Cloud-Edge Microservice Architecture for DNN-based Distributed Multimedia Event Processing

Felipe Arruda Pontes and Edward Curry

Insight SFI Research Centre for Data Analytics, Data Science Institute, National University of Ireland Galway
{felipe.arruda.pontes,edward.curry}@insight-centre.org
https://dsi.nuigalway.ie

Abstract. The rise of Big Data, Internet of Multimedia Things (IoMT), and Deep Neural Network (DNN) enabled the growth of DNN-based Computer Vision solutions to Multimedia Event Processing (MEP) applications. When these are applied to a real-world scenario we notice the importance of having a system with a satisfactory speed that can fit in the limited resources of most IoMT devices. However, most solutions for distributed MEP are dependent on a Cloud architecture, which makes these applications migration to the Edge more challenging. As a response to this, we present a microservice architecture for DNN-based distributed MEP over heterogeneous Cloud-Edge environments. We describe our solution that allows for an easier deployment both on the Edge and on the Cloud. We show that choosing the proper tools for an Edge-Friendly solution can lead to 100 times less resource utilisation. Our preliminary investigation shows promising results, with a reduction in energy consumption by 8% with a minor drawback of 15% in throughput in the Edge and a negligible increase in energy consumption on the Cloud.

Keywords: Cloud-Independent · Edge-Friendly · Distributed Computing · Multimedia Event Processing · Deep Neural Networks.

1 Introduction

Alongside the increase of Big Data and Internet of Multimedia Things (IoMT), we can observe the rise of Multimedia Event Processing (MEP) applications. This is mostly because MEP is useful for handling continuous streams of data that are present in a Big Data scenario, and it provides a framework for the constant multimedia event streams generated from IoMT devices. Another common characteristic of Big Data and IoMT is the fact that they work well with distributed computing architecture, and since both are highly connected to the concepts of Cloud and Edge computing respectively, combining them also presents an interesting scenario where it is common to see this mixed Cloud-Edge environment.

The general decision of having Deep Neural Network (DNN)-based Computer Vision (CV) solutions as part of MEP applications for Big Data with IoMT accompanies the consolidation of using DNN models for most CV problems. On one side, DNN models are known to require many resources, on the other side, many Edge devices are resource-constrained, with restrictions in energy consumption, CPU/GPU, and memory. Thus, Cloud-Edge Heterogeneity becomes another important characteristic for distributed MEP applications because it is essential to take into consideration the different aspects and limitations of both Cloud and Edge devices.

However, migrating the available distributed MEP solutions to the Edge brings many challenges since they are mostly made with a focus on specific Cloud infrastructures (e.g.: Amazon AWS or Microsoft Azure) [10, 2]. As a response to this, we propose a microservices architecture for distributed MEP over heterogeneous Cloud-Edge environments, which is both Edge-Friendly and Cloud-Independent at the same time. Microservices architecture (MSA) is one of the new trends in distributed systems architecture, and have been used by several prominent companies such as Netflix, Amazon, and Uber, in addition, they are accepted as a reliable solution for the overall problems of distributed systems.

In this work, we detail our architecture design and its impact in terms of energy and speed. We describe our tooling decisions and show that choosing the appropriate tools for an Edge-Friendly solution can lead to **100 times less resource utilisation** than a non-optimal tool. Our architecture shows promising preliminary results of **8% of energy reduction** with only a **minor reduction of 15% on the overall speed**.

2 Motivation and Related Works

The basic required components in a MEP application are **sources**, **stream manager**, **stream processor** and **sinks**. An example of the dataflow is depicted in Figure 1. The use of Computer Vision (CV) techniques to help with Occupational Health and Safety (OHS) in construction sites has been recently explored [6]. The use of a mixed Edge-Cloud is important when we consider that construction sites may be located in isolated regions without access to infrastructure. Some of the problems in this area range from accidents prediction and prevention to safety rule violation alerts. In this case, it is possible to identify the lack of safety helmets or hi-viz vest in dangerous locations by using cameras, Object Detection (OD) models to analyse the video stream from the cameras, and generate an alert to the OHS supervisor, as shown in Figure 1.

On the use of bandwidth-efficient MEP for drones, Wang et al. [9] uses similar DNN models to ours. However, they focus on the weight of the devices and the models' speed, without taking into account the energy consumption. The EdgeWise system [5] gives stream processing optimisations for mixed Cloud-Edge environments, but it differs from our work since it does not take into consideration Cloud-Edge environment heterogeneity.

Microservice architecture (MSA) is a system architecture style where an application is decomposed into small and autonomous parts that work together and around business capabilities, with decentralised control of languages and data

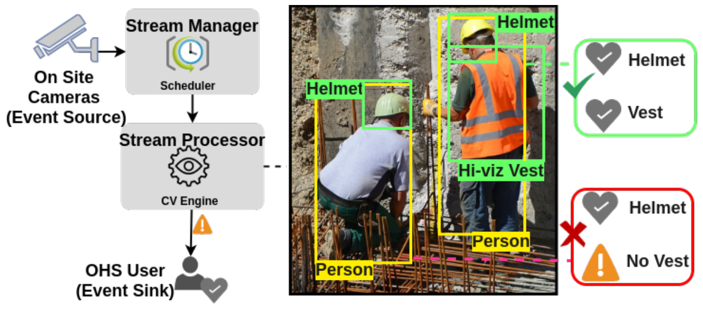

Fig. 1. Multimedia Event Processing on OHS for Construction Sites Safety Rules alerts using OD

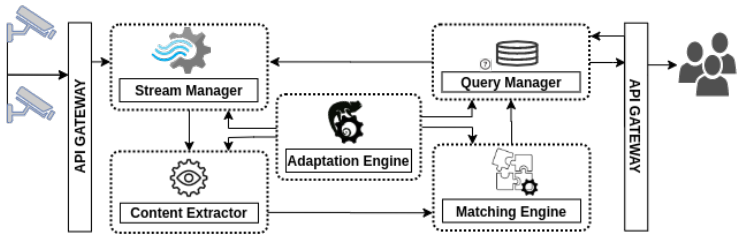

Fig. 2. Microservice architecture for DNN-based Distributed Multimedia Event Processing.

[4]. Sprocket [1] implements MSA for MEP and takes advantage of well-defined patterns from Amazon AWS. The approach is highly dependent on a single cloud infrastructure, making it harder to apply it to the Edge.

3 MEP Framework Design

Microservices Decomposition: Our approach for defining the boundaries of our MSA is to use a Domain-Driven Design (DDD) decomposition [4] and to improve it over multiple development iterations. This process supported us on avoiding common MSA anti-patterns, such as having the wrong cuts in our architecture. We ended up producing five main sub-domains (see Table 1 and Figure 2), each containing multiple sub-domains with their respective microservices (19 in total).

Underlying Tooling. Below there is a summary of the selected tools for our MEP framework together with the reason for each choice.

– **Docker**: To get the most of MSA, and to have a Cloud/Edge-Friendly solution, we are using Docker as our containerisation. Docker allows us to have a compartmentalised, independent, scalable, and reproducible development

Table 1. Our main sub-domains details

Sub-Domain	Purpose
API Gateway	Connection of external entities, such as publishers and subscribers.
Query Manager	Parsing, maintaining, optimising, and planning the user queries.
Stream Manager	Pre-processing of publisher streams, scheduling and dispatching events.
Content Extraction	DNN models for extracting features from the video streams.
Matching Engine	Manages the matching of the extracted features with the users queries.
Adaptation Engine	Incorporates self-adaptive behaviour into the system.

and deployment process. It can be used in both Cloud and Edge devices with a small footprint in resource usage.

- **Streaming**: We selected Redis rather than Kafka because in addition to Redis providing most of the stream functionality as Kafka, when comparing their docker images, Redis uses **100 times less memory, 10 times less disk space and 5 times less CPU** than Kafka when the systems are idle.
- **JSON**: We decided to use JSON to serialise our internal messages. This schemaless format fits our unstructured data from the video streams, it is simple to use, and it is part of Python default libraries.
- **CV**: We selected Python and Tensorflow to work with the DNN models. They are widely used both in the academia and in the industry, and there are versions built especially for common Edge device, with the option of easily setting the amount of GPU allocated for each model. To represent the video streams, we are using Video Event Knowledge Graphs (VEKG) [10].
- **Microservice Monitoring**: We chose Jaeger for the distributed event tracing. It provides a unified front for analysing the path and time of each event on the system. And to monitor the Quality of Service metrics of our MSA we use Prometheus. It also provides a centralised view of each microservice (MS) current metrics. Both tools are used by our Adaptive Engine to ensure that the framework adaptation plans are more precise.

4 Framework Evaluation

4.1 Study Requirements

We selected energy and speed as our study metrics. **Speed** is essential in scenarios where there is a need for a quick response to events identified in the system [9]. **Energy**'s importance comes in three-fold, first in its economical impact, especially for big companies that manage massive cloud infrastructures in their data-centres. Second, for its ecological significance, since recent studies show that 14% of the worldwide energy consumption is for data-centres alone [3]. Being especially true regarding DNN-based models since the carbon footprint of DNN models can produce as much C02 as five cars would produce in a lifetime [7]. And third, for the context of resource-constrained Edge devices, where energy is often a limited resource.

Table 2. Device Specifications

Device	CPU	Memory RAM	GPU	Disk
Jetson TX2	Dual-core Denver 2 64-bit, quad-core ARM A57 complex	8 GB 128-bit LPDDR4 1866MHz - 59.7 GB/s	NVIDIA Pascal 256 CUDA cores(FP16)	32 GB eMMC 5.1
Dedicated Server	Intel i9-9900K 8 Cores	32 GB Corsair 163301 2x 16 GB DDR4 3200 MHz	MSI GeForce RTX 2080 TI GAMING X TRIO 11GB	500 GB SSD & 4 TB HD

4.2 Methodology

To measure the impact of Energy and Speed that our framework adds to a DNN-based OD operation on a mixed Cloud-Edge environment, we needed first to execute different state-of-the-art DNN-based OD models with the images from our chosen dataset and calculate their energy consumption and speed in different Edge and Cloud scenarios, and then choose the model that had the best speed in the Edge environment.

Next, we tested our MEP framework architecture in a mixed Cloud-Edge environment. We started a single node of our framework in our Dedicated Server (Cloud environment) without a content extraction microservice. Then, we started a single Object Detection MS in the Jetson (Edge environment) with GPU enabled, using the previously selected DNN model, and had the MS connected to the Cloud node. This way, we get a network of microservices which is composed of a mixed Cloud-Edge environment. Once these services were ready, a publisher was connected to the framework, where each event published represents an image from the dataset. Finally, we compare the framework results against the baseline values of the DDN model execution without our framework.

4.3 Execution Environments

We selected two devices (see Table 2). The first is a Jetson TX as the Edge device. The second, for the Cloud environment, is a Dedicated Server. This setup presents different environments for exploration, encompassing both Edge and Cloud. These environments are: i) **Cloud-Baseline**: Cloud environment at stand-by; ii) **Edge-Baseline**: Edge environment at stand-by; iii) **Edge-SSD-Model**: Edge with GPU enabled with only the DNN model running; iv) **Edge-OD-Service**: Edge with GPU enabled with the DNN model running inside our solution's MS; v) **Cloud-MEP**: Cloud environment running the rest of our architecture.

4.4 Evaluation Method

For our evaluation we followed the same protocol on all experiments, with the method for measuring each one of the targeted metrics as follows:

Speed: We analyse the models' prediction speed for each image and calculate their averages. This is converted to the models' throughput in Frames Per Seconds (FPS), which is a measure of quantity per unit time (Seconds). For our baseline speed, we are using the stand-alone DNN models running without our framework. In this case, the metric is exported to each experiment output. For our framework results, the speed is gathered from the event tracing service and exported to a JSON file once the experiments are done.

Energy Consumption: For energy consumption measurement, we follow on the work of Walker et al. [8]. We are connecting our devices into two smart power plugs monitors that can estimate the energy usage every 10 seconds, and send it via radio frequency to a smart home gateway device that will save it.

During the experiments' execution, we record the starting and ending timestamps. Later, during the evaluation, we get the energy consumption records that match the experiments start/end timestamps and calculate the average of the energy consumption value for the experiment as a whole. We also analysed both Cloud and Edge devices at stand-by, that is, without running anything on them, to get the baseline energy consumption that these machines consumed by being on. To do that, we gathered the energy consumption of the Jetson and the Dedicated Server for 5 minutes and calculated the average and standard deviation of them. The results were: **2 Watts** (standard deviation of 0) and **72.1 Watts** (standard deviation of 0.3) for the Jetson and the Dedicated Server respectively.

4.5 Object Detection Models and Dataset

We started with three state-of-the-art OD DNN models for our initial comparison: SSD-MobilenetV1, Faster RCNN-InceptionV2, Faster RCNN-Inception-ResnetV2-Atrous. The models are pre-trained on the COCO 2017 image dataset [1] and were gathered from the official Tensorflow model collection. The configurations for the DNN models were: The batch size of 1, GPU memory limit of 70% (except for Faster RCNN-Atrous which was 18%), image input size of 300x300 pixels and detection threshold of 0.5. After analysing the results from the models, we selected the SSD DNN model to test our framework against, since it had the best speed in the Edge device. Curiously, this model showed some non-intuitive behaviour, with its usage on the GPU being more economic in terms of energy and with a lower throughput than when running only on the CPU. This only reiterates how heterogeneous Cloud-Edge environments can affect the performance of an application.

Since these models were pre-trained on the COCO 2017 Training dataset, we decided to use the COCO 2017 Validation as our OD dataset. This way, we would not need to implement class label mappings. In COCO 2017 Validation dataset, there are 80 classes and 5000 images.

4.6 Framework impact on Speed and Energy

By analysing the event traces from the OD service running on the Jetson, we could calculate their average time on different processes. In this case, the *"Process Data Event"* process represents the full process of extracting content in the OD MS, starting from the moment that each imaging event is read and finishing

[1] COCO dataset: https://cocodataset.org/

Table 3. Comparison of Energy and Speed from the different environments studied

Environment	**Energy**	**Throughput**	**Process**	**Speed**
Edge-Baseline	2.0 Watts	–	–	–
Edge-SSD-Model	6.6 Watts	1.3 FPS	Model Execution	0.7692 Seconds
Edge-OD-Service	6.1 Watts	1.1 FPS	Process Data Event	0.9027 Seconds
			Serialise and Write Event	0.0166 Seconds
			Tracer Injection	0.0001 Seconds
Cloud-Baseline	72.1 Watts	–	–	–
Cloud-MEP	72.3 Watts	–	Rest of MEP	1.3030 Seconds

after the event is sent to the next service in the data-flow. This process is broken into, first, *Serialise and Write Event*, where the current event is serialised into JSON format and written to the next service stream in the dataflow; and second, *Tracer Injection*, where the last event trace from the service is added to an event before it leaves the service. This way the next service can retrieve the last event trace id, making the event trace flow in Jaeger clearer to follow.

Table 3 shows that the OD service in this setup had a lower throughput than the bare model, **losing 15% of the throughput** the model originally had. This is expected since we are using a lazy load approach for retrieving the images to reduce the size of the event messages through the system. Our imaging events that are read by the service in the Edge only contains the ID of the image stored in the Cloud Redis server, thus the service needs to retrieve each image from the Cloud before it can load it up into the model, which incurs some latency due to the network communication. We also observe that the use of JSON, Redis and Jaeger did not add much overhead in terms of latency on the resource-constrained Edge device.

We can see that adding our framework shell around the DNN model did not increase the amount of energy usage in the Edge. With an average of 6.1 Watts, it indicates that our solution leads to **8% of reduction in energy consumption** when compared to running the model on its own in the Edge, as can be seen on Table 3. This is probably caused by the network communications while retrieving the image from the Cloud to the Edge device. This leaves the CPU and GPU idle, thus reducing the amount of energy consumed. And for the Cloud environment, this was also negligible when compared to the stand-by baseline.

5 Conclusion

This paper has discussed the importance of a Cloud/Edge-Friendly architecture for DNN-based distributed MEP applications over heterogeneous Cloud-Edge environments; tooling decision can have a direct impact on the usability of resource-constrained Edge devices, greatly benefiting real-world scenarios such as when implementing OHS for construction sites.

The paper proposes an MSA for distributed MEP over heterogeneous Cloud-Edge environments; this system is both Edge-Friendly and Cloud-Independent at the same time. Some preliminary results of the proposed system were presented, starting with an analysis of how different OD models perform in heterogeneous Cloud and Edge scenarios. Initial exploration shows promising results on the impact that the proposed architecture solution impact has in a mixed Cloud-Edge deployment. The solution **reduces the energy consumption by 8%** with only a minor **drawback of 15% in throughput** in the Edge environment, while the energy usage in the Cloud is negligible. At the same time, the overhead for deployment in the different scenarios is very small, requiring only specific changes in the node configuration file.

Further testing of this solution is planned in a broader range of scenarios, such as a complete Edge node that can run independently from any Cloud node, as well as with a varying range of workloads with multiple publishers and subscribers. A self-adaptive scheduler for the DNN-based tasks is being developed which will take into account the different characteristics of the DNN models and the deployment environments. This scheduler will then be applied in a real-world case study for OHS in construction sites.

Acknowledgement. This work was supported by Science Foundation Ireland under grant SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund.

References

1. Ao, L., Izhikevich, L., Voelker, G.M., Porter, G.: Sprocket: A serverless video processing framework. In: ACM Symposium on Cloud Computing. ACM (2018)
2. Aslam, A., Curry, E.: Towards a generalized approach for deep neural network based event processing for the internet of multimedia things. IEEE Access **6** (2018)
3. Belkhir, L., Elmeligi, A.: Assessing ict global emissions footprint: Trends to 2040 & recommendations. Journal of Cleaner Production **177**, 448–463 (2018)
4. Fowler, M., Lewis, J.: Microservices a definition of this new architectural term. URL: http://martinfowler. com/articles/microservices. html (2014)
5. Fu, X., Ghaffar, T., Davis, J.C., Lee, D.: Edgewise: a better stream processing engine for the edge. In: 2019 USENIX Annual Technical Conference (2019)
6. Seo, J., Han, S., Lee, S., Kim, H.: Computer vision techniques for construction safety and health monitoring. Advanced Engineering Informatics **29**(2) (2015)
7. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243 (2019)
8. Walker, G., Taylor, A., Whittet, C., Lynn, C., Docherty, C., Stephen, B., Owens, E., Galloway, S.: A practical review of energy saving technology for ageing populations. Applied ergonomics **62**, 247–258 (2017)
9. Wang, J., Feng, Z., Chen, Z., George, S., Bala, M., Pillai, P., Yang, S.W., Satyanarayanan, M.: Bandwidth-efficient live video analytics for drones via edge computing. In: 2018 ACM Symposium on Edge Computing. pp. 159–173. IEEE (2018)
10. Yadav, P., Curry, E.: Vidcep: Complex event processing framework to detect spatiotemporal patterns in video streams. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 2513–2522. IEEE (2019)