Competitive Edge Cloud Object Detection

Bobby Dhillon
University of California
Santa Cruz
bosdhill@ucsc.edu

Samaa Gazaaz
University of California
Santa Cruz
sgazzaz@ucsc.edu

Faisal Nawab
University of California
Santa Cruz
fnawab@ucsc.edu

ABSTRACT

Many Internet of Things (IoT) and edge applications require real-time processing and data access, such as video surveillance, autonomous vehicles, virtual reality, and augmented reality (V/AR). We introduce a system architecture that leverages the edge in real-time object detection of a video stream, storage of and access to detection results, and actions triggered on the detection of objects while utilizing the cloud for archival purposes. We propose a demonstration that adapts our system architecture to host a reactive application that showcases the need for real-time data processing and storage on the edge. Our proposed demonstration will allow users to experiment with both an edge and cloud architecture, where there are two forms of competition: between users who desire to beat each other's scores, and between the cloud and edge architectures to provide faster object detection verification and video streams.

PVLDB Reference Format:
Bobby Dhillon, Samaa Gazzaz, and Faisal Nawab. Competitive Edge Cloud Object Detection Database. *PVLDB*, 12(xxx): xxxx-yyyy, 2020.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

Many modern Internet of Things (IoT) applications rely on the efficient, real-time processing of video streams. Applications include Virtual and Augmented Reality (V/AR), video surveillance, self-driving cars, and more. A challenge that faces these applications is the lack of real-time image processing and storage in the cloud, as a result of high wide-area latency and bandwidth costs. We introduce an IoT object detection system that utilizes the edge and the cloud for different purposes. The edge is utilized for real-time object detection of a video stream, storage and access to detection results, and actions triggered on detection of objects. The cloud, on the other hand, is utilized for archival purposes. The IoT object detection system was motivated by IoT edge data management and processing at the edge [1, 2, 4], and IoT application use cases such as home automation and surveillance.

We propose a demonstration that will adapt our IoT object detection system architecture and allow the user to play a game that involves real-time detection of different animals in a video. When the user detects an animal or set of animals while playing the video, the user's detection must be verified before the video ends in order for the user to earn points. We allow the users to experiment with two different architectures in verifying and playing back object detection results: data processing, storage, retrieval, and action-on-detect in the edge, and data processing, storage, retrieval, and action-on-detect in the cloud. The user is ranked and scored separately for each architecture on the architecture's leaderboard. The real-time demonstration application would emphasize the need for the quick processing and data retrieval from the edge versus the processing and data retrieval from the cloud that will incur wide-area latency. We aim to use the demo to motivate our IoT object detection system and design, and utilize the *action-on-detect* feature of our architecture. The action-on-detect feature models interactive uses of image detection in mobile games and edge/IoT applications where a detection leads to a real-time action.

In this proposal, we present the IoT object detection system architecture (section 2), the adapted demonstration system (section 3), and the demonstration and its interface (section 4).

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

2. IMAGE DETECTION SYSTEM

The IoT object detection system consists of an edge, cloud, and client node, and the application which is the system user. The client node serves as a source of image frames and is on the same wireless network as the edge node, whereas the cloud node is hosted in a public or private cloud provider (for example, Amazon AWS). The application is independent of the system and can be located anywhere. The edge node contains a pre-trained model for objection detection, and a data store for the image frame entries, which are image frames along with metadata such as labels that resulted from the object detection model.

The edge node periodically archives image frames to the cloud node while retaining the image frame metadata, serving as cache and metadata log for the application. The edge node also has a set of actions that may be triggered on detection of an object, which is configured and handled by the application. For example, an action-on-detect could be if a person is detected, turn on the light. When the edge node

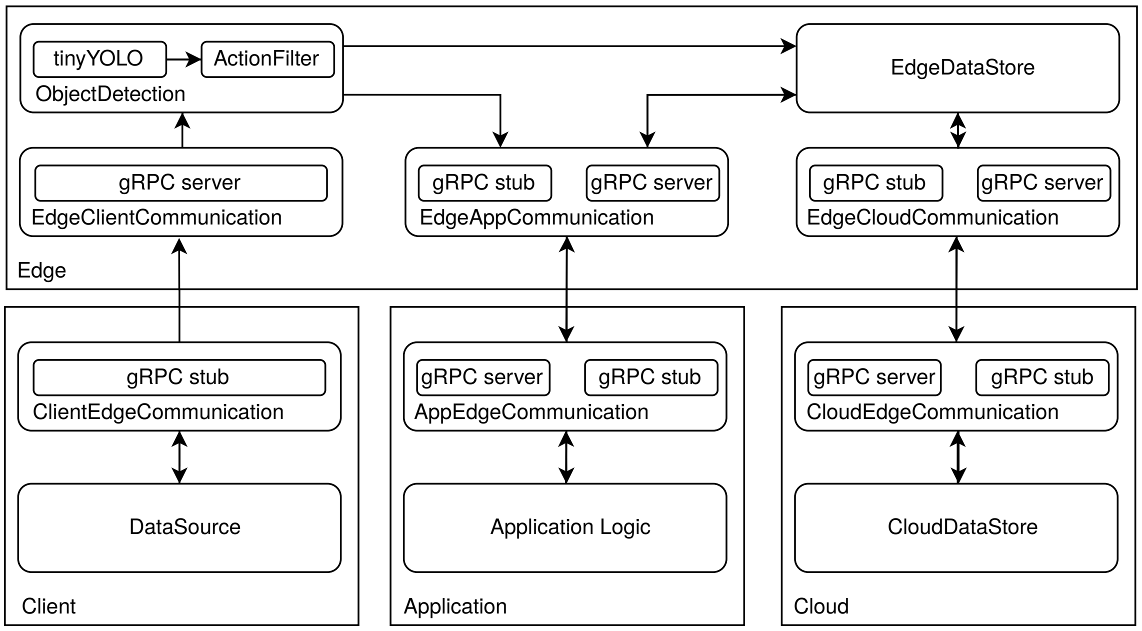


Figure 1: IoT object detection system architecture

detects a person object, the edge node triggers an action to turn on the light that is handled by the application.

The action-on-detect design gears the system towards IoT applications, such as smart home automation and surveillance [4].

2.1 Client

The client node in Figure 1 acts as a source of image frames for the edge node to process for object detection, storage, and action-on-detect. The client node consists of two components: the DataSource component, and the ClientEdgeCommunication component. The DataSource extracts image frames from a source, for example a web camera, or an mp4 video. The EdgeClientCommunication component uses a gRPC stub to upload the image frames from the DataSource component to the edge node. The image frames are received by the the gRPC server in the EdgeClientCommunication component. Since the client node acts only as a source of image frames, the results of the image classifications are impertinent to the client node and so the data flows strictly upstream.

2.2 Edge

The edge node in Figure 1 serves the purpose of processing the image frames ingested from the client node, storing the results of processing the image frames, and triggering actions on detection of objects in the image frames for the application to handle. The edge node consists of the following components: the EdgeClientCommunication, EdgeAppCommunication, EdgeCloudCommunication, ObjectDetection, and EdgeDataStore components.

(a) EdgeClientCommunication. The EdgeClientCommunication component receives the uploaded image frames from the client node on the gRPC server, and then passes them to the ObjectDetection component.

(b) ObjectDetection. The ObjectDetection component detects the object labels in the image frames and uses the Tiny-YOLO NN Model, which is a smaller and faster version of the original YOLO2 [3]. The detection results are partitioned into two sets with the ActionFilter: image frames that have interesting objects, and image frames that have uninteresting objects or no detected objects.

For the image frames with interesting objects, the ActionFilter component will issue a write request to the EdgeDataStore component with an image frame entry that consists of the image frame and its metadata. In this case, the image frame's metadata consists of a map from labels to detection confidences, a map of labels to box dimensions, the interesting image frame's time to live value (ITTL), and the time of detection. If there is any action associated with the detection of the interesting objects, the ActionFilter component will pass the image frame entry and action to the EdgeAppCommunication component. The EdgeAppCommunication component will then call the appropriate RPC implemented by the gRPC server located in the AppEdgeCommunication in the application to handle the object actions.

For the image frames with uninteresting or no detected objects, the ActionFilter component will make a write request to the EdgeDataStore component with an image frame entry that consists of the image frame and its metadata. In this case, the image frame's metadata consists a map from labels to detection confidences or none if there are no objects, the map of labels to bounding box dimensions or none if there are no objects, the uninteresting image frame's time to live value (UTTL), and the time of detection.

(c) EdgeCloudCommunication. The EdgeCloudCommunication component uses a gRPC stub to periodically archive batches of image frames and their metadata to a gRPC server located in the cloud node's CloudEdgeCommunication component. When archiving the image frame entries, the image frames are removed from the entries and uploaded while the image frame entry's metadata is retained in the EdgeDataStore component. The archival period depends on the image frame entry's ITTL or UTTL values, where UTTL is less than ITTL.

(d) EdgeDataStore. The EdgeDataStore component acts as an image frame entry cache, providing fast access to recently processed frames. The EdgeDataStore stores image frame entries, which are image frames and their associated metadata, or just image entry metadata if the image entry was archived. The EdgeDataStore component interfaces with the EdgeAppCommunication, EdgeCloudCommunication, and ObjectDetection components. The EdgeDataStore component handles write requests from the ObjectDetection component, read, remove, and update requests from the EdgeCloudCommunication component, and read requests from the EdgeAppCommunication component. In the case where the EdgeDataStore component receives a read for an image frame entry that has been archived, the EdgeCloudCommunication component will issue a read request to the CloudEdgeCommunication component and update the entry in the EdgeDataStore.

(e) EdgeAppCommunication. The EdgeAppCommunication component passes actions triggered by the ActionFilter component using the appropriate RPC in the gRPC stub to the gRPC server located in the AppEdgeCommunication component. The EdgeAppCommunication component also handles RPCs from the application's AppEdgeCommunication gRPC stub, which can request image frame entries by the image frame's metadata, or access only image frame entry metadata. This allows fast access for applications to stream image frames, filter and fetch specific image frame entries depending on metadata, or fetch only image frame entry metadata.

2.3 Cloud

The cloud node can be thought of as an archival data storage for the edge node. The cloud node has just two components: the CloudEdgeCommunication component, and the CloudDataStore component.

(a) CloudEdgeCommunication. The CloudEdgeCommunication component receives the image frames and metadata to be archived on its gRPC server from the gRPC stub located in the EdgeCloudCommunication component in the edge node. After receiving the image frames and metadata, it issues a write request to the CloudDataStore component to persist the image frame entries. The gRPC server in the CloudEdgeCommunication component implements a read RPC that is called from the gRPC stub located in the EdgeCloudCommunication component that reads the requested image frame entry from the CloudDataStore and returns it.

(b) CloudDataStore. The CloudDataStore component handles write and read requests for image frame entries from the CloudEdgeCommunication component. It is used as persistent storage for the image frame entries that were archived from the edge node.

2.4 Application

The application node in Figure 1 is defined to be the IoT object detection system's user. The application consists of just two components: the AppEdgeCommunication component, and the Application Logic component. The Application Logic component serves as an abstraction for the remainder of the application.

The AppEdgeCommunication component issues read requests for image frame entries and image frame entry metadata, which allows for streaming and fast access to metadata. The AppEdgeCommunication component also handles actions through RPCs implemented in its gRPC server and is called from the gRPC stub in the EdgeAppCommunication component. An RPC action call will include image frames, their metadata, and the action, with the implementation logic of the action handler defined by the application.

3. DEMONSTRATION SYSTEM

We adapt the IoT image detection system to host a reactive application for our proposed demonstration. The demonstration user detects animals in a video and has the detections verified by the cloud and edge in real-time, and then the cloud and edge return a video stream of image frames detected for the duration of the video. The demonstration will showcase the need for processing and data management at the edge, and further motivate our original system design to integrate action-on-detect functionality. In our demonstration system, we omit the cloud archival process and instead opt for the same architecture on both the cloud and edge nodes, while implementing the client and application on the same machine.

3.1 Client

The client node and application will be consolidated into the a single client, and the ClientEdgeCommunication component will contain a gRPC stub for the cloud node as well as for the edge node. The Application Logic component will be a web application that displays the video for the users to detect animals on. In order for the user to log animal detections, the user must press and hold each animal's associated key for the duration of animal's appearance in the video. For example, when the user detects a coyote and a lion, she will press down the c and l keys for the duration of their appearances in the video. As she releases the keys, the frames that were displayed in the video for the duration of the key presses will be stored in memory as the user's detections.

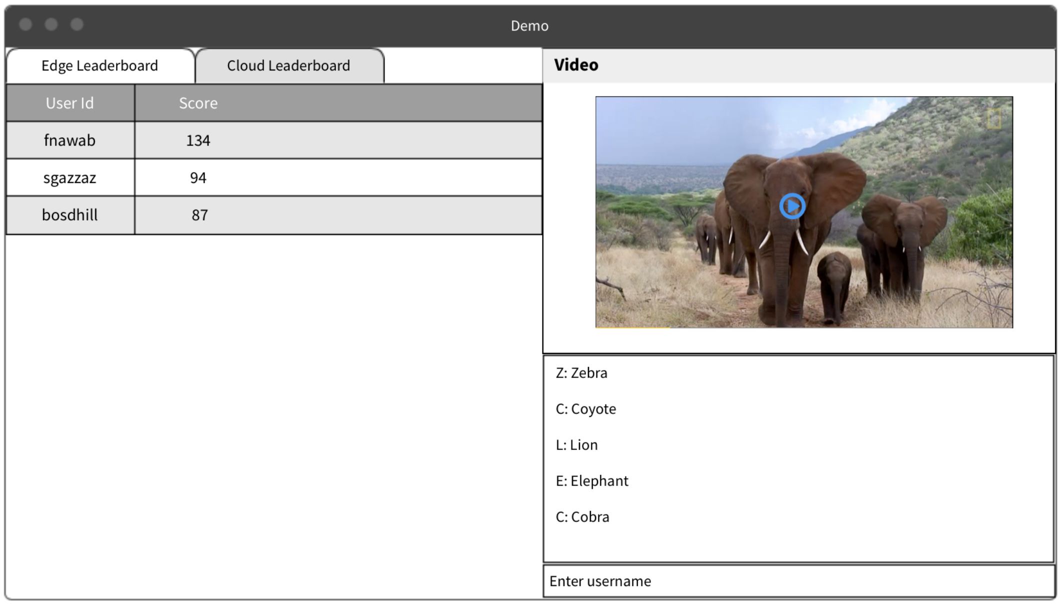
The image frames the user detected animals on will be sent to either the cloud or the edge, each of which have an action-on-detect for each animal. If the cloud or edge detects animals in the image frames, an associated action will be triggered and handled by the client. The action handlers in the client will compare the user's animal detections to the cloud or edge node's animal detections, and increment the user's score by 1 for each image frame that has a matching animal detection.

3.2 Edge and Cloud

The edge and cloud nodes will have similar implementations, and will no longer require the EdgeCloudCommunication components. The Tiny-Yolo NN model in the cloud node's ObjectDetection component will be replaced with YOLO2 [3], since the cloud node will have access to more compute resources than the edge node.

4. DEMONSTRATION

The user will play the animal detection game that consists of two rounds, with the edge for the first round and cloud for the second round. Each round requires the user to log her animal detections for a video. The user will have her detections verified and her score computed by the cloud or



User Id	Score
fnawab	134
sgazzaz	94
bosdhill	87

Z: Zebra
C: Coyote
L: Lion
E: Elephant
C: Cobra

Figure 2: Proposed demonstration user interface

edge in real-time, and then view the objects detected by the cloud or edge during the round as a stream of image frames once the video ends. The user will then have her score added to the cloud or edge leaderboard in Figure 2, and can view her rank and score for each architecture.

4.1 User Interface

The user will be presented with two main views seen in Figure 2: the leaderboard view on the left, and the video player view on the right. The user interface for the video player view will include a text entry box for the user to enter her username and will be used to track her score on the leaderboard.

The video player view includes a sub-view that contains the key maps for the different animal types, for example z for zebra, c for coyote, etc. that will allow the user to know which keys to press when animals are presented in the video frames.

4.2 User Interaction

First, the user must enter her name in the text entry box for the video player to become enabled in Figure 2. Once the user presses play, she will start the first round of the demonstration with the cloud and will need to log her animal detections by pressing each animal's associated key for the duration of animal's appearance in the video (z for zebra, c for coyote, etc.). As she releases the keys, the frames the user logged detections for will be uploaded to the cloud for object detection, and the user's score will be computed on the client in real-time by handling the cloud's action-on-detect. Once the video stops, the cloud will stream image frames it stored for the user to view. When the stream terminates, the client updates the cloud's leaderboard in Figure 2 with the user's score. The video becomes playable again and allows the user to play the same round but with the edge. The demonstration completes after the edge round.

After the edge round, the user will notice the sluggish frame rate and short duration of the stream returned from the cloud when compared to that of the edge.

5. CONCLUSIONS

We introduced an IoT object detection system that utilizes the edge for object detection, storage of detection results, data access, action-on-detect, and archival of stale image frames to the cloud. We proposed an adaption of our IoT object detection system to host a demonstration that showcases the need for edge processing and data storage with a reactive application. Instead of showing the user plain metrics, we proposed a demonstration that will allow users to compete with each in real-time object detection, and have the edge and cloud compete in real-time verification of the user's object detection. Our proposed demonstration would provide an engaging user experience and reinforce the paradigm of data processing and management at the edge.

6. REFERENCES

[1] Y. Guo, B. Zou, J. Ren, Q. Liu, D. Zhang, and Y. Zhang. Distributed and Efficient Object Detection via Interactions Among Devices, Edge, and Cloud. *IEEE Transactions on Multimedia*, 21(11):2903–2915, Nov. 2019. Conference Name: IEEE Transactions on Multimedia.

[2] D. O'Keeffe, T. Salonidis, and P. Pietzuch. Frontier: resilient edge processing for the internet of things, June 2018.

[3] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242 [cs]*, Dec. 2016. arXiv: 1612.08242.

[4] R. Wolski, C. Krintz, F. Bakir, G. George, and W.-T. Lin. CSPOT: portable, multi-scale functions-as-a-service for IoT. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing - SEC '19*, pages 236–249, Arlington, Virginia, 2019. ACM Press.