Data collection and utilization framework for edge AI applications

Rexha, Hergys; Lafond, Sebastien

Published in:
2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)

DOI:
10.1109/WAIN52551.2021.00023

Published: 01/01/2021

Document Version
Accepted author manuscript

Document License
Publisher rights policy

Link to publication

Please cite the original version:
Rexha, H., & Lafond, S. (2021). Data collection and utilization framework for edge AI applications. In *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)* (pp. 105-108). Article 9474354 IEEE. https://doi.org/10.1109/WAIN52551.2021.00023

Data Collection and Utilization Framework for Edge AI Applications

Hergys Rexha, Sébastien Lafond
Åbo Akademi University
hergys.rexha@abo.fi, sebastien.lafond@abo.fi

***Abstract*—As data being produced by IoT applications continues to explode, there is a growing need to bring computing power closer to the source of the data to meet the response-time, power dissipation and cost goals of performance-critical applications in various domains like Industrial Internet of Things (IIoT), Automated Driving, Medical Imaging or Surveillance among others. This paper proposes a data collection and utilization framework that allows runtime platform and application data to be sent to an edge and cloud system via data collection agents running close to the platform. Agents are connected to a cloud system able to train AI models to improve overall energy efficiency of an AI application executed on a edge platform. In the implementation part we show the benefits of FPGA-based platform for the task of object detection. Furthermore we show that it is feasible to collect relevant data from an FPGA platform, transmit the data to a cloud system for processing and receiving feedback actions to execute an edge AI application energy efficiently. As future work we foresee the possibility to train, deploy and continuously improve a base model able to efficiently adapt the execution of edge applications.**

I. Introduction

Edge computing is a fast-growing technology trend, which involves pushing compute capabilities to the edge. Edge computing can be described as a distributed computing paradigm that brings computation and data storage closer to the location needed to improve response times, save bandwidth, and improve security.

Edge systems are the deterministic embedded communication and real-time control engines that reside at the edge of the network and closest to the physical world of factories and other industrial environments, e.g., motion controllers, protection relays, programmable logic controllers, and similar systems. Clock frequencies in gigahertz, larger memory sizes, higher numbers of input/output ports, and the latest encryption engines might seem to offer solutions for future requirements. However, when dealing with the timescale of industrial equipment, which has critical subsystems that operate on a scale of hundreds of microseconds (or less) and need to operate in factories and remote locations for decades, relying solely on a cutting-edge multicore embedded processor is risky. A much higher degree of freedom in scaling is desperately needed, at for example Industrial edge system, due to the timescales involved. Also there is a need for a more consistent approach that allows determinism, latency, and performance to be easily managed. At the heart of the current industrial revolution is the roll-out of machine learning (ML) algorithms, specifically deep neural networks (DNNs). They achieve impressive results in computer vision and speech recognition, and are increasingly being adopted for other tasks. DNNs are first trained on a labeled dataset, and afterwards can be used for inference on previously unseen data as part of an application. The large compute and storage requirements associated with DNN deployment necessitate acceleration. Furthermore, different constraints might be imposed on accuracy, cost, power, model size, throughput, and latency depending on the use case. Real-time and safety-critical applications such as augmented reality, drone control, and autonomous driving are not suitable for offloading to the cloud due to low-latency requirements and data transmission overhead. In cloud-computing and ML-as-a-Service contexts, data centers face ever-increasing throughput requirements to process astronomical scales of data [13], bringing additional challenges in energy efficiency to minimize operating expenses. While cloud service latency is less critical compared to embedded scenarios, it still translates directly into customer experience for interactive applications. Traditionally, machine learning research was focused on improving the accuracy of the models without particular regard to the cost of inference. This is evident in the older networks like AlexNet and VGG, which are now considered large and with many parameters [2]. However, as machine learning and DNNs move into practical applications, compute and memory requirements become a major concern.

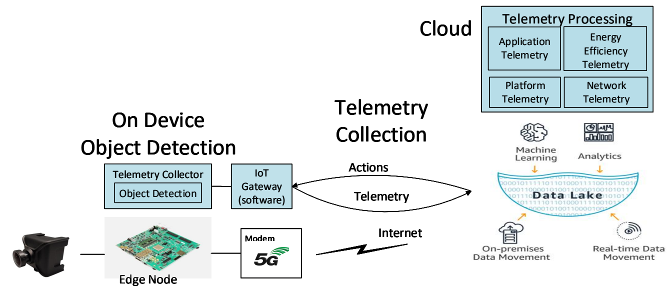

Fig. 1. Basic schematic of the telemetry framework

II. Use case and Architecture Overview

The assumed scenario for this work is the following: an industrial system (it could be for example a patrolling robot or a manufacturing conveyor) is streaming live a video over a 5G network and requests as a service detected objects from the video stream. The object detection service is executed from the Multi-access Edge Computing (MEC) of the used 5G base station.

The main assumptions of the work are the following: (a) an FPGA platform can provide lower latency than more traditionally used GPU platforms for this type of application. This is due to the datapath architecture of the FPGA and DPU, which does not require to first "flood" a large number of Streaming Multiprocessors (SM) as in a GPU; (b) taking advantage of the DPUs, we can reach higher throughputs in terms of number of processed frames per second; (c) the FPGA platform will provide a better energy efficiency solution compared to CPU and GPU based solutions.

The main goals of the work are the following:

- Propose and implement a telemetry collection framework that complements the use case scenario described above.
- Evaluate achievable latency, throughput and energy efficiency of common edge platform alternatives for the selected use case and proposed architecture.
- Show the benefits off customizing computations at the edge with the intelligence from the cloud side created with the collected telemetry data.

A. *Telemetry Framework*

There is a need for big data analytics and machine-learning based AI technologies for the operational automation of factories and other industrial environments. These use cases deploy edge systems for real-time control of the operations. The collection of large amounts of data is required from different system components like applications, edge platform and network. The single-sourced and static data acquisition cannot meet this data requirements. It is therefore desirable to have a framework that integrates multiple telemetry approaches from different components. The telemetry framework brings a solution to this problem. The main focus of this work is to provide an end-to-end description and evaluation of the proposed telemetry architecture which is described in Figure 1. The framework can be divided in two parts: the edge part which is described on the left of the Figure 1, and the cloud part which is on the right side. At the edge side of the schematic we have a highly heterogeneous platform which is equipped, either with GPU or with re-configurable hardware (FPGA). The platform is hosting an intelligent application which uses a convolutional neural network (CNN) for performing real-time video inference, and an agent which is collecting several metrics from the application, platform, and network called the telemetry agent. Metrics of various components of the platform are collected and formatted as a JSON object and sent to the other part of the framework. On the cloud part the data is analyzed and actions are taken as a feedback controlling the behaviour of the intelligence performed on the edge side. A more detailed description of the telemetry framework is available from [6].

III. Edge platform technologies

To support our assumptions on the achievable latency and performance of FPGA platforms for CNN-based edge applications, we evaluate two different platforms for the role of the edge node: a Nvidia Jetson AGX Xavier (as a representative GPU-based platform) and a Xilinx ZCU102 (as a representative FPGA-based platform). The Xavier is an embedded GPU platform which promise to offer high compute density and good energy efficiency for AI related applications. The Xavier is equipped with 512 CUDA cores with Volta architecture GPU running at 1.37GHz and a 16GB LPDDR4X @ 2133MHz memory with a bandwidth of 137 GB/s, and a flash storage eMMC 32GB. The Xilinx Zynq UltraScale+ MPSoC ZCU102 board has a 16nm XCZU9EG FPGA, an on-board 4GB 64bit DDR4 RAM with a peak bandwidth of 136Gb/s.

We aim at testing the AI inference capabilities, and the power dissipation of the two platforms while running neural network algorithms. The experiments are conducted using Yolov3 and SSDResnet50Fpn algorithms for object detection which perform inference on a 420p video file. In Table I we report the measurements done for both platforms for metrics such as end-to-end delay (EE latency) to process a single frame and number of frames per second processed for a single dissipated watt (FPS/Watt) while running two popular object detection algorithms such as Yolo and SSD. The neural network is fed with the same video file and the power is measured on the entire platform. FPGA architecture is able to achieve good latency in time-sensitive jobs due to the circuit-level customizations on its massively parallel computing units. From the results shown on the table there is a clear advantage of the FPGAs platforms versus GPUs to be used especially in streaming applications, this is noticeable in terms of latency and energy-efficiency. The SSD_Resnet_50_FPN is a heavier model compared to Yolo, requiring 178.4 Gops compared to 65.63 Gops of the other side. Based on this evaluation, the ZCU102 FPGA-based board will be used a the edge device in the rest of the paper.

Platform	Algorithm	EE latency (ms)	FPS/Watt
Xavier AGX	Yolov3	120	0,3
	SSD_Resnet50_fpn	250	0,17
ZCU102	Yolov3	29,4	1,48
	SSD_Resnet50_fpn	200	0,37

TABLE I
Inference characteristics of the considered edge platforms

IV. Experimentation Methodology

As described in section II the telemetry framework consist mainly in two parts: the Edge and Cloud side.

A. *Edge Side*

On the edge side we are executing a CNN-based video inference application which is quantized and pruned for running on a FPGA device. We also collect the parameters which will make up our telemetry data through an agent that is running on the device. The agent collects telemetry data from three categories as described below:

1) Application Telemetry: Latency of the application, FPS.

2) Model Efficiency Telemetry: Computational Unit utilization, Memory Throughput, CPU utilization, Memory utilization, AI model efficiency.

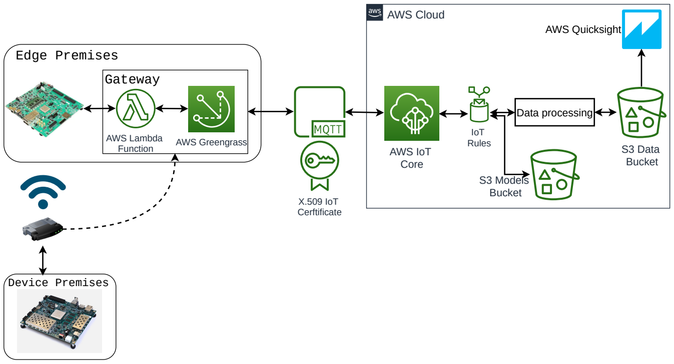

Fig. 2. Proposed Telemetry Infrastructure

3) Energy Efficiency Telemetry: Dissipated power, Temperature of the module, FPS/Watt.

4) Communication Network Telemetry: From the network side the agent collects parameters of the 4G modem used in the experimental framework and these telemetry is sent transparently to the cloud. The parameters collected include: RSSI, RSRQ, RSRP, Temperature, DL/UP.

B. Cloud Side

The second part of the architecture consists of the cloud side which offers services for securely receiving the telemetry data, enriching those data with additional information (e.g. timestamps), analyzing the information contained in the data, providing feedback actions to the edge platform based on some defined triggers and additional services such as further processing, storage, and analytics. The practical implementation used in this paper is based in the Amazon Web Services (AWS) cloud environment.

C. Telemetry Architecture

In Figure 2 we can see the actual components in the deployment of the telemetry architecture for both the edge and cloud sides. In the edge platform we have deployed the AWS IoT Greengrass software which provides the environment for running lambda functions to control the hardware platform and the application running on the platform.

The workflow of the process is as follows: At first on the edge premises the telemetry agent is running and collecting metrics from the application, AI neural network and hardware platform. The collected information is packed into a JSON object and sent to the Greengrass core (GGC) located on the edge side. The GGC is registered with AWS IoT Core on the cloud side and uses the MQTT protocol to forward the JSON objects to the cloud. The IoT Core provides a Device Gateway which manages active device connections and a powerful Message Broker which routes the messages with low latency. Once a message is received we use AWS IoT Rules to send messages to further data processing and aggregation before storing them to the S3 data bucket. Other rules are created to call specific lambda functions on the edge which perform actions like checking the achieved FPS by the object detection application and if the value is above 30 fps, lowering the clock frequency of the platform processing unit in order to save power. Another rule checks for model efficiency, which is the model fps divided by the ratio of peak accelerator rate and model workload, and if the number is below a certain threshold triggers a lambda function on the edge which downloads a new model from a S3 models bucket, located in the Cloud, to the edge premises to perform the inference with the new model. In the telemetry framework in Figure 2 the video stream is transmitted to the edge platform from the device via a 4G or 5G connection. Beside the application, ML model, and edge platform telemetry we also collect network telemetry as explained above, which is sent to the cloud. This data can be exploited for training machine learning models able to predict the connection bandwidth from parameters collected from the router. There are several policies that could decide the location of the inference. By exploiting the telemetry data collected from the router we can predict the bandwidth of the connection and decide whether it is reliable to send the video stream to the edge. There are several research work which show the possibility to predict the current connection bandwidth based on parameters like RSRP, RSRQ, and historic throughput [14, 1, 7]. In the case that the bandwidth is high enough, the stream can be transmitted to the edge premises for faster inference, otherwise the edge will decide to push the inference on the device itself, resulting in slower inference time. The decision on where to actually run the inference in this case will be made on the cloud side based on the received telemetry, and from the model results which predicts the available bandwidth. The offloading decision, from edge to device, could be made by the edge system also, which in case of high levels of utilization can decide to send the intelligent application to the device.

V. Related Work

There is a wide research work regarding the usage of data analytics in making smart and fast decisions especially in wireless networks [4, 8, 9, 10, 3]. Mainly the advances in IoT and hardware/software technology have given the opportunity to collect real-time data from user equipment or core devices which are valuable in making decisions that will impact the performance, adaptability, efficiency of the end-to-end system. This collection of works emphasis more the need for gathering telemetry data from different components of the end-to-end application system. Beside the telemetry data collection there is also to consider the edge component, which in many cases is used to bring resources closer to device side and is a central actor in the real-time applications as in [5, 12, 11]. In this paper we propose a framework that includes different telemetry data, gathered from the edge platform, application, network, and machine learning model with goal of providing feedback to the edge or device premises plus creating a data lake for training machine learning models at the cloud side.

VI. Experimental Results

A. Latency measurements

Devices connect to AWS IoT and other services through AWS IoT Core. Through AWS IoT Core, devices send and receive messages using device endpoints that are specific to the used AWS account. There are two main communication protocols for sending the data to the message broker in the IoT Core service. One is MQTT, which is a lightweight and widely adopted messaging protocol that is designed for constrained devices, and the other is HTTPS over websockets. To evaluate the proposed architecture, we measured the achievable message latency when reaching the IoT Core and measured any possible difference between the communication protocols. All

Protocol	Mean Lat.(ms)	Min(ms)	Max(ms)	Std. dev(ms)
MQTT	516,44	218	1652	169,45
HTTP	565,75	181	6600	415,91

TABLE II
LATENCY MEASUREMENTS OF SENDING DATA TO THE CLOUD

measurements were done with the Edge platform connected to a commercial 4G network, and the IoT core deployed in eu-west-2 region (London).

As shown in Table II, for the case of MQTT the average latency is lower regardless of the fact that with Greengrass there is an additional delay of the core software. Also the spikes in case of HTTP are quite high bringing a real need for local processing on the edge instead of relying only on the cloud.

VII. CONCLUSIONS AND FUTURE WORK

This paper proposes an edge/cloud telemetry collection and utilization framework for applications where reliability, latency, power efficiency and high computational capacity is critical. For instance, vehicle safety as well as vehicular visual and non visual sensing systems could be potential use cases. We evaluate GPU based platform against FPGA platform for the role of edge node in an AI computer vision application and set up our framework with the FPGA platform induced by latency and power efficiency numbers provided. We define the cloud side components of the data lake architecture which will serve later as valuable input for training machine learning networks at the cloud side. At the end we discuss about reaction time of cloud side of the framework and FPGA implementation issues which is good to consider when developing AI-based application on re-configurable platforms. As a future work we foresee the creation of intelligent engines on the cloud side based on the data collected through the telemetry framework.

ACKNOWLEDGMENT

This work has been partly funded by the European Commission through the projects EU-TW 5G-DIVE (Grant Agreement no. 859881).

REFERENCES

[1] N. Bui and J. Widmer. “Data-Driven Evaluation of Anticipatory Networking in LTE Networks”. In: *IEEE Transactions on Mobile Computing* 17.10 (2018), pp. 2252–2265. DOI: 10.1109/TMC.2018.2809750.

[2] Y. Chen et al. “Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.2 (2019), pp. 292–308. DOI: 10.1109/JETCAS.2019.2910232.

[3] F. Hussain et al. “Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges”. In: *IEEE Communications Surveys Tutorials* 22.2 (2020), pp. 1251–1275. DOI: 10.1109/COMST.2020.2964534.

[4] P. Jiang et al. “An Intelligent Information Forwarder for Healthcare Big Data Systems With Distributed Wearable Sensors”. In: *IEEE Systems Journal* 10.3 (2016), pp. 1147–1159. DOI: 10.1109/JSYST.2014.2308324.

[5] Z. Khan and J. J. Lehtomäki. “FPGA-Assisted Real-Time RF Wireless Data Analytics System: Design, Implementation, and Statistical Analyses”. In: *IEEE Access* 8 (2020), pp. 4383–4396. DOI: 10.1109/ACCESS.2019.2962200.

[6] Hergys Rexha and Sebastien Lafond. *Data Collection and Acceleration Infrastructure for FPGA-based Edge AI Applications*. 2021. arXiv: 2103.06518 [cs.LG].

[7] J. Schmid et al. “A Deep Learning Approach for Location Independent Throughput Prediction”. In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. 2019, pp. 1–5. DOI: 10.1109/ICCVE45908.2019.8965216.

[8] J. Wang et al. “Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach”. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017, pp. 1–9. DOI: 10.1109/INFOCOM.2017.8057090.

[9] L. Wang and S. Cheng. “Data-Driven Resource Management for Ultra-Dense Small Cells: An Affinity Propagation Clustering Approach”. In: *IEEE Transactions on Network Science and Engineering* 6.3 (2019), pp. 267–279. DOI: 10.1109/TNSE.2018.2842113.

[10] L. Wang and S. Cheng. “Data-Driven Resource Management for Ultra-Dense Small Cells: An Affinity Propagation Clustering Approach”. In: *IEEE Transactions on Network Science and Engineering* 6.3 (2019), pp. 267–279. DOI: 10.1109/TNSE.2018.2842113.

[11] Shang Wang et al. “Live Video Analytics with FPGA-Based Smart Cameras”. In: *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 9–14. ISBN: 9781450369282. DOI: 10.1145/3349614.3356027. URL: https://doi.org/10.1145/3349614.3356027.

[12] W. Xia et al. “A Deep Learning Framework for Optimization of MISO Downlink Beamforming”. In: *IEEE Transactions on Communications* 68.3 (2020), pp. 1866–1880. DOI: 10.1109/TCOMM.2019.2960361.

[13] Xilinx. *Xilinx Powers Alibaba Cloud FaaS with AI Acceleration Solution for E-Commerce Business*. URL: https://www.xilinx.com/publications/powered-by-xilinx/xilinx-alibaba-case-study.pdf.

[14] C. Yue et al. “LinkForecast: Cellular Link Bandwidth Prediction in LTE Networks”. In: *IEEE Transactions on Mobile Computing* 17.7 (2018), pp. 1582–1594. DOI: 10.1109/TMC.2017.2756937.