Serverless Empowered Video Analytics for Ubiquitous Networked Cameras

Miao Zhang, Fangxin Wang, Yifei Zhu, Jiangchuan Liu, and Bo Li

Abstract

Ubiquitous deployment of networked cameras has boosted the prevalence of video analytics. Leveraging machines to automatically analyze captured videos has become the driving force behind a variety of contemporary applications. However, it is known to be resource-hungry with highly dynamic demands, which mismatches the existing monolithic cloud service deployment with coarse-grained resource allocation. Recent advances in serverless computing, which offers ultra-fast and fine-grained autoscaling, would become a game-changer. This article closely examines the potential and challenges of serverless computing in building modern video analytics applications. We accordingly present an integrated framework with geo-distributed resources, and identify the critical design issues toward its implementation. We further discuss a series of promising research directions in this field.

Introduction

Networked cameras have been deployed at a staggering rate. For instance, the number of surveillance cameras installed in the world was expected to increase from 770 million in 2019 to 1 billion in 2021 (https://www.cnbc.com/2019/12/06/one-billion-surveillance-cameras-will-be-watching-globally-in-2021.html). Leveraging machines to automatically analyze videos captured by these cameras has fuelled the development of video analytics [1, 2], which has been the driving force behind a wide variety of contemporary applications, such as intelligent transportation, smart retail, and mobile vision systems.

Figure 1 shows a typical video analytics application that answers queries about the number of a specific object (e.g., vehicles) passing through a camera's field of view. It involves multiple visual computing primitives [3], such as object detection and object tracking. All of them are notorious for high resource demand and long processing time, especially when deep neural networks (DNNs) are introduced for high accuracy [2, 3]. Although cloud and edge resources have been explored for video analytics [1–3], existing efforts typically allocate and manage coarse-grained resources manually at a virtual machine (VM) level, which can hardly match the fine-grained video content dynamics and unpredictable usage patterns. There have been recent efforts toward camera-to-camera [4] or camera-to-cloud [1, 5] collaboration so as to reduce latency and improve resource utilization. Yet their monolithic deployment architectures hamper the flexibility and scalability, for the partial execution results can hardly be shared among multiple video queries, which is particularly severe for resource-constrained edges.

Since AWS introduced AWS Lambda (https://aws.amazon.com/lambda/) in 2014, serverless computing [6], represented by Function as a Service (FaaS) offerings, has been revolutionizing the way to build modern cloud-native applications. Recent years have witnessed its successful applications, from building backends for the Web, Internet of Things (IoT), and mobile applications, to providing automated infrastructures for on-demand and real-time data processing [7]. Nearly half of AWS users had adopted AWS Lambda in 2020 (https://www.datadoghq.com/state-of-serverless-2020/), and similar trends have been observed in other serverless platforms (https://www.datadoghq.com/state-of-serverless/), such as Google Cloud Functions (GCF) (https://cloud.google.com/functions), and Microsoft Azure Functions (https://azure.microsoft.com/en-us/services/functions/). We believe that serverless computing has great potential for building new-generation video analytics platforms, given its fine-grained autoscaling, its built-in microservice architecture with highly parallelizable compute units, and its truly pay-as-you-go pricing strategy. Unfortunately, unlike lightweight Web or IoT applications, the computing primitives of video analytics are generally much heavier, particularly with advanced learning, and the video queries are highly heterogeneous, which can hardly be unified. These present a series of distinct challenges when meeting with serverless computing.

In this article, we first review the state-of-the-art video analytics solutions and discuss their trade-offs. We then closely investigate the potential when serverless computing becomes the foundation for video analytics. We accordingly present an integrated framework with geo-distributed resources and elaborate on the key design issues of its implementation. We finally identify several future directions that are worthy of investigation.

Video Analytics: State-of-the-Art Solutions and Challenges

Table 1 summarizes the representative solutions to date, which seek to accommodate the huge and highly dynamic resource demands of video analytics from different aspects.

Digital Object Identifier:
10.1109/MNET.101.2000668

Miao Zhang and Jiangchuan Liu are with Simon Fraser University; Fangxin Wang is with The Chinese University of Hong Kong, Shenzhen; Yifei Zhu is with Shanghai Jiao Tong University; Bo Li is with Hong Kong University of Science and Technology.

From Retrospective to Live

Retrospective video analytics queries the recorded videos for post-event intelligence. It works when query events are only known after the videos are captured, and/or only a small fraction of the historical videos are to be queried. DNN model compression and approximation have been widely used in this context. For instance, Focus [2] accelerates large-scale retrospective video analytics by training specialized, faster DNN models to approximate the accuracy of full, expensive models. Most retrospective video analytics systems are built on VM clusters, with coarse-grained and often manual-scaling resources. Though not to be in real-time, analyzing a large-scale historical video dataset with low latency and cost remains challenging in response to unpredictable queries.

To enable real-time interactions (e.g., in augmented reality applications) or decision making (e.g., in intelligent traffic systems), live video analytics arises to process camera feeds in real-time [1]. Streaming camera feeds over dedicated network links to a remote resource-rich cloud data center is the initial attempt. Achieving effective resource management is a significant challenge in this context. Video analytics applications typically expose several general configuration knobs, such as video resolution, frame rate, and models. Different configurations require different resources to achieve real-time responses while leading to different accuracies. Configuration tuning is thus a core technique to balance resources and accuracy. For example, VideoStorm [3] manages cluster resources for massive live video queries by tuning multi-dimensional configuration knobs.

From Cloud to Cloud-Edge

Users' increasing appetite for Ultra-High-Definition (UHD) videos has exacerbated the scarce network bandwidth issue of live video analytics. Edge computing provides a natural solution by bringing computing resources to video sources' proximity. Edge video analytics further extends from edge servers to cameras as the cameras are becoming "smart" with onboard computing resources (https://aws.amazon.com/deeplens/). *DNN model compression and approximation* are also widely used in this context to enable in-situ video analytics on these resource-constrained devices.

Recent years have witnessed the development of cross-camera video analytics [4] that facilitates the building of efficient space-time object tracking systems over an array of collaborative cameras. An integrated deployment across cameras, edges, and the cloud has been conceived as an ideal solution [1]. Reducing data transfer overhead across these heterogeneous components is critical for efficient collaboration. One prevalent technique is *frame filtering*. For example, `FilterForward` [5] executes lightweight classification algorithms on cameras or edge devices to filter out irrelevant frames; since only query-related frames are backhauled to the powerful backends for processing, it preserves accuracy with considerably reduced bandwidth use. As DNN architectures become increasingly deeper, *DNN model splitting* has also been introduced, which executes a portion of a model on the camera and transfers the intermediate results (e.g., high-level features) to the cloud for further inference [8].

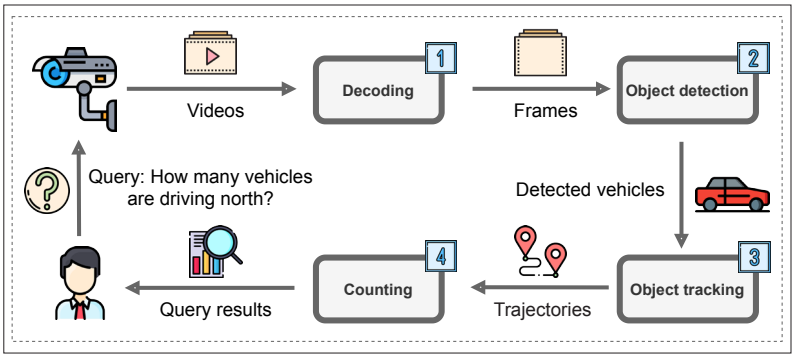

FIGURE 1. An object counting pipeline. The decoding primitive decodes frames from the input camera stream and passes them to the object detection primitive, which detects a specific object (e.g., vehicle in the figure) from the decoded frames. The object tracking primitive is subsequently applied to obtain the moving trajectories of the detected objects, and these trajectories are then fed to the counting primitive for directional counting.

Frame compression further allows cameras or edge devices to transfer low-resolution frames to the cloud, which then recovers the resolution via super-resolution [9].

Despite these efforts, building resource-efficient live video analytics systems remains an elusive goal. The unpredictability of video queries and video content dynamics can cause fine-grained resource demand variations. These variations can hardly be addressed by systems built on coarse-grained computing infrastructures. Moreover, the monolithic deployment architectures of existing systems couple different components tightly. It impairs flexibility and scalability, especially in an edge context where one needs to handle multiple tenants with constrained resources.

When Video Analytics Meets Serverless Computing

Serverless Computing: Emergence and Attractiveness

The last decade has witnessed the success of cloud computing represented by low-level VMs (e.g., Amazon EC2; https://aws.amazon.com/ec2/). As shown in Fig. 2, the VM-based architecture relieves users of physical infrastructure investment but also stresses them with complex virtual resource management and monitoring. To set up a VM cluster in the cloud, developers have to address such issues as predetermining the number and types of VMs in the cluster, routing requests to balance the load, and scaling up or down in response to workload variations. These are known to be barriers to the general cloud users [10].

As the microservices architecture and containerized deployment become popular, cloud providers further abstract infrastructures and propose serverless computing. It enables cloud users to run their applications without thinking about servers (runtimes). As a general implementation of serverless computing, FaaS is popularizing the serverless paradigm [6] and has been offered by major cloud providers, under the name of AWS Lambda, Google Cloud Functions, and Microsoft Azure Functions. As the public cloud extends to the edge, serverless functions can be executed in the content delivery network (e.g., AWS Lambda@Edge (https://aws.amazon.com/lambda/edge/)) and IoT devices (e.g., AWS IoT Green-

System	Type	Architecture	Core Techniques	Description
`Focus` [2]	retrospective	data center	model approximation	Training specialized, faster DNN models to approximate the accuracy of full, expensive models.
`VideoStorm` [3]	live	data center	configuration tuning	Tuning configuration knobs (e.g., frame rate, resolution, and models) to achieve resource-accuracy trade-off.
`FilterForward` [5]	live	edge-to-cloud	frame filtering	Training micro-classifiers to filter out irrelevant frames on edge nodes, backhauling relevant frames to the cloud.
`split-brain` [8]	live	edge-to-cloud	model splitting	Splitting DNNs into two parts with the first part evaluated at the edge and the other part evaluated in the cloud.
`CloudSeg` [9]	live	edge-to-cloud	frame compression	Sending low-resolution videos to the cloud, and then recovering the resolution via super-resolution.

TABLE 1. Representative networked video analytics solutions.

grass (https://aws.amazon.com/greengrass/)). Open-source projects (e.g., Apache OpenWhisk (https://openwhisk.apache.org), and Kubeless (https://kubeless.io)) further create possibilities for the private deployments of serverless functions. Specifically, the popularity of serverless computing can be attributed to its following attractiveness.

Fully Managed Infrastructures: In FaaS platforms, users decompose monolithic applications into small, short-lived, and stateless functions, each implementing a microservice. As the unit of computation, functions are code snippets typically written with popular high-level programming languages, such as Python, Node.js, and Go [6]. For deployment, users register functions to the FaaS platform with minimal configuration efforts (e.g., specify memory only) and declare events to trigger their executions. The FaaS platform is responsible for handling every triggering request, scaling resources precisely with the size of workloads, ensuring fault tolerance and service availability. With offloaded operational responsibilities, users can focus on application development, thus improving agility, innovation, and time-to-market, as shown in Fig. 2.

Fine-grained Autoscaling: Benefiting from the event-driven programming model of serverless computing, the resource allocation and release in FaaS platforms are automatically driven by input workloads. Specifically, each triggering event for a function is served by a dedicated function instance, which executes the function code with the input message in a specialized container or sandbox [7, 6]. Function instances can spin up or down in tens of milliseconds, significantly faster than VMs that typically require many seconds to startup [6, 11]. Such fine-grained autoscaling responds quickly to input workload dynamics, making FaaS a perfect match for unpredictable and sporadic workloads.

Truly Pay-as-you-go Pricing Strategy: A function instance of FaaS is only invoked for handling a triggering request and put into sleep immediately after completion. Developers are charged for their function codes' execution duration, and there is no charge for idle function instance time. The fine-grained billing timescale of FaaS, typically 1 millisecond, makes it a truly pay-as-you-go service [6].

Given the aforementioned advantages, serverless computing has attracted considerable attention from both industry and academia [6, 7]. It is employed to address an increasing variety of workloads. Typical industrial use cases include building backends for Web and IoT applications, processing streaming data in real-time, developing request-response microservices, and infrastructure automation [7]. Recently published research projects also attempt to unlock serverless computing's potential in handling more challenging workloads, such as data analytics [12] and machine learning training [13].

Serverless Video Analytics: Opportunities

Traditional video analytics systems have suffered from the lack of fine-grained computing infrastructures, which can be fortunately provided by serverless computing. For example, recent years have witnessed several attempts to introduce the serverless paradigm in data center video processing [14]. We next illustrate the opportunities brought by empowering video analytics with serverless computing in detail.

Accelerating Video Analytics through Thousands of Function Instances: Serverless computing is well-suited for embarrassingly parallel jobs [10] since it provides fine-grained, readily available infrastructures. This creates new possibilities for video analytics since there are inherently parallelizable structures in videos, such as groups of pictures (GOPs) and frames. After decoding a video into frames, we can invoke massively concurrent function instances. Each function instance runs the same image-based computer vision algorithm to process a frame, thus reducing the processing latency and monetary cost. Since a variety of vision algorithms are developed to target a single image (e.g., object detection, face recognition, and image classification), this kind of parallelism can be harnessed broadly in video analytics.

Handling Unpredictable Usages by Workload-Driven Resource Reshaping: Large-scale deployed low-cost cameras operate in 24 x 7, producing a staggering volume of video data. Up to 100 MB uncompressed data can be gen-

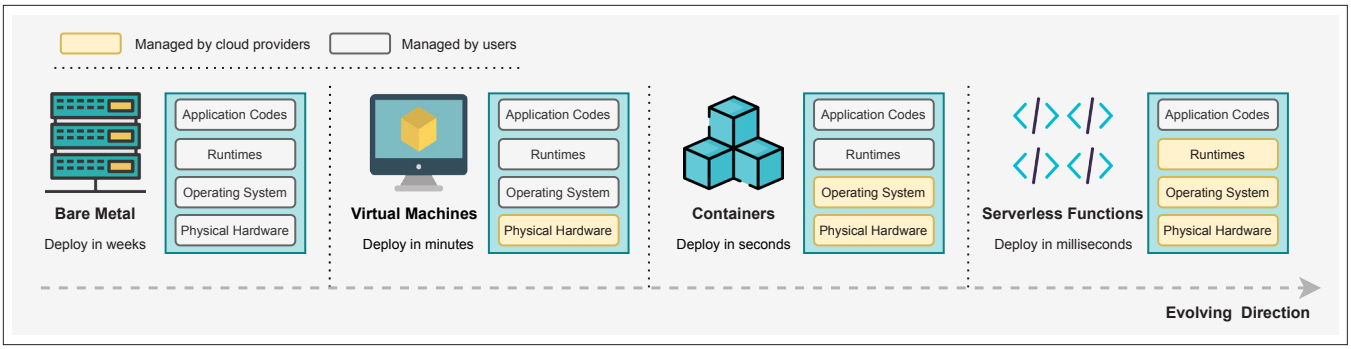

FIGURE 2. The evolution of application deployment architecture. Users take diminishing responsibilities and can focus on the development of application codes with serverless computing. The increasingly lightweight virtualization technologies significantly improve the development and deployment speeds, agility, portability, and time-to-market.

erated every second from a real-world HD camera (https://www.youtube.com/watch?v=1EiC9bvVGnk) (1080p, 30 FPS). Only a small fraction of them may have events of interest. Hence, maintaining a dedicated VM cluster can be of prohibitively high costs. Yet accidental video queries can occur at any time (e.g., in response to an AMBER alert to find an abducted child) with no particular patterns, rendering resource reservation based solutions slow (due to insufficient resources) or inefficient (due to overprovisioning). Serverless computing is a natural fit in this case since functions are invoked in an event-driven way. Variations in the input workload (e.g., a sudden surge of video query requests) automatically trigger allocation or deallocation of resources. Figure 3a compares the monetary cost of VMs and serverless functions when serving unpredictable object detection queries on four video clips collected from the aforementioned camera. To enable real-time responses, the VM instance is provisioned based on peak workloads, and functions instances are configured with sufficient memory. As suggested by the figure, with improved resource utilization, serverless functions greatly reduce the monetary cost by 65.17 percent of its VM counterpart.

Adapting Fine-Grained Video Content Dynamics: Primitives in a video analytics application are often tightly coupled. The input workload of each primitive and further resource demands are affected by video content dynamics. Take the object counting pipeline (Fig. 1) as an example. If the object detection primitive detects no object of interest on decoded frames, there is no need to execute the downstream primitives. To understand the impact of coupling, we run this pipeline to query the number of vehicles or pedestrians on the aforementioned real-world camera stream. We use two metrics to quantify the video content dynamics:

- Average objects per frame (AOPF), which indicates how many queried objects appear in a frame on average.
- Average unique object per frame (AUOPF), because the same object can appear in multiple successive video frames.

As can be seen from Figs. 3b and 3c, coarse-grained VM-based allocation can hardly keep pace with the video content, which typically changes in minutes or even shorter timescales. In contrast, serverless function instances can be invoked to process tiny workloads, for example, a video chunk, a frame, or even an image cropped from a frame. Consequently, the downstream functions in a pipeline can flexibly and agilely respond to upstream functions' results.

SVAG: Geo-Distributed Serverless Video Analytics

SVAG Overview

To fully unleash the potential of serverless computing in video analytics, we envision a Serverless Video Analytics framework with Geo-distributed resources (SVAG) (Fig. 4), through which analytics pipelines can be executed across IoT devices, edge nodes, and the cloud. In `SVAG`, developers can decouple the monolithic code into a pipeline of serverless functions, each implementing a microservice. For instance, the application shown in Fig. 1 can be decoupled into a pipeline of four serverless functions corresponding to its four primitives. The same serverless pipeline can be deployed in multiple IoT devices, edge nodes, and the cloud to enable possible collaborations in the same hierarchy (e.g., device-to-device) and between distinct hierarchies (e.g., edge-to-cloud). After debugging in local and live environments, the serverless pipelines will be ready to serve video queries.

`SVAG` allows users to issue queries on pre-collected or live videos. Users can specify their performance (e.g., latency, monetary cost, and accuracy) goals for a specific video query. They can issue distinct video queries on the same or different camera streams concurrently. `SVAG` then orchestrates the corresponding analytics pipelines across geo-distributed infrastructures. Unlike other geo-distributed video analytics frameworks [1] that manage coarse-grained resources with pre-reservation, `SVAG` offers fine-grained resources in an on-demand manner. To enable the resource-accuracy trade-off, it exposes multiple configurable parameters (e.g., frame rate, resolution, and models), which can be dynamically specified at runtime by function invocation events. According to available resources and performance goals, the original video or intermediate execution results may be redirected multiple times to appropriate places (i.e., IoT devices, edge nodes, and the cloud) for further processing. For video queries with stringent latency constraints, corresponding serverless function instances may be pre-warmed to mitigate the influences of function instance cold-start (https://docs.aws.amazon.

com/lambda/latest/dg/configuration-concurrency.html).

Preliminary Evaluation

Based on our existing work on fine-grained and adaptive partitioning of cloud-edge workloads [15], we have implemented a slim version of SVAG over AWS Lambda and AWS IoT Greengrass. Our current prototype consists of five surveillance cameras with very limited computing capability, one edge node, and a cloud. Assume the network bandwidth between the cameras, the edge node, and the cloud is sufficient for real-time streaming.

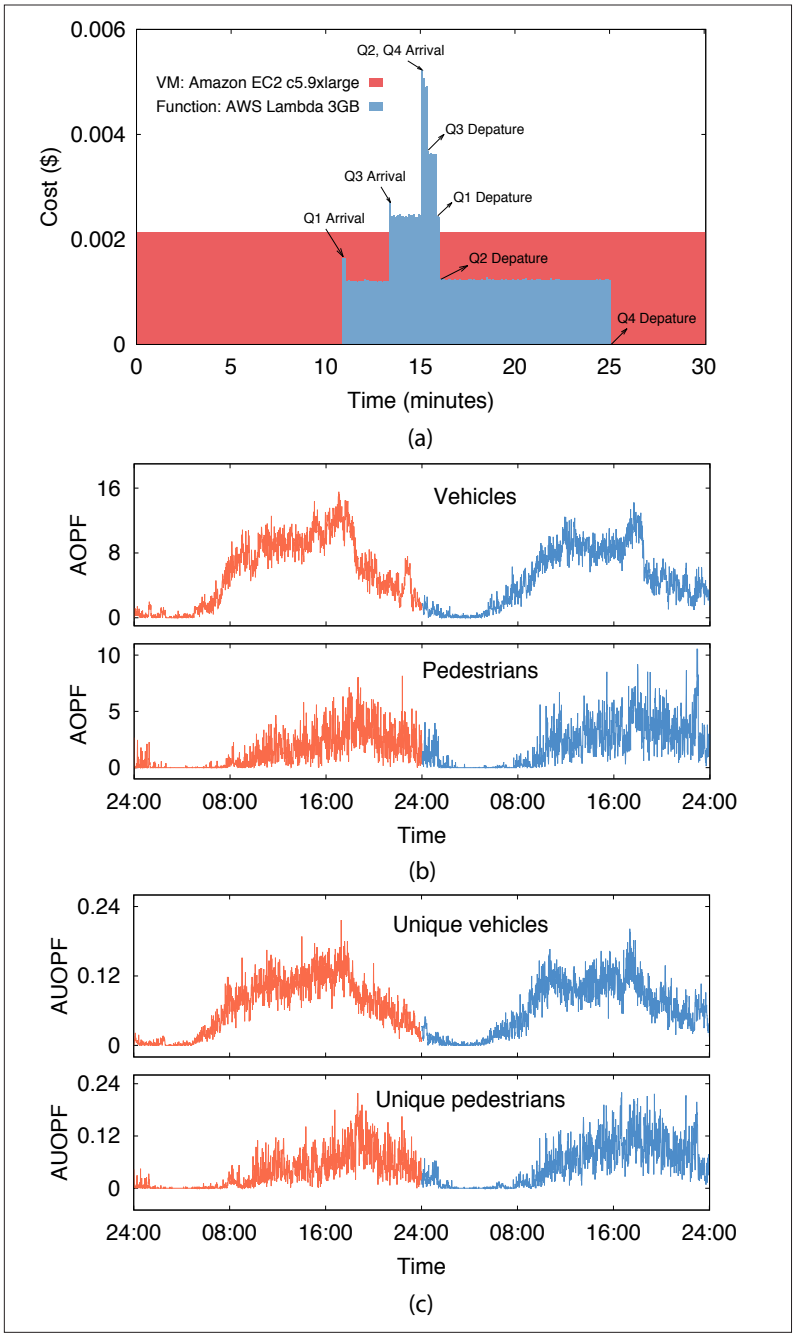

FIGURE 3. Video analytics statistics on real-world camera streams: a) the respective costs of using VMs and serverless functions to answer randomly generated object detection queries on four video streams (Q1-Q4) in 30 minutes; b) and c) depict the corresponding average objects per frame (AOPF) and average unique object per frame (AUOPF) for querying the number of vehicles (top) and pedestrians (bottom) on a crossroad camera stream.

We use two video clips collected from a real-world traffic camera (https://www.youtube.com/watch?v=1EiC9bvVGnk) (traffic streams) and three video clips collected from a roadside restaurant camera (https://www.youtube.com/watch?v=s-bZNL98Z0GE) (restaurant streams) to drive the experiments. Users can issue vehicle counting queries on the traffic streams and customer age and gender queries on the restaurant streams.

The performance goal of all queries is to analyze streams in real-time while minimizing the cloud-edge data transfer overhead and cloud expenditure. We implement a lightweight controller to orchestrate geo-distributed pipeline functions. It integrates a content-aware predictor to handle fine-grained video content dynamics. The predictor can forecast the content-variant resource demands of each query and help determine which version (edge or cloud) of the function to be invoked. To evaluate the pipeline analysis speed, we calculate the percentage of the pipeline output frame rate over the pipeline input frame rate, which can be viewed as a normalized throughput of the pipeline, that is, 100 percent throughput implies real-time analytics. We also examine the amount of data transferred between the edge node and the cloud and the monetary costs for executing functions in the cloud (cloud expenditure).

To fairly evaluate the performance with different system configurations, we compare SVAG with an *Edge-only* scheme only relying on functions deployed on the edge node to analyze streams, and a *Cloud-only* scheme only invoking functions deployed in the cloud. In this experiment, all camera streams are persistently queried for a one-hour duration. The comparison results are shown in Fig. 5a. It can be seen that although there are no data transfer overhead and cloud expenditure for the *Edge-only* scheme, it cannot support real-time analytics due to insufficient resources. On the other hand, the *Cloud-only* scheme achieves real-time processing at relatively high costs. By contrast, the geo-distributed design has the best of both worlds. Specifically, SVAG reaches real-time analytics with only 25.6 percent data transfer overhead and 13.1 percent cloud expenditure of the *Cloud-only* scheme.

We further examine the scalability and adaptability of SVAG by randomly querying the camera streams. The queries arrive as a Poisson Process of a 12-minute mean interarrival time, with the duration being randomly set from 1, 5, or 10 minutes. Fig. 5b shows a query pattern, where each row indicates if there is a query on a specific camera stream or not (e.g., QS1 corresponds to queries on the first camera stream) in a given time window. As can be seen from Fig. 5c, SVAG can flexibly and agilely schedule resources in response to the fine-grained input workload variations. For example, when the edge node is overloaded (e.g., from minute 3 to minute 8), SVAG smartly pushes partial workloads to the cloud for real-time analytics. Otherwise, only functions deployed on the edge node are invoked (e.g., from minute 13 to minute 18), leading to zero cost.

Key Design Issues Toward Full Implementation

Considering the design spirit of serverless computing and video analytics characteristics, before fully realizing SVAG, one has to address the following

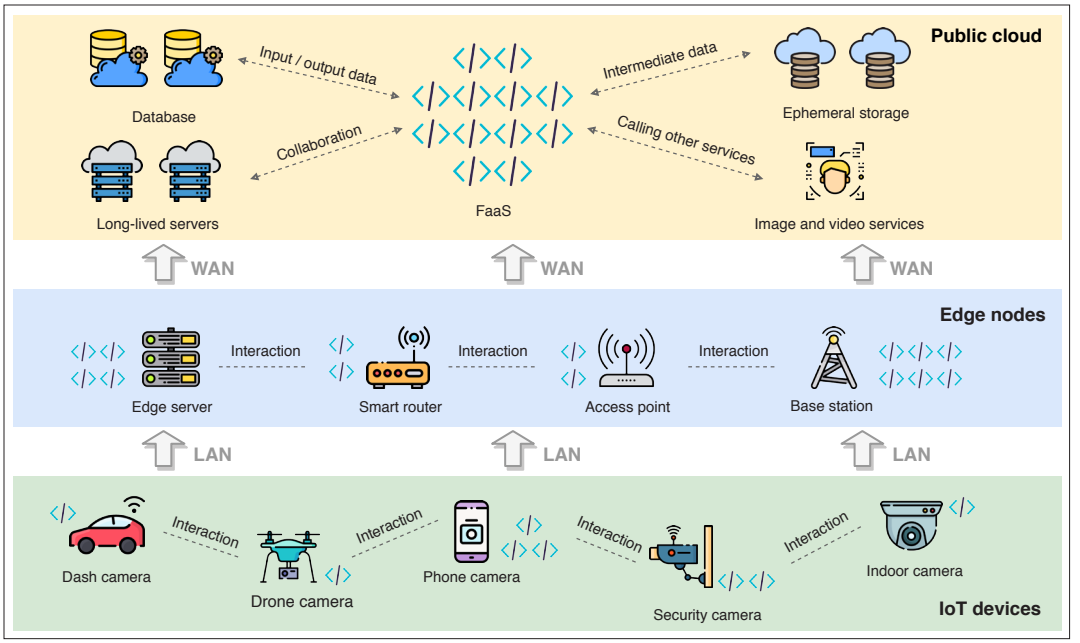

FIGURE 4. An overview of SVAG. There are three hierarchies in this framework, from IoT devices with cameras and onboard computing resources, to heterogeneous resource-constrained edge nodes, to the resource-rich public cloud. Serverless functions are the smallest unit of configuration and placement. Interactions and collaborations within and between hierarchies are enabled for the geo-distributed execution of a serverless video analytics pipeline.

key design issues within current FaaS platforms and beyond.

Orchestration of Video Analytics Pipelines: Video queries can have different performance goals, for example, low latency, low cost, high accuracy, or any combination of them. A function in a video query pipeline has runtime configurable parameters that typically need different resources to meet specific goals. Multiple replicas of a function can be deployed in various IoT devices, edge nodes, and the cloud, but only one of them will be invoked to process a particular input workload. These geo-distributed computing infrastructures that host the replicas provide heterogeneous compute, storage, and network resources while requiring distinct data-shipping distances. Thus, it is already challenging to choose a suitable replica for a single function, not to mention the execution orchestration of massively concurrent pipeline functions.

FaaS platforms do not guarantee the completion order of concurrent function instances. However, the temporal context information can be paramount for visual primitives executing across-frame analysis, for example, object tracking and action recognition. As a result, the orchestration service should be able to handle the synchronization issue. For instance, multiple function instances for the same object detection function can be invoked concurrently, with each processing a frame, to accelerate the processing. Since these function instances' completion order is unpredictable, synchronization mechanisms should be provided to ensure that the downstream object tracking function can process frames in the correct order. Although cloud providers already offer serverless function orchestration services like AWS Step Functions (https://aws.amazon.com/step-functions/), they tend to be slow, expensive, and only applicable to pure cloud-based applications.

Complex Model Inference: To comply with the abstraction of serverless computing, that is, minimizing management efforts from developers, most FaaS offerings expose only memory size as the configurable knob for developers to specify the computing power of function instances. Unlike VM instances, the computing power of serverless function instances is relatively low. For example, the maximum configurable memory size for a GCF function is 8 GB, corresponding to a 4.8 GHz CPU quota (https://cloud.google.com/functions/pricing), and there are typically no hardware accelerators (e.g., GPU) for serverless functions. These limits preclude today's serverless functions from running computer vision primitives employing complex DNN models.

The solution suggested by cloud providers is calling cloud vision APIs (e.g., Amazon Rekognition (https://aws.amazon.com/rekognition/)) from serverless functions, that is, outsourcing heavy liftings to other cloud services. This solution inevitably harms flexibility and locks developers into proprietary APIs. Furthermore, when calling external services, the function instance's resources are locked down to wait for the services to complete. Unfortunately, the invocation of external vision APIs typically requires several seconds to return. It is a long time for the lifecycle of function instances, thus leading to low resource efficiency.

Massive Intermediate Data Sharing: Different functions in the same video analytics pipeline can exchange highly varied amounts of ephemeral intermediate data. Unfortunately, function instances are short-lived, stateless, and unaddressable, impeding direct communications between them. De facto solutions address this issue by exploiting a general-purpose cloud storage service (e.g., Amazon S3 (https://aws.amazon.com/s3/)).

The ideal ephemeral storage services for serverless video analytics should be serverless (i.e., requiring zero administrative effort and scaling automatically as fine as serverless functions.), low-

cost, and high-throughput. Unfortunately, existing serverless storage services, such as Amazon S3 and Amazon DynamoDB (https://aws.amazon.com/dynamodb/), are designed to provide high durability and fault-tolerance for long-term storage. Their costs for high-throughput intermediate data sharing can be quite high. For example, uploading one-day-long decoded video frames (30 FPS) to Amazon S3 will result in 2,592,000 PUT requests of writing frames, at a monetary cost of US$12.96. Thus, designing high-throughput and low-cost ephemeral storage optimized for serverless video analytics remains a significant challenge.

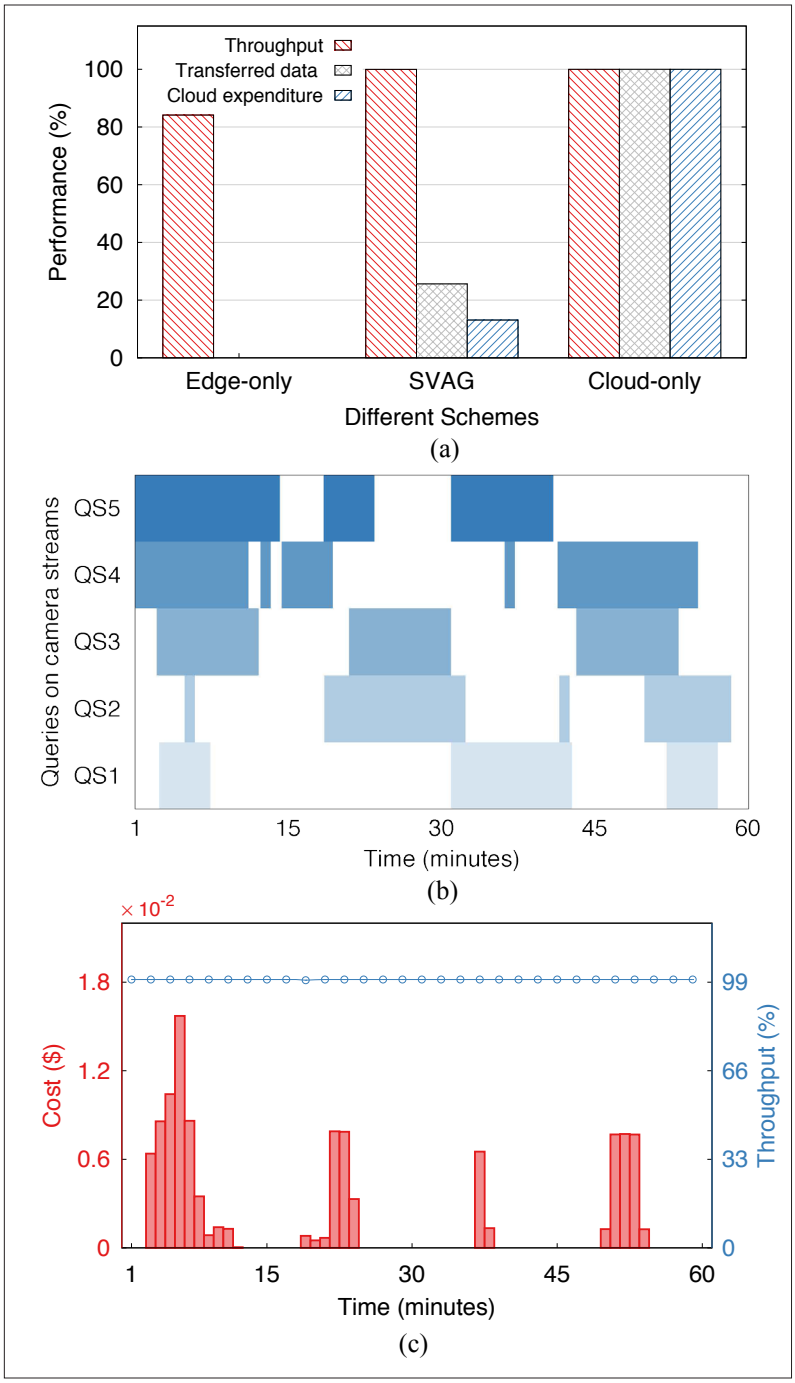

FIGURE 5. Performance under different query patterns. The transferred data and cloud expenditure values in a) are normalized by that of the *Cloud-only* scheme, and the throughput values are calculated by averaging across all queries. The colorized time windows in each row of b) indicate the existence of queries on a specific camera stream. c) assumes it costs US$0.1 to transfer 1 GB data between the edge node and the cloud and sums up the money paid for data transfer and cloud expenditure to obtain the cost values.

Future Directions

Serverless computing is still under rapid development, as is serverless-empowered video analytics. We now highlight some future directions that we believe to be important toward its pervasive deployment.

Geo-Distributed Function Orchestration

Decoupling the functions in a monolithic implementation is the first step toward a serverless deployment. Having more decoupled functions improves scalability and flexibility but complicates the function orchestration. Conversely, fewer decoupled functions can simplify the function orchestration but lead to poor scalability and high processing latency. The choice involves multiple factors and becomes even more challenging with distributed services and resources.

SVAG expects to meet the heterogeneous performance goals of video queries. Note that unpredictable video queries can be submitted, leading to continuous variations in available resources of IoT devices and edge nodes. Resorting to remote resources is not always viable, as unreliable communication channels may lead to unacceptable latencies. On the other hand, fine-grained video content dynamics bring considerable instability to the resource demands and intermediate data size of a pipeline. Tuning the configuration and replica choice of pipeline functions to accommodate the dynamics requires non-trivial efforts. To make reasonable decisions for concurrent video queries, it requires up-to-date knowledge about available resources, network conditions, video content, query types, and importantly, their interactions.

Function-VM Hybridization

For retrospective video analytics, FaaS is a perfect match for video queries with highly data-parallelizable operations; for live video analytics, the splendid use case of FaaS is bursty queries on cold video streams. Because of resource limits, today's FaaS offerings are not ideal for video analytics tasks that require tremendous resources to ensure low latency or high accuracy. Furthermore, compared with VM instances, function instances have a higher price per unit resource, making them less attractive for persistent queries. It would be interesting to combine both functions and VMs toward a hybrid video analytics framework.

Intermediate Data Optimization

As we have seen earlier, intermediate data sharing between serverless functions in video query pipelines can incur significant latency and cost. It is necessary to optimize video queries to reduce the intermediate data size. For example, distinct video queries on the same stream may share partial execution results, reducing unnecessary data exchanges caused by independent computations. Ephemeral storage services also play an important role in intermediate data sharing. There have been significant studies for storage in generic data analytics applications [12], but not for videos. The intermediate data in video analytics often contain massive redundant information and are not easy

to regenerate. Their processing functions are also much more computation-intensive. Insights into video content, vision models, and query patterns are expected to design efficient ephemeral storage.

Security and Privacy Preservation

Video content is known to be sensitive and may contain much more private information than other types of data, for example, the interior of one's house. Unfortunately, following the microservice and event-driven paradigms, serverless video analytics systems expose an larger attack surface than their monolithic counterparts since functions can consume data from a wide variety of event sources (e.g., cloud storage and message queue). The geo-distributed architecture further exacerbates the risks of confidential data leakage. For privacy-preserving purposes, efficient encryption algorithms are necessary to secure the data in transit over the network and at rest on devices. Effective device authentication and appropriate access control policies are critical to ensure that confidential data is only revealed to intended entities in the geo-distributed architecture. As the underlying resources (e.g., CPU and memory) are shared by multiple tenants, protecting data from attacks while it is being processed is also a concern. Trusted execution environments (TEEs) (https://azure.microsoft.com/en-us/solutions/confidential-compute/) have been introduced in VMs to protect data in use. We expect it to be embedded in serverless computing, further preserving the privacy of video content being analyzed.

Conclusion

Video analytics plays an essential role in our daily life. Due to the lack of fine-grained autoscaling computing infrastructures, achieving resource-efficient video analytics is exceptionally challenging. Serverless computing is revolutionizing the way we build applications and opens up new possibilities for video analytics. In this article, we have explored these possibilities and envisioned a unified framework, **SVAG**, which empowers video analytics with geo-distributed serverless computing. We have discussed its key design issues within the current FaaS platforms and beyond, which inspire further explorations on serverless video analytics.

Acknowledgment

This research is supported by a Canada NSERC Discovery Grant; in part by the Key Area R&D Program of Guangdong Province with grant No. 2018B030338001 and the Outstanding Talents Training Fund 202002; and in part by the RGC RIF grant R6021-20 and RGC GRF grants under the contracts 16207818 and 16209120.

References

[1] G. Ananthanarayanan *et al.*, "Real-Time Video Analytics: The Killer App for Edge Computing," *IEEE Computer*, vol. 50, no. 10, 2017, pp. 58–67.
[2] K. Hsieh *et al.*, "Focus: Querying Large Video Datasets with Low Latency and Low Cost," *Proc. 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI'18)*, 2018, pp. 269–86.
[3] H. Zhang *et al.*, "Live Video Analytics at Scale with Approximation and Delay-Tolerance," *Proc. 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*, 2017.
[4] J. Jiang *et al.*, "Networked Cameras are the New Big Data Clusters," *Proc. 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo'19)*, 2019, pp. 1–7.
[5] C. Canel *et al.*, "Scaling Video Analytics on Constrained Edge Nodes," *Proc. 2nd SysML Conference (SysML'19)*, 2019.
[6] E. Jonas *et al.*, "Cloud Programming Simplified: A Berkeley View on Serverless Computing," arXiv preprint arXiv:1902.03383, 2019.
[7] A. Agache *et al.*, "Firecracker: Lightweight Virtualization for Serverless Applications," *Proc. 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*, 2020, pp. 419–34.
[8] J. Emmons *et al.*, "Cracking Open the DNN Black-Box: Video Analytics with DNNs Across the Camera-Cloud Boundary," *Proc. 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo'19)*, 2019, pp. 27–32.
[9] Y. Wang *et al.*, "Bridging the Edge-Cloud Barrier for Real-Time Advanced Vision Analytics," *Proc. 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'19)*, 2019.
[10] E. Jonas *et al.*, "Occupy the Cloud: Distributed Computing for the 99%," *Proc. 2017 Symposium on Cloud Computing (SoCC'17)*, 2017, pp. 445–51.
[11] S. Fouladi *et al.*, "Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads," *Proc. 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)*, 2017, pp. 363–76.
[12] Q. Pu, S. Venkataraman, and I. Stoica, "Shuffling, Fast and Slow: Scalable Analytics on Serverless Infrastructure," *Proc. 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*, 2019, pp. 193–206.
[13] J. Jiang *et al.*, "Towards Demystifying Serverless Machine Learning Training," *Proc. 2021 Int'l. Conf. Management of Data (SIGMOD/PODS'21)*, 2021, pp. 857–71.
[14] L. Ao *et al.*, "Sprocket: A Serverless Video Processing Framework," *Proc. ACM Symposium on Cloud Computing (SoCC'18)*, 2018, pp. 263–74.
[15] M. Zhang *et al.*, "Towards Cloud-Edge Collaborative Online Video Analytics with Fine-Grained Serverless Pipelines," *Proc. 12th ACM Multimedia Systems Conference (MMSys'21)*, 2021, pp. 80–93.

Biographies

Miao Zhang received her B.Eng. degree from Sichuan University, Chengdu, China, in 2015, and her M.Eng. degree from Tsinghua University, Beijing, China, in 2018. She is currently a Ph.D. student at Simon Fraser University, British Columbia, Canada. Her research areas include cloud and edge computing, and multimedia systems and applications.

Fangxin Wang received his B.Eng. degree from Beijing University of Posts and Telecommunication, China, in 2013, his M.Eng. degree from Tsinghua University, China, in 2016, and his Ph.D. degree from Simon Fraser University, British Columbia, Canada, in 2020. He is currently an assistant professor at the Chinese University of Hong Kong, Shenzhen. His research interests include cloud and edge computing, multimedia systems and applications, networking optimization, and deep learning.

Yifei Zhu received his M.Phil. degree from Hong Kong University of Science and Technology in 2015, and his Ph.D. degree in computer science from Simon Fraser University, Canada, in 2020. He is currently an assistant professor at Shanghai Jiao Tong University. His research areas include edge computing and multimedia networking.

Jiangchuan Liu [S'01, M'03, SM'08, F'17] is a professor at Simon Fraser University, BC, Canada. He is a Fellow of The Canadian Academy of Engineering and an IEEE Fellow. He received the B.Eng. degree (cum laude) from Tsinghua and the Ph.D. degree from HKUST. He has served on the editorial boards of *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Multimedia*, *IEEE Communications Surveys and Tutorials*, and *IEEE Internet of Things Journal*. He was a Steering Committee member for *IEEE Transactions on Mobile Computing* and Steering Committee Chair for IEEE/ACM IWQoS. He was TPC Co-Chair for IEEE INFOCOM'2021.

Bo Li is a Chair Professor in the CSE Department, Hong Kong University of Science and Technology. He made pioneering contributions in multimedia communications and Internet video broadcast, which attracted significant attention and investment from industry and received the Test-of-Time Best Paper Award from IEEE INFOCOM'2015. He has received six Best Paper Awards from IEEE, including INFOCOM'2021. He was the Co-TPC Chair for IEEE INFOCOM'2004. He is a Fellow of IEEE. He received his Ph.D. from the ECE Department, University of Massachusetts at Amherst, and his B.Eng. degree (summa cum laude) in computer science from Tsinghua University, Beijing, China.