Research statement

Beomyeol Jeon (bj2@illinois.edu)

Machine learning (ML) has shown its capability and has become essential in many areas, such as e-commerce, healthcare, finance, and agriculture. Despite substantial growth in ML, the real impact of ML requires building reliable, efficient, and scalable systems that support ML operations. ML systems, however, have been facing various *constraints*. For example, the fast growth of ML capabilities and popularity has pushed the *limits* of current computation environments, such as limited device memory and constrained cluster resources. Also, ML applications confront other types of constraints beyond computation environments, e.g., isolated data silos due to privacy concerns. More constraints will show up in the future. To realize the promises of ML, e.g., technology advances via foundation models, ML-driven automation, and AI-driven medical advancements, addressing such constraint challenges in ML systems is essential.

My research goal is to design ML systems that can run successfully and efficiently in the presence of constraints. Achieving this goal helps broaden domains where ML technologies demonstrate their effectiveness and contribute to breakthroughs in ML, leading to paradigm shifts. During my Ph.D. study, I have started to work on a few steps to accomplish this goal by proposing ML system designs with the following four constraints: (i) I built a fast operator placement system for ML graphs on *memory-constrained* devices via an algorithmic approach, (ii) I built an intelligent SLO-aware autoscaling framework for time-varying ML inference jobs on a *constrained on-premises* cluster via sloppifying utility functions and workload prediction, (iii) I proposed a decentralized aggregation protocol for federated learning with *privacy constraints* via controlling communication patterns, and (iv) I have been working on an efficient automatic system configuration optimization framework for Graph Neural Network training on *constrained* serverless computing environments via a gray-box optimization algorithm and system optimization techniques.

In my research, I had collaboration opportunities beyond my research group, including researchers at IBM Research, researchers at Nokia Bell Labs, and Ph.D. students at other universities. These collaborations made me understand real-world problems better and leverage domain expertise from different areas, leading to impactful research contributions.

1 Past Work and Ongoing Work

Device Memory Constraints for Large ML Graphs [2, 3]. The increasing size of Machine Learning (ML) models and scale of training datasets is quickly outpacing available GPU memory, e.g., GPT-3, Megatron-LM, etc. At the same time, ML training is gravitating towards being run among small collections of *memory-constrained* devices, e.g., edge devices, mobile devices. etc. These two trends cause scenarios wherein a single device memory is insufficient to run ML applications. This problem is traditionally solved by adopting model parallelism, wherein the ML model graph is split across multiple devices. Today, the most popular way to accomplish model parallelism is to use learning-based approaches (e.g., Reinforcement Learning) to generate the device placements. While learning-based approaches minimize training step times, these approaches require a very long time (e.g., a few hours) and substantial compute resources (e.g., tens of GPU hours) to generate a placement plan. Also, learning-based placement algorithms are not generalizable, where placement quality is sensitive to changes of target devices and models.

I have developed Baechi [2, 3], a fast device placement system for ML graphs by adopting a traditional algorithmic approach. We adapt classical literature from parallel job scheduling to propose two *memory-constrained* algorithms (m-SCT and m-ETF) and prove their optimality under certain conditions. There are multiple challenges in applying our algorithms to popular ML frameworks (TensorFlow and PyTorch), e.g., excessive communication overhead, a massive number of operators to place, etc. To address these challenges, we propose multiple optimization techniques, such as co-placement, operator fusion, and co-

adjust placements. In evaluation, BAECHI generates placement plans in time 654×–206K× faster than today's learning-based approaches, and yet the step times of the placed models are only up to 6.2% higher than expert-based placements. BAECHI is open-sourced and has been used for extensive evaluation in other work [1].

Resource Constraints in On-premises ML Inference Clusters [4]. Enterprises are increasingly deploying on-premises clusters wherein ML inference engines run directly over a container orchestration framework, e.g., Kubernetes. To reduce the capital and operating expenses, multiple users share resources of a common cluster. However, the time-varying workload (query rate) of any given ML inference application has led to overprovisioning of these clusters, resulting in underutilization of *limited resources* and Service Level Objective (SLO) violations of ML inference tasks. Existing techniques attempt to address this problem by changing resource applications of applications. Yet, all these systems either: (i) tackle only a single job at a time or (ii) do multi-tenancy but need the availability of unlimited resources as in a cloud.

I have built DEUNGDAE [4], the first autoscaling scheduler intended for fixed-size on-premises ML inference clusters wherein the total resources are limited, and where jobs (models) that have time-varying resource demands compete for resources given their SLOs. DEUNGDAE auto-distills each job's latency SLO into utility functions, "sloppifies" these utility functions to make them amenable to mathematical optimization, automatically predicts workload via probabilistic prediction, and dynamically makes implicit cross-job resource allocations, in order to satisfy cluster-wide objectives, e.g., total utility, fairness, and other hybrid variants. Trace-driven cluster deployments show that DEUNGDAE achieves 1.7×–22× lower SLO violations than state-of-the-art systems.

Data Privacy Constraints in Federated Learning [5]. The powerful ML capabilities come from the availability of training data in the big data era, but not all data can be utilized for training ML models. For example, privacy-sensitive data such as medical records and financial information, are stored in *isolated* data silos. Federated Learning (FL) is a promising method to enable ML training over isolated data silos. In FL, each participant trains the same shared ML model by computing gradients within its data silo and sending the gradients to a central server. The central server aggregates the gradients, updates the model, and distributes it to participants. Although there is no raw data transfer outside the data silo, recent studies show that an adversary can infer privacy-sensitive information from the leaked model parameters. Existing techniques (e.g., Secure Multiparty Computation (SMC), Differential Privacy (DP), and combinations of both) require a trusted third party for secret key generation, demand significantly more (e.g., 5–7 orders of magnitude) computation overhead than plain computation, and sacrifice trained model quality due to introduced noise.

I have developed SECUREDD-FL [5], a new decentralized aggregation protocol for federated learning based on the Alternating Direction Method of Multiplier (ADMM) algorithm. Unlike prior work that provides privacy guarantee via DP or SMC, our algorithm takes a different approach whereby we control a communication pattern among participants in ADMM aggregation to prevent them from inferring other participants' privacy-sensitive data. We also have proposed an efficient algorithm to generate communication groups inspired by combinatorial design theory. Our algorithm is proven to guarantee privacy against *honest-but-curious* adversaries. Experimental results with the benchmark datasets show that SECUREDD-FL efficiently trains ML models with a privacy guarantee while maintaining the trained model quality comparable to the centralized FL algorithm.

Serverless Computing Constraints in Automating GNN Training System Configurations [6]. Graph neural networks (GNN) is an ML method that processes graph-structured data and has shown its capabilities in many domains, including social media, e-commerce, biology, etc. A challenge in training GNNs is finding optimal system configurations to minimize training time and/or monetary cost, due to various input graph characteristics (e.g., the numbers of nodes and edges, sparsity). Existing GNN training systems leave configurations as users' responsibilities, which makes users pick suboptimal configurations or manually find optimal ones via extensive measurements. I observe that serverless computing (e.g., AWS Lambda, Azure Functions, Google Cloud Functions, IBM Cloud Functions) is a good fit for automating system configurations for GNN training. Serverless frameworks provide low provisioning time (a few milliseconds) and support

massive parallelism, which enables dynamic optimal configuration search for different scale graphs. Its pay-as-you-go billing model allows cost-efficient training. Unfortunately, the serverless computing paradigm has *constraints* in computation environments, such as no direct communication among compute instances, limited lifetime, ephemeral computation resources, limited network bandwidth, etc. Prior works that tune serverless applications require costly extensive measurements and assume different execution models, which makes them inapplicable to GNN applications.

I am working on a GNN training system that dynamically and automatically tunes system configurations to minimize training times and/or monetary costs on AWS Lambda. Our framework uses (i) an offline optimization based on simulation by capturing GNN training application behaviors and efficiently exploring different system configurations, and (ii) an online optimization based on a gray-box optimization algorithm that finds optimal configurations fast on AWS Lambda environment compared to black-box optimization methods. Preliminary results show that our framework finds optimal system configurations faster than baselines.

2 Future Work

My goal is to make systems execute successfully and efficiently even on environments with various *constraints*. While I have shown how to realize that during my PhD study, I believe I have only started to achieve my research goal. Below, I outline future research directions that I am excited to work on.

Heterogeneous Hardware Constraints in Distributed ML Applications. The success of AI/ML technologies has brought new types of hardware for AI/ML, such as Tensor cores in GPUs, TPUs, and FPGAs. The adoption of these new devices has made on-premises clusters more heterogeneous. The limited resource contention caused by increasing ML uses makes heterogeneous hardware uses inevitable. Prior works assume homogeneous environments or even heterogeneity-aware existing works assume that a model can fit in a single device. The foundation models (e.g., GPT-4, LLaMA) require memory beyond single device memory, which makes existing approaches inapplicable. Simply extending homogeneous solutions is challenging because of different computation powers and connectivity among devices.

Distributed ML training and inference on heterogeneous hardware opens many research questions. How to split and place partial ML graphs over heterogeneous devices to minimize training times and inference latencies? How to perform device placement fast and efficiently? How to improve heterogeneous resource utilization while maximizing ML training and inference system performance? I believe addressing this problem also contributes to sustainable computing by efficiently utilizing old devices and extending hardware lifespans. By leveraging my experience and lessons learned from Baechi and Deungdae, I aim to explore new ways to utilize heterogeneous hardware for ML training and inference systems.

Breaking Isolation Constraints among Distributed ML Applications. More ML applications are deployed in a distributed manner to accelerate training times and overcome device memory constraints. Distributed ML applications commonly show inefficient resource utilization because of dependency in a computation graph and synchronization barriers. Prior work has proposed asynchronous parallel training, which allows ML training to use stale model parameters to increase resource utilization. However, synchronous training is often preferred in the ML community because asynchronous training shows degraded convergence rates and non-deterministic results.

An idea that I would like to explore is a new way to improve resource utilization by breaking *isolation* among ML applications. Can we utilize idle resources caused by the operator dependency and the synchronization barriers with *other* ML applications? For example, inference tasks for smaller models can run efficiently by using idle resources in distributed training. This approach brings up many research questions. For instance, how to pick the right-sized tasks and schedule them on idle devices? How to estimate idle times and execution times of tasks quickly but accurately? Also, allocating multiple resources on a device can be limited due to memory. How to address this memory limit intelligently? My experience from previous works (e.g., estimating ML application execution times) would be solid building blocks to tackle this problem.

References

[1] Ubaid Ullah Hafeez, Xiao Sun, Anshul Gandhi, and Zhenhua Liu. Towards optimal placement and scheduling of DNN operations with Pesto. In *Proceedings of the 22nd International Middleware Conference*, Middleware '21, page 39–51, Québec city, Canada, 2021. Association for Computing Machinery.

[2] **Beomyeol Jeon**, Linda Cai, Chirag Shetty, Pallavi Srivastava, Jintao Jiang, Xiaolan Ke, Yitao Meng, Cong Xie, and Indranil Gupta. Baechi: Fast device placement of machine learning graphs. *arXiv preprint arXiv:2301.08695*, 2023.

[3] **Beomyeol Jeon**, Linda Cai, Pallavi Srivastava, Jintao Jiang, Xiaolan Ke, Yitao Meng, Cong Xie, and Indranil Gupta. Baechi: Fast device placement of machine learning graphs. In *Proceedings of the 11th ACM Symposium on Cloud Computing*, SoCC '20, page 416–430, Virtual Event, USA, 2020. Association for Computing Machinery.

[4] **Beomyeol Jeon**, Diana Arroyo Chen Wang, Alaa Youssef, and Indranil Gupta. SLO-aware ML inference autoscaler for fixed-size on-premises clusters. *Under Review at a System Conference*, 2024.

[5] **Beomyeol Jeon***, S M Ferdous*, Muntasir Raihan Rahman, and Anwar Walid. Privacy-preserving decentralized aggregation for federated learning. In *Proceedings of the 1st International Workshop on Distributed Machine Learning and Fog Network (co-located with INFOCOM '21)*, FOGML '21, pages 1–6. IEEE, 2021.

[6] **Beomyeol Jeon**, Yongjoo Park, and Indranil Gupta. Automating resource allocation for graph neural training on serverless frameworks. *Currently Under Preparation*, 2024.