\in\mathbb{R}^{d_1}} \max_{y\in\mathbb{R}^{d_2}} \left\{ F(x,y) = \frac{1}{N}\sum_{i=1}^{N} f_i(x,y) \right\}$$

We focus on nonconvex settings, where $f_i(x,y)$ is nonconvex over $x\in\mathbb{R}^{d_1}$ and concave or nonconcave over $y\in\mathbb{R}^{d_2}$.
Under various different settings, *e.g.* **NC** (NonConvex) + **SC** (Strongly Concave) / **PL** / **C** (Concave), we designed novel optimization algorithms whose sample and communication complexities obtain best known results.

Settings	Algorithm	Sample	Communication
NC-C	Baseline	$O\left(N^{-1}\varepsilon^{-8}\right)$	$O(\varepsilon^{-7})$
	Ours	$O\left(N^{-1}\varepsilon^{-8}\right)$	$O\left(\varepsilon^{-6}\right)$
NC-SC	Baseline	$O\left(\kappa^4 N^{-1}\varepsilon^{-4}\right)$	$O(\kappa^3\varepsilon^{-3})$
	Ours	$O\left(\kappa^3 N^{-1}\varepsilon^{-3}\right)$	$O\left(\kappa^2\varepsilon^{-2}\right)$

[a]Solving a Class of Non-Convex Minimax Optimization in Federated Learning, NeurIPS '23

III. Federated CSO

CSO (Conditional Stochastic Optimization) has many applications in invariant learning, AUPRC maximization, and meta-learning. We consider the following federated CSO problem [a]:

$$\min_{x\in\mathbb{R}^d} \left\{ F(x) = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\xi^n} f^n_{\xi^n}\left(\mathbb{E}_{\eta^n|\xi^n} g^n_{\eta^n}(x,\xi^n)\right) \right\}$$

where $\mathbb{E}_{\xi^n} f^n_{\xi^n}(\cdot)$ is the outer function on the n-th device with random ξ^n, and $\mathbb{E}_{\eta^n|\xi^n} g^n_{\eta^n}(\cdot,\xi^n)$ is the inner function *w.r.t.* the conditional distribution of $\eta^n \mid \xi^n$.
We start from proposing FCSG, which is the first algorithm that tackles federated CSO, to further integrating variance reduction techniques that matches the lower-bound complexity.

Algorithm	Sample	Communication
FCSG (Ours)	$O\left(\epsilon^{-6}\right)$	$O\left(\epsilon^{-3}\right)$
Theoretical Lower Bound	$O\left(\epsilon^{-5}\right)$	-
Acc-FCSG-M (Ours)	$O\left(\epsilon^{-5}\right)$	$O\left(\epsilon^{-2}\right)$

[a]Federated Conditional Stochastic Optimization, NeurIPS '23

IV. Federated AUPRC Optimization

Imbalanced **Binary** classification (*i.e.* positive (negative) data # / total data # ≤ 20%) **Metric:** (1) Accuracy (×), (2) AUROC (✓), (3) AUPRC (✓)
Cross-entropy loss is usually used to optimize accuracy since it is the surrogate function of accuracy. We study how to design a surrogate loss function for AUPRC and introduce an algorithm for AUPRC maximization in the large-scale distributed online setting [a].

$$\min_{\mathbf{x}} F(\mathbf{x}) = \mathrm{E}_{\xi\sim\mathcal{D}^+}\left[f(g(\mathbf{x};\xi)\right] = \mathrm{E}_{\xi\sim\mathcal{D}^+}\left[f\left(\mathrm{E}_{\xi'\sim\mathcal{D}} g\left(\mathbf{x};\xi,\xi'\right)\right)\right]$$

The network system of N worker nodes $\mathcal{G} = (\mathcal{V},\mathcal{E})$ is represented by a doubly stochastic matrix $\underline{\mathbf{W}} = \{\underline{w}_{ij}\} \in \mathbb{R}^{N\times N}$. By setting different network topology $\underline{\mathbf{W}}$, our analysis subsumes different types of collaborative training including federated learning.

[a]Serverless Federated AUPRC Optimization for Multi-Party Collaborative Imbalanced Data Mining, KDD '23

V. Client-Centric FL

We propose Client-Centric FL algorithms, in which we enable several features e.g. **arbitrary client participation**, **asynchronous server aggregation**, and **heterogeneous local computing**, which are ubiquitous in real-world systems but missed in most existing FL works.

Algorithm 1 Client-Centric FL

1: **for** $t \in \{1,\ldots,T\}$ **do**
2: At Each Client (Concurrently)
3: Retrieve x_μ from the server and its timestamp, set $x^i_{\mu,0} = x_\mu$.
4: Select local iteration number $K_{t,i}$, which is time-varying and device-dependent.
5: Δ^i_μ = **LocalOPT** $(i,\eta_l,K_{t,i},x_\mu)$
6: Normalize and send $\Delta^i_\mu = \frac{\Delta^i_\mu}{K_{t,i}}$
7: At Server (Concurrently)
8: Collect m local updates $\{\Delta^i_{t-\tau_{t,i}}, i\in\mathcal{S}_t\}$, where $\tau_{t,i}$ is the random delay of the client i's local update
9: Aggregate $\Delta_t = \frac{1}{|\mathcal{S}_t|}\sum_{i\in\mathcal{S}_t}\Delta^i_{t-\tau_{t,i}}$
10: x_{t+1} = **ServerOPT** $(x_t,\Delta_t,\mathbb{H})$;
11: **end for**
12: **Output:** x_T

We comprehensively study the property of client-centric FL when **ServerOPT** enables momentum and adaptive learning rates.