2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)

Shepherd: Seamless Stream Processing on the Edge

Brian Ramprasad*, Pritish Mishra*, Myles Thiessen*, Hongkai Chen*, Alexandre da Silva Veith*, Moshe Gabel*
Oana Balmau†, Abelard Chow‡, Eyal de Lara*

‡Huawei abelard.chow@huawei.com
†McGill University oana.balmau@cs.mcgill.ca
*University of Toronto {brianr,pritish,mthiessen,chk,aveith,mgabel,delara}@cs.toronto.edu

2022 IEEE/ACM 7th Symposium on Edge Computing (SEC) | 978-1-6654-8611-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/SEC54971.2022.00011

***Abstract*—Next generation applications such as augmented/virtual reality, autonomous driving, and Industry 4.0, have tight latency constraints and produce large amounts of data. To address the real-time nature and high bandwidth usage of new applications, edge computing provides an extension to the cloud infrastructure through a hierarchy of datacenters located between the edge devices and the cloud.**

Outside of the cloud and closer to the edge, the network becomes more dynamic requiring stream processing frameworks to adapt more frequently. Cloud based frameworks adapt very slowly because they employ a stop-the-world approach and it can take several minutes to reconfigure jobs resulting in downtime.

In this paper, we propose Shepherd, a new stream processing framework for edge computing. Shepherd minimizes downtime during application reconfiguration, with almost no impact on data processing latency. Our experiments show that, compared to Apache Storm, Shepherd reduces application downtime from several minutes to a few tens of milliseconds.

***Index Terms*—stream processing, reconfiguration, late binding, hierarchical edge computing, seamless**

I. Introduction

Next generation applications such as autonomous driving, augmented/virtual reality, smart technologies, interactive games, and Industry 4.0 produce massive scales of data that must be analyzed in a timely fashion [1]. Stream processing frameworks are often used to address this need [2].

Stream processing applications are often structured as a dataflow graph. Vertices can be sources that generate streams of data tuples, or operators that execute a function over incoming data streams. Sinks, are a special type of vertices that consume the processed data but are terminal and represent the end of the flow. Traditionally, all application components are placed in the cloud to take advantage of powerful datacenters. Unfortunately, this approach is not compatible with next generation applications, since sending data over wide-area links to the cloud results in high bandwidth usage and high application latency.

Edge computing expands cloud computing with a hierarchy of computational resources located along the path between the edge and the cloud [1], [3]. In this paper, we argue that efficient use of edge computing for stream processing requires support for seamless reconfiguration and deployment of application operators without disrupting application execution. In particular, throughput should be stable, and latency should not spike during the reconfiguration. Seamless reconfiguration is required to enable efficient resource sharing

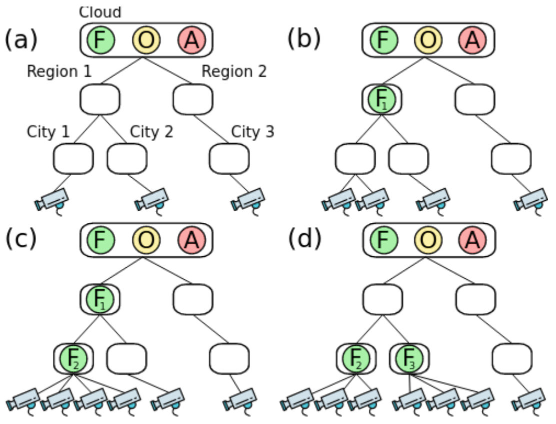

Fig. 1. Seamless Reconfiguration: A video analytics service is dynamically reconfigured to adapt to the change in workload at the edge of the network. This makes more efficient use of the network by reducing the number of video frames transmitted over the WAN.

between applications running on edge datacenters. The smaller size of an edge datacenter, leads to higher costs for storage and computation relative to the cloud (e.g., AWS Wavelength [4] is 40% more expensive than EC2). It is therefore impractical (and may not be even possible due to limited resources) to run all applications continuously on the edge.

Figure 1 illustrates the benefits of dynamic reconfiguration for an application running on a hierarchy of edge datacenters with three levels: cloud, region, and city. The application provides analytics about traffic on city roads by performing object detection on video frames produced by a network of motion-activated street cameras. The application consists of a sequence of three operators: [F], a frame filter operator that removes frames that do not significantly change compared to the previous frame; [O], an object detector operator; and finally, [A], an aggregation operator that computes statistics.

Initially, all operators are deployed on the cloud as the number of active cameras is small, and the network cost of transmitting the raw images to the cloud is low (Figure 1a). As more cameras become active in cities [1,2] located in Region 1, network traffic grows and it becomes more cost-effective to filter frames closer to the source by deploying operator [F_1]