2024 Forum on Specification &amp; Design Languages (FDL) | 979-8-3315-0457-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/FDL63219.2024.10673864

Platform Design for Privacy-Preserving Federated Learning using Homomorphic Encryption

Wild-and-Crazy-Idea Paper

Hokeun Kim
Arizona State University
hokeun@asu.edu

Younghyun Kim
Purdue University
younghyun@purdue.edu

Hoeseok Yang
Santa Clara University
hoeseok.yang@scu.edu

***Abstract*—Federated learning (FL) has been increasingly widely used for distributed and privacy-preserving machine learning (ML) environments, as the raw training data can stay local to clients while leveraging model updates from individual clients. Homomorphic encryption (HE) technologies can provide additional privacy protection for FL by encrypting the model update parameters while allowing model aggregation on a remote server. Although HE-enabled FL seems to be a promising privacy-preserving ML solution, it requires significantly more computational and memory resources, requiring a dedicated hardware and software platform. In this paper, we discuss preliminary but concrete and realizable research ideas for analyzing the requirements for HE-enabled FL and for designing a hardware and software platform. Furthermore, we propose a platform co-design process that considers various design stages and challenges in the platform co-design.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Federated learning (FL) [1] is a set of machine learning (ML) techniques devised for multiple entities collaborating to solve ML problems, expected to solve multiple critical challenges in traditional distributed systems, such as privacy, communication costs, and robustness [2]. Privacy challenges in distributed learning environments are mitigated by FL as it preserves clients' private data by only sharing the metadata of the locally trained ML model to be aggregated for updating the shared ML model rather than sharing the raw data. However, it still suffers from privacy concerns as the metadata of locally trained ML models can still reveal sensitive information of the local data to attackers [3], for example, through the inference attack for the given model parameters [4]. These privacy concerns have been one of the major obstacles to adopting FL in critical applications, including digital healthcare [5].

Among efforts to address privacy concerns in FL, homomorphic encryption (HE) or fully homomorphic encryption (FHE) technologies [6] have become a potentially promising solution to protect FL clients' privacy from aggregation servers. One example approach is BatchCrypt [7], which efficiently encrypts a batch of quantized gradients for model updates using HE. The emergence of significantly cost-efficient HE algorithms, such as CKKS [8], is also facilitating the adoption of HE, which has previously been prohibitively costly in computation. HE allows for model aggregation operations on remote servers without exposing the actual model parameters, thus ensuring stronger privacy protection. HE-based privacy solutions to FL provide stronger cryptographic guarantees compared to other solutions, for example, differential privacy-based FL [9]. Thus, integrating FL with HE is now considered a promising solution with great potential to enable privacy-enhanced FL in a wide range of use cases.

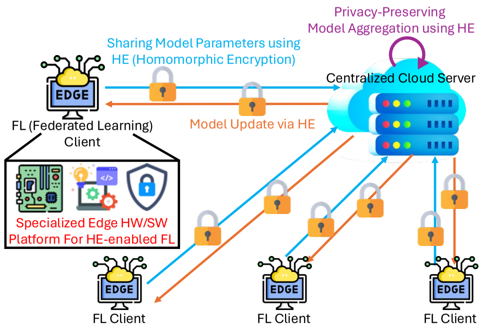
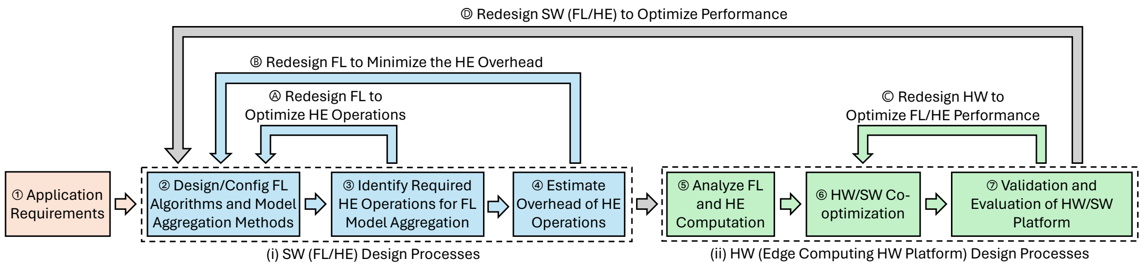


Fig. 1. Overview of an FL application with HE running on FL clients on the edge and the model aggregation server on the cloud. Our goal is to design a specialized HW-SW platform for the FL client running on the edge.

However, performance-wise challenges still need to be addressed to enable HE-based FL systems. Both FL and HE are computation-heavy and memory-hungry applications, requiring underlying computational architecture for high-performance computing (HPC), including components such as GPUs, high bandwidth memory (HBM) DRAM technology [10], and hardware accelerators [11], [12]. Also, FL and HE have very different types of workloads requiring heterogeneous architecture on the same system. Specifically, FL, like other ML applications, requires a high volume of low-precision computation that TPUs [13] or GPUs can accelerate, whereas HE demands massive memory bandwidth for the encryption and decryption process in addition to computation on large numbers, including the bootstrapping process [14] which incurs enormous computation overhead.

In this position paper, we discuss the design ideas and steps toward a dedicated hardware-software platform for FL with HE. Specifically, we elaborate on the research efforts required to design such a platform, including (1) application-specific profiling and analysis for hardware requirements (Section IV-A), (2) design space exploration for heterogeneous high-performance hardware components (Section IV-B), (3) design of optimized accelerators for FL and HE workloads (Section IV-C), (4) acceleration through approximate computing (Section IV-D), and (5) middleware and runtime for optimal usage of the underlying hardware (Section IV-E).



Ⓓ Redesign SW (FL/HE) to Optimize Performance
Ⓑ Redesign FL to Minimize the HE Overhead
Ⓐ Redesign FL to Optimize HE Operations
Ⓒ Redesign HW to Optimize FL/HE Performance
① Application Requirements
② Design/Config FL Algorithms and Model Aggregation Methods
③ Identify Required HE Operations for FL Model Aggregation
④ Estimate Overhead of HE Operations
⑤ Analyze FL and HE Computation
⑥ HW/SW Co-optimization
⑦ Validation and Evaluation of HW/SW Platform
(i) SW (FL/HE) Design Processes
(ii) HW (Edge Computing HW Platform) Design Processes

Fig. 2. Overall timeline of the proposed hardware/software (HW/SW) platform design and redesign processes for HE-enabled FL Applications.

II. System and Threat Models

Fig. 1 illustrates an overview of the target system and the platform. The target system to be designed as a platform is an edge computing device running an FL client. Each FL client will be connected to a powerful, centralized server, possibly on the cloud. The FL clients will periodically, for example, every few hours, share ML model parameters, such as gradients, which will be encrypted using HE. The centralized server will perform an FL algorithm for model aggregation [15] (e.g., FedAvg [16], FSVRG [17], BatchCrypt [7]) on the shared model parameters encrypted by HE. The aggregated model, still encrypted by HE, will be sent back to each FL client as a model update. We acknowledge that there are FL models based on decentralized model aggregation [18]; in this paper, we will focus on the discussion on centralized FL.

System Model: The primary target system for the platform design in this paper is an FL client using HE running on the edge. In addition, we also consider the design requirements for the cloud server performing model aggregation. The system configuration parameters include (1) an ML model and algorithm, (2) an FL algorithm that determines the model aggregation method and ML parameters to be shared, (3) an HE algorithm to protect the shared ML parameters, (4) runtime and middleware for running software and communication, and (4) a hardware architecture of the edge device.

Threat Model: The target system's threat model assumes that the FL clients trust each other and share keys for HE. Our threat model also assumes the honest-but-curious aggregation server model, similar to the threat model used by Ada-PPFL [19] and BatchCrypt [7], which is widely used for privacy-preserving approaches in FL. Specifically, FL clients trust the aggregation server on the cloud in terms of computation; that is, the aggregation server will perform the correct computation honestly. However, FL clients do not want to share any sensitive information that can potentially expose privacy, including the model parameters used for model aggregation. Thus, we assume the aggregation server does not have knowledge of the HE keys used by FL clients.

III. Design Challenges and Requirements

Fig. 2 illustrates the overall timeline of the proposed processes of hardware/software (HW/SW) design and redesign for an edge platform running FL clients using HE. To begin with, ① we first specify the requirements of the FL application, including the problem domain, data types, ML model's inputs and outputs, required performance, etc. Then, (i) SW design (FL and HE) should be done, followed by (ii) the design of HW components.

A. Software Design Challenges

The software design part involves the ② design and configuration of FL algorithms and model aggregation methods for the specified application, ③ figuring out the matching HE operations to perform FL algorithms and model aggregation, and ④ the estimation of the overhead caused by the HE operations. As HE operations can be extremely expensive, especially multiplication, we may have to Ⓐ redesign the FL algorithms to optimize the required HE operations or to Ⓑ minimize the overhead of HE operations.

For example, FedAvg [16] is a widely used basic model aggregation method when the model parameters at the ith round are p_i, and the kth client's weight is w_k, the updated model parameters for the next round are defined as a simple weighted sum:

$$p_{i+1} \leftarrow \sum w_k \cdot p_i$$

where $0 \leq k < N$ of total N clients, $\sum_{k=0}^{N-1} w_k = 1$, and $w_k \geq 0$ for all k. This involves multiplication for model aggregation. To eliminate the multiplication on the server side, we can let the clients know their weights in advance and have the clients send their parameters multiplied by their weights so that the server only needs to do less expensive HE operations, which in this case are additions.

At each round, the central server chooses a subset of n clients out of the total N clients to report their model parameters, such that $n < N$ and $\sum_{k=0}^{n-1} w_k = 1$. Thus, the overhead for each client to perform HE to send encrypted model parameters is reduced to n/N. We can also adjust the period of model updates. If we use a longer period of model updates, each FL client and the server need to perform HE operations less frequently. The size and complexity of the ML model is another knob that can change the HE and HW requirements significantly. If we use a smaller and lighter model with fewer model parameters, we can reduce the number of HE operations and thus the HW cost.

B. Hardware Design Challenges

While recent algorithmic advancements have achieved a significant speedup in HE computation [20], there is a multipleorders-of-magnitude gap in the computation efficiency between HE and unencrypted computing [21]. The major challenge in HE computation is the enormous amounts of computation and memory transactions that traditional computer architectures cannot efficiently provide [22]. More specifically, the large size of the ciphertext poses a number of problems. Because of the sheer length of the ciphertex that does not fit in on-chip memory, the HE computation requires a large off-chip memory and generates massive off-chip memory transactions. Reducing off-chip memory transactions, such as data reuse and computing-in-memory, can greatly improve performance.

Since the edge nodes in the proposed system are relatively resource-constrained in terms of power and cost compared to the cloud, the design of optimal HW architecture for the edge nodes through HW-SW co-design is critical for the proposed FL-HE ensemble. This process, as illustrated in Fig. 2, involves ⑤ the estimation of HW requirements or overheads based on the specifications of FL and HE operations. Specifically, factors such as the model update period, memory, and execution time overheads of HE operations are critically modeled based on key sizes. Subsequent steps include ⑥ HW-SW partitioning, and ⑦ validation via co-simulation or performance estimation, emphasizing the necessity of design space exploration (DSE) to pinpoint the optimal HW configuration setup. Ⓒ Redesign of HW or Ⓓ SW may be necessary to further optimize or accelerate the FL and HE performance.

IV. Hardware and Software Platform

In this section, we present several software and hardware strategies that we adopt in our platform.

A. *Profiling and Analysis for Hardware Requirements*

The first step to achieving research goals is to profile and analyze the workload of FL and HE. Specifically, we will first build a practical FL application involving sensitive data using datasets available online. For example, we can leverage the public datasets on electric power grids, such as the ones from Texas A&M University[1] or Columbia University[2], to set up an FL model for predicting power usage over time.

Next, we will build a distributed FL application using HE for model aggregation. We will use the state-of-the-art FL frameworks and platforms such as Flower Framework [23] and NVIDIA FLARE [24] in addition to HE or FHE libraries such as OpenFHE [25] to realize the prototype application. By varying the aggregation algorithms, model update frequency, and model pruning/compression techniques for FL and using different HE algorithms, we will profile the prototype system and analyze the performance requirements for hardware running FL with HE.

B. *Design Space Exploration*

While design space exploration (DSE) should be conducted not only in high-performance computing (HPC) environments and the cloud [26], it is also essential in edge nodes, where resource constraints are even more constrained. Based on the profiling and analysis results on FL with HE, we will perform DSE for cost-effective architecture involving heterogeneous hardware components that are designed for FL and HE. This exploration is driven by a quantitative modeling of overheads tailored to the FL/HE specifications [27], HE operation types, and key sizes, ensuring efficient and accurate performance analysis. In particular, number theoretic transform (NTT) and multi-scalar multiplication (MSM) components, which are commonly used in HE accelerators, may have different optimized forms depending on the given HE requirements. Therefore, it is necessary to employ a model-based design methodology that can support various degrees of parallelism [28]. Also, while performing DSE for the edge HW platform, it is essential to incorporate sophisticated real-time scheduling techniques that quantitatively analyze the Quality of Service (QoS) of inference services. This is crucial as edge nodes must not only handle model updates but also concurrently provide inference services.

We will initiate our DSE with commercial-off-the-shelf (COTS) hardware components that primarily include CPUs and GPUs, albeit with limited programmability. This initial phase will necessitate extensive profiling on different CPUs and GPUs. As we progress, we plan to extend our target to FPGAs and leverage open-source accelerators as starting points. For ML-guided DSE, we will leverage existing open-source tools such as Colmena [29][3].

C. *Design of Optimized Accelerators*

Performance analysis and DSE of hardware components will open up opportunities to further customize the hardware for the computational workload of FL and HE. We will utilize open-source accelerators as seed architectures, modifying and optimizing them based on the outcomes of the DSE. This optimization process will include adjustments to memory size, key size, and the degree of component duplication according to the specific requirements identified during DSE. This tailored approach ensures that the accelerators are precisely aligned with the operational demands and performance targets of the specified FL/HE use case.

Since the volume of encrypted data is substantial in HE, communication and memory often become a bottleneck in the hardware design. Consequently, optimizing the memory structure is of paramount importance. The existing HE accelerators are prototyped on high-end FPGA, with HBM [11] or without HBM [12], due to its memory-intensive algorithm behavior. Given these considerations, the accelerators derived from the results of DSE will be generated in RTL, optimized for the given FL/HE use case. This optimization will specifically tailor features such as on-chip memory size, bus bandwidth, the parallelism of components, and the utilization of High Bandwidth Memory (HBM), ensuring that the accelerator meets both performance efficiency and cost-effectiveness in its deployment on edge platforms.

[1]https://electricgrids.engr.tamu.edu/

[2]https://wimnet.ee.columbia.edu/portfolio/synthetic-power-grids-data-sets/

[3]https://colmena.readthedocs.io/en/latest/

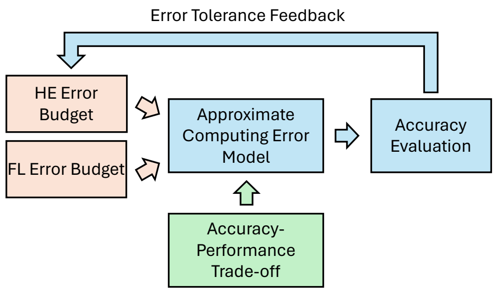


Fig. 3. Approximate computing that we apply to HE and FL acceleration.

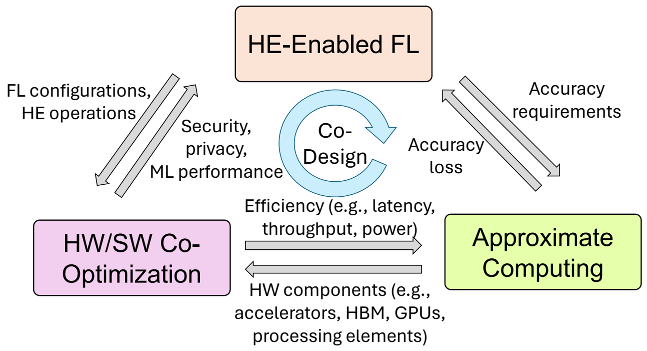

HE-Enabled FL
FL configurations, HE operations
Security, privacy, ML performance
Co-Design
Accuracy requirements
Accuracy loss
HW/SW Co-Optimization
Approximate Computing
Efficiency (e.g., latency, throughput, power)
HW components (e.g., accelerators, HBM, GPUs, processing elements)

Fig. 4. Proposed interactive, iterative HW/SW co-design process of platform for HE-enabled FL using approximate computing.

D. Acceleration through Approximation

Error tolerance is the key property that can be exploited to accelerate FL and HE computation, and approximate computing is a promising solution to this. Approximate computing aims to improve performance and efficiency by relaxing the exactness constraint on computation. Various approximate computing techniques have been proposed [30], and many of them can help accelerate FL and HE. For example, approximate multiplication-less integer FFTs (Fast Fourier Transforms) and IFFTs (inverse FFTs) to exploit error tolerance and speed up computation [31]. Another example of approximate computing is a systolic DNN accelerator with approximate multipliers to accelerate FL [32].

We will adopt various approximate computing techniques to our platform to accelerate FL and HE computation as shown in Fig. 3. Unary computing, which allows the use of extremely efficient arithmetic hardware, can be utilized. Matrix multiplication is a fundamental operation in HE computation; thus, general matrix multiplication (GEMM) based on efficient unary computing such as [33] will be an effective solution. HE can also benefit from dynamic accuracy control of approximate computing. Some approximate computing hardware supports dynamic adjustment of the accuracy level, where the accuracy can be traded off for performance and efficiency [34]–[36]. Such approaches can be used to perform adaptive bootstrapping based on the available error budget in HE.

E. Middleware and Runtime for Optimized Usage of the Underlying Hardware

The software stack in HPC environments plays a critical role [37] for task scheduling [38] and I/O management. We will apply hardware/software co-design approaches [39] when developing the runtime. That is, in parallel to the hardware architecture research, we plan to work on the design of middleware and runtime for optimized usage of the underlying hardware. This includes efficient memory management policies, scheduling of various computational, memory, and I/O operations, and fine-grained control of hardware components. We also plan to leverage FPGA to enhance the runtime's performance by delegating performance-critical parts of the runtime to FPGA.

In particular, since the edge nodes must continue to provide inference services without serious QoS degradation even during model updates, a real-time scheduling method that accounts for both model update tasks and inference needs to be implemented at the runtime software level. This will involve the introduction and implementation of isolation scheduling techniques [40] on multi-core or heterogeneous systems to ensure seamless service delivery.

For validation, we will perform integrated co-verification of hardware and software stack [41] to ensure the proposed approach achieves cost-effective, power-efficient, and workload-specific platform design for both FL and HE, using the practical FL applications using sensitive, large-scale data.

V. Concluding Remarks

In this paper, we explore research challenges and ideas in the platform design for HE-enabled FL. The security requirements imposed by the target FL use case and model update frequency should be taken into account in the HW-SW optimization. Moreover, since we use approximate computing components in the HE accelerator design, the approximated components should be carefully optimized along with the accuracy requirements of the chosen HE algorithm. This co-design is illustrated in Fig. 4 in summary.

This approach's application areas include large-scale, high-value systems dealing with sensitive and private data, where we can afford relatively high-performance computing power for both FL and HE. These applications include critical social and industrial infrastructure such as privacy-preserving FL among private hospitals for healthcare, power grids for optimized power usage and generation, transportation and urban mobility systems, and collaborative malware/intrusion detection. In terms of technical impacts, we aim to advance the HPC platform design methodology for software with mixed or even conflicting requirements beyond FL and HE, maximizing the underlying heterogeneous architecture.

Acknowledgment

This work was supported in part by ATTO Research, the NSF I/UCRC for Intelligent, Distributed, Embedded Applications and Systems (IDEAS) and from NSF grant #2231620, and the NSF under award No. CNS-2435327.

References

[1] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[2] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information processing & management*, vol. 59, no. 6, p. 103061, 2022.

[3] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[4] A. Pyrgelis, C. Troncoso, and E. De Cristofaro, "Knock knock, who's there? Membership inference on aggregate location data," in *Network and Distributed System Security (NDSS) Symposium*, 2018.

[5] N. Rieke *et al.*, "The future of digital health with federated learning," *NPJ Digital Medicine*, vol. 3, no. 1, p. 119, 2020.

[6] P. Martins, L. Sousa, and A. Mariano, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–33, 2017.

[7] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning," in *2020 USENIX Annual Technical Conference*, 2020, pp. 493–506.

[8] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology–ASIACRYPT*, 2017, pp. 409–437.

[9] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[10] H. Jun, J. Cho, K. Lee, H.-Y. Son, K. Kim, H. Jin, and K. Kim, "HBM (High Bandwidth Memory) DRAM technology and architecture," in *IEEE International Memory Workshop (IMW)*, 2017, pp. 1–4.

[11] N. Samardzic *et al.*, "F1: A fast and programmable accelerator for fully homomorphic encryption," in *IEEE/ACM International Symposium on Microarchitecture (Micro)*, 2021, pp. 238–252.

[12] J. Zhang, X. Cheng, W. Wang, L. Yang, J. Hu, and K. Chen, "FLASH: Towards a high-performance hardware acceleration architecture for cross-silo federated learning," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1057–1079.

[13] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.

[14] D. Micciancio and Y. Polyakov, "Bootstrapping in FHEW-like cryptosystems," in *Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2021, pp. 17–28.

[15] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Workshop on Distributed Infrastructures for Deep Learning (DIDL)*, 2018, pp. 1–8.

[16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[17] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[18] L. Yuan, L. Sun, P. S. Yu, and Z. Wang, "Decentralized federated learning: A survey and perspective," *arXiv preprint arXiv:2306.01603*, 2023.

[19] J. Le, D. Zhang, X. Lei, L. Jiao, K. Zeng, and X. Liao, "Privacy-preserving federated learning with malicious clients and honest-but-curious servers," *IEEE Transactions on Information Forensics and Security*, 2023.

[20] J.-P. Bossuat, C. Mouchet, J. Troncoso-Pastoriza, and J.-P. Hubaux, "Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys," in *Advances in Cryptology – EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert, Eds. Springer International Publishing, 2021, pp. 587–617.

[21] W. Jung, E. Lee, S. Kim, J. Kim, N. Kim, K. Lee, C. Min, J. H. Cheon, and J. H. Ahn, "Accelerating fully homomorphic encryption through architecture-centric analysis and optimization," *IEEE Access*, vol. 9, pp. 98 772–98 789, 2021.

[22] B. S. Latibari, K. I. Gubbi, H. Homayoun, and A. Sasan, "A survey on fhe acceleration," in *IEEE Dallas Circuits and Systems Conference (DCAS)*, 2023, pp. 1–6.

[23] D. J. Beutel *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[24] H. R. Roth *et al.*, "NVIDIA FLARE: Federated learning from simulation to real-world," *arXiv preprint arXiv:2210.13291*, 2022.

[25] A. Al Badawi *et al.*, "OpenFHE: Open-source fully homomorphic encryption library," in *Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, 2022, pp. 53–63.

[26] C. Gómez, F. Martınez, A. Armejach, M. Moretó, F. Mantovani, and M. Casas, "Design space exploration of next-generation HPC machines," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 54–65.

[27] T. Ye, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Performance modeling and fpga acceleration of homomorphic encrypted convolution," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2021, pp. 115–121.

[28] L. Schor, H. Yang, L. Bacivarov, and L. Thiele, "Expandable process networks to efficiently specify and explore task, data, and pipeline parallelism," in *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, 2013, pp. 1–10.

[29] L. Ward *et al.*, "Colmena: Scalable machine-learning-based steering of ensemble simulations for high performance computing," in *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, 2021, pp. 9–20.

[30] Y. Kim, J. S. Miguel, S. Behroozi, T. Chen, K. Lee, Y. Lee, J. Li, and D. Wu, "Approximate hardware techniques for energy-quality scaling across the system," in *International Conference on Electronics, Information, and Communication (ICEIC)*, 2020, pp. 1–5.

[31] L. Jiang, Q. Lou, and N. Joshi, "MATCHA: a fast and energy-efficient accelerator for fully homomorphic encryption over the torus," in *ACM/IEEE Design Automation Conference (DAC)*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 235–240.

[32] K. Pfeiffer, K. Balaskas, K. Siozios, and J. Henkel, "Energy-aware heterogeneous federated learning via approximate systolic dnn accelerators," 2024.

[33] D. Wu, J. Li, R. Yin, H. Hsiao, Y. Kim, and J. S. Miguel, "UGEMM: Unary computing architecture for gemm applications," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2020, pp. 377–390.

[34] J. Melchert, S. Behroozi, J. Li, and Y. Kim, "SAADI-EC: A quality-configurable approximate divider for energy efficiency," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2680–2692, 2019.

[35] D. Wu, T. Chen, C. Chen, O. Ahia, J. S. Miguel, M. Lipasti, and Y. Kim, "SECO: A scalable accuracy approximate exponential function via cross-layer optimization," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[36] T. Kemp, Y. Yao, and Y. Kim, "MIPAC: Dynamic input-aware accuracy control for dynamic auto-tuning of iterative approximate computing," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, p. 248–253.

[37] D. Boehme, T. Gamblin, D. Beckingsale, P.-T. Bremer, A. Gimenez, M. LeGendre, O. Pearce, and M. Schulz, "Caliper: performance introspection for HPC software stacks," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2016, pp. 550–560.

[38] Y. Fan, Z. Lan, P. Rich, W. E. Allcock, M. E. Papka, B. Austin, and D. Paul, "Scheduling beyond CPUs for HPC," in *International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 2019, pp. 97–108.

[39] J. S. PARKERe and L. TANG, "On the role of co-design in high performance computing," *Transition of HPC Towards Exascale Computing*, vol. 24, p. 141, 2013.

[40] G. Giannopoulou, P. Huang, R. Ahmed, D. B. Bartolini, and L. Thiele, "Isolation scheduling on multicores: model and scheduling approaches," *Real-time systems*, vol. 53, pp. 614–667, 2017.

[41] P. Herber, "The rescue approach-towards compositional hardware/software co-verification," in *IEEE HPCC, CSS, ICESS*, 2014, pp. 721–724.