Platform Design for Privacy-Preserving Federated Learning using Homomorphic Encryption

Wild-and-Crazy-Idea Paper

Hokeun Kim*, Younghyun Kim‡, Hoeseok Yang§

***Assistant Prof. @Arizona State University**

‡Associate Prof. @ Purdue University

§Associate Prof. @ Santa Clara University

Forum on specification and Design Languages (FDL) 2024
@ KTH, Stockholm, Sweden

Session 3: Design Optimization and Exploration

Thursday, September 5, 2024

Disclaimer

- This is a Wild-and-Crazy-Idea Paper
- Ambitious, concrete, and realizable ideas/plans not implemented yet (some are work-in-progress)
- Futuristic and immature plans needing more investigation and discussion

Background – Federated Learning

- Common architecture of federated learning (FL)

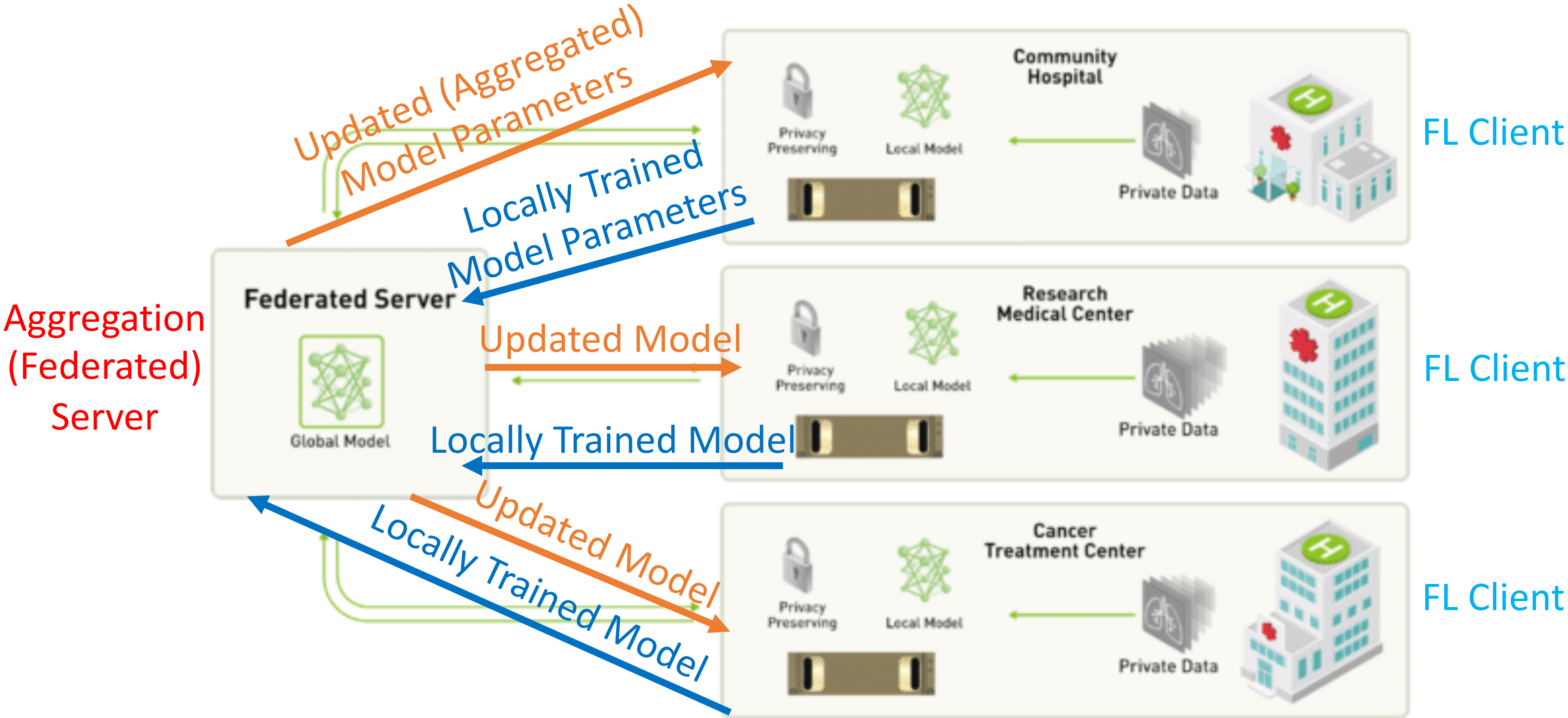


Image from https://blogs.nvidia.com/blog/what-is-federated-learning/

Background – Federated Learning

- Benefits of federated learning (FL)
  - Raw training data stays locally; only model info is shared
    - Example 1: Patients' medical records do not leave the hospital
    - Example 2: Home IoT data (sensor data, images, activity logs) is not shared with the cloud
  - Enhanced privacy compared to the centralized ML model
  - Less communication overhead for sending bulky raw data

- But, aggregation server can still learn sensitive info[1]
  - Inference of sensitive information using shared model parameters[2]

[1] Sharma, and Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, 2020.
[2] Pyrgelis *et al.*, "Knock knock, who's there? Membership inference on aggregate location data," in *NDSS* 2018.

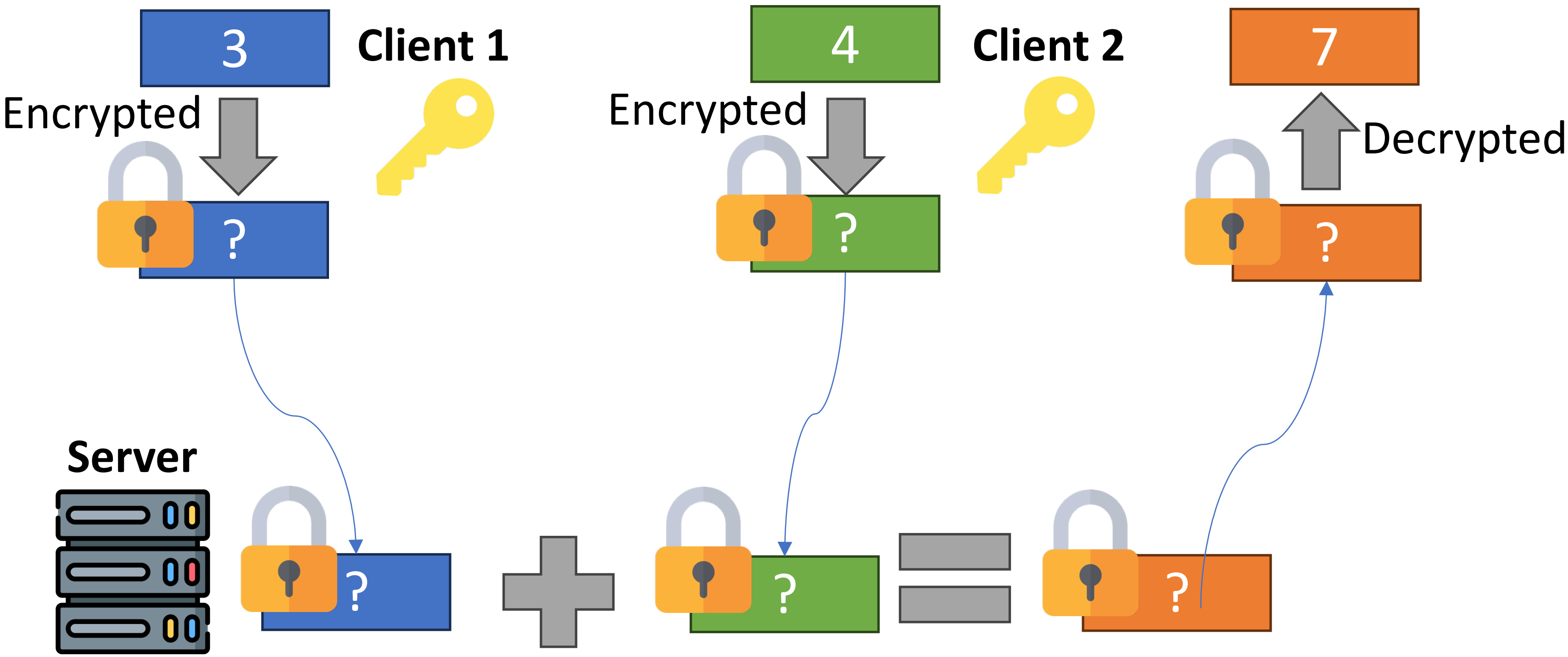
Background
– Further Privacy-Preserving FL

- Multiple ways to protect the privacy of the ML models shared by FL clients
  - Differential privacy[1]
    - Statistical approach
  - Homomorphic encryption (HE)[2]
    - Cryptography-based approach

[1] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE TIFS*, 2020.
[2] C. Zhang, *et al.*, "BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning," in USENIX ATC 2020.

- Homomorphic encryption (HE)
  - Stronger guarantees compared to differential privacy
  - However, much more expensive (than diff. privacy)

Background
– Homomorphic Encryption (HE)



- Arithmetic computation over ciphertext (encrypted data) w/o knowing crypto key or plaintext (unencrypted data)

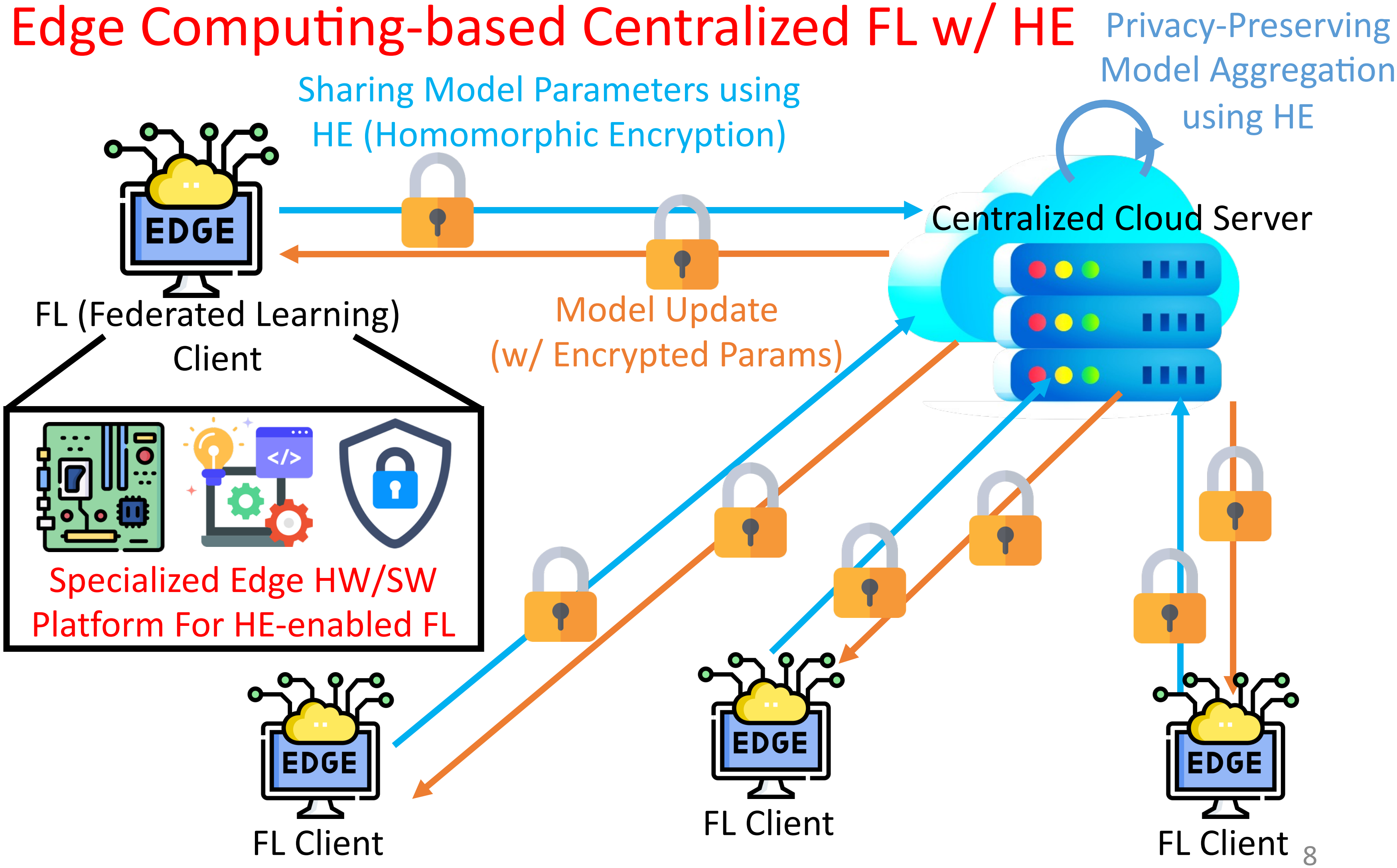
*Common arithmetic operations supported by HE are addition and multiplication.

Background – HE-enabled FL

- ML parameters (weights) are encrypted with HE
- Encrypted parameters are sent to aggregation server
- Model aggregation is essentially a weighted sum computation with multiplications (✖) & additions (✚)
  - E.g., FedAvg
  - $p_{i+1} \leftarrow \sum_{k=0}^{N-1} w_k \cdot p_i$
- ✖ and ✚ can be performed on encrypted data
- Aggregation server can perform model aggregation without knowing plaintext parameters from FL clients

Overall System Model

- Edge Computing-based Centralized FL w/ HE



Target System for Platform Design: FL client w/ HE running on the edge

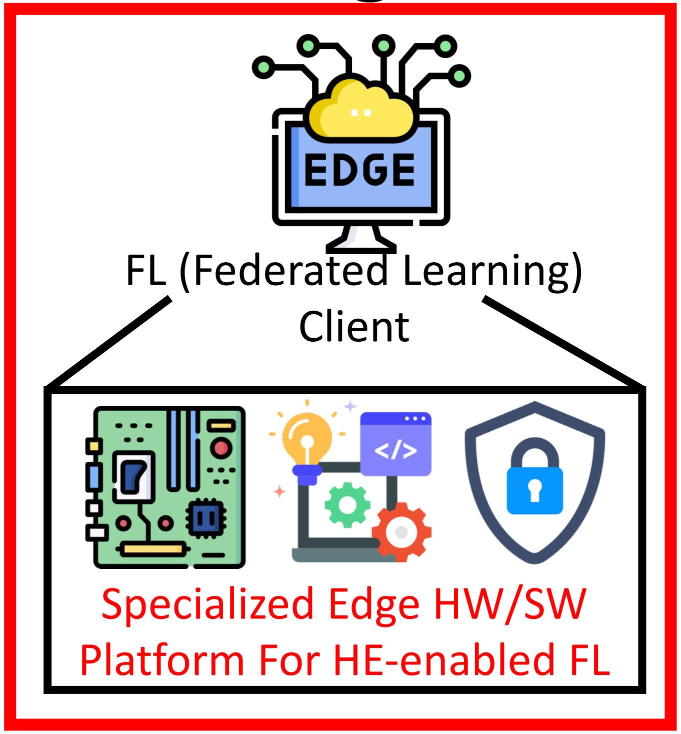
Heterogeneous workloads
& requirements!

(1) ML (FL)

- High volume of low-precision comp.
- Accelerated by TPUs/GPUs

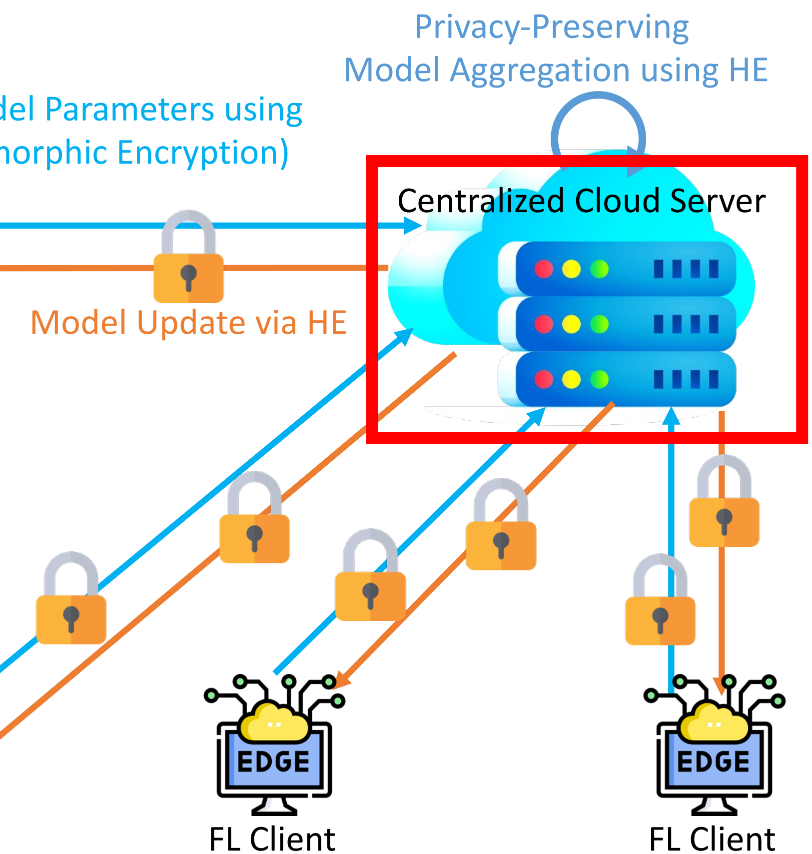
(2) HE

- Massive memory bandwidth
- Computation heavy (e.g., bootstraping*)
- Requires HBM* DRAM and HPC*



- Bootstraping: Process of refreshing ciphertex
- HBM: High bandwidth memory
- HPC: High performance computing

Threat Model

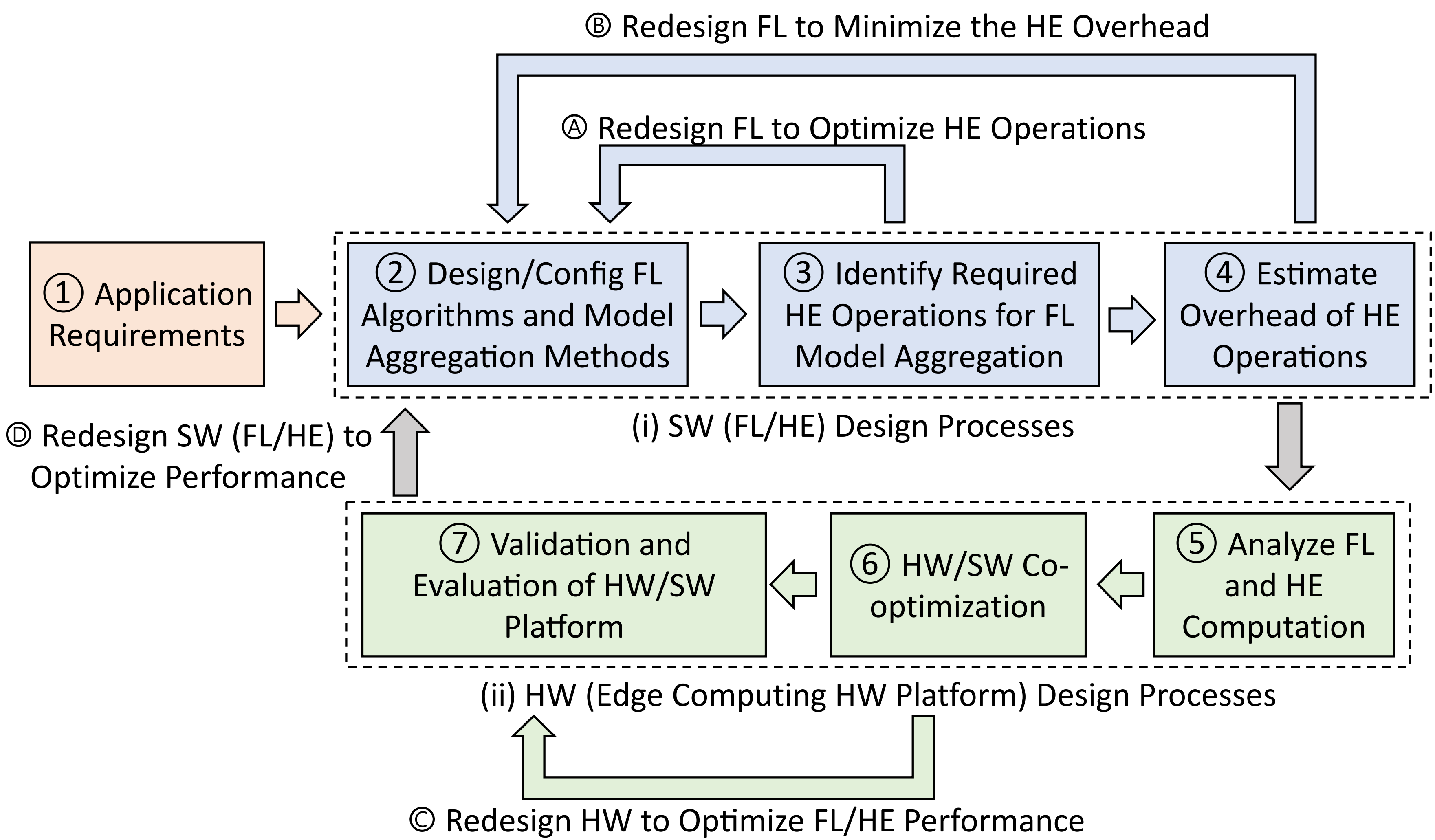


Honest-but-curious server model

- Widely used model in privacy-preserving FL approaches[1][2]
- (Honest) aggregation server is trusted for computation
- However, FL clients still do not want to expose ML model parameters (sensitive info)

[1] J. Le , *et al.*, "Privacy- preserving federated learning with malicious clients and honest-but-curious servers," IEEE TIFS 2023.
[2] C. Zhang, *et al.*, "BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning," in USENIX ATC 2020.

Proposed Platform Design Process

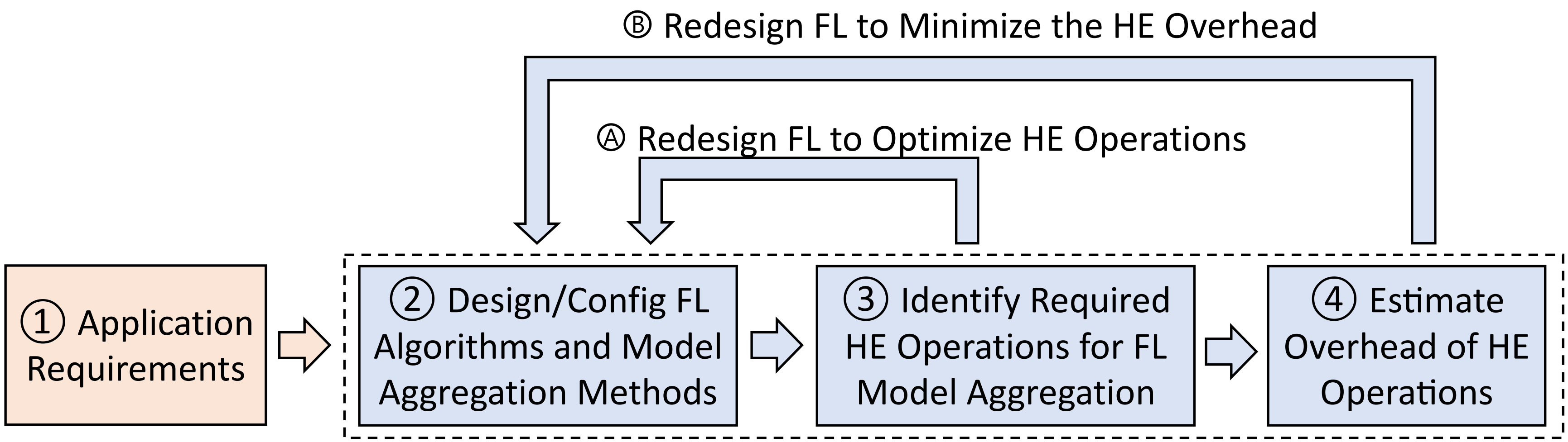


Specifying FL Application Requirements

① Application Requirements

- First and critical step
- Problem (application) domain
- Data types
- ML model's inputs/outputs
- Required performance
  (e.g., latency, accuracy, cost, etc.)

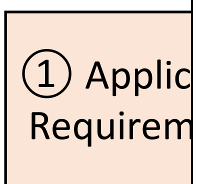
Software Design



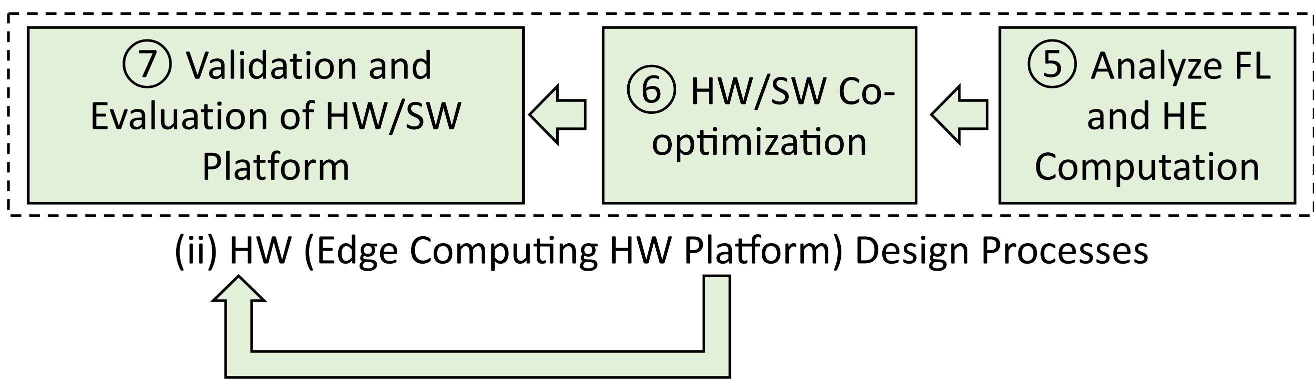
(i) SW (FL/HE) Design Processes

- Possible SW optimizations
  - $p_{i+1} \leftarrow \sum_{k=0}^{N-1} w_k \cdot p_i$
  - ✖ is more expensive than ✚ in HE
  - Let FL clients know their weights & have them send weighted params
  - Reduces server computation time, thus, reduces model update latency
- Design/config parameters
  - Aggregation algorithms
  - Update from a subset of FL clients
  - Model update frequency
  - Size/complexity of ML models
  - HE algorithms

Hardware Design

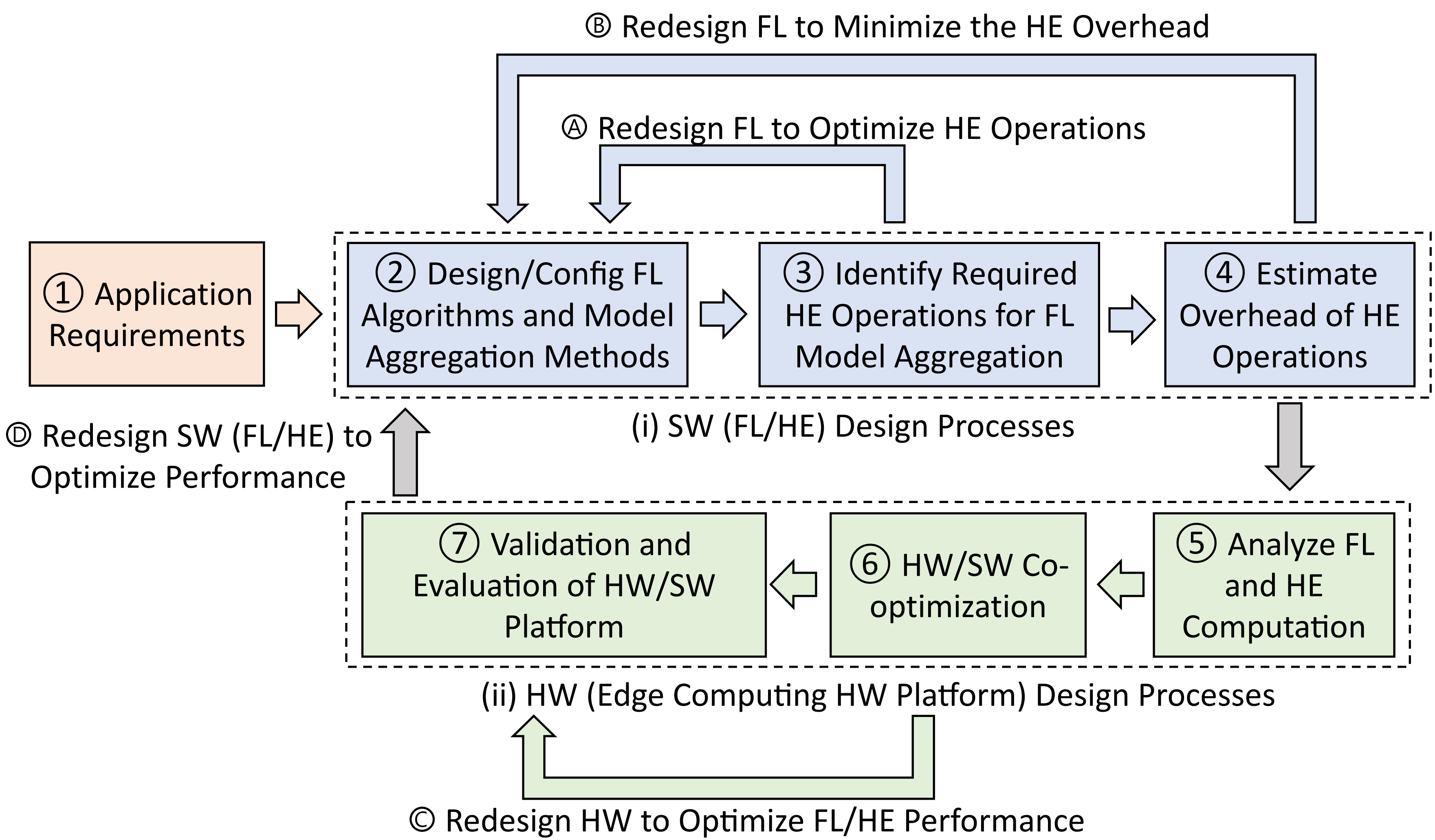


- Design challenges
  - Heterogenous computation/mem requirements
  - Traditional computer arch can't support HE efficiently
  - Enormous ciphertext length (does not fit in on-chip mem, e.g., caches)
  - Necessary to reduce off-chip mem transactions (e.g., data reuse, in-mem computation, etc.)

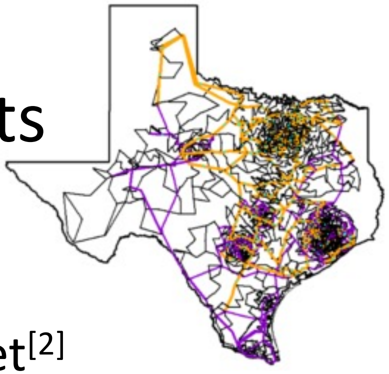


© Redesign HW to Optimize FL/HE Performance

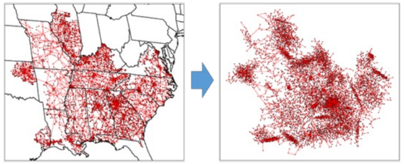
Iterative SW/HW Design Process



HW/SW Platform Design Strategies



- (1) Profiling & analysis of HW requirements
  - Public dataset examples for FL training
    - Texas A&M University Electric Grid Datasets[1]
    - Columbia University Synthetic Power Grid Data Set[2]
  - Prototype FL + HE with open-source software
    - FL: Flower[3], NVIDIA FLARE[4]
    - HE: OpenFHE[5]



[1] https://electricgrids.engr.tamu.edu/
[2] https://wimnet.ee.columbia.edu/portfolio/synthetic-power-grids-data-sets/
[3] https://flower.ai/
[4] https://developer.nvidia.com/flare
[5[https://www.openfhe.org/

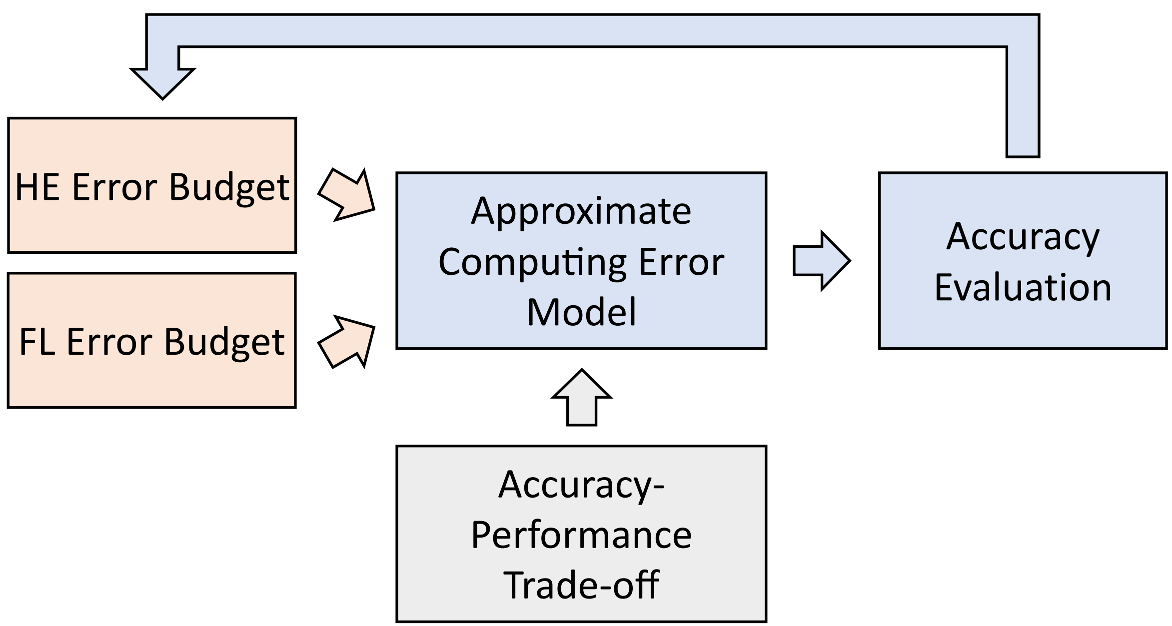
HW/SW Platform Design Strategies

- (2) Design space exploration (DSE)
  - Using profiled HW requirements & COTS HW components
  - Develop dedicated HW using FPGA if necessary
  - Commonly used components in HE acceleration
    - Number theoretic transform (NTT)
    - Multi-scalar multiplication (MSM)
- (3) Design of accelerators
  - Based on DSE results, especially bottlenecks that can be accelerated by HW
  - Start with the previously proposed HE accelerator ideas, e.g., F1[1] (with HBM), FLASH[2] (without HBM)

[1] N. Samardzic *et al.*, "F1: A fast and programmable accelerator for fully homomorphic encryption," Micro 2021
[2] J. Zhang *et al.*, "FLASH: Towards a high-performance hardware acceleration architecture for cross-silo federated learning," NSDI 2023

HW/SW Platform Design Strategies

- (4) Acceleration via approximate computing
  - FL (ML) applications are error tolerant
  - HE with approximate computing – enhanced speed, power efficiency, relaxed HW requirements

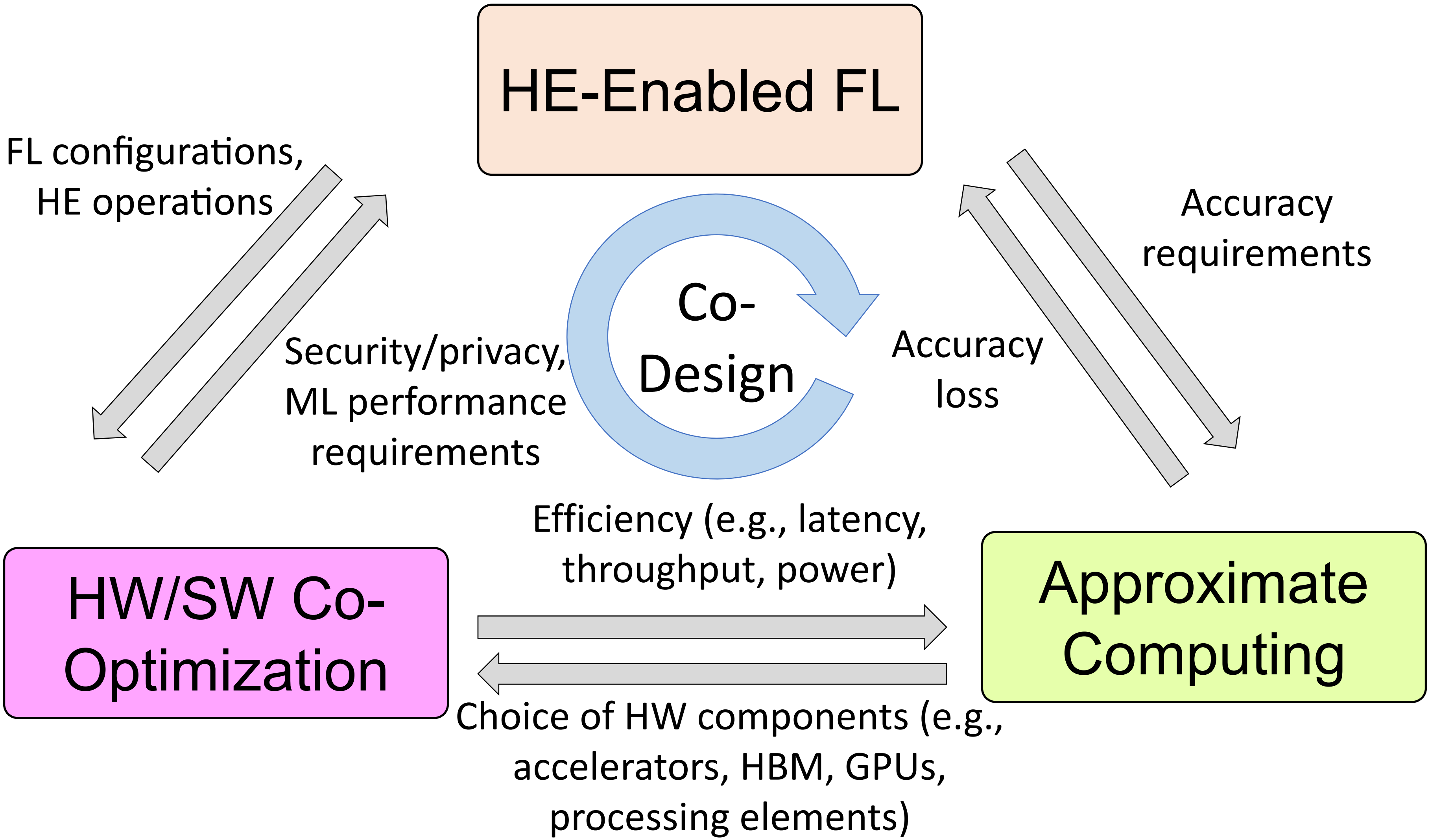


HW/SW Platform Design Strategies

- (5) Middleware and runtime for optimized usage of the underlying hardware
  - Software stack in HPC environments is critical[1]
  - E.g., task scheduling and I/O management
  - Memory management policies
  - Scheduling of computation, mem usage, and I/O ops.
  - Fine-grained control of hardware components
  - Balance inference service vs. training/model update (HE)

[1] D. Boehme *et al.*, “Caliper: performance introspection for HPC software stacks,” SC'16

Strategies as Codesign



Application Areas

- ML applications with a limited volume of sensitive, high-value local data
  - No need for FL if the local data volume is enough
  - Not worth HE if nonsensitive or little value data
- FL involving entities that can afford the cost of a dedicated HW/SW platform
  - Benefit of platform > cost of platform
- Possible application domains
  - Healthcare/medical insurance
  - Critical infrastructure (power grids, transportation, etc.)
  - Intrusion/malware detection

Thank you for your attention!

Summary

- Privacy-preserving FL with HE is promising
- FL with HE has high mem/computation requirements
- Dedicated SW/HW platform is needed
- FL with HE has great optimization potential in SW/HW
- We plan to address the SW/HW platform design challenges with the aforementioned five strategies

Contact Information for Further Discussions

- https://hokeun.github.io, hokeun@asu.edu