Scheduling Inference Workloads on Distributed Edge Clusters with Reinforcement Learning

***Abstract*—Edge networks are crucial for real-time applications, as they provide close proximity to data sources for processing inference tasks. However, the constrained nature of edge networks presents challenges in managing inference workloads. This paper focuses on scheduling inference queries on DNN models in edge networks at short timescales, highlighting the need for a dynamic scheduling policy. ASET, a reinforcement learning-based scheduling algorithm, provides the best performance compared to static policies.**

I. Introduction

The popularity of Deep Neural Networks (DNNs) has increased in recent years, particularly in applications like Augmented/Virtual Reality (AR/VR), cognitive assistance, and video surveillance. DNN model training is typically done offline in centralized data-centers or distributed via federated learning [1]. However, DNN inference tasks are typically performed online with constraints in terms of accuracy, throughput, and latency, which may differ across applications. Providing an inference service [2]–[8] requires addressing several challenges, such as selecting the appropriate model variant, processing unit, and nodes and resources. This requires management at different timescales, with schedulers selecting computing instances for new requests and orchestrators optimizing model placement across nodes. Edge computing is crucial for DNN-based applications with stringent delay or bandwidth requirements, but realizing DNN inference at the edge poses additional challenges due to complex networks and limited resources. Recent work [9], [10] combines edge computing and deep learning, but none analyzes inference workload optimization considering different application constraints in realistic edge network settings.

This paper discusses the scheduling of DNN inference requests, considering accuracy, throughput, and latency constraints in realistic edge deployment settings. It presents several static scheduling policies and proposes ASET, an adaptive scheduling algorithm based on Reinforcement Learning. ASET improves performance when resources are distributed across the edge network, increasing the percentage of successfully handled queries. The study demonstrates that different applications may benefit differently from each scheduling strategy.

II. Related Work

Recent studies [2], [3], [5]–[8] have investigated the provisioning of on-demand inference services, focusing on cloud inference workload, edge inference workload, and tasks scheduling. Most existing solutions address the common scenario of scheduling queries over data center resources, but they only consider network latency and resource constrained clusters. Edge inference workload combines deep neural networks with edge computing to overcome scalability and latency limitations. Some authors propose an approach to schedule tasks across multiple edge servers, minimizing end-to-end latency. However, this approach is not suitable for interactive or critical scenarios like virtual reality and autonomous driving. Tasks scheduling [11]–[21] has emerged due to the increasing number of real-time applications requiring powerful resources and fast processing time. Many works propose scheduling mechanisms for orchestrating computational resources, prioritizing tasks, and managing dependencies to meet the performance requirements of edge environments. Deep Reinforcement Learning has emerged as a powerful approach for real-time decision-making, allowing adaptive and efficient optimization even for online task scheduling and resource allocation. Recent solutions, such as TapFinger, train a multi-agent reinforcement learning algorithm in combination with heterogeneous attention network graphs to support each agent's optimal decision.

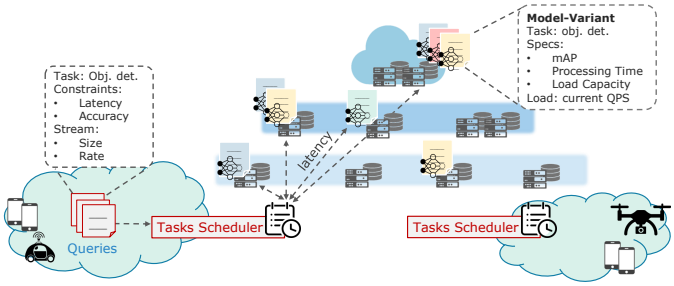
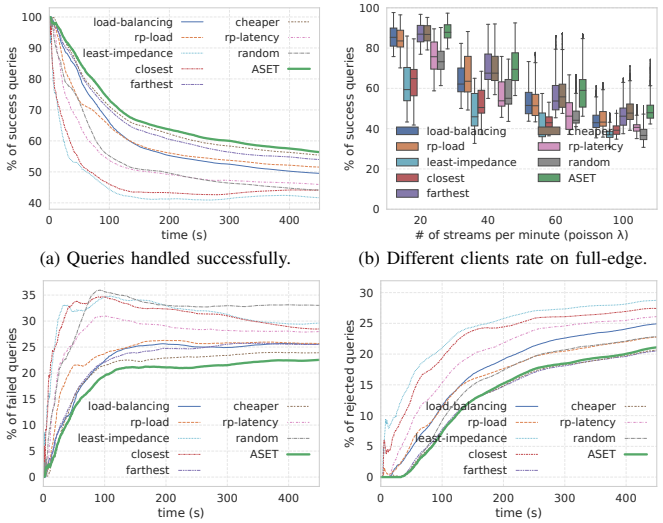


Fig. 1: The scheduler dispatches streams of queries on available model variants based on their constraints and geographical position of clusters.

III. ASET Scheduling Algorithm

The adaptive scheduling approach aims to learn the optimal policy based on current system conditions, such as applications, network topology, and stream arrivals as shown in Figure 1. This is formulated as a Reinforcement Learning problem, where an intelligent agent tries to learn the optimal policy selection strategy according to the observed state of the environment. An RL policy estimates a probability distribution of each possible action that cumulatively maximizes a reward, typically maximizing the fraction of queries served successfully.



(a) Queries handled successfully. (b) Different clients rate on full-edge.

(c) Queries delivered with QoS violations. (d) Queries rejected for lack of resources.

Fig. 2: Performance of ASET compared with static policies for the full-edge topology. (a) (c) and (d) show averages of multiple runs with $\lambda = 60$.

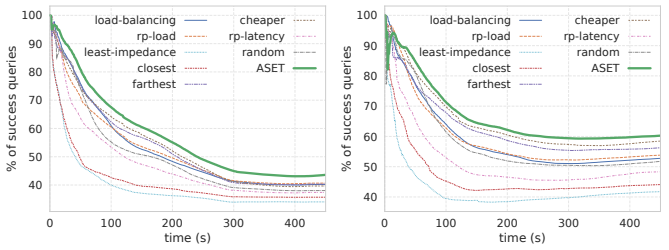
The goal of the proposed adaptive scheduling is to learn an optimal sequence of static network scheduling policies that maximizes the percentage of successfully dispatched streams. The agent collects various observations from the edge-cloud infrastructure, building up the current state of the environment. The agent evaluates a discrete set of actions and chooses an action.

The agent infers the optimal policy sequence based on the system conditions, seeking an optimal binding between workloads and model variants that maximizes the percentage of success queries. The policy, selected by the agent at time t, is used to dispatch all incoming streams during the subsequent time window. The resulting overall scheduling policy dynamically maps a stream i to a model variant v and its deployment on cluster n. The policy learned by the ASET agent leads to a particular static policy sequence, which corresponds to any function employed to estimate the optimal sequence of actions that the agent should perform at each time window.

IV. Performance Evaluation

We evaluate ASET using a prototype implementation of an edge inference system that will be released upon acceptance of the paper.The prototype is used for small-scale experiments to profile representative models and their variants, followed by large-scale experiments on a simulator to compare ASET's performance to static scheduling policies.

Edge deployment. The results so far suggest that a good distribution of computing resources is a key factor to improve against static scheduling policies. As shown in Figure 2, the benefits of using a dynamic scheduling approach become more concrete in a full-edge topology, where resources are better distributed on multiple smaller clusters in different locations. In fact, Figure 2a shows that the dynamic approach of ASET is able to achieve a constant improvement over any static policy, with a higher success ratio over time. In particular, Figures 2cd show that, while maintaining the same rejection rate as the best



(a) Burst variation of λ from 20 to 100. (b) Steady variation of λ between 60 and 20.

Fig. 3: Performance of ASET varying the requests rate over time with two different load variation patterns (full-edge topology).

static-policy, ASET effectively reduces the number of queries that are handled violating one or more QoS requirements. Moreover, Figure 2b shows that an ASET agent trained only for $\lambda = 60$ can also generalize on different requests rate, even supporting a load of more than 1600 queries per second ($\lambda = 100$) on a single antenna.

Dynamic input rate. We have performed some additional experiments to evaluate how the system behaves in dynamic situations where the requests rate varies over time. For this purpose, we have set up some dynamic runs where the lambda value changes every 150 seconds: a first pattern simulates a particularly fast variation with values of 20, 60, and 100 clients per minute (Figure 3a); a different pattern simulates a more steady scenario where the requests rate first moves from 60 to 40 clients per minute, then drops to 20, and finally slowly goes back to 60 (Figure 3b). Similar to previous plots, the outcomes for this set of experiments are shown averaging values over time for multiple runs (Figure 3). Results in both figures show that having a dynamic requests arrival even introduces a bigger margin for improvement that ASET effectively exploits reaching the highest percentage of queries handled successfully. This appears particularly evident in the case where the variation between client arrivals is faster and bigger (Figure 3a). Static policies perform well in stable system loads, but struggle in dynamic scenarios. Adaptive algorithms like ASET are more suitable for these situations, as they learn optimal system optimization under different conditions. ASET training is effective under previously unknown dynamic conditions.

V. Conclusions

The paper introduces ASET, an adaptive algorithm based on Reinforcement Learning, for scheduling inference workloads at the network edge. It solves the problem of exploiting scattered clusters of resources to serve inference queries from multiple edge applications. ASET optimizes the binding between inference stream requests and available DNN models across the network, maximizing throughput and satisfying inference accuracy and end-to-end delay requirements. The approach was evaluated over a large ISP network topology and heterogeneous edge applications, showing that ASET improves performance compared to static policies.

REFERENCES

[1] J. Konecny, H. B. McMahan, F. Yu, P. Richtarik, A. Theertha Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Neural Information Processing Systems (NeuRIPS)*, 2016.

[2] R. S. Kannan, L. Subramanian, A. Raju, J. Ahn, J. Mars, and L. Tang, "Grandslam: Guaranteeing slas for jobs in microservices execution frameworks," in *Fourteenth EuroSys Conference*, 2019.

[3] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *ACM Special Interest Group on Data Communication*, 2019.

[4] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017.

[5] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "Infaas: Managed & model-less inference serving," *arXiv:1905.13348*, 2019.

[6] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke, "Tensorflow-serving: Flexible, high-performance ml serving," *arXiv:1712.06139*, 2017.

[7] D. Chappell, "Introducing azure machine learning," *A guide for technical professionals, sponsored by microsoft corporation*, 2015.

[8] "Ai platform of google cloud," https://cloud.google.com/ai-platform.

[9] J. Chen and X. Ran, "Deep learning with edge computing: A review," *IEEE*, 2019.

[10] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *IEEE Symposium on Edge Computing*, 2018.

[11] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Resource provisioning in fog computing: From theory to practice," *Sensors*, vol. 19, no. 10, p. 2238, 2019.

[12] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[13] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021.

[14] Y. Zhao, Y. Liu, Y. Peng, Y. Zhu, X. Liu, and X. Jin, "Multi-resource interleaving for deep learning training," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 428–440.

[15] G. Bartolomeo, M. Yosofie, S. Bäurle, O. Haluszczynski, N. Mohan, and J. Ott, "Oakestra: A lightweight hierarchical orchestration framework for edge computing," in *USENIX Annual Technical Conference (USENIX ATC 23)*, 2023, pp. 215–231.

[16] M. Cheong, H. Lee, I. Yeom, and H. Woo, "Scarl: Attentive reinforcement learning-based scheduling in a multi-resource heterogeneous cluster," *IEEE Access*, vol. 7, pp. 153 432–153 444, 2019.

[17] H. Lee, J. Lee, I. Yeom, and H. Woo, "Panda: Reinforcement learning-based priority assignment for multi-processor real-time scheduling," *IEEE Access*, vol. 8, pp. 185 570–185 583, 2020.

[18] Y. Li, X. Zhang, T. Zeng, J. Duan, C. Wu, D. Wu, and X. Chen, "Task placement and resource allocation for edge machine learning: a gnn-based multi-agent reinforcement learning paradigm," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[19] Y. Bao, Y. Peng, and C. Wu, "Deep learning-based job placement in distributed machine learning clusters with heterogeneous workloads," *IEEE/ACM Transactions on Networking*, vol. 31, no. 2, pp. 634–647, 2022.

[20] Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Meng, and W. Lin, "DL2: A Deep Learning-Driven Scheduler for Deep Learning Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 1947–1960, 2021.

[21] B. Paeng, I.-B. Park, and J. Park, "Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups," *IEEE Access*, vol. 9, pp. 101 390–101 401, 2021.