Navigating the Visual Fog: Analyzing and Managing Visual Data from Edge to Cloud

Ragaad Altarawneh, Christina Strong, Luis Remis, Pablo Muñoz, Addicam Sanjay, Srikanth Kambhatla
{ragaad.altarawneh, christina.r.strong, luis.remis, pablo.munoz, addicam.v.sanjay, srikanth.kambhatla}@intel.com

Intel Corporation

Abstract

Visual data produced at the edge is rich with information, opening a world of analytics opportunities for applications to explore. However, the demanding requirements of visual data on computational resources and bandwidth have hindered effective processing, preventing the data from being used in an economically efficient manner. In order to scale out visual analytics systems, it is necessary to have a framework that works collaboratively between edge and cloud. In this paper, we propose an end-to-end (E2E) visual fog architecture, designed for processing and management of visual data. Using our architecture to extract shopper insights, we are able to achieve application specified real time requirements for extracting and querying visual data, showing the feasibility of our design in a real-world setting. We also discuss the lessons we learned from deploying an edge-to-cloud architecture for video streaming applications.

1 Introduction

With over 70 exabytes of live video expected to be flowing over the Internet by 2021, video analytics is becoming an increasingly important source for deriving actionable insights in various use cases like digital surveillance, retail analytics, factory automation, and smart cities. Despite its abundance and ubiquity, visual data is still significantly under-utilized due to the computational requirements and the outstanding volume of collected data. While today's solutions have made advances in converting video data into actionable information at scale [3], these solutions often have fixed function capabilities and are limited to the use of cloud infrastructure.

An example of these limitations can be found in retail analytics, where the fixed functionality is often something trivial such as counting the number of shoppers. A mismatch between in-store inventory and shopper buying patterns costs retailers collectively nearly $800 billion globally. A subset of retail analytics, called shopper insights, focuses on the problem of deriving suggested actions based on the in-store behavior of shoppers [6]. Providing shopper insights requires far more functionality, as well as flexibility, than simply counting people. To further complicate matters, given the limited bandwidth from edge data sources such as cameras, it is often not feasible to send all video streams to the cloud [2,7]. While this presents a problem in using cloud infrastructure alone, it is also impractical to rely solely on edge devices, due to the significant computational resource requirements of visual data. Edge computing has the additional drawback of not being economically scalable, as having on-premise servers translates to recurring expenses to the users. The fog paradigm uses edge and cloud resources collaboratively for improved resource management and network bandwidth utilization [1,2].

The goal of this paper is to address the problem of domain-specific video analytics by applying the fog paradigm in the context of retail shopping. We propose an E2E architecture that analyzes streaming visual data and manages the resulting metadata. To achieve this, we combine two components: a) the Streaming Analytics Framework (SAF), which provides an easy-to-use API for building visual analytic pipelines that can span from edge to cloud, and b) the Visual Data Management System (VDMS [4]), a data management system specifically designed for visual data. Using shopper insights as our primary use case, we show how to develop an application that makes use of our E2E framework to analyze streaming data using SAF, and then store the resulting data and metadata in VDMS. We explain how to enhance and provide context for the analyzed data through a shopper insights application. We evaluate our framework using a set of queries that provide insights to a store (or set of stores) and prove the feasibility of the framework for such an application.

2 End to End Streaming Architecture

E2E streaming data management architecture refers to an edge-to-cloud design for workload and data management. Figure 1 shows the data flow of our E2E architecture. In this E2E architecture, live video streams originate from video cameras located in the shop. Each of these streams is analyzed by the

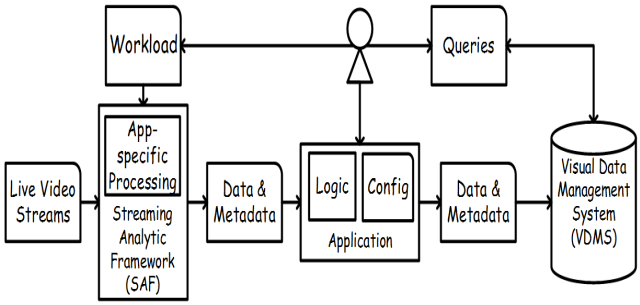

Figure 1: Edge-to-cloud design of E2E streaming data management framework.

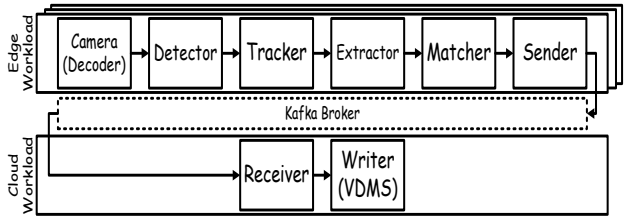

Figure 2: The tracker pipeline blocks, built using SAF.

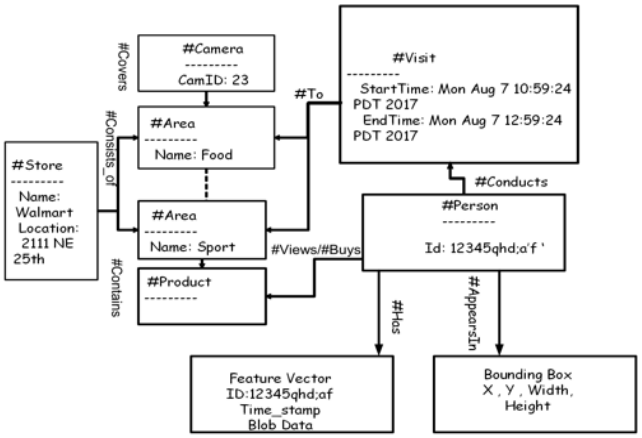

Figure 3: VDMS example schema.

SAF in an application specific manner to generate data and metadata, which are optionally enhanced by an application. Message passing is realized by the Apache Kafka protocol; the choice of protocol may depend on the scale of the deployment and the network topology. Finally, the information is stored in and managed by the Visual Data Management System (VDMS).

2.1 Streaming Analytics Framework (SAF): Visual Workload Management

E2E architecture requires the creation and distribution of video analytic workloads across devices from edge to cloud with dynamic workload creation using an easy-to-use and flexible interface. The Streaming Analytics Framework[1] (SAF) processes live video streams into a combination of data and metadata using a specific pipeline that can be easily configured by the developer. In Figure 2, we show the tracker pipeline that was used in the retail application context. The processed data and metadata from SAF is enhanced by the application, which provides additional logic and context about the data and metadata. With the rich, easy-to-use APIs for both SAF and VDMS, an application developer can create a distributed video analytics workload spanning from the edge to the cloud using SAF and provide additional context to the resulting data. An end user can subsequently query VDMS to retrieve information relevant to the application.

2.2 Visual Data Management System (VDMS)

To store and manage the data from SAF, we use the Visual Data Management System (VDMS [4]), a novel data management solution that treats visual data such as images, videos, and feature descriptors as first class objects in the system. The goal of VDMS is to enable efficient access of big-visual-data to support visual analytics. This is achieved by searching for relevant visual data via metadata stored as a graph, as well as enabling faster access to visual data through new machine-friendly storage formats. VDMS follows a server-client model, supporting multiple clients concurrently, and uses JSON-based API calls that allow an application to interact with VDMS easily and efficiently. The API explicitly predefines certain visual primitives associated with images, videos, and feature vectors. While VDMS uses a graph database to store metadata, the API is not graph specific. VDMS differs from existing large scale photo serving, video streaming, and textual big-data applications in its primary focus of supporting machine learning and data analytic workloads that use visual data. The VDMS architecture consists of three main components, that are available as open source tools[2].

2.3 E2E Context Aware Application

The third component of the E2E architecture is the application itself. While SAF can perform analytics and VDMS can store data and metadata, it is also necessary to have an application that provides context to the analyzed data. For SAF, the application should be responsible for indicating which operators are needed and how they should be connected in order to achieve its objective, as shown in Figure 2. The application should identify what its metadata schema would be in order to

[1] https://github.com/viscloud/saf

[2] https://github.com/IntelLabs/vdms

add entities with appropriate labels and properties to VDMS (e.g. Customer, Product), see Figure 3. It should also perform any sort of configuration steps such as setting up store areas, cameras and other details known in advance of the streaming execution.

Further, the application can provide additional functionality that the basic versions of SAF and VDMS may not support. An example of this would be if the application wanted to be able to store the feature vectors of people found in the streaming videos in VDMS. It is not necessary to store every feature vector seen. Rather, the application should summarize a set of feature vectors that belong to the same person and store that in VDMS. Since it is the application that knows feature vectors belong to people, and that there is a high probability that a set of them belong to the same person, it is the application that should handle this operation. This sort of logic, that provides information that is specific to the application, can be implemented in the application logic.

3 Visual Fog Instantiation

In this section, we describe in detail the realization of our use case, shopper insights. We start by providing a set of queries that provide insights into shopper behavior, and identify the initial configuration of the framework and resulting schema provided to VDMS. Finally, we explain some of the application logic needed to satisfy a few of the more complex queries.

3.1 Shopper Insights (Retail Application)

Online retailers have benefited from real time insights collected from users, such as how long users spend looking at a given product and what types of products a user looks at most often. The purpose of shopper insights in retail analytics is to provide brick and mortar retailers with a similar set of tools in order to keep up with their online competitors.

To achieve this, data about the in-store behaviors and experiences of shoppers must be collected. The process of converting this data into insights for improved in-store shopper marketing should be continuous and automated. We have identified a set of queries that can provide insights about shoppers and the way they interact with the store. These queries are listed below, in order of increasing complexity:

Visits per Store: Provides retailers with information on how many people visited the store in a given time range.

Find the sequence of areas for a specific visitor: In this context, we report the list of areas that a person visited and their time.

Returned Customer: Find if a customer has visited before using the feature vector of that person

Trip Summary: Lists all the areas a customer visited in a given trip. This can help with more targeted advertisements or understanding behaviors.

Hotspot Identification: Identifies areas in which the largest number of customers have gathered. This can help store managers with ways to rearrange the store and spread out products to enable ease of viewing.

Peak Time Identification: Given a specific duration, identifies the time frame that has the highest number of visitors. We designed this query to iterate through possible durations at a minute scale. For example, if the specified duration is an hour and the peak hour is from 2:13pm to 3:13pm, the application will identify the time frame at that granularity. We show how an instantiation of our E2E framework helps us capture responses to these queries on a real time basis and evaluate the performance of the framework in the following section. These queries represent a sample of what can be achieved with a general framework, and are a good subset to demonstrate how the framework can dynamically help gather such insights.

3.2 Framework Configuration

Information that is known in advance, such as the layout of a store, is populated at the initialization of the application in a configuration step. This type of information is updated occasionally, such as when the store is reconfigured or more cameras are added. Dynamic information, about customers and where they are in the store, is what SAF collects and sends to VDMS.

This dynamic information, together with the information about the store, forms a schema that enables VDMS to efficiently store and retrieve data. Figure 3 shows an example schema containing some of the entities relevant to the shopper insights workload, such as areas in a store and camera information, and associated properties for each entity. It also shows entities created from the data provided by SAF, as well as how they relate to the other pieces of the schema.

Storing information in VDMS in accordance with the schema enables efficient queries over the data provided by SAF. Moreover, if SAF starts providing new information about the people inside the store (such as ethnicity or emotions), VDMS can easily extend the schema without losing the previous data and start supporting new kinds of queries. This is one of the features that makes VDMS powerful and well suited for this kind of applications.

3.3 Retail Application Logic

To enable more complex shopper insights, the application needs to provide additional logic about the data collected by SAF. We discuss two different types of data that are contributed by the application logic: constructed data and enhanced data. We look at how the application handles re-identification as an example of enhanced data, and how the

application creates a new type of metadata as an example of constructed data.

Enhanced Data: Re-identification is the process by which a collection of feature vectors is searched for a query feature vector, usually corresponding to a person or object, to establish if there is a similar feature vector in the collection. The similarity is usually expressed and measured in terms of geometric distances (Euclidean distance, inner product, etc.). This operation is also called feature matching, and the similarity search is often performed using a k-nearest neighbor (KNN) algorithm. VDMS naively supports the ability to perform efficient feature matching, thus enabling this application to perform re-identification of a person based on extracted feature vectors from SAF. We use VDMS feature vector functionality to store or query each new feature vector coming from SAF. Consider Figure 4, where all the steps necessary for re-identification are shown. First, SAF extracts a feature vector from a bounding box of a person in a frame (Step 1), and sends it to VDMS (Step 2), where it is stored for future reference. Later in time, SAF extracts a second feature vector (Step 3), and issues a query to VDMS to perform feature vector matching (Step 4). VDMS responds with a set of k-nearest neighbors – if any were found – to the query feature vector (Step 5). The response consists of a list of pairs IDs, distance, where the ID corresponds to the person and the distance corresponds to how similar that person is to the query feature vector.

Using that list, the application assesses whether or not the feature vector corresponds to a new customer or an existing one, based on a threshold on the distance. If the returned distance is less than the threshold value, the query feature vector is considered to be an Alias of an existing person, and the new feature vector is connected to the existing person in VDMS. Otherwise, if the returned distance is greater than the specified threshold, a new person is added to VDMS and the queried feature vector is connected to the new person.

Constructed Data: In order to keep track of a specific instance of a customer visiting a store, the application keeps track of that customer's movements through the store, from entrance to exit. Once the customer has exited, the application creates a new piece of data called a visit, which connects the person with the areas in the store she went to between the entrance and the exit. This is illustrated in Figure 3, where a person entity is connected to a visit entity, which in turn is connected to an area in a store. This allows the application to track where a customer goes in the store within a single time period, since a person may visit the same store multiple times. This constructed data provides the ability to identify hot-spots in the store (by iterating over the areas in a store and finding unique visits connected with them), as well as providing a summary of the customer's trip to the store. In addition, it could support customer-specific targeted reminders or advertisements. For example, if a customer has two visits in the same day, but the second visit only consists of a single

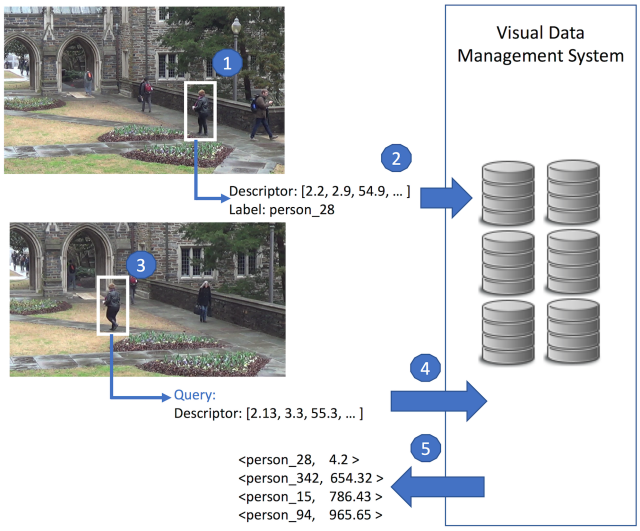

Figure 4: An illustration of how similar feature vectors are found using the SAF-VDMS integration.

area, the application could identify the item that was bought in the second visit and suggest that item to the customer next time he visits.

4 Results

The objective of this evaluation is to validate that our proposed E2E architecture is feasible for deploying shopper insights at scale while meeting the real-time challenges described in the Introduction. We set an aggressive goal of achieving sub-second times for processing video streams, insertion rate throughput, insertion rate latency, and query/sec rate in scalable setup over different store sizes.

4.1 Experimental Setup

We designed the following experimental setup to simulate the behavior of real scenarios. We used two server machines with a dual-socket Intel(R) Xeon(R) CPU E52699 v4 @ 2.20GHz and 128GB of RAM, running Ubuntu 16.04.4 LTS. Figure 5 shows the interaction between the different systems. System 1 simulates streaming instances that feed data to the Kafka broker in System 2. These streaming instances are SAF pipelines that run Docker images on System 1, and are producing data in parallel. In our set-up, the number of streaming instances represents the number of cameras that produce data. In each Docker image, the camera processor reads 1080p video files from the DukeMTMC [5] dataset, at 60 fps. Moreover, each Docker image takes the camera name that it is connected to it as a parameter, and writes to a separate Kafka topic associated with that camera. Therefore, docker-image1 is connected to camera1 and writes data into the SAF-camera1 topic in the Kafka broker. This segregation is important; as the SAF

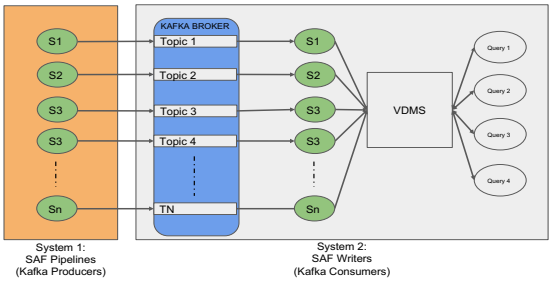

Figure 5: Experimental Set-up.

writers in System 2 will read the data from the specified camera topic without overlapping with other clients. System 2 hosts SAF writers, the Kafka broker, and VDMS, as well as handling clients that wish to query the metadata.

4.2 Performance Metrics

Our goal was to see how much we could stress the system without compromising the real-time performance constraints. Therefore, we used the queries listed in Shopper Insights and evaluated performance using the following metrics:

Query Throughput measures the amount of shopper insights queries/sec that VDMS can be handled in real time less than one second.

Insertion Rate was measured by the following sub-metrics:

1 - Insertion Throughput: This measures the average transaction/sec for one SAF writer to complete a successful transaction, while other writers are performing either a writing operation or a matching operation. Each transaction consists of the metadata that represents one person, a bounding box, and a feature vector vector of 1000 dimensions. This is designed to simulate the aggressive, real settings of our pipeline.

2 - Matching Throughput: This measures the average number of feature vectors matched per second while other writers are performing a writing or matching operation.

4.3 Experimental Results

We ran experiments collecting streaming data for one hour for 5, 15, 25, and 30 cameras. Increasing the number of cameras should increase the complexity of the resulting metadata store in VDMS. This is due to the fact that the visit entity that we build for each visiting person should be larger as the customer moves around a larger number of areas. Remember that in our example schema in Figure 3, areas are connected to cameras in a one-to-one connection. For a store with 15 cameras, we assume that we have 15 areas: each camera is covering one area without overlapping with the other camera views.

Query Throughput To measure the number of the smart retail queries that VDMS can handle concurrently, we designed a test that launches all of these queries at once and measures the number of successfully finished queries once the first one

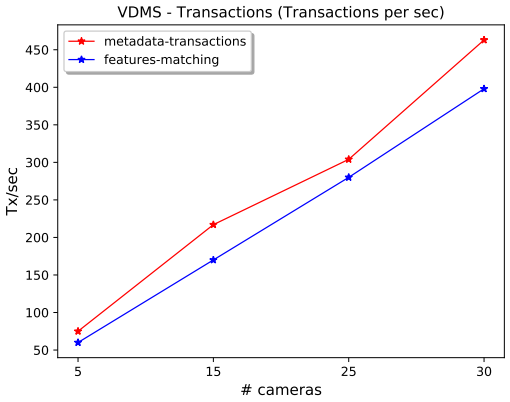

Figure 6: Throughput for metadata (and feature vector blob) insertion rate (red) and feature vector matching query (blue), when varying the number of cameras.

of them is finished. We found that VDMS is able to answer 150 queries/sec for 5 cameras, 140 queries/sec for 15 cameras, 137 queries/sec for 25 cameras, and 142 queries/sec for 30 cameras. We observed that the throughout in each case is not significantly different due to the capability of VDMS to handle multiple request concurrently. To validate this finding, we ran two tests to measure the latency of each individual retail query, one while the queries are running concurrently; the second one while each query is running for 10 time individually. We observed that there is no significant difference between the latency of running a single query versus running concurrent queries, due the VDMS concurrency feature.

Insertion Rate To measure the insertion rate, we measured the time a single writer took to execute a transaction successfully, while the other writers were either writing the metadata and blob representing the feature vector of the detected bounding box or finding a match of a feature vector. In order to simulate realistic behavior of the retail pipeline, all operations are running concurrently. This means VDMS was responsible for executing both insertion operations and matching operations simultaneously. Moreover, as the number of cameras (and therefore writers) increases, the transaction size increases proportionally.

We measured the time that it took each single writer to finish a transaction successfully (send and receive a response), then calculated the estimated throughput of our setup based on those times. Figure 6 (left) shows how VDMS can scale and deliver higher throughput as the number of operations increases. For measuring the matching rate, we measured the latency of each writer to perform a successful feature vector matching operation in a database that has 50,000 feature vectors. Figure 6 shows how VDMS can efficiently perform feature vector matching even with 30 writers concurrently issuing queries and pushing new feature vectors.

5 Discussion Topics

In this paper, we proposed a novel visual fog framework for ad-hoc video analytic with easy-to-use visual workload and data management frameworks. We have described an instantiation of it for a retail analytic use-case. The lessons we learned from our experiences are summarized as the following:

Edge device resource utilization techniques: This criteria is related to the hardware/software capabilities of the edge device. Having multiple deep-learning pipelines running on the same device simultaneously requires heavy usage of resources either on the software or the hardware side. This requires designing new dynamic scheduling techniques to enhance the resource utilization at the edge side.

Feature vector summarizing techniques: Continually pushing feature vectors (high dimensional data), which is relatively redundant per bounding-box, causes unnecessary overhead on both the application and the back-end side. This requires new techniques to summarize/filter multi-dimensional data like feature vectors in real time.

Re-identification of a person based on changing feature vectors: From the retail scenario, we found that relying on the feature vector to re-identify the person is currently a sub-optimal solution to the re-identification problem. The feature vector keeps changing based on the surrounding illumination, angle of the camera, person's pose, or the emotion of the person. Re-identifying a person more accurately using data from the feature vector is an active research topic.

Preserve privacy: One of the main challenges in our work, especially in the retail scenario, is how we can preserve the privacy of shoppers. A common solution is to use only the metadata and the feature vector data, with the assumption that the actual video data is what would violate the privacy concern of shoppers. However, this means potentially missing details from the actual video data that could be useful. It is necessary to identify a middle ground in order to get the most out of the video data without losing the trust of our end-users.

Conclusions We think our proposed E2E framework could be considered one of the early examples that discuss the relations between the edge and the cloud within the retail context. The functionality of our E2E infrastructure is expanding, as we are adding an offline video processing framework in order to support adding more in-depth details about the input video, such as the gender distribution of shoppers and identifying the most interesting products in a store. Moreover, we aim to enhance the usability of the overall framework by providing a better user interface, enabling users from different background to use our framework to ask more queries seamlessly. We highlighted shopper insights in this paper, but we envision the adoption of our E2E architecture in various use cases including, but not limited to, smart cities, agriculture, and industrial real-time applications. Our next step is to deploy and benchmark our current setup in these applications.

References

[1] Flavio Bonomi, Rodolfo A. Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *MCC@SIGCOMM*, 2012.

[2] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescapⓒ. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, 56:684 – 700, 2016.

[3] Yao Lu, , and Srikanth Kandula. Visflow: A relational platform for efficient large-scale video analytics, June 2016.

[4] Luis Remis, Vishakha Gupta-Cledat, Christina R. Strong, and Ragaad Altarawneh. VDMS: an efficient big-visual-data access for machine learning workloads. *CoRR*, abs/1810.11832, 2018.

[5] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. *CoRR*, abs/1609.01775, 2016.

[6] Venkatesh Shankar, J. Jeffrey Inman, Murali Mantrala, Eileen Kelley, and Ross Rizley. Innovations in shopper marketing: Current insights and future research issues. *Journal of Retailing*, 87:S29 – S42, 2011. Innovations in Retailing.

[7] S. Yang, O. Tickoo, and Y. Chen. A framework for visual fog computing. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, May 2017.