CKN: An Edge AI Distributed Framework

Sachith Withana and Beth Plale

Department of Intelligent Systems Engineering
Indiana University Bloomington, IN, USA
{*swithana, plale*}*@indiana.edu*

***Abstract*—Edge AI is the use of artificial intelligence at the Edge which allows for real-time processing and low latency in making AI decisions. In a resource-limited environment of edge computing, it is crucial to efficiently manage the deployment and use of inference models while at the same time optimizing user experience. Our framework supports AI at the Edge through a standalone monitoring framework that builds and maintains a context for an edge application through monitoring data and control streams between a device and edge server. Its objective is to maximize user expressed Quality of Experience. In this brief abstract we introduce the CKN framework and show the monitoring framework's effectiveness in response to changes in edge needs.**

***Index Terms*—Edge-cloud continuum, provenance, streaming, Edge AI, Knowledge Graphs**

I. Introduction

As Artificial Intelligence (AI) continues to advance, its benefit at the Edge is becoming more evident. EdgeAI combines the benefits of Edge Computing, such as faster processing times and better privacy, with the ability to make decisions closer to the data source. However, Edge resources are often limited (reliability, power, etc.), which means that AI models must be deployed efficiently based on specific requirements such as accuracy, latency, and privacy. It is crucial for Edge Systems to optimize their model deployment to provide a better Quality of Experience (QoE) for connected devices.

To address this challenge, we introduce a framework that enables efficient deployment of AI models at the Edge, even with highly dynamic device requirements. Our framework, described more fully in [10], builds and maintains a context through time of what could be called the ecosystem of the device and edge server. That is, through monitoring both data and control streams at both device and edge server, and processing that information into a provenance and time-based picture of the ecosystem, the framework is able to anticipate changes in needs and preemptively react. The data model of the ecosystem is a graph, making it amenable to analysis through deep learning.

In this abstract we give an overview of the design of the framework and briefly describe the experiment that we carry out to evaluate the framework's potential. The experiment uses an image classification task, and evalulates the framework's ability to swap out an inference model for a "better" one adaptively as need changes. It works as follows: a device sends an image to the edge server. This series of images from the device make up the device's data stream. Along with the image, the device requests a particular latency (time to complete the classification) and model accuracy. This series of requests forms the device's control stream. We further monitor the response from the Edge Server back to the device. The response provides the achieved accuracy and latency.

The request for a certain model latency and accuracy are expressed as Quality of Experience (QoE); QoE we give as having two attributes: Quality of Latency (QoL) and Quality of Accuracy (QoA). Both are an integer value [0..1] where a QoA or QoL of 1 is a request for the highest possible qualities of each and 0 is the lowest. QoE is then the weighted values of QoA and QoL. In our experiment, both QoA and QoL are weighted equally.

Using the QoE stream to represent a set of changing needs, and monitoring the response from the Edge Server back to the device of satisfied QoE, our framework can detect when an inference model is no longer meeting the needs of the device. When this occurs, we enact a change at the Edge Server to place an inference model that better meets the needs of the device.

We call the framework CKN for Cyberinfrastructure Knowledge Network. Full details of CKN and the experiment can be found in [10].

II. Design

The overall environment for the framework is of devices communicating with some number of devices communicating with an edge server.

A CKN daemon exists in each Edge Server from where it captures the request (image classification) and control stream (Quality of Experience). Accompanying the QoE is identifying information about the device such as the device ID. The daemon takes the information along with identifying information about the edge server (e.g., server ID) and streams this information through a distributed message broker where these events are aggregated and stored in the Knowledge Graph of the system.

The device-edge context (historical provenance data) is analyzed using a time-series Deep Learning Model to predict a better model for placement based on incoming resource constraints. The selected model can then be triggered if it is already resident at the edge server, or in the future can be downloaded from the ICICLE Model Commons on demand as network latency permits, and the ICICLE Model Commons [7] comes into being.

Numerous questions are raised by this framework having to do with how frequently a model is replaced (one does not want

to put the Edge Server into a state of constant churn), where is the optimal location of CKN knowledge graph given its size and analysis demands. In the proof of concept evaluation that we carried out, we are interested in initial feasibility alone.

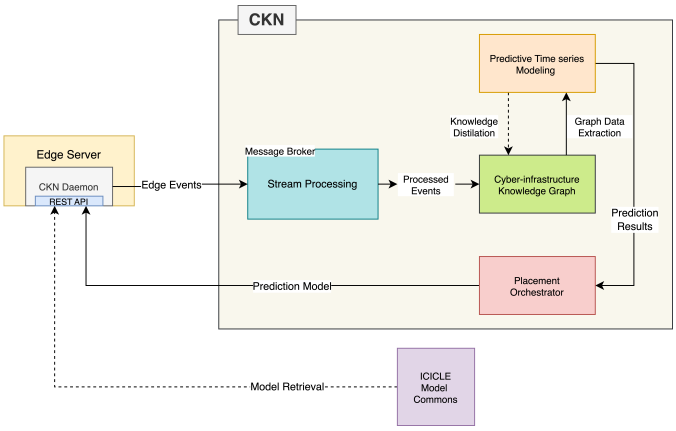

Figure 1. CKN architecture for the Edge-cloud continuum

The five major components or aspects of CKN are described here and component relationships captured in Figure 1.

III. Evaluation

In order to evaluate the predictive capability of the KG, we create a synthetic and deterministic workload (available at [11]). The workload encodes abrupt changes in QoE needs to measure how long it takes for our framework to react. The workload consists of 1500 time windows, each window having a duration of 60 seconds. The number of requests in each window ranges from 100 to 1000, giving us a wide range of scenarios to examine. The images from the ImageNet [1] is used for the classification task.

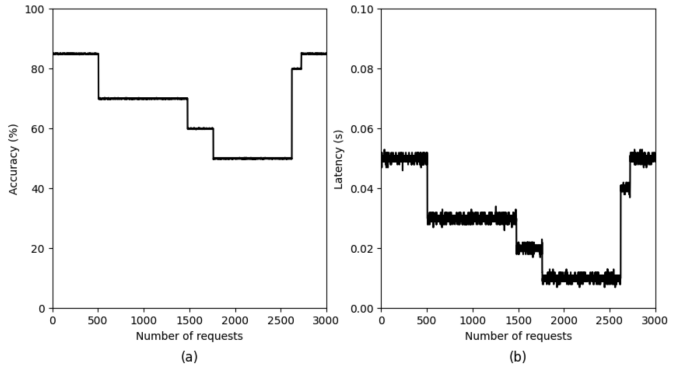

Figure 2. Workload behavior for first 3000 data points showing (a) requested accuracy and (b) latency

The behavior of the workload is demonstrated in Figure 2 for the first 3000 requests. Changes in requested accuracy fall on time window boundaries. The first time window is 500 requests long and the requested accuracy is 82%. At the boundary between the first and second time windows, the requested accuracy drops to 70%. The time window for the second window is 1000 requests long. Similarly latency is .5 +/- (.03) at the first time window and drops to .3 for the second time window. This change might occur when faster processing is needed. Suppose at event 500, the device needs faster processing and is willing to accept lower accuracy for it.

For the classification models, we leveraged publicly available models trained on the ImageNet dataset as shown in Table I.

Table I
Inference Models

Model Name	Acc@1		Mean Latency (s)	Mean Accuracy
ShuffleNet [8]	72.996	3.5M	0.0133	50.18%
DenseNet [5]	76.896	20.0M	0.1388	40.02%
GoogleNet [9]	69.778	6.6M	0.0599	32.81%
MobileNet [4]	67.668	2.5M	0.0120	47.55%
ResNet [3]	82.284	60.2M	0.2068	49.54%
ResNeXt [12]	77.618	25.0M	0.0953	53.82%
SqueezeNet [6]	58.178	1.2M	0.0206	46.61%

The mean accuracy and latency values were measured for the system for precise model placement. Jetstream2 [2] community cloud was leveraged to carry out the experiments evaluating the framework. We compared our algorithm (called ”predictive” in table) to three other versions:

- Predictive Algorithm leverages the CKN time-series deep learning model to select the model for the placement,
- Optimal Algorithm assumes perfect foresight on the part of the selection algorithm. That is, the system knows the future constraint values and decides the models to be placed,
- Random Placement places the models randomly,
- No-choice Algorithm uses the best choice from Table I throughout the experiment.

All the evaluations start with a default model in place for the first time-step.

An illustrative result of the experiment (taken from [10] and shown in Figure 3, shows the results for Quality of Experience values for the system for each of the four algorithms. One can read the figure as follows: the optimal algorithm has perfect foresight so it represents the right inference model being where it needs to be when it needs to be there. As expected, random choice of inference model for placement performs the worst. The predictive algorithm closely follows the optimal algorithm for QoE. The no-choice algorithm (hand picked at outset because it is thought to perform the best under all circumstances) lags in several circumstances.

IV. Discussion and Future Work

There are a host of questions to be answered as next steps for this work. Our measured latencies, which were different from the published latencies, meant disappointingly little difference between model latencies. This will result in the favoring of the “no choice” option which is simple and cost effective. Further, we are applying the framework outside

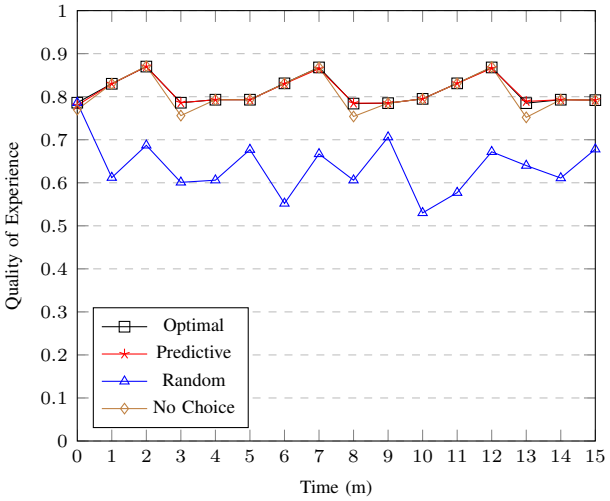

Figure 3. Quality of Experience for different placement algorithms (Jetstream 2)

a synthetic dataset and a cloud environment (the latter where networking latencies are not a factor and device reliability and variability is not accounted for).

We are further exploring extending the notion of QoE beyond simple performance measurements to include attributes having to do with trustworthiness of the inference model. That is, choice of an inference model needs to go beyond simply performance and model accuracy. The CKN framework is envisioned as a standalone tool yet one that is part of the ICICLE integrated infrastructure environment. The strength of the framework is the adaptability that it provides at the edge, thus realizing one of ICICLE's major objectives of "plug and play" infrastructure. CKN derives this strength through an intelligent data model that captures ecosystem context at the edge and thereby is adaptive to user need, including specificity with accountability and trust in mind.

V. Acknowledgements

The inspiration for this framework came as a result of discussions that began at the CENTRA5 workshop in Porto Portugal between the authors and Prof. Hana Khamfroush of the University of Kentucky and Nathaniel Hudson of the University of Chicago. This research is funded in part through the National Science Foundation under award #2112606 ICICLE AI Institute.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[2] David Y. Hancock, Jeremy Fischer, John Michael Lowe, Winona Snapp-Childs, Marlon Pierce, Suresh Marru, J. Eric Coulter, Matthew Vaughn, Brian Beck, Nirav Merchant, Edwin Skidmore, and Gwen Jacobs. Jetstream2: Accelerating cloud computing via jetstream. In *Practice and Experience in Advanced Research Computing*, PEARC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE conf on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[5] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[6] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[7] AI Institute for Intelligent CyberInfrastructure with Computational Learning in the Environment (ICICLE). https://icicle.osu.edu/.

[8] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proc. European conference on computer vision (ECCV)*, pages 116–131, 2018.

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[10] Sachith Withana and Beth Plale. Ckn: An edge ai distributed framework. In *2023 IEEE 19th Int'l Conf on e-Science (e-Science)*, pages 1–10, 2023.

[11] Sachith Withana and Beth Plale. CKN Edge AI Dataset for Image inference at the Edge (CEAD). (1.1) [Data set].Zenodo.https://doi.org/10.5281/zenodo.8023205, June 2023.

[12] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc IEEE conf on computer vision and pattern recognition*, pages 1492–1500, 2017.