# Real-Time Telemetry Systems for Multidimensional Streaming Data:
## A Case Study on Video Viewership Analytics

*Submitted in partial fulfillment for the requirements for*
*the degreee of*
*Doctor of Philosophy*
*in*
*Electrical & Computer Engineering*

**Antonios Manousis**

B.S., Electrical and Computer Engineering, National Technical University of Athens

Carnegie Mellon University
Pittsburgh, PA

October 2021

*To my parents.*

# Acknowledgments

In August of 2015, a younger and much different version of me hopped on a plane, excited to begin a new life chapter. Six years later, I am writing these words, having completed an outstanding PhD program and getting ready for what is coming next. It's a personal milestone that I am very proud of because the journey has been challenging at times, like a roller-coaster with its many highs and lows. Despite that, completing the PhD program has been an immensely rewarding and transformative experience for me. And as with every meaningful journey, this one involved many wonderful people to whom I am deeply thankful.

First and foremost, I am grateful to my advisor, Vyas Sekar. Vyas has been by my side from the get-go as a tireless research mentor, counselor, and career advisor while giving me the freedom to chart my own researh trajectory. I have always been impressed by Vyas' clarity of thought, his structured and rigorous approach when tackling complex questions, his creativity and his laser-sharp focus (in and out of the squash courts). I hope that osmosis has done its trick and that I adopted some of these traits in my personal work style. But most importantly, I admire Vyas' constant drive to evolve as a mentor. People often wrongly assume that being a good mentor is straightforward, or even that a PhD prepares you in advance for that role. Vyas showed me what that work looks like. Thank you, Vyas; One day I hope to become as good and honest a mentor as you have been to me.

In completing this dissertation, I am also personally and intellectually indebted to Justine Sherry. Justine is without a doubt one of the smartest and most inspiring people I have ever had the fortune to work with and has been the catalyst that jumpstarted my first, first-author research paper after a long and unproductive lull. Thank you, Justine for your help in making me a better researcher, for the coffees that you got me, for lending a listening ear when I needed it, and for introducing me to Ira Glass.

I would also like to thank my committee members, Alan Liu and Hui Zhang. Alan and I wrote our first SIGCOMM paper together and many years later joined forces again when it became clear that the implications of this 2016 paper would be instrumental in my thesis work. Thank you Alan, for your time and help. I was incredibly fortunate to work with Hui at Conviva, on a project that later became PROTEAS, one of the key components of this thesis. Thank you, Hui for opening this door for me.

To my project collaborators Rahul Anand Sharma, Harshil Shah, Henry Milner, Yan Li, Zhuo Cheng and Ran Ben-Basat, thank you for your help in turning vague research ideas into actual papers. To my (extended) lab mates Maria Apostolaki, Aqsa Kashaf, Daehyeok Kim, Sekar Kulandaivel, Hun Namkung, Rahul Shama, Soo-Jin Moon, Seyed Kaveh Fayazbakhsh, Junchen Jiang, Nirav Atre, Adithya Philip, Hugo Sadok, Ranysha Ware, Zhipeng Zhao, Chris Canel, Miguel Ferreira, Margarida Ferreira, thank you for inspiring me with

your smarts and talent and for helping me improve through our conversations.

During my time in Pittsburgh, I was also very fortunate to have in my life an incredible group of people that became my second family. To Joe Sweeney, Natalia Castillejo, Dimitris Stamoulis, Stamatis Athiniotis, Marilina Raices, Gabriele Farina, Pietro Simeoni, Amritanshu Pandy, Onur Kibar, Meric Isgenc, Luca Colombo, Ifigeneia Apostolopoulou, Anna Apostolopoulou, Maria Vlachostergiou, Bechara Maroun, thank you for being such dear friends. To my friends back home, thank you for being constant positive influences and for always inspiring me with your ambition and drive. There's so much I could write about how these people have enriched my life; I can only hope that I have been able to give some of that back to them. Special thanks go to two friends and future professors, Marios Kogias and Charalampos Avraam. Marios set an example that guided me when, as an undergraduate student at NTUA, I was figuring out my next steps. Charalampos' honest and no-fluff advice style kept me accountable throughout the PhD, in times when it felt as if I was losing my focus.

To my beloved Francesca, pure serendipity brought us both here to Pittsburgh, but we chose to travel countless miles and explore all sorts of new places (literally and metaphorically) ever since. Thank you for your love, for putting up with my never-ending deadlines, my moods and all my idiosyncrasies; you have been a rock and I am so excited for this next chapter that is now opening up for us.

Last, but by no means least, my very special gratitude goes to my family: my father Petros Manousis, my mother Sofia Kyriakopoulou and my sister Nefeli Manousi for their never-ending love and support. My parents have put their heart and soul in making sure my sister and I achieve our goals and aspirations. This thesis is for them.

A.M.

The work presented in this thesis has been supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, ERDF Project AIDA (POCI-01-0247-FEDER-045907), NSF awards 1440056, 1440065, and 1700521, CNS-1513764 and CNS-1565343, the Kavcic-Moura research award as well as the Gerontelis Foundation Fellowship, the Deans Graduate Fellowship and the Frank J. Marshall Graduate Fellowship.

**Thesis Committee**

Vyas Sekar (Chair)
Justine Sherry
Zaoxing Liu
Hui Zhang

# Abstract

Large-scale infrastructures across various domains (*e.g.,* Internet services, sensor farms, operations monitoring *etc.*) produce ever-increasing amounts of streaming data. As these data streams contain invaluable operational insights, operators invest heavily on telemetry frameworks to extract these insights and use them towards ensuring their infrastructure's reliability and growth. In this dissertation, we focus on telemetry for streaming video infrastructures. In particular, this work is motivated by a previously unexplored aspect of video telemetry, namely viewership analytics. That is, detecting and diagnosing video viewership anomalies, simultaneously, across multiple subpopulations of viewers.

This dissertation aims at enhancing video operators' toolbox with novel telemetry capabilities for viewership analytics. Nevertheless, designing telemetry workflows for viewership analytics proves challenging on multiple fronts. First, increases in volume and dimensionality of incoming data streams result in a combinatorial explosion of data subpopulations to monitor and, as a result, in prohibitive cost and resource overheads for operators. Second, the contextual and non-stationary nature of viewership complicates the detection and diagnosis of viewership anomalies. Last, the need to simultaneously monitor ever-increasing numbers of subpopulations of viewers complicates extracting the few critical, and often highly localized, events of interest needed to provide actionable insights to operators.

Our work addresses these challenges through the design and implementation of a suite of practical tools for video viewership analytics. First, we introduce Hydra, a novel sketch-based analytics framework for efficient and general analytics over multidimensional data streams. We show that HYDRA offers robust accuracy guarantees at one tenth (or less) of the operational cost of exact analytics frameworks and does so with query latencies that are up to 20× lower than existing alternatives. In Proteas, our second contribution, we leverage key structural insights of viewership in order to enable accurate detection and insightful diagnosis of viewership anomalies. We show that our approach ensures low numbers of false positives and outperforms the closest state-of-the-art alternatives. Last, we illustrate how these insights can be combined in the design of an end-to-end telemetry framework. Through extensive analysis driven by real-world datasets, we demonstrate that our findings can yield substantial cost and resource benefits over existing solutions. Additionally, we discuss their potential applicability in different domains, in addition to video.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Large-scale infrastructures across various domains (*e.g.,* Internet services, sensor farms, operations monitoring, *etc.*) produce massive and ever-increasing amounts of streaming data. As these data streams contain invaluable operational insights, operators rely heavily on real-time telemetry frameworks to extract these insights and use them towards ensuring their infrastructure's reliability and growth. For instance, the analysis of machine-generated server logs of a Facebook data center can shed light into the health and operational status of the service, whereas the analysis of Facebook user data can expose trends and behavioral patterns that are invaluable from a business perspective.

In this thesis, we scope our focus on telemetry for video streaming services. Video streaming services are a key component of today's Internet and play an increasingly important role in our everyday lives for entertainment, communication, and news. A recent study by Cisco [22] predicts that by 2022, streaming video traffic will make up more than 82% of all consumer Internet traffic — 4 times higher than it was in 2017. In addition, the advent of the COVID-19 global pandemic has triggered fundamental changes in viewing behaviors and spurred a tipping point in video streaming growth that shows no signs of reversal [8]. In this increasingly competitive streaming landscape, the survival of video streaming services critically depends on their ability to provide their customers with engaging content and a seamless viewing experience. For example, research shows that even short increases in video buffering rate can lead to $\sim$40% less time watching online videos and cause substantial

losses for ad-based content providers [104].

For these reasons, video streaming operators invest heavily in telemetry capabilities that, in general, focus on two fronts; The first is to ensure that any service disruptions that might impact viewer engagement are promptly detected and mitigated. For example, a telemetry framework for video QoE that processes per-viewer measurements of various video-related metrics (*e.g.,* bitrate, buffering time, *etc.*) issues alerts when different *subpopulations of viewers* are experiencing service disruptions. The second is to shed light into viewer habits and preferences to optimize ad-placement and content production.

This work is motivated by an important, but previously unexplored aspect of video telemetry, that of detecting and diagnosing *video viewership anomalies* across multiple subpopulations of viewers. In developing these capabilities, we also tackle a broader shortcoming of existing telemetry, *i.e.,* enabling efficient and scalable telemetry across a combinatorial explosion of data subpopulations of interest. Overall, we design a suite of new designs and tools that improve the state-of-the-art in telemetry frameworks and implement them in the context of an end-to-end framework for viewership analytics.

## 1.1 Motivating Example — Viewership Analytics

Video streaming providers monitor the operational status and overall health of their infrastructure through various Quality-of-Experience (QoE) metrics (*e.g.,* bitrate, buffering *etc.*). Their goal is to promptly detect and diagnose anomalous incidents (*e.g.,* ISP outages, buggy players) [6, 26] that might affect subpopulations of their viewership base and to inform appropriate mitigation efforts [102, 104]. Yet, while existing QoE-based monitoring workflows are invaluable, they also suffer from critical blind spots. Our conversations with analysts working in this domain have highlighted the need for *viewership-based analytics* (viewership is defined as the the timeseries of the count of active viewers) to complement existing monitoring workflows and to proactively identify anomalies that would be missed by more conventional QoE metrics [102, 104]). We discuss a few illustrative case studies:

1. *Platform failures.* In a recent incident, a buggy update of a content provider's authentication service disconnected viewers and prevented them from logging back in for one hour. During this time, video QoE metrics remained unchanged for the remaining viewers and as a result no QoE-based anomalies were raised. However, the viewership of the content provider indicated a sharp drop. For this incident, the provider was only informed about it through its Twitter support page!
2. *Video encoding errors.* During the live broadcast of a popular annual US event, video encoding errors resulted in severe image pixelation and audio being out of sync. Despite the corrupted content, QoE metrics such as buffering or video start failures remained unaffected during the transmission and the incident went unnoticed until the next day when the provider looked at its unexpectedly low total viewership numbers.
3. *Insufficient streaming resources.* In high-stakes live transmissions, such as sports events, video QoE alerts (*e.g.,* high buffering) are particularly common. These alerts, while indicating the presence of a problem, cannot often explain its underlying root-cause; *e.g.,* is this an outage or a sudden flash crowd that puts unexpected load on streaming resources? Viewership-based analytics can enable operators to disambiguate and, thus take action minimizing the incident's impact.

**Opportunity** Given that sustaining viewer engagement is one of operators' overarching goals, these anecdotal case studies demonstrate that there is ample opportunity to enhance existing telemetry for video QoE through a framework that tracks video viewership and provides actionable insights to the operator through *detection and diagnosis of viewership anomalies.*

## 1.2 Design Blueprint for a Viewership Analytics Framework

Seen in a general context, anomaly detection and diagnosis are not new problems and have been examined in several prior efforts in networked systems (*e.g.,* [33, 45, 50, 53, 106, 110, 131]) and in data mining (*e.g.,* [36, 62, 114, 115, 135, 140, 143, 155]). This allows us to conceptualize a high-level view of the design of a framework for video viewership analytics as follows:

**Input Data** The framework processes streams of *video session summaries i.e.,* per-viewer summaries of viewer activity and streaming video quality. Each data point in the stream is *multidimensional i.e.,* it consists of measurements of various video-QoE metrics along with metadata that describe the video session across domain-specific dimensions, such as ISP, CDN, City, Device and more. This is a standard data format for streaming video telemetry as described in prior work (*e.g.,* [102]) and illustrated in Figure 1.1.

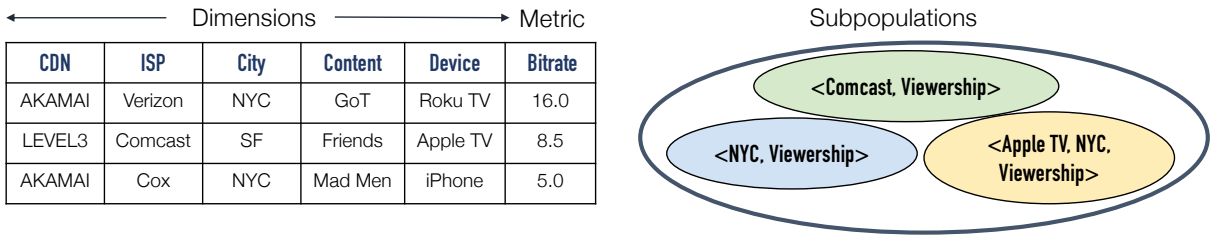



Figure 1.1: View of multidimensional video data

**System Workflow** At a high level, the viewership analytics framework should implement two logical operations: (i) viewership estimation and (ii) alerting of viewership anomalies. We describe them below:

1. *Viewership Estimation.* During this step, the framework ingests a batch of incoming video session summaries and clusters them across combinations of dimension values to represent *subpopulations of viewers e.g.,* NYC-based viewers on iPhones (Figure 1.1).

Then, for each subpopulation of viewers, the framework estimates its viewership count *i.e.,* the total number of active viewers. Over time, the system creates a timeseries of viewership values per subpopulation.

2. *Alerting.* At this phase, the framework uses these estimations to monitor viewership for anomalies. This component implements two sub-operations:
   (a) *Anomaly Detection.* This step leverages the estimated viewership counts in order to periodically detect viewership anomalies on a per-subpopulation basis. It then outputs a list of anomalous subpopulation of viewers.
   (b) *Anomaly Diagnosis.* Last, the diagnosis component, processes detected anomalies in order to translate them to insightful and actionable insights for operators. This component focuses on root-cause attribution *i.e.,* mapping detected anomalies to plausible and possible root-causes in order to inform appropriate mitigation efforts from operators, if possible.

**Viewership analytics are an instance of multidimensional telemetry.**

Note that viewership analytics are an instance of *multidimensional telemetry.* That is, the system leverages multidimensional data in order to perform fine-grained viewership estimation and anomaly detection on a per-subpopulation basis. The multidimensional telemetry paradigm constitutes a positive development for operators as it enables more fine-grained visibility into subpopulations of their data. Specifically, in the context of video viewership, this translates to being able to accurately detect and diagnose video viewership anomalies across different subpopulations of viewers.

## 1.3 Challenges of Viewership Analytics Workflows

Unfortunately, realizing the above workflow of multidimensional telemetry for viewership analytics raises several important challenges.

**Challenge #1: Existing telemetry cannot efficiently scale with data subpopulations.**

As video streaming services mature, the data they produce become increasingly multi-dimensional [102]. This high data dimensionality results in a combinatorial explosion of subpopulations to monitor and in prohibitive ingestion- and query-time overheads for telemetry frameworks. In addition, the need of operators to analyze input data through potentially multiple summary statistics induces extra compute/memory overheads, proportional to the number of statistics of interest. In this respect, as we will extensively discuss in Chapter 3, existing frameworks can be fundamentally limited in terms of the trade-off across accuracy, scalability (across subpopulations), and generality (across summary statistics they can estimate).

**Challenge #2: Contextual anomalies and non-stationarity complicate detection.**

A sudden drop/rise in the timeseries of viewership is not always anomalous; *e.g.,* when a popular show ends, viewership is expected to drop sharply and should *not* be marked as anomalous. Therefore, anomaly detection needs to consider the *context* in which a candidate anomaly appears [95]. While this contextual nature of anomalies is not unique to viewership, it suggests we use model-based techniques for detection [62]. Unfortunately, viewership is also non-stationary *i.e.,* the timeseries' statistical properties change over time, making it difficult to accurately capture them with a model.

**Challenge #3: Anomaly redundancy complicates diagnosis and contributes to alert fatigue.**

In practice, we want to monitor many subpopulations of viewers; *e.g.,* did viewers using RokuTV have an outage? Note, however, that a single logical event (*e.g.,* RokuTV outage) can simultaneously manifest as an anomaly across multiple viewership groups (*e.g.,* for NYC-RokuTV, NYC-Comcast, SF-RokuTV). Naively raising alerts for all these groups will overwhelm the analyst and complicate the diagnosis, thus raising the need for

anomaly summarization. Unfortunately, techniques used in prior summarization and diagnosis works (*e.g.,* [33, 45, 50, 53, 102, 106, 110, 131]), while insightful, cannot account for the features of viewership that determine how a single incident manifests across multiple groups.

## 1.4 A Taxonomy of Alternatives

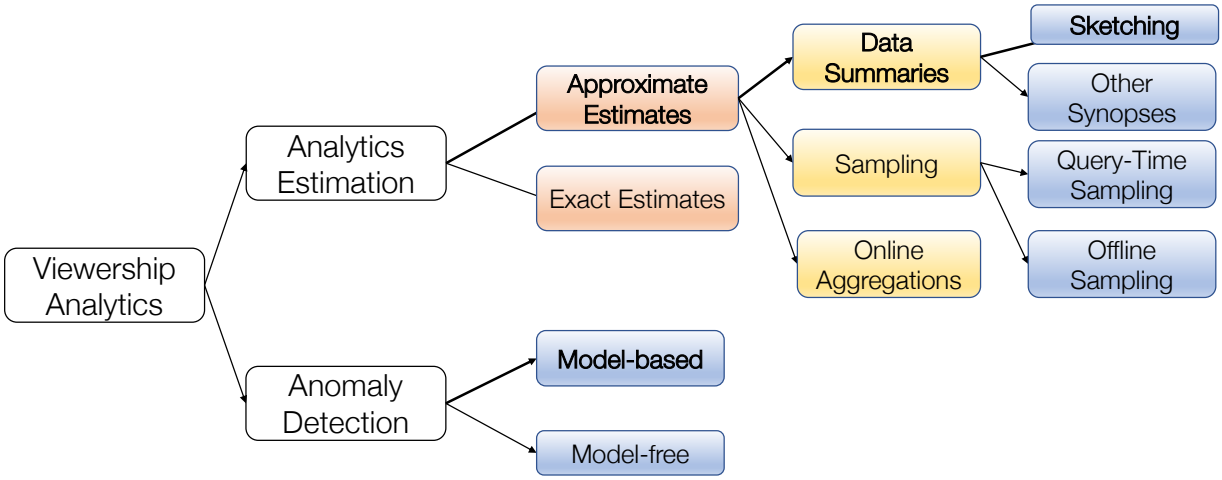



Figure 1.2: A taxonomy of alternatives

Having summarized the system design overview as well as the key challenges faced by operators, we now introduce a taxonomy of alternatives (Figure 1.2) for viewership analytics frameworks. This taxonomy also helps us put our contributions in context; the high-level approach taken in this work in specified in bold text in Figure 1.2. We defer a more detailed analysis of related work to Chapter 2 and examine each component of the telemetry framework separately.

**Analytics Estimation** We classify existing analytics estimation approaches based on their design decisions in order to achieve scalability:

1. *Horizontal Scaling for exact analytics* In this category we find analytics frameworks whose distributed design enables reducing estimation latency through horizontal scal-

ing of server resources (*e.g.,* Spark [166], Hive [146] Hadoop [141], Dremel [121], Druid [161], Flink [60]). These frameworks can provide *precisely exact* analytics and their clusters scale along with the input data. However, as we will discuss in more detail in Chapter 2, these approaches cannot cope with the overheads of multidimensional analytics.

2. *Approximate analytics* To account for the inefficiencies of exact analytics, approximate analytics frameworks tradeoff estimation accuracy for improved estimation latencies. There are two broad categories of approximate frameworks, namely frameworks implementing sampling (query-time or offline) [30, 31, 35, 43, 61, 66, 106, 121, 128, 142, 146] and frameworks implementing online aggregation or other data summarization techniques [71, 72, 76, 84, 118, 165].

Given that scalability and interactive response times are some of our design objectives, we take an *approximate analytics approach.* Additionally, given our strong requirements for accuracy guarantees from our estimations, we opt for approaches that involve precomputations and, in particular, *sketch-based analytics* [71, 72, 76, 84, 118, 165].

**Anomaly Detection** Regarding detection of viewership anomalies, we see two broad classes of approaches, namely model-based [51, 64, 68, 145, 167] and model-free ones [41, 123, 155, 168]. As we discuss extensively in Chapter 4, the specific characteristics of viewership, point in the direction of *model-based* detection.

## 1.5 Thesis Overview

### 1.5.1 Thesis Statement

*Viewership analytics workflows are a necessary tool in video providers' telemetry toolkit. However, the ever-increasing multidimensionality of video data and the contextual and non-stationary nature of viewership have created both scalability and anomaly detection*

*challenges. This thesis (1) presents novel sketch-based algorithms to enable scalable viewership (and other analytics) estimation workflows, (2) leverages key structural insights of viewership in order to enable accurate anomaly detection and insightful diagnosis, and (3) presents the design of a novel end-to-end viewership analytics workflow.*

### 1.5.2 High-level Approach Overview

In this thesis, we envision a framework for real-time video viewership analytics that implements both analytics estimation as well as alerting. To that end, we develop a suite of novel solutions for the viewership estimation and alerting (*i.e.,* detection and diagnosis) workflows. More specifically:

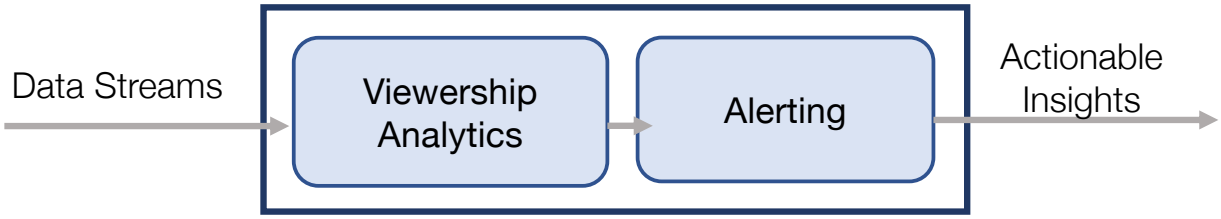



Figure 1.3: System overview

1. *Viewership Estimation.* We propose a novel, sketch-based workflow that enables scalable, interactive, and provably accurate approximate estimations of viewership (among a broader set of summary statistics) across subpopulations of viewers. This component ingests multidimensional video data streams and aggregates them across different combinations of dimensions values that represent different subpopulations of viewers. Our sketching primitives enable efficient summarization of subpopulations of viewers in sub-linear space (to the number of subpopulations) as well as approximate, yet provably accurate, viewership estimations.
2. *Alerting.* The alerting component uses the viewership estimations to detect and diagnose viewership anomalies across subpopulations of viewers. For detection, we design model-based approaches that leverage domain-specific insights about viewership in

order to detect anomalous viewership behavior on a per-subpopulation granularity. For diagnosis, we design a practical mechanism that scopes detected anomalies to a small set of root-cause viewership groups. Then using a periodically updated, signature-based library, we match detected anomalies with candidate root-causes in order to produce insightful alerts.

### 1.5.3 Thesis Contributions and Key Results

#### Contribution #1: A scalable, sketch-based multidimensional analytics framework (Chapter 3)

We present HYDRA, a novel approximate analytics framework for efficient and general analytics over multidimensional data streams. HYDRA is based on two key ideas:

1. *Enable scalability with a "sketch-of-sketches":* To tackle the combinatorial explosion of subpopulations, we design HYDRA-sketch, a "sketch of sketches" primitive that enables efficient data stream summarization. This enables a reduction in the framework's ingestion-time memory cost of one to two orders of magnitude compared to Spark- and Druid-based alternatives while at the same time offering robust and provable accuracy guarantees.
2. *Ensure high-fidelity estimations and generality across statistics with universal sketching:*To provide high-fidelity estimations for a general set of summary statistics, we leverage the power of universal sketching [118]. Unlike canonical sketch-based approaches that deploy one custom sketch type per statistic [84, 147, 148], a universal sketch allows for estimating multiple different summary statistics with only one sketching structure using polylogarithmic space.

We analytically prove that combining the above sketch-of-sketches idea with universal sketches ensures polylogarithmic space complexity to the data subpopulations for a given target accuracy. Last, our analysis provides practical strategies for configuring HYDRA.

*Key Results:* We show that HYDRA offers robust accuracy guarantees (estimation error

<5% with 90% probability), comparable to those of exact analytics frameworks at 1/10 of their operating cost or less. HYDRA's memory footprint scales sub-linearly with dataset size and number of data subpopulations, resulting in a $7-20\times$ query latency improvement compared to Spark- and Druid-based alternatives.

**Contribution #2: An alerting framework for video viewership anomalies (Chapter 4):**

PROTEAS is an alerting framework that detects and diagnoses viewership anomalies and builds on the following *structural insights*:

1. *Shape persistence for anomaly detection:* Despite the non-stationarity of viewership, we find that its *underlying shape* remains invariant over longer periods of time, and can thus be leveraged as a basis for detection. By modeling this key structural invariant using custom Gaussian Processes [139], we enable accurate, timely and robust anomaly detection of viewership anomalies across multiple viewership groups.
2. *Hierarchical group dependencies and spatiotemporal anomaly signatures:* A single logical event *propagates* predictably across viewership groups. Using practical heuristics, we extract the set of candidate groups that best explain an anomalous incident. Moreover, while viewership anomalies might be the result of many different root causes, we observe that anomalies resulting from similar incidents share common spatiotemporal features, allowing us to compile a library of *anomaly signatures*. This enables associating each viewership incident to a small set of possible root causes for further investigation.

*Key Results:* Under normal operating conditions, PROTEAS issues a practical (for the analyst) number of alerts with low numbers of false positives (precision >86%) and practically no false negatives. PROTEAS outperforms the closest state-of-the-art alternatives by Twitter [95] and Netflix [25] both quantitatively in terms of precision and qualitatively in terms of trust shown by expert expert analysts to PROTEAS's alerts over those issued by prior work (> 95%). Our synthetic analysis shows that PROTEAS's anomaly summarization component prunes out >99% of redundant anomalies and accurately identifies the root-

cause viewership groups by classifying anomalies in 4 broad classes of events. PROTEAS identified 3 large-scale technical outages that affected up to 50% of the content provider's viewers and were not caught by the existing alerting workflows of a major video analytics provider. PROTEAS tracked flash crowds during popular live events (*e.g.,* sports games) before the corresponding QoE alerts were raised, thus exposing the need for resource re-provisioning. Last, PROTEAS uncovered qualitative insights into the pervasive changes in viewership caused by COVID-19.

**Contribution #3: An end-to-end multidimensional telemetry workflow for anomaly detection**

Combining the design insights of HYDRA and PROTEAS, we prototype INTEGR, a scalable and efficient end-to-end multidimensional telemetry framework for timeseries anomaly detection. Through INTEGR, we demonstrate the feasibility of leveraging the provably accurate, yet approximate, analytics estimations of HYDRA in order to supply downstream tasks such as anomaly detection.

*Key Results:* We compare INTEGR's accuracy against a baseline telemetry framework that runs timeseries anomaly estimation using precisely estimated analytics. We view the baseline's alerts as ground truth and estimate INTEGR's false positives and negatives relative to that ground truth. In the case of INTEGR, when approximate analytics estimation is configured to achieve at least 95% accuracy, we observe that across two real-world datasets, precision and recall are consistently above 95%. This is a strong indicator that approximate estimations have negligible impact on the accuracy of anomaly detection. In addition, we show that INTEGR brings the same resource and cost benefits as HYDRA.

### 1.5.4 Generalizability of Observations

While the focus of this thesis has been on video streaming and, specifically, on viewership analytics, we argue that many of our findings can be applied or extended to domains other than video. Indeed, HYDRA's sketch-based approach to multidimensional telemetry

is general-purpose and not video-specific. Regarding PROTEAS, while its shape-based anomaly detection approach is derived from domain-specific insights, we speculate that there exist various domains that exhibit similar structural patterns. For example, one such domain could be the electric grid where power demand might exhibit the same diurnal patterns as demand for streaming video. We discuss these future work-related questions in Chapter 6.

## 1.6 Thesis Organization

This thesis proceeds as follows: In Chapter 2, we motivate the need for multidimensional analytics, formulate the problem and discuss the limitations of the existing telemetry landscape. In Chapter 3 we present HYDRA, our sketch-based framework for multidimensional analytics. In Chapter 4 we introduce PROTEAS, a real-time alerting framework for video viewership anomalies. Chapter 5 discusses how we bring the insights of HYDRA and PROTEAS together to design and implement an end-to-end telemetry framework for multidimensional data. Finally, in Chapter 6, we summarize the lessons we learnt and propose future work.

# Chapter 2

# Background & Related Work

In this chapter, we provide some background on each of the two components of a viewership analytics framework, namely viewership estimation and alerting. First, we focus on viewership estimation. As viewership estimation is an instance of multidimensional telemetry, we first formalize the multidimensional telemetry domain and provide an overview of related work. Then, we zoom in on the alerting component of viewership analytics, introduce some key definitions and as well as related work.

## 2.1 Multidimensional Telemetry

By means of a series of motivating examples, below we formalize key notions of multidimensional telemetry. Then, we outline related work.

### 2.1.1 Example Use Cases of Multidimensional Telemetry

**Video streaming QoE** To sustain viewer engagement and to maintain ad- and subscription-driven revenue streams, streaming video providers monitor the health of their infrastructure through various video-specific Quality-of-Experience metrics (*e.g.,* bitrate, buffering ratio, join time, viewership *etc.*). The operators' goal is straightforward: detect and diagnose incidents of interest or service disruptions and inform mitigation efforts. Examples of such

incidents abound: Increases in buffering rate that can result in high drops in viewer engagement. ISP outages or buggy video player updates that can drive groups of viewers away or unexpected flash crowds and sudden increases in content demand that when undetected, can generate technical disruptions due to insufficient resource provisioning.

To achieve the aforementioned goal, operators invest heavily in multidimensional telemetry frameworks. These frameworks collect video session summaries *i.e.,* multidimensional data records that measure video QoE across various dimensions of interest (*e.g.,* ISP, CDN, CITY *etc.*) and, through appropriate analysis, provide operators with fine-grained visibility into the perceived QoE of different *subpopulations of viewers*. As a consequence, the telemetry framework can ideally detect and alert the operator on disruptions that might be affecting different subpopulations of viewers so that they're addressed in a timely fashion.

**Resource utilization in data centers** To ensure the sustained reliability of their infrastructure, data center operators analyze in real time measurements of hardware resource utilization that characterize different subpopulations of servers, containers or applications. For instance, one important priority of data center operators is to ensure that hardware resources are appropriately distributed across different applications and that failures are promptly detected and diagnosed. Let us consider the following examples of real-world outages: In a recent incident, a small group of oversubscribed AWS EC2 servers unexpectedly shut down due to overheating. As a result, an entire AWS availability zone covering the greater Tokyo area, experienced degraded EBS volume performance. The overheating was due to a control system failure that caused multiple, redundant cooling systems to fail in parts of the affected Availability Zone [1]. In a different incident, due a buggy configuration update reduced the number of hosts running the EC2 DNS resolver service. As a result, the remaining healthy instances, experienced above normal utilization and, thus, EC2 network connectivity and DNS resolution in the availability experienced sporadic outages [2].

Similar to the video-QoE usecase, these examples highlight the need of data center operators to ensure fine-grained visibility into different subpopulations of their resources (*i.e.,* servers, containers, applications *etc.*). Many of the aforementioned failures could have been

averted given more efficient multidimensional telemetry infrastructures in place.

**Network flow monitoring** Network management is a multi-faceted and complex operation that involves tasks such as traffic engineering [79, 118], attack and anomaly detection [138] or forensics [156]. To implement each of these tasks, network operators need an accurate and timely analysis of various statistics on different application-level performance metrics (*e.g.,* flow distributions, per-flow packet sizes, latency, *etc.*) across different subpopulations of flows, *i.e.,* network flows grouped across combinations of packet header fields.

Similar use-cases can also be identified in various other domains and applications, such as sensor-based deployments, A/B testing [94, 105], exploratory data analysis [46, 150], operations monitoring [29], sensor-based deployments [162], *etc.* The proliferation of multidimensional data in large-scale infrastructures makes multidimensional telemetry increasingly relevant.

### 2.1.2 Formalizing Multidimensional Telemetry

At a high level, the goal of multidimensional telemetry is to enable the estimation of a broad set of summary statistics simultaneously across multiple subpopulations of data. We observe that, across the board, the multidimensional telemetry problem exhibits three properties:

**Property #1: Multidimensional data.** Analytics are estimated on multidimensional data. We define a multidimensional data record as $x = (d_1, \ldots, d_D, m)$, where $d_i$ is the value of a dimension $D_i$ and $m$ is the value of metric $M$. For instance, in video streaming, QoE metrics of interest might be bitrate or video buffering time whereas dimensions might be the location of the viewer, their video player device, their ISP or CDN. The metrics and dimensions are domain- and use case-specific.

**Property #2: Analytics on data subpopulations.** The desired analytics are estimated in parallel across subpopulations of the input data. A subpopulation $Q_j$ is a collection

of data records $\{x_i\}$ such that all $x_i \in Q_j$ match on a subset of dimension values. With a slight abuse of notation, we formally define $Q_j$ using this set of dimension values, *i.e.,* $Q_j = \{D_{j,1} = d_{j,1} \wedge \cdots \wedge D_{j,l} = d_{j,l}\}$, where $\{D_1, \ldots, D_l\} \subseteq \{D_1, \ldots, D_D\}$. In video streaming, an exampleo of a data subpopulation is NYC viewers on AppleTV.

**Property #3: Multiple statistics of interest.** For each subpopulation, the operator might want to estimate various summary statistics, such as heavy hitters, entropy, cardinality, *etc.* A query $q_k$ specifies a data subpopulation $Q_j$ (or a set thereof), and a statistic $g$ to estimate using the values $m_i$ of $x_i \in Q_j$.

### 2.1.3 Existing Approaches for Multidimensional Telemetry

Despite their applicability across multiple domains, existing multidimensional frameworks suffer from a well-known scalability challenge (discussed in more detail in Chapter 3). More specifically, as data streams become increasingly multidimensional, telemetry frameworks are expected to simultaneously monitor a combinatorial explosion of data subpopulations [46, 84] which results in big, often prohibitive, overheads in terms of operating cost and resources. Below, we look at how state-of-the-art telemetry frameworks attempt to address this challenge. We broadly taxonomize existing frameworks in two categories based on their underlying mechanism to achieve scalability.

**Horizontal resource scaling** In this category we find well-known (SQL and NoSQL) analytics engines whose distributed design enables reducing estimation latency through horizontal scaling of server resources (*e.g.,* Spark [166], Hive [146], Hadoop [141], Dremel [121], Druid [161], Flink [60]). These frameworks can provide *precisely exact* analytics and their clusters scale along with the input data. However, as data volume and dimensionality grow, (1) deploying these clusters becomes increasingly expensive and (2) the continuous addition of resources results in decreasing performance gains due to data shuffling overheads [35].

**Approximate estimates** An orthogonal line of work reduces analytical processing latency through *approximate* analytics estimations. These include both sampling and non-sampling-based approaches. Below we discuss each one of these two broad classes in more detail:

1. *Data Sampling.* These frameworks down-sample the original dataset to reduce response time, with prior work exploring various sampling approaches; *Query-time sampling* frameworks extend SQL through sampling operators enabling the operator to express at query-time sampling semantics on the original dataset (*e.g.,* uniform sampling of entire dataset) [43, 61, 121, 128, 146]. Query-time sampling, while it generally allows programmers to utilize the full semantics of SQL, it does not offer any accuracy guarantees or response time bounds. *Pre-computed sampling* approaches on the other hand, use a pre-processing step to create samples that are used to answer a narrower (but still useful) set queries. While they offer low latency, a key drawback of these approaches is their ability to only offer accuracy estimates (in the form of confidence bounds) a posteriori *i.e.,* after the query has executed. Additionally, creating and maintaining these samples is expensive. BlinkDB, for instance, leverages stratified sampling to build samples only for data partitions that commonly appear in aggregation queries [30, 31, 35, 142]. Therefore, pre-computed sampling approaches are often limited to only a narrow set of queries, thus lacking generality. In addition, given that sampling approaches cannot guarantee accuracy, this results resulting in low user trust [66, 107].
2. *Online Data Aggregations (OLA).* This class of works propose the idea of providing approximate answers which are constantly refined during query execution. It provides users with interfaces enabling users to stop the query once they are satisfied with the resulting accuracy. However, as streaming data comes in random order, providing apriori accuracy guarantees while maintaining performance constraints is infeasible [63, 69, 90, 93].
3. *Data Summaries.* There exists a great deal of work on developing custom data synopses (*e.g.,* sketches, histograms, data cubes) that queries are applied on. *Sketches*

in particular are data summaries offering sound apriori accuracy guarantees and are natural fits for streaming data applications as they only need to make one pass over the incoming data stream. However, sketches also present several disadvantages that raise questions regarding their ability to meet the requirements of a multidimensional analytics engine; First, sketches are, in general, tailor made for a specific type of aggregation (entropy, quantiles *etc.*) [71, 72, 76, 118, 165]. Additionally, common practice in prior work [84] suggests that there needs to be a sketch per subpopulation. Intuitively, as the number of possible dimensions in the dataset increases, the resulting combinatorial explosion of groups makes the task of allocating a sketch per statistic and subpopulation infeasible.

Nevertheless, this classification is not strict and it is not uncommon to encounter frameworks that combine the above capabilities. For instance, both Apache Spark and Druid allow for data summarization at ingestion time such that incoming data are stored as a key-value store where the keys are distinct $(Q_j,\ m_i)$ tuples and the values are their respective counts. These hybrid approaches enable data reduction without compromising the frameworks ability to offer precise estimations.

## 2.2 Detection and Diagnosis of Viewership Anomalies

### 2.2.1 Formalizing Viewership Analytics

**Viewership** We define viewership of a subpopulation of viewers (*i.e.,* a viewership group) as the timeseries of the count of active viewers for that subpopulation. Figure 2.1 illustrates the timeseries of daily viewership for a particular viewership group across four consecutive weeks.

**Viewership anomalies** We define as a viewership anomaly, broadly, as any unexpected change in the timeseries of viewership (*e.g.,* sudden rise or drop) that warrants technical intervention in order to be mitigated. For instance, Figure 2.1 illustrates an example of a

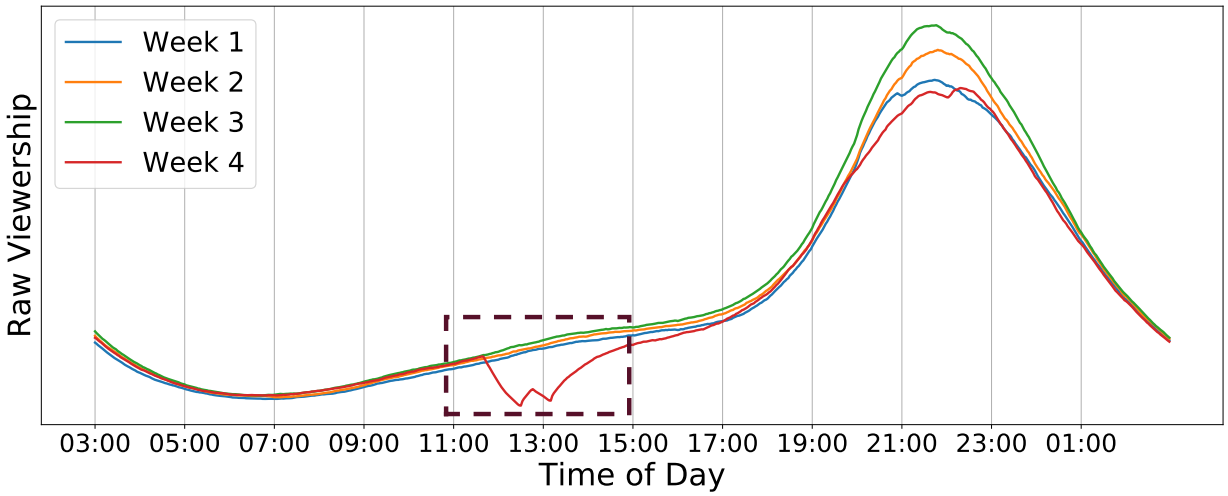



Figure 2.1: Motivating example of viewership anomaly. Real viewership values not shown.

large scale incident that manifested itself as a multi-hour long viewership drop. The figure depicts the timeseries of viewership for the same weekday across 4 consecutive weeks. As we can observe, there is a visible and sharp viewership drop in Week 4. The incident affected viewers of a major US-based content provider on RokuTV and was attributed to platform malfunction. During that period, no QoE alerts were raised and the issue went undetected. Viewership was the only indicator of the presence of this technical disruption.

**Operator requirements** Our discussions with analysts at a video analytics firm indicated that video operators have several requirements from an viewership analytics framework. We discuss those below:

1. *Automated detection workflows.* Given the large number of subpopulations that the alerting component needs to monitor in parallel, operators need automatic detection workflows. More specifically, operators ideally need a framework that minimizes the manual effort needed per-subpopulation while ensuring accurate anomaly detection.
2. *Concise alerts to minimize alert fatigue.* As mentioned previously, one single logical event might manifest as multiple redundant anomalies across subpopulations. To avoid releasing multiple, often redundant, alerts to the operator, the alerting workflow should be able needs to scope the anomalous incidents detected to the subpopulation

the event originates from. This will minimize alert fatigue for the operator and ideally ensure that the finite number of human analyst cycles will be optimally utilized.

3. *Interactive alerting latencies.* Ideally, operators need to be issued alerts as soon as the incident of interest occurs. Therefore, the alerting workflow needs to operate at interactive timescales.

### 2.2.2 Existing Approaches for Detecting and Diagnosing Viewership Anomalies

To the best of our knowledge, we are the first to formally investigate methods for anomaly detection and diagnosis in the context of video viewership. Nevertheless, below we provide a broad taxonomy of timeseries anomaly detection techniques as well as an overview of prior works focusing on root-cause attribution.

**Detection** There is a plethora of candidate approaches for timeseries anomaly detection. We broadly taxonomize these approaches in two categories:

1. *Model-based approaches.* These approaches leverage persistent statistical properties of the timeseries in order to build a model that captures them and uses them as a predictor of future timeseries values. There is a variety of model-based approaches for timeseries anomaly detection, including classical approaches such as exponential smoothing [64], autoregressive models [167], STL decomposition [51, 68, 145] *etc.*

2. *Model-free Approaches.* Model-free approaches opt for the more-lightweight strategy of relying on short history windows in order to detect anomalies. Similar to model-based approaches, this category contains a variety of well-known approaches, such as control charts [155], EWMA [168], CUSUM [123] or change-point detection [41].

**Diagnosis** Previous research has also focused on extensively analyzing root-causing in systems and networks. In particular, a well-established approach to diagnosis (which we also employ within this dissertation) involves building libraries of anomaly signatures. Nev-

ertheless, prior works on diagnosis that broadly leverage signatures are generally domain- and usecase-specific [33, 45, 50, 53, 106, 110, 131].

As we discuss extensively in Chapter 4, in our work we opt for model-driven anomaly detection and diagnosis based on anomaly signatures. However, as we will see, viewership exhibits several properties that prevent the use of well-known prior efforts.

# Chapter 3

# Enabling Efficient and General Subpopulation Analytics in Multidimensional Data Streams with HYDRA

Large-scale infrastructures across various domains (*e.g.,* Internet services, sensor farms, operations monitoring) produce data streams that are growing both in data volume and dimensionality [15, 44, 46, 134]. Increasingly, these data streams are *multidimensional*; *i.e.,* they contain measurements of metrics along with complex metadata that describe said measurements across domain-specific dimensions. For instance, video streaming services analyze measurements of viewer quality of experience or viewership in order to detect issues that might impact viewer engagement across dimensions, such as ISP, CDN, Device, City, *etc.* [102, 104]. Similarly in other domains such as network and data center monitoring, we see similar trends [1, 2, 118].

In these settings, analysts need *interactive* and *accurate* estimates of diverse summary statistics across *multiple data subpopulations*. For instance in video streaming, we need to monitor different statistics (*e.g.,* cardinality, histogram, entropy) of viewer quality metrics

(*e.g.,* bitrate, buffering time) across subpopulations of viewers (*e.g.,* viewers in NYC on Comcast, *etc.*) [104]. Similarly, in network monitoring, operators need to analyze network flows grouped by combinations of their 5-tuple (srcIP, dstIP, srcPort, dstPort, protocol) [118].

Providing interactive and accurate estimates across multiple summary statistics and subpopulations is challenging on two fronts. First, the high data dimensionality results in a combinatorial explosion of subpopulations to monitor and in prohibitive ingestion- and query-time overheads. Second, the multiple summary statistics induce compute and/or memory overheads, proportional to the number of statistics of interest. In this respect, existing frameworks are fundamentally limited in terms of the tradeoff across accuracy, scalability (across subpopulations), and generality (across summary statistics they can estimate). Exact analytics frameworks (*e.g.,* Spark [166], Hive [146], Druid [161]) that rely on horizontal resource scaling entail poor cost-performance tradeoff as datasets become larger. While approximate analytics [66] (*e.g.,* sampling- or sketch-based analytics) can trade off estimation accuracy for lower cost and improved interactivity, these too suffer undesirable tradeoffs. For instance, sampling-based approaches provide generality across metrics and can handle many subpopulations, but cannot offer accuracy guarantees. On the other hand, sketch-based analytics (*e.g.,* [30, 34, 67, 84, 85, 147, 148, 165]) can offer robust accuracy guarantees, but cannot address the combinatorial explosion of data subpopulations and also incur per-statistic effort.

In this chapter, we present Hydra, a novel sketch-based analytics framework for efficient and general analytics over multidimensional data streams. Hydra is based on two key ideas:

1. To tackle the combinatorial explosion of subpopulations, we design a "sketch of sketches" primitive that enables efficient data stream summarization. This enables a reduction in the framework's ingestion-time memory cost of one to two orders of magnitude compared to Spark- and Druid-based alternatives while at the same time offering robust and provable accuracy guarantees.

2. To provide high-fidelity estimations for a general set of summary statistics, we leverage the power of universal sketching [118]. Unlike canonical sketch-based approaches that deploy one custom sketch type per statistic [84, 147, 148], a universal sketch allows for estimating multiple different summary statistics with only one sketching structure using polylogarithmic space.

We analytically prove that combining the above sketch-of-sketches idea with universal sketches ensures polylogarithmic space complexity to the data subpopulations for a given target accuracy. This analysis also provides practical strategies for configuring HYDRA.

We prototype HYDRA on top of Apache Spark. Our choice of framework was based on the ease of implementation as, in practice, HYDRA's design is not tied to any existing platform. We implement several practical system optimizations to further improve HYDRA's empirical accuracy and runtime.

We evaluate HYDRA using two real-world datasets; (1) a 2h-long CAIDA traces from the equinix-NYC vantage point, collected in January 2019 [3, 4] and (2) a real-world trace of video QoE from a video analytics provider capturing the perceived QoE of viewers of a US-based content provider [7].[1] To further evaluate the sensitivity of HYDRA-sketch, we also leverage a synthetic multidimensional dataset drawn from a Zipf (Pareto) distribution with different parameter values [85, 148].

We compare HYDRA against 5 baselines: A native Spark-SQL implementation for precisely exact analytics, a Spark-based sampling implementation that uniformly samples incoming data, a sketch-based approach that allocates one universal sketch instance per data subpopulation and two key-value based implementations (on Apache Spark and Apache Druid) that pre-aggregate data at ingestion time and provide precisely accurate analytics . Our

[1] **Ethical Considerations**: All video QoE data used in this work are covered by NDAs prohibiting any re-sharing with 3rd parties even for research purposes. Further, video QoE data have been reviewed and validated by the operator with respect to GPDR compliance (*e.g.,* no identifier can be associated to person), and data processing only looks at QoE measurements per combination of session dimension values. No personal and/or contact information was available in the data used for this study. The data were properly anonymized and no identifying/personal data were used during our analysis.

evaluation shows that:

1. HYDRA offers robust accuracy guarantees (error <5% with 90% probability), comparable to those of exact analytics frameworks at 1/10 of their operational cost or less.
2. HYDRA's memory footprint scales sub-linearly with dataset size and number of data subpopulations, resulting in 7-20× query latency improvement compared to Spark- and Druid-based alternatives.
3. HYDRA's configuration heuristics ensure close to optimal accuracy-memory tradeoff and our performance optimizations improve end-to-end runtime by 30%.

## 3.1 Evaluating the Limitations of Prior Work

In the previous chapters, we provided background on multidimensional telemetry and hinted at the scalability challenge existing frameworks face. To empirically evaluate this claim, we analyze the cost and performance of existing works (discussed in Chapter 2) during *Data Ingestion* and *Query Estimation*. Specifically, for ingestion, we look (1) at the computational overhead of processing multidimensional streams and (2) at the data resident cost after ingestion. For estimation, we look at the additional CPU and memory complexity per query for a given accuracy target (when setting one is possible).

| Notation | Definition |
|---|---|
| $V$ | Dataset Size |
| $Q$ | Number of data subpopulations |
| $D$ | Number of Data Dimensions |
| $C$ | Cardinality of each data dimension |

Table 3.1: Notation for analysis of multidimensional analytics problem

As shown in Table 3.1, let us denote the dataset size (in terms of number of data records) as $V$ and let $Q$ be the number of data subpopulations. Suppose the number of dimen-

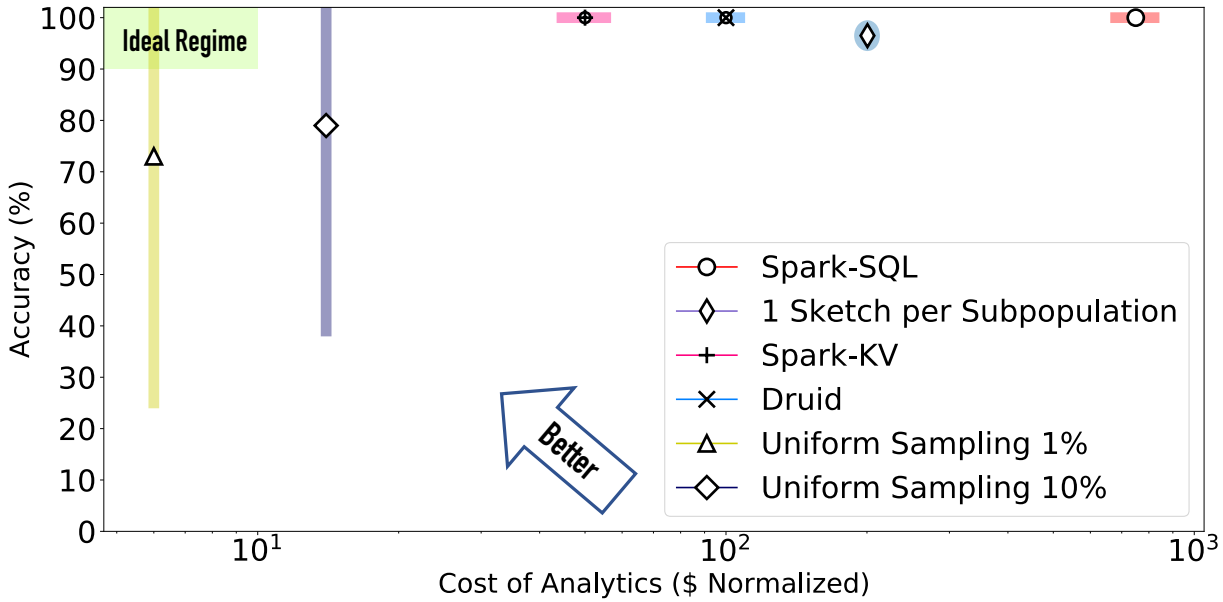



Figure 3.1: Cost-accuracy analysis for existing analytics systems.

sions is $D$ and each dimension has cardinality $C$. Therefore, a multidimensional data record belongs in $2^D$ different subpopulations and there are $\mathcal{O}(C^D)$ subpopulations in the dataset. In practice, we find that $\mathcal{O}(2^D \times V)$ is a tighter empirical bound for $Q$ and we will use that moving forward. In practice, each multidimensional data record belongs in $2^D$ subpopulations, and in our datasets we do not generally see all possible values per dimension. Finally, let us assume that the framework needs to estimate $\mathcal{O}(S)$ different statistics. Therefore, the total number of summary statistics queries expected to be estimated is $\mathcal{O}(Q \times S) = \mathcal{O}(2^D \times V \times S)$.

We generally observe that ingestion incurs a memory and CPU cost of $\mathcal{O}(Q)$ or $\mathcal{O}(V)$, depending on whether the framework pre-aggregates data (*e.g.,* Apache Druid using the data roll-up feature [161]). Similarly, the estimation time overheads can be as high as $\mathcal{O}(Q \times S)$ or $\mathcal{O}(V \times S)$ respectively. In summary, we observe that existing frameworks incur significant memory and compute overheads either at ingestion time or at estimation time, due to their linear scaling with respect to $S$ and/or exponential scaling with respect to $D$.

**Empirical analysis** To empirically corroborate this analytical intuition, we evaluate the cost-accuracy tradeoff for several analytics frameworks when used in a multidimensional context (Figure 3.1). We quantify frameworks' accuracy as a function of their operational cost when asked to estimate in real-time 4 summary statistics from a 120GB real-world dataset with approximately 5.6 million data subpopulations. Following the typical cloud billing model [20], we use the total runtime times the number of cluster nodes used (20) as a proxy for the $ cost. We provide a detailed description of our experimental setup and baselines in §3.5. Ideally, we need a framework whose cost-accuracy tradeoff lies in the shaded green region, *i.e.,* it offers the accuracy of a precise analytics frameworks at the cost of sampling. However, we observe that the cost gap between the cheapest (1% uniform sampling) and the most expensive baselines (precisely accurate Spark-SQL) is two orders of magnitude wide. Furthermore, a sketch-based approach where the framework allocates one sketch per subpopulation, while cheaper than Spark-SQL, remains expensive as it allocates exponentially many sketch instances, thus incurring high memory overheads.As discussed in §3.5, this baseline uses universal sketching *i.e.,* a sketch type that can simultaneously estimate all 4 statistics of interest per subpopulation with one sketch. Last, precise baselines that summarize data at ingestion time, such as Apache Druid and Spark (denoted as Spark-KV) lie in the middle between Spark-SQL and sampling.

**Key takeaway** Meeting the requirements for multidimensional analytics is challenging due to both the overheads incurred by the combinatorial explosion in data subpopulations and the number of summary statistics the framework needs to simultaneously monitor. In practice, we see that existing frameworks cannot strike a reasonable balance between operators' requirements for coverage across subpopulations, high-fidelity general analytics and real-time estimations. This motivates us to rethink how we can support such analytics workloads at scale.

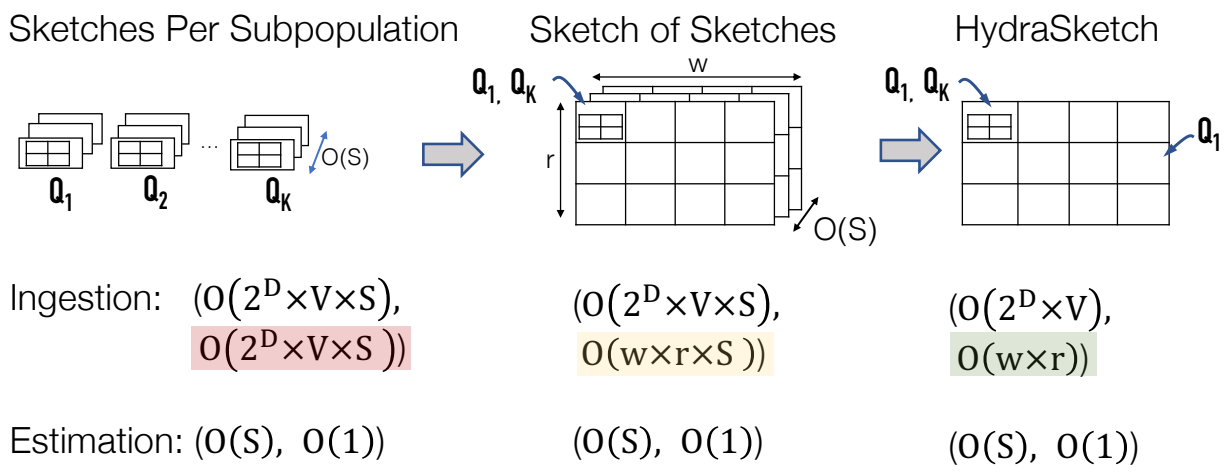



Figure 3.2: Comparison of ingestion and estimation cost (CPU time, memory) for different sketch-based analytics designs.

## 3.2 HYDRA: System Overview

Ideally, we want to strike an optimal balance between cost and accuracy guarantees, while supporting many statistics over all subpopulations. In this section, we give a high-level view of HYDRA and of the key design contributions to reduce these overheads without compromising accuracy.

### 3.2.1 Key Ideas

To enable strong accuracy guarantees, we take a sketch-based approach at scale. However, as we saw, allocating $\mathcal{O}(S)$ sketch instances per data subpopulation cannot address the combinatorial explosion of subpopulations and the large number of required summary statistics. To avoid these overheads, we introduce two key ideas, as illustrated in Figure 3.2. Note that the figure highlights the theoretical improvements in space complexity from HYDRA's design ideas.

**Idea 1: Controlling the combinatorial explosion of subpopulations with a "sketch of sketches".** Our first idea is to reduce the prohibitively large $\mathcal{O}(Q) = \mathcal{O}(2^D \times V)$ ingestion-

time memory overhead of sketch-based telemetry through a novel "sketch of sketches". We show that by maintaining a $w \times r$ array of sketch instances (shown in Fig. 3.2), where $w \times r \ll 2^D \times V$, HYDRA reduces the memory cost of estimating $\mathcal{O}(S)$ statistics from $\mathcal{O}(2^D \times V \times S)$ to $\mathcal{O}(w \times r \times S)$. The intuition is that unlike canonical sketch-based approaches that maintain one sketch instance per subpopulation [84], we can multiplex multiple subpopulations into one sketch instance and then query that instance with predictable estimation error.

**Idea 2: Combining a sketch of sketches with universal sketching to estimate multiple summary statistics.** To reduce the need for instantiating sketches of different types to cover each of the $\mathcal{O}(S)$ summary statistics, HYDRA leverages the power of *universal sketching* [58, 118]. Universal sketching allows us to replace $\mathcal{O}(S)$ sketches with a single sketch algorithm, which enables the simultaneous estimation of multiple different statistics per subpopulation. This design choice further reduces the framework's space complexity from $\mathcal{O}(w \times r \times S)$ to $\mathcal{O}(w \times r)$.

### 3.2.2 HYDRA's Workflow

To support multidimensional workloads at scale, we envision HYDRA as a distributed framework consisting of one frontend and multiple worker nodes. Its input are 1) streams of batched multi-dimensional data, ingested in parallel at the worker nodes and 2) estimation queries provided by the operator to the frontend node. Data ingestion happens at the worker nodes, whereas the frontend node is responsible for configuration dissemination, sketch merging and statistic estimations. Each worker node summarizes its incoming data stream in one local instance of HYDRA-sketch which is configured to ensure specific accuracy guarantees and memory footprint (§3.3.6). These sketches are then collected at the frontend node, merged into one unified HYDRA-sketch instance for answering queries. Data is ingested in batches or *epochs*. HYDRA executes two logical operations per epoch; (1) Data Ingestion, and (2) Query Estimation.

1. **Data Ingestion:** HYDRA compactly summarizes information per data subpopula-

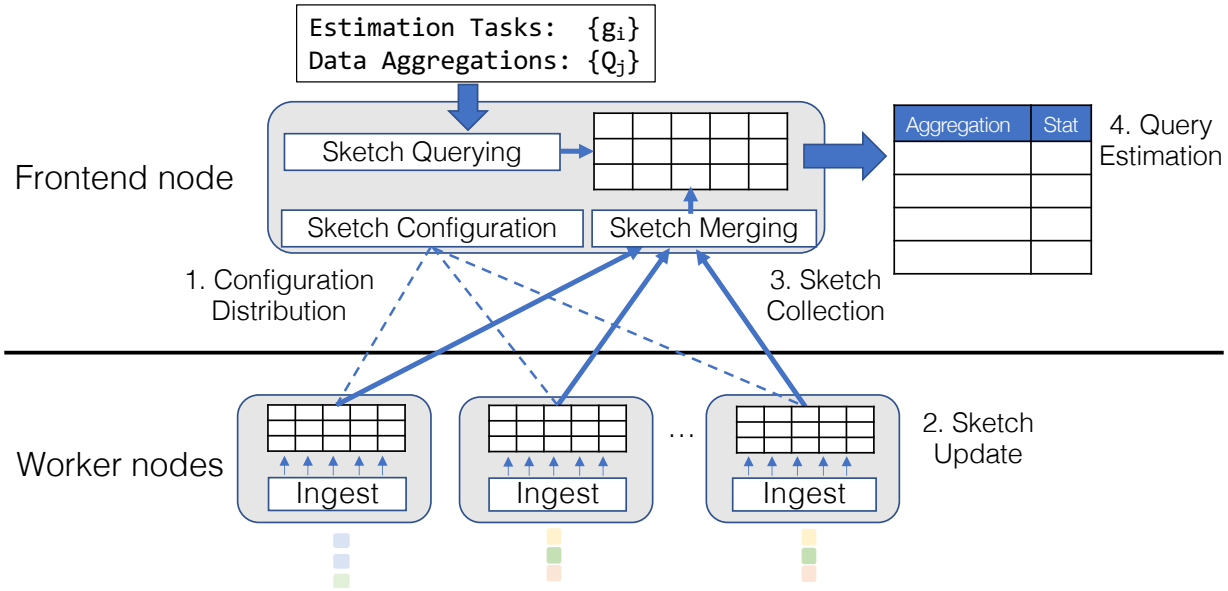



Figure 3.3: HYDRA's example workflow.

tion during data ingestion in HYDRA-sketch instances. For every incoming data item, HYDRA first identifies what subpopulations the data point belongs in and correspondingly updates HYDRA-sketch.

2. **Query Estimation:** Query estimation runs at the frontend node which needs a global view over the data stored at the worker nodes. At the end of the epoch, the frontend node periodically collects all HYDRA-sketch instances from the worker nodes and merges them into one HYDRA-sketch instance which is then used to estimate the desired statistics. Queries can either be one-time or continuous (*i.e.,* repeated every epoch).

## 3.3 HYDRA: Detailed Design

In this section, we focus on HYDRA-sketch, the sketching primitive that reduces HYDRA's space complexity to sub-linear in the number of subpopulations and makes it independent to the number of statistics. We first provide background on sketching to set up the intuition for HYDRA-sketch. We then introduce the basic HYDRA-sketch algorithm, formally prove

| Notation | Definition |
|---|---|
| $w$ | Number of sketches per 2D-sketch row |
| $r$ | Number of rows in 2D-sketch |
| $(\epsilon, \delta)$ | $0 < \epsilon < 1$ as additive error and $\delta$ is the probability that the result error is not bounded by $\epsilon$ (failure probability) |
| $w_{US}$ | Number of counters per universal sketch row |
| $r_{US}$ | Number of universal sketch rows |
| $(\epsilon_{US}, \delta_{US})$ | $0 < \epsilon_{US} < 1$ as additive error in universal sketch and $\delta_{US}$ is the failure probability |
| $L$ | Number of universal sketch layers |
| $k$ | Number of keys in universal sketch heavy hitter heaps |

Table 3.2: HYDRA notation. The upper subsection of the table introduces notation specific to the sketch-of-sketches and the lower on universal sketches.

its error bounds, and devise HYDRA-sketch configuration strategies. Table 3.2 summarizes the notation we use in this section.

### 3.3.1 Background on Sketching

Let $\mathcal{S}_{m,n}$ denote a data stream with length $m$ and $n$ distinct keys. Suppose we want to estimate a summary statistic based on the frequencies of the keys (*e.g.,* entropy, cardinality, frequency moments, *etc.*). The natural design is to maintain a key-value counter data structure, where we track the frequency per key and, later, use this structure to estimate the desired statistic. For instance, for frequency estimation, we maintain and increment one counter per key. While correct, this approach's space complexity is linear in $n$ and, thus, not space efficient (Figure 3.4).

**Hash-based mappings for space efficiency** To ensure sub-linear (in $n$) space complexity, sketching algorithms do not maintain per-key state but, instead, map more than one keys in

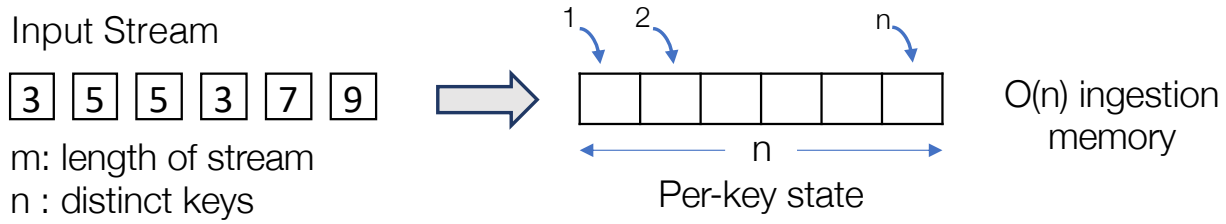



Figure 3.4: Maintaining per-key state is not space efficient

the same counters through hashing. For instance, a simple sketch for frequency estimation consists of $w$ integer counters, where $w \ll n$. Based on the hash of the key, an element gets mapped to a counter, which is then incremented to maintain an estimate of that key's frequency. Naturally, *colliding* multiple keys in the same counters introduces some estimation error (Figure 3.5).

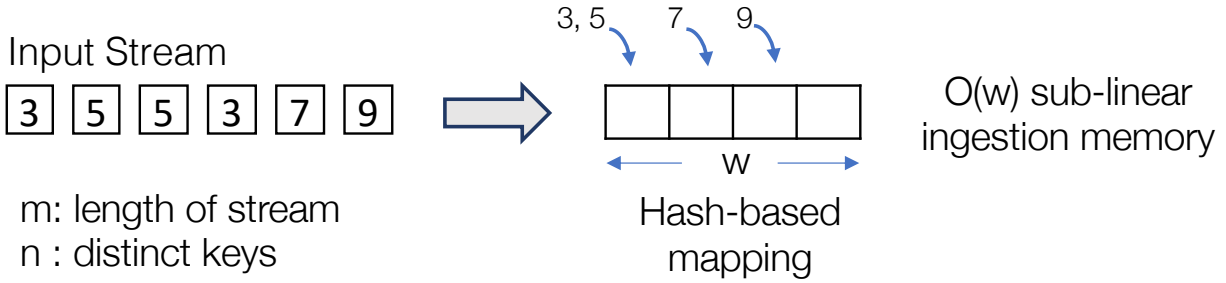



Figure 3.5: Hashing enables sub-linear memory complexity

**Multiple independent updates for tighter error bounds** As defined, this basic mechanism offers weak accuracy guarantees, *i.e.,* it only provides a small probability that the estimation error will lie within a desirable range of error values [40]. To overcome this, sketches use independent instances (*e.g.,* $r$ arrays) of the counter structure of length $w$. Each vector of length $w$ has its own hash function and the $w$ hash functions are pairwise independent. Thus, ingesting a stream element now translates to $r$ update operations (*e.g.,* incrementing $r$ integer counters instead of one). For each key, this sketch produces $r$ different estimates of the statistic of interest. Depending on the algorithm the final estimate will be some function summarizing $r$ estimates (*i.e.,* min, median *etc.*) (Figure 3.6) [72].

This amplifies the probability that the estimation error lies within the desired range.

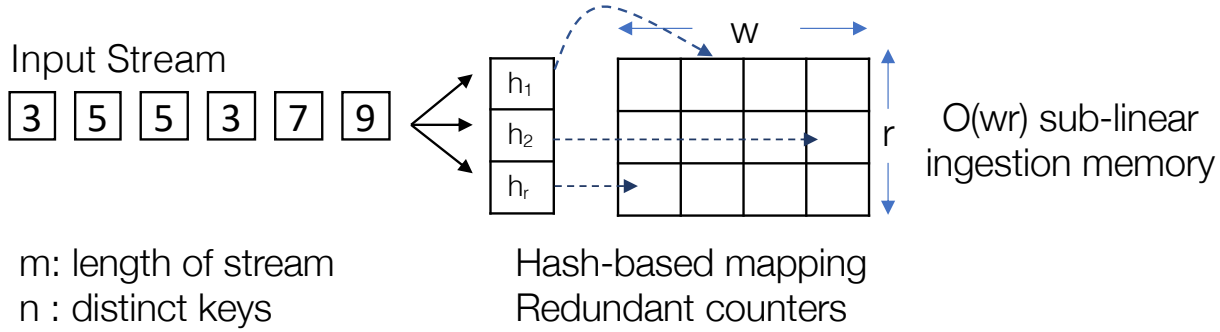



Figure 3.6: Independent hashing and counters improve accuracy guarantees

### 3.3.2 Tackling the Combinatorial Explosion of Data Subpopulations

For now, let us make the simplifying assumption (which we relax later) that our system only needs to estimate one summary statistic (*e.g.,* entropy) per data subpopulation.

Similar to Figure 3.4, a starting point for our design would be to maintain per-subpopulation state, *i.e.,* allocate one sketch instance for each of the $\mathcal{O}(2^D \times V)$ distinct subpopulations. However, this approach is not scalable as it requires as many sketches as the exponentially many possible data subpopulations.

To avoid keeping per-subpopulation state, we borrow from the first intuition that we saw in the sketch construction in the background (fig. 3.5). The basic sketch construction avoids maintaining per-key state by allowing multiple keys to explicitly collide in a hashed key-value store whose size is less than the number of unique elements.

Extending this intuition to our setting is a bit tricky; the basic sketch is maintaining a single counter per array entry but we want to able to estimate some statistical summary of an subpopulation instead. Instead of keeping a single counter per array entry, we maintain a sketch-per-entry. This brings us to the following construction (Figure 3.7). We consider a single array of $w$ (*e.g.,* $w \ll 2^D \times V$) *sketches*. For each $(Q_j,\ m_i)$ pair, we hash the $Q_j$ and map it to one of the $w$ sketches, thus colliding multiple subpopulations to the same

sketch. Then, we update the sketch with $m_i$ and at query time, we estimate the statistic for $Q_j$.

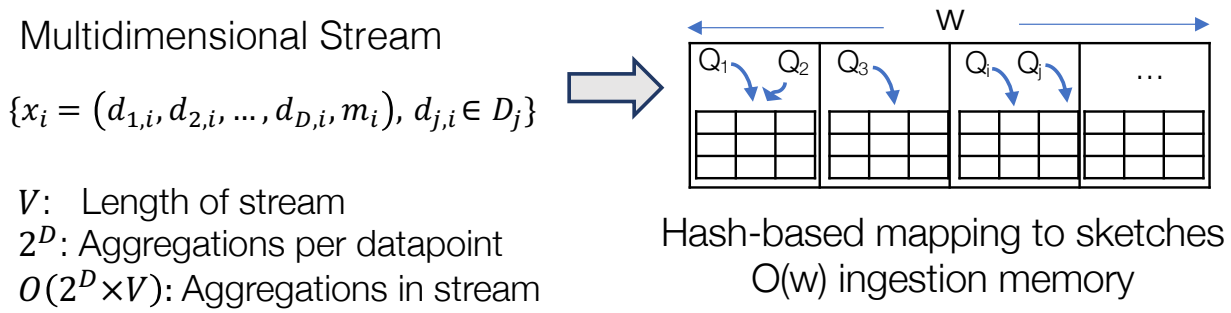



Figure 3.7: Hash-based mapping of subpopulations to a vector of sketches.

Analogous to the basic sketch from the background, by mapping multiple subpopulations to one sketch this baseline construction will have some estimation error. To control this, we extend the idea of using redundant counter vectors and pairwise-independent hashes shown in Figure 3.6. That is, we use *r arrays of w sketches* and use $r$ pairwise-independent hash functions to map each subpopulation to one sketch per row (Figure 3.8). At query time, we return the median of the $r$ estimates.

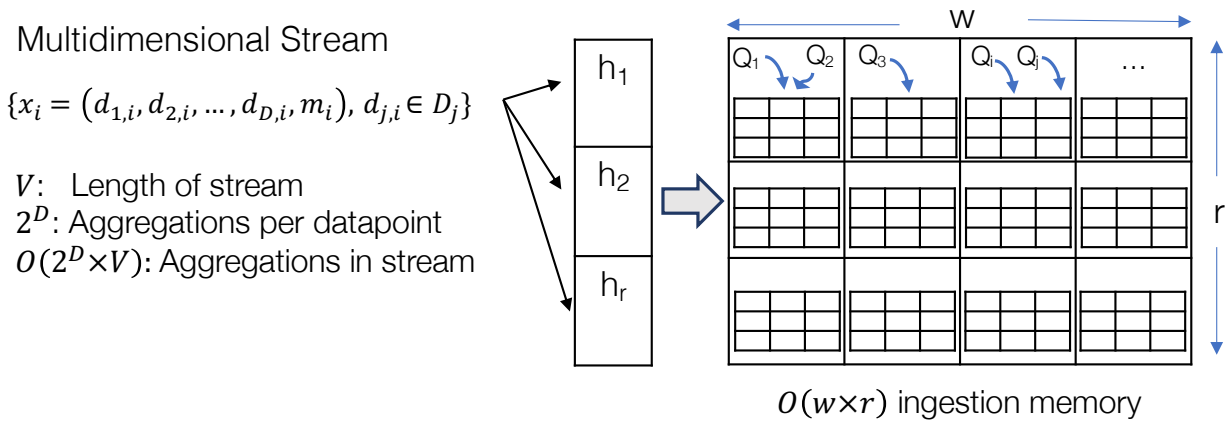



Figure 3.8: Redundant sketch vectors and pairwise-independent hashes for tighter error bounds.

In summary, we see that this analogous to the 2D array of counters in a simple sketch, our sketch-of-sketch construction maintains a 2D array of sketches to track multiple subpopu-

lation. This approach can reduce the memory cost of ingestion to $O(w \times r)$ *i.e.,* sub-linear in subpopulations. In §3.3.5, we formally prove theoretically rigorous memory-accuracy tradeoffs for this construction.

### 3.3.3 Ensuring Generality Across Statistics

The above discussion is based on the simplifying assumption that we need to only estimate one summary statistic. Sketching algorithms are generally designed to estimate one (or few) statistical properties of the data stream. Thus, to support $O(S)$ different summary statistics, we need to create $O(S)$ sketch-of-sketches instances. This raises two natural concerns. First, the total memory cost of this solution becomes $O(w \times r \times S)$, *i.e.,* linear to the number of summary statistics of interest. Second, the framework cannot offer generality as it cannot estimate summary statistics that are not already allocated; *e.g.,* some future analysis might require estimating the entropy of a metric but the framework has not instantiated an entropy-specific sketch-of-sketch instance.

Our insight is that the sketch of sketches structure can be combined with universal sketching [118] to achieve the desired generality across summary statistics. A universal sketch is a sketching primitive that enables the simultaneous estimation of multiple different, apriori unknown, statistics with one sketch instance. To ensure generality across statistics, instead of deploying a sketch-of-sketches per statistic, we can deploy only one sketch of universal sketches. We formally demonstrate the feasibility of this insight in §3.3.5 and, thus, show that HYDRA's ingestion cost drops to $\mathcal{O}(w \times r)$.

**Background on universal sketches** A universal sketch can estimate any summary statistic that belongs to a broad class of functions, known as *Stream-PolyLog* [57, 58, 118]. We denote each function in *Stream-PolyLog*, as $G$-sum $= \sum g(f_i)$, where $f_i$ is the frequency of the $i$-th unique element in the input stream $\mathcal{S}$ and $g$ is a function defined over $f_i$. If $g$ is monotonically increasing and upper bounded by $\mathcal{O}({f_i}^2)$, then $G$-sum can be computed by a single universal sketch with polylogarithmic memory.

The basic building block of universal sketches are L2-Heavy Hitter (HH) sketches *e.g.,* Count-sketch [122]. Each count-sketch maintains $r_{CS}$ arrays of $w_{CS}$ counters each, $r_{CS}$ pairwise-independent hash functions and a max-heap keeping track of the top-$k$ Heavy Hitters in the sketch; When updating each count-sketch with a new data item, the sketch updates a randomly located counter in every row based on the corresponding hash index to keep track of that data item's frequency. The top-$k$ HH heap is subsequently updated to reflect the addition of the new item. A universal sketch consists of $L$ layers of count-sketches. Each count sketch applies an independent 0-1 hash function $h_{l \in [0,\ L)}$ to the input data stream to sub-sample at every layer (from the previous layer). These layers then track the heavy hitters, *i.e.,* the important contributors to the $G$-sum.

The intuition here is that the layered structure of universal sketch is designed for sampling representative elements with diverse frequencies and these elements can be used to estimate $G$-sum with bounded errors. If only one layer of heavy hitter sketch were used, the estimations would lack representatives from less frequent elements. The heavy-hitters at each layer are processed iteratively from the bottom layer to the top and the recursively aggregated result is used to compute the desired statistic. This is an unbiased estimator of $G$-sum with bounded additive errors (Theorem 1).

**Theorem 1** ([58, 118])**.** Given a stream $\mathcal{S}$ of length $m$ with $n$ distinct keys, let us consider a Universal Sketch $US$ with $L = \mathcal{O}(\log n)$ layers. If each layer of $US$ provides an $(\epsilon_{US},\ \delta_{US})$-L2 error guarantee, then $US$ can estimate any $G$-sum function $G \in$ Stream-Polylog to within a $(1 \pm \epsilon_{US})$ factor with probability $1 - \delta_{US}$. Satisfying a $(\epsilon_{US},\ \delta_{US})$-L2 error guarantee requires $O(logn)$ Count-Sketch instances with $w_{CS} = \mathcal{O}(\epsilon_{US}^{-2})$ columns and $r_{CS} = \mathcal{O}(\log \delta_{US}^{-1})$ rows.

### 3.3.4 The HYDRA-sketch Algorithm

Combining these ideas gives us the HYDRA-sketch algorithm as shown in Algorithm 1.

1. **Updating** HYDRA**-sketch:** Updating HYDRA-sketch with a data record,

$$x_i = \langle d_{1,i},\ d_{2,i},\ \ldots, d_{D,i},,\ m_i \rangle$$

is a three-step process. At the first, "fan-out" stage, we compute the $O(2^D)$ aggregations $\{Q_1, \ldots, Q_{2^D}\}$ that $x_i$ belongs in. Then, we map each $Q_j$ to $r$ universal sketches instances using $r$ pairwise-independent hash functions $h_{k\in[0,r)} : Q_j \to [0, w)$. For the $k^{th}$ row, the index of the universal sketch to update $US_k$ is the hash of $Q_j$ using hash function $h_k$. Last, we update each $US_k$ with the metric value $m_i$.

2. **Querying** HYDRA**-sketch:** The querying algorithm for HYDRA-sketch takes as input a statistic $g$ and an aggregation $Q_j$ *i.e.,* the aggregation to estimate $g$ on. Querying consists of 2 steps. The first involves identifying the set of $r$ universal sketch instances $\{US_k\}$ that $Q_j$ maps to. Then $g$ is estimated from each $US_k$, and the median value of these estimations is returned.

### 3.3.5 Accuracy Guarantees

Theorem 2 formally states the accuracy bounds of HYDRA-sketch.

**Theorem 2.** Let us assume that each Universal Sketch $US$ can approximate the $G$-sum, for a monotone function $g$ within a $(1 + \epsilon_{US})$-factor with probability $1 - \delta_{US} > 1/2$. Further, let $G(S)$ be the $G$-sum when applied to the entire stream $S$ and $G_i$ when applied to the target subpopulation $Q_i$. Then HYDRA-sketch with $w = \mathcal{O}(\epsilon^{-1})$ columns and $r = \mathcal{O}(\log \delta^{-1})$ rows, for user defined parameters $\epsilon,\ \delta$, provides an estimate $\widehat{G_i}$ that with probability $1 - \delta$ satisfies:

$$G_i(1 - \epsilon_{US}) \leq \widehat{G_i} \leq G_i(1 + \epsilon_{US}) + \epsilon \cdot G(S) \tag{3.1}$$

*Proof.* To bound the error of our algorithm, we analyze the frequency vector $f_j$ of the stream of elements mapped to each Universal Sketch instance $US_j = h_j(Q_i)$, where $Q_i$

**Algorithm 1:** HYDRA-sketch Algorithm

```
1  Generate r pairwise independent hash functions: h_1 ... h_r : [n] → [0, w)
2  function UPDATE(Data point ⟨d_1, d_2, ..., d_D, m_i⟩, HYDRA-sketch HS)
3      r ← HS.r; w ← HS.w; s ← HS.sketch
4      {Q_1, ..., Q_{2^D}} ← FANOUT(⟨ d_1, d_2, ..., d_D⟩)
5      for each Q_j do
6          for k ← 0 to r − 1, in parallel do
7              US_k ← s[k][h_k(Q_j)]
8              US_k.UPDATE(m_i)

9  function QUERY(HS, Q_j, g)
10     r ← HS.r; w ← HS.w; s ← HS.sketch
11     estimates ← [ ]
12     for k ← 0 to r − 1, in parallel do
13         US_k ← s[k][h_k(Q_j)]
14         estimates[k] ← US_k.ESTIMATE(Q_j, g)
15     final ← MEDIAN(estimates)
16     return final
```

is the queried subpopulation. The frequencies of all $m_i \in Q_i$ are guaranteed to appear in $f_j$, since the UPDATE algorithm of §3.3.4 maps them to $US_j$.

Let $\mathfrak{Q} = \{Q_1, \ldots\}$ denote all groups in the input stream $\mathcal{S}$, and let

$$\mathfrak{Q}_j = \{Q_k \in \mathfrak{Q} \mid h_j(Q_k) = h_j(Q_i)\}$$

denote the set of groups mapped to $US_j$. That is, $G(\mathcal{S}) = \sum_{Q_k \in \mathfrak{Q}} \sum_{m_k \in Q_k} g(f_{m_k})$.

The target quantity, which we wish to estimate, is $G_i \coloneqq \sum_{x \in Q_i} g(f_m)$, *i.e.,* the $g$-sum of

the group $Q_i$, while the $US_j$ processes all groups in $\mathfrak{Q}_j$ and thus approximates

$$\sum_{Q_k \in \mathfrak{Q}_j} \sum_{m_k \in Q_k} g(f_{m_k}) = G_i + \sum_{Q_k \in \mathfrak{Q}_j \setminus \{Q_i\}} \sum_{m_k \in Q_k} g(f_{m_k}).$$

For all $j \in \{0, \ldots, r-1\}$, denote by $\widehat{G_{i,j}}$ the estimate of $US_j$, and denote the *noise* added by the other groups as

$$N_j = \sum_{Q_k \in \mathfrak{Q}_j \setminus \{Q_i\}} \sum_{m_k \in Q_k} g(f_{m_k}).$$

Notice that, since any group in $\mathfrak{Q} \setminus \{Q_i\}$ has a probability of $1/w$ of being in $\mathfrak{Q}_j$, its expectation satisfies that:

$$E[N_j] = \frac{\sum_{Q_k \in \mathfrak{Q} \setminus \{Q_i\}} \sum_{m_k \in Q_k} g(f_{m_k})}{w} \leq \frac{G(\mathcal{S})}{w}.$$

Therefore, according to Markov's inequality, for any $c \in R^+$, $\Pr[N_j \geq c \cdot \frac{G(\mathcal{S})}{w}] \leq 1/c$. Next, by the correctness of the universal sketch, we have that,

$$\Pr[\widehat{G_{i,j}} \notin [(G_i + N_j)(1 - \epsilon_{US}), (G_i + N_j)(1 + \epsilon_{US})]] \leq \delta_{US}.$$

Since $g$ is part of $G$-sum $\in$ *Stream-PolyLog*, it must be monotone, and thus $N_j \geq 0$. This means that with probability of at least $1 - \delta_{US} - 1/c$ both $\widehat{G_{i,j}} \in [G_i(1 - \epsilon_{US}), (G_i + N_j)(1 + \epsilon_{US})]$ and $N_j < c \cdot \frac{G(\mathcal{S})}{w}$ holds, and thus

$$G_i(1 - \epsilon_{US}) \leq \widehat{G_{i,j}} \leq G_i(1 + \epsilon_{US}) + \frac{c}{w}(1 + \epsilon_{US})G(\mathcal{S}). \tag{3.2}$$

Therefore, we pick $w = c \cdot (1 + \epsilon_{US}) \cdot \epsilon^{-1}$ and a $c$ value such that $1 - \delta_{US} - 1/c > 1/2$, to get that

$$\Pr\left[G_i(1 - \epsilon_{US}) \leq \widehat{G_{i,j}} \leq G_i(1 + \epsilon_{US}) + \epsilon \cdot G(S)\right] > 1/2.$$

Recall that the algorithm's query sets $\widehat{G_i} = \text{median}_c \widehat{G_{i,j}}$ and that the $r$ rows are i.i.d. and thus a standard Chernoff bound yields that

$$\Pr\left[G_i(1 - \epsilon_{US}) \leq \widehat{G_i} \leq G_i(1 + \epsilon_{US}) + \epsilon \cdot G(S)\right] \geq 1 - \delta. \qquad \square$$

### 3.3.6 HYDRA-sketch Configuration

The structure of HYDRA-sketch (Figure 3.9) depends on 6 parameters: 2 parameters determining the structure of the array of sketches (*i.e.,* $r$ and $w$) and 4 for the Universal Sketches (*i.e.,* $L$, $w_{CS}$, $r_{CS}$, $k$). The configuration parameters of HYDRA-sketch determine its empirical accuracy and its practical memory footprint. For instance, larger values of $w$ and $r$ while they ensure better estimation accuracy, they also result in a larger HYDRA-sketch.

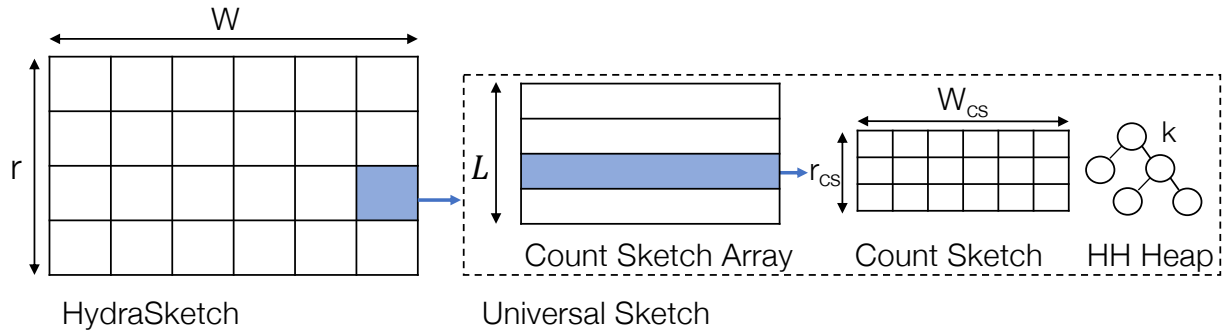



Figure 3.9: HYDRA-sketch structure and configuration parameters.

To achieve a good tradeoff between empirical accuracy and performance, we follow a two-step approach that leverages Theorems 1 and 2. First, we control the impact of the $\epsilon \cdot G(\mathcal{S})$ factor in Eq. (3.1) by appropriately configuring $w$ and $r$. Then, we focus on the estimation error of each Universal Sketch instance, $\epsilon_{US}$.

**Controlling $\epsilon \cdot G(\mathcal{S})$** The $\epsilon \cdot G(\mathcal{S})$ additive error factor can be seen as HYDRA's *estimation bias*. To control the bias through the configuration of the array of sketches, we need to answer two questions; First, what are acceptable values for $\epsilon$ and second, how to translate $\epsilon$ to a practical HYDRA-sketch configuration (*i.e.,* $r$ and $w$ values)?

However, from Eq. (3.1), we observe that $\epsilon$ controls the magnitude of the estimation bias and, thus, determines the level of accuracy HYDRA-sketch can offer to subpopulations of different $G$-sum values. To see why, consider the following example: For a subpopulation $Q_i$, assume $G_i = 0.01 \cdot G(\mathcal{S})$. If $\epsilon = 1 \gg 0.01$, then from Eq. (3.1), $\widehat{G_i}$ is dominated by the $G(\mathcal{S})$ factor and we cannot have any confidence to HYDRA-sketch's estimation of $Q_i$. On

the other hand, if $\epsilon = 0.0001 \ll 0.01$, then the bias is negligible compared to $G_i$. Therefore, to ensure narrow error bounds, the smaller each subpopulation is (in terms of $G$-sum), $\epsilon$ needs to be set at smaller values.

From the proof of Eq. (3.2), we see that $w = c \cdot (1 + \epsilon_{US}) \cdot \epsilon^{-1}$ where $c$ is chosen such that $1 - \delta_{US} - 1/c > 1/2$. We observe that $w$ is inversely proportional to $\epsilon$, which means that to ensure narrow error bounds for small data subpopulations, $w$ (and, thus, the memory footprint of HYDRA-sketch) can become very large. To that end, if we want HYDRA-sketch to have practical memory footprint, we need to set a "threshold" with respect to the size of subpopulations we can provide accuracy guarantees. Therefore, we set $\epsilon = G_{min}/G(\mathcal{S})$, where $G_{min}$ is the $G$-sum of the smallest aggregation for which we want to provide accuracy guarantees. For example for $\epsilon = 0.001,\ \delta_{US} = 0.1,\ \epsilon_{US} = 0.05$ (we discuss these universal sketch parameters later), we get that $w \geq 300$.

From Theorem 2, we see that $r = \mathcal{O}(\log \delta^{-1})$. For (3.1) to hold with 90% probability, we set $\delta = 0.1$. This, in turn, translates to $r \approx 3$. For a 95% probability, $r \approx 5$.

**Controlling universal sketch error** To configure $r_{CS}$, we note that for error probabilities at 90 and 95%, $r_{CS}$ becomes 3 and 5 respectively. To estimate $w_{CS}$, from Theorem 1 we can observe that for $\epsilon_{US} = 0.05$ *i.e.,* for 95% accuracy, we can set $w_{CS}$ to $O(1/0.05^2) \approx 400$. Last, we configure the number of levels ($L$) maintained in each universal sketch instance and the number of heavy keys ($k$) needed to store at each level's heavy hitter heap. From Theorem 1, $L$ needs to be $\mathcal{O}(\log n)$, where $n$ is related to the average number of distinct subpopulations summarized at each universal sketch. The value of $k$ is empirically set to $k = \mathcal{O}(1/\epsilon_{US}) \approx 20$.

In §3.5, we show that these rules indeed provide us with the configuration that achieves the optimal tradeoff for empirical accuracy and memory cost across different datasets.

## 3.4 Implementation and Optimizations

This section discusses practical challenges we tackled to optimize the HYDRA prototype. First, we present our baseline system implementation and introduce an accuracy-improving heuristic for HYDRA-sketch. Then, we discuss potential system bottlenecks and introduce practical optimizations to mitigate them.

### 3.4.1 Baseline Implementation and Workflow

We implement HYDRA's workflow (discussed in §3.2.2) as an Apache Spark plugin [166]. The choice of Spark is a practical one, as the framework's extensibility allowed us to prototype design alternatives easily. However, HYDRA's workflow can easily fit into different analytics frameworks *e.g.,* Druid [161].

In our distributed implementation, worker nodes perform data ingestion and the frontend node performs sketch collection, merging, and query estimation. As we make no assumptions about what subsets of the input stream are processed by each worker node, centralized query estimation is a natural choice to ensure a global view of the data (instead of enforcing a policy where all data records of a given subpopulation are ingested at the same worker node).

At the worker nodes, Spark first splits input data into multiple partitions of $\approx$ 64MB and allocates one HYDRA-sketch instance for each partition before it is updated by that partition of data. Both the input partition and the HYDRA-sketch instances are treated as Spark RDDs. When a failure happens during execution, spark will automatically restart the job and re-ingest. Once a data partition is fully ingested, the corresponding HYDRA-sketch instance is serialized and sent over TCP to the frontend node.

After receiving several HYDRA-sketch instances from worker nodes, the frontend node deserializes them and merges them into a global HYDRA-sketch instance. Note that merging is executed continuously at the frontend node and in parallel with ingestion at the worker

nodes. The operator interacts with the frontend node's query API to estimate the desired summary statistics from the merged HYDRA-sketch instance and return query results to the operator.

### 3.4.2 An Accuracy-improving Heuristic

Recall from Algorithm 1 (line 8) that after $Q_j$ is mapped to a universal sketch, that sketch instance only stores the frequencies of metric values $m_i$. This design, however, does not keep track of which subpopulation $Q_j$ each $m_i$ maps to. As a result,a universal sketch will return the same estimations for all subpopulations whose data it stores. Our heuristic is simple: Instead of updating each universal sketch with $m_i$, we can use a more fine-grained key, *i.e.,* the concatenation of the metric value and its corresponding subpopulation. Algorithm 2 (line 8) highlights this change in the corresponding sketch method. This way, heavy hitter heaps will maintain heavy counts for each $(Q_j,\ m_i)$ pair and will be able to differentiate between them at query time.

**Algorithm 2:** HYDRA-sketch Algorithm with Heuristic

```
1 Generate r pairwise independent hash functions: h_1 ... h_r : [n] → [0, w)
2 function UpdateHeuristic(Data point ⟨d_1, d_2, ..., d_D, m_i⟩, HydraSketch
    HS)
3     r ← HS.r; w ← HS.w; s ← HS.sketch
4     {Q_1, ..., Q_{2^D}} ← Fanout(⟨d_1, d_2, ..., d_D⟩)
5     for each Q_j do
6         for k ← 0 to r − 1, in parallel do
7             US_k ← s[k][h_k(Q_j)]
8             US_k.update(Q_j, m_i)
```

### 3.4.3 Reducing Runtime Bottlenecks

To illustrate HYDRA's bottlenecks, we run our system prototype using a real-world CAIDA trace (which is also used in §3.5). We configure HYDRA-sketch using the strategies from §3.3.6 and measure the CPU time of different HYDRA operations. Table 3.3 summarizes our observations.

| **Worker Nodes** | | **Frontend Node** | |
|---|---|---|---|
| **Operation** | **CPU time** | **Operation** | **CPU Time** |
| Heap Updates | 34% | Sketch Merging | 24% |
| Hashing | 29% | Deserialization | 23% |
| Read Stream | 15% | Communication | 12% |

Table 3.3: Analysis of HYDRA's bottlenecks.

1. *Worker Nodes.* During data ingestion, we observe that approximately 15% of CPU time is dedicated to data reading and fan-out and approximately 75% is spent on sketch updates. The remaining time is spent on framework-specific operations. Breaking down sketch-update time further, we observe that hashing consumes 29% of this time, and the remaining 34% is spent on updating the top-$k$ heavy hitter heaps. These numbers are are consistent with our expectation. For every data point, HYDRA-sketch runs $\mathcal{O}(r \times L)$ hashing and heap update operations, both of which are CPU-heavy.
2. *Frontend Node.* At the frontend node, ∼60% of CPU cycles is spent on sketch merging, deserialization and communication. In particular, we see that the communication channels from the frontend to the worker nodes take up approximately 12% of CPU time. Data deserialization and sketch merging take up 50% of CPU time.

The above analysis highlights that to fully extract HYDRA's potential, we need to address non-trivial compute and communication bottlenecks. Below, we look at mitigating their

impact on HYDRA's end-to-end performance though several practical optimizations.

### 3.4.4 Implementation Optimizations

**Reducing overheads at worker nodes** Reducing compute overheads in HYDRA's worker nodes boils down to reducing hash computations and heap updates. To that end, we introduce two optimizations to HYDRA-sketch's design:

1. **One large hash per ($Q_j$, $m_i$) Pair:** Updating HYDRA-sketch with a ($Q_j$, $m_i$) pair requires $\mathcal{O}(r \times L)$ hash computations, $r$ to identify the universal sketches to update and up to $L$ per universal sketch. In HYDRA, we reduce the number of hash operations to $\mathcal{O}(1)$ by computing one large 128-bit hash and breaking it down into substrings of variable lengths and we treat each substring as a separate hash. Prior analysis [81, 109] shows that different substrings from the same long hash provide sufficient independence.
2. **One layer update:** In prior universal sketching implementations, the algorithm keeps a heap to track frequent keys per layer. For each datapoint update, the universal sketching needs to update one or more of its layers (two layers on average). In HYDRA-sketch, we follow the paradigm of [162] and observe that only the lowest sampled layer is required to update per datapoint. This technique reduces the number of layers updated from two to one per datapoint, while providing an algorithmically equivalent implementation. In §3.5, we show that this optimization improves total HYDRA runtime by 10%.

**Reducing overheads at frontend node** To accurately estimate queries, HYDRA's frontend node fetches the HYDRA-sketch instances, merges them into a global HYDRA-sketch and uses that to estimate the desired metrics. For instance, if a HYDRA cluster consists of $N$ worker nodes and one frontend and within one data epoch, there are $Q$ subpopulations

to query, the total query time is:

$$t_{Query} = t_{Fetch} + (N-1)t_{Merge} + \frac{Q}{\alpha}t_{Estimate} \tag{3.3}$$

Eq. 3.3 assumes that $N$ HYDRA-sketch instances are simultaneously fetched and estimation is parallelized by a factor of $\alpha$. Therefore, minimizing overheads at the frontend node, involves minimizing each of the three terms in Eq. 3.3. $t_{Fetch}$ can be minimized with appropriate compression and we apply the two optimization techniques we used at the worker nodes for $t_{Estimate}$. Below, we discuss our approach for reducing $t_{Merge}$.

Merging two HYDRA-sketch instances involves summing up corresponding counters, re-computing the heavy elements, and re-populating the heavy hitter heaps. However, our analysis shows that we can focus on merging only the heavy hitters in the heaps instead of merging all the sketch counters. This incurrs negligible accuracy loss while improving system runtime by 8%.

## 3.5 Evaluation

We evaluate HYDRA's end-to-end performance using both real-world and synthetic datasets. We also provide a sensitivity analysis of HYDRA's design and evaluate our configuration strategies, and performance optimizations. Our key findings are:

1. HYDRA offers guaranteed estimation accuracy that is comparable to that of traditional, exact analytics engines (*e.g.,* >95% accuracy with 90% probablity) for a broad set of summary statistics at 1/10 of their $ cost.
2. HYDRA's memory usage scales sub-linearly with both dataset size and data subpopulations. Further, HYDRA's query latency is 7-20× smaller than existing analytics engines.
3. HYDRA's sketch configuration strategies ensure high estimation accuracy and low memory footprint, often up to two orders of magnitude lower than that of an unoptimized configuration.

4. HYDRA's performance optimizations improve end-to-end system runtime by 30% compared to a deployment that uses the basic HYDRA-sketch design.

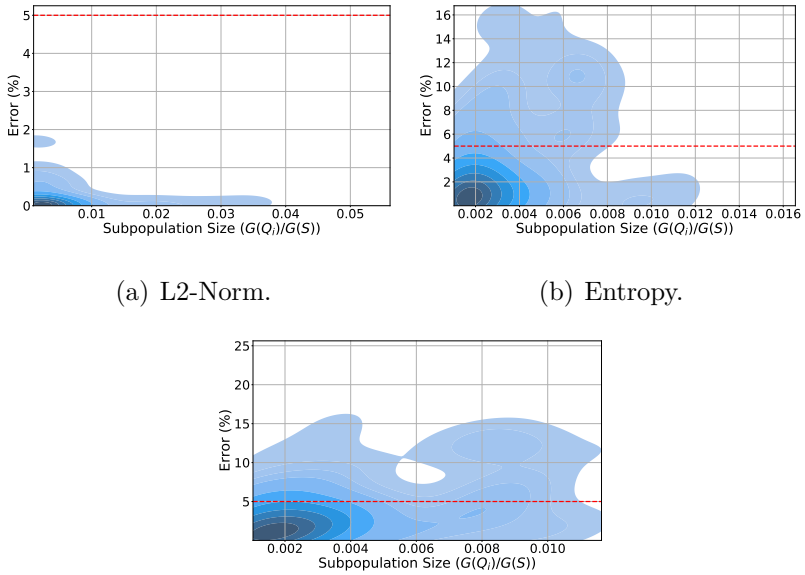
### 3.5.1 Experimental Methodology

**HYDRA testbed** We run HYDRA on a 20-node cluster on m5.4xlarge AWS servers [20]. For the end-to-end performance evaluation, we deploy optimally configured HYDRA-sketch instances to ensure 95% estimation accuracy with at least 90% probability for data subpopulations for which $G(Q_j)/G(\mathcal{S}) >= 0.001$. We also use the performance optimizations mentioned in §3.4. The input data consists of CSV files that are ingested from AWS S3.

**Datasets** We evaluate HYDRA using two large real-world datasets and a synthetic trace. Each dataset maps to a different usecase that can benefit from efficient multidimensional analytics. First, we use CAIDA flow traces [3] collected at a backbone link of a Tier1 US-based ISP. The total trace is up to 130GB in size and flow data can be clustered in up to approximately 5.6M subpopulations. Second, we use a real-world trace of video session summaries corresponding to one major US-based streaming-video provider. The size of the video-QoE trace is approximate 5GB, with data that we cluster in up to 700k subpopulations. Third, we generate synthetic traces following Zipf distribution with varying skewness (*e.g.,* 0.7 to 0.99).

**Summary statistics** We evaluate HYDRA's accuracy using a set of 4 summary statistics, including Cardinality, Entropy, L1 and L2 Norms. For each subpopulation, we compute the precise value of each statistic as ground truth and then estimate the relative error with respect to HYDRA's accuracy.

**Evaluation baselines** For our experiments, we compare HYDRA against the following baselines: (1) **Spark-SQL:** This is a traditional SQL implementation where incoming data record is stored as a row in one (logical) data table; (2) **Spark-KV:** In this custom baseline, we transform incoming data at ingestion time and maintain a Key-Value store where the




(a) L2-Norm.

(b) Entropy.

(c) L1-Norm.

Figure 3.10: Error distribution for different data subpopulations per statistic. Red line indicates 5% acceptable error threshold.

keys are distinct $\langle Q_j,\ m_i \rangle$ tuples and the values are their respective frequency counts; (3) **Uniform Sampling:** We implement 10% uniform sampling at ingestion time and then apply the Spark-KV approach to the sub-sampled data; (4) **Druid:** This is similar to Spark-KV but uses Druid's built-in capabilities (*i.e.,* data roll-up) to generate the key-value store; (5) **One Universal Sketch per subpopulation:** We implement a simple sketch-based approach by allocating one universal sketch per subpopulation.

### 3.5.2 End-to-End Evaluation of HYDRA

To evaluate HYDRA end-to-end we investigate whether the system meets operators' requirements. To that end, we ask three key questions; (1) Does HYDRA enable scalable coverage across data subpopulations at a reasonable cost to the operator? (2) Does HYDRA provide high fidelity estimations across a broad set of summary statistics? (3) Does HYDRA offer interactive query latencies? We answer these questions below.

**Scalable coverage across subpopulations** To estimate HYDRA's operational cost, we estimate the $ cost of querying 4 summary statistics for the CAIDA dataset (100GB size, 5.6$M$ subpopulations) on our 20-node AWS cluster. Specifically, we measure the ingestion and query times for HYDRA and baselines and compute their normalized $ cost. Figure 3.11 depicts HYDRA's cost-accuracy tradeoff. HYDRA's cost is approximately two orders of magnitude smaller than Spark-SQL and one order of magnitude smaller than a Druid baseline that leverages the framework's built-in data roll-up feature. We observe that HYDRA's operational cost is on par with a sampling approach that uniformly samples 1% of all data but whose error can be very large. In the case of the smaller video-QoE dataset, HYDRA is only 3× cheaper than the Spark-SQL baseline and approximately as costly as Spark-KV. We attribute this smaller gap to the smaller size of the dataset.

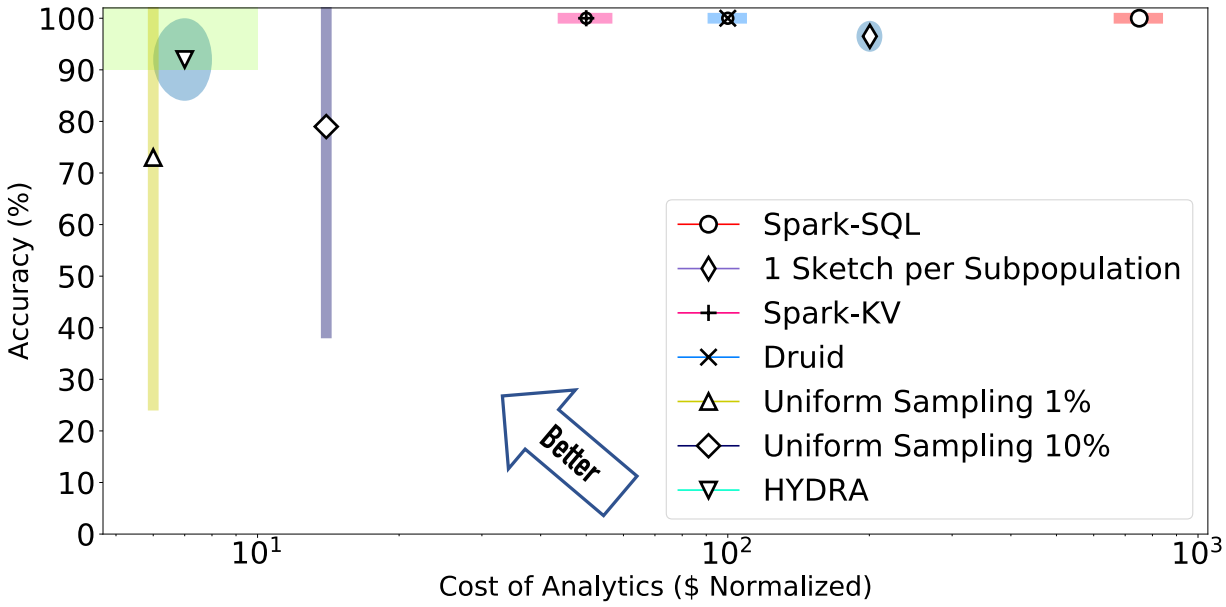



Figure 3.11: End-to-end cost analytics. The green-shaded region indicates the ideal operating regime for HYDRA.

**Generality and fidelity across summary statistics** To evaluate HYDRA's ability to ensure generality and high fidelity estimations across statistics, we estimate four different sets of summary statistics. In Figure 3.12 we depict for each statistic its mean error and standard

deviation. For all application sets HYDRA operates under the same resource budget and configuration as described previously. Indeed, we find that estimating multiple summary statistics does not incur accuracy reduction compared to when individual statistics are estimated, thus highlighting HYDRA's generality and high fidelity. This is because the information maintained in the universal sketches is statistic-agnostic and is equally used for any (one or more) statistics of interest. We observe the same behavior for the video-QoE dataset with a mean error across statistics of ∼6%.

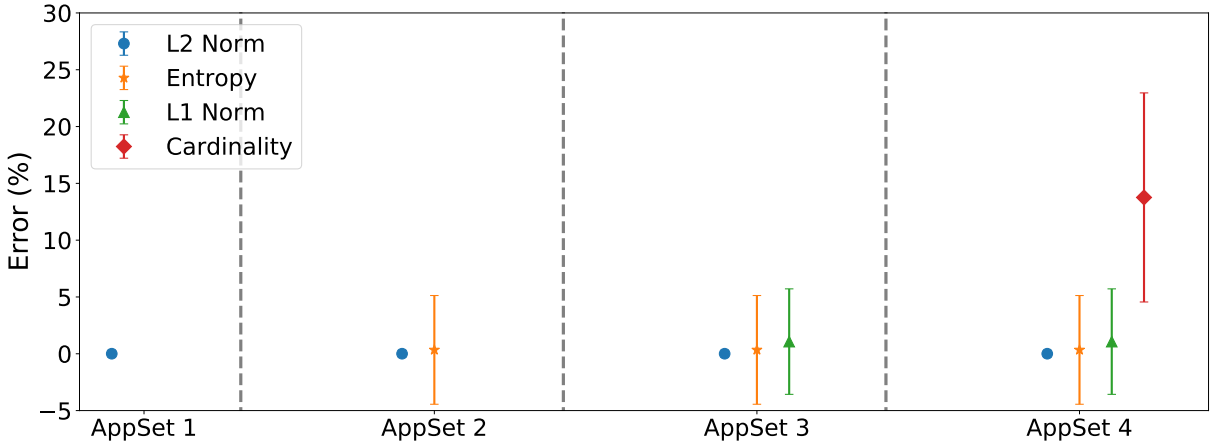



Figure 3.12: HYDRA's estimation error for the CAIDA dataset.

Figure 3.10 depicts the distribution of estimation error values for 3 summary statistics as a function of the subpopulation's normalized $G$-sum *i.e.*, $G(Q_i)/G(\mathcal{S})$. Recall that the $\epsilon$ value of HYDRA-sketch determines the lowest threshold (in terms of $G$-sum) for which HYDRA offers accuracy guarantees. In our experiments, we configure HYDRA-sketch to ensure 95% accuracy with 90% probability with $\epsilon = 0.001$ and observe that our prototype's empirical accuracy meets the expected accuracy guarantees with the majority of subpopulations. A larger $\epsilon$ would constrain the analysis to larger subpopulations. A smaller $\epsilon$ would result in the opposite behavior.

**Interactive estimation latencies** Figure 3.13 illustrates HYDRA's runtime (ingestion and query time) as a function of the dataset size and the number of data subpopulations for the CAIDA dataset. We can see that HYDRA's query time is <10sec for 5.6 million

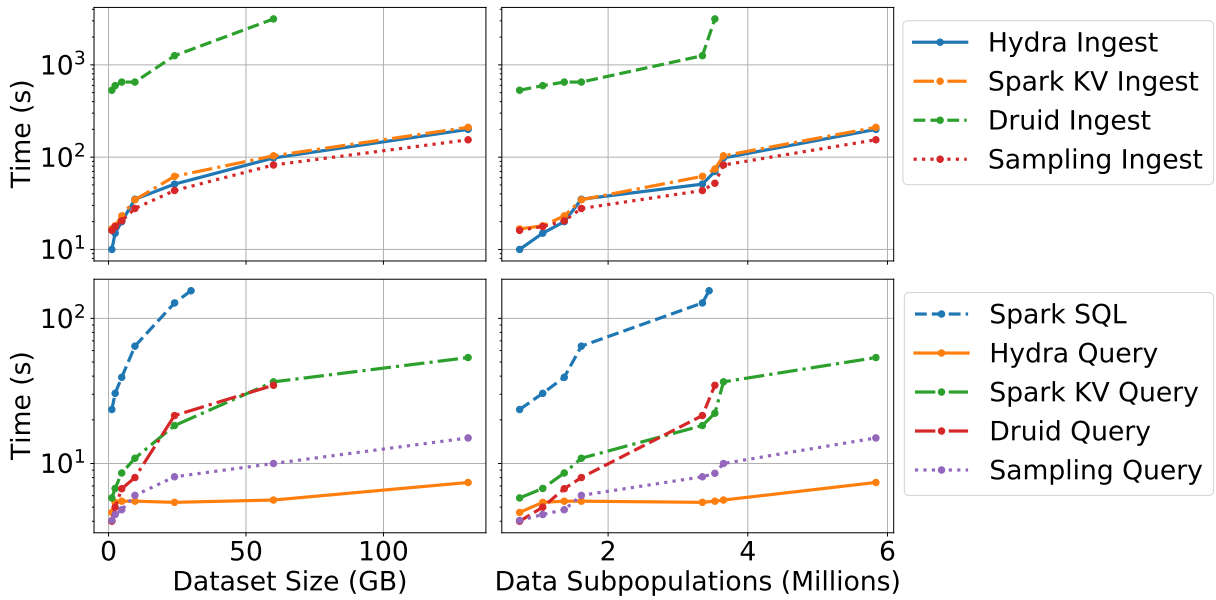



Figure 3.13: Runtime for CAIDA dataset

data subpopulations, almost one order of magnitude (7×) smaller than that of Spark-KV. Note that we do not report Spark-SQL results for dataset sizes larger than 25GB because execution was always prematurely terminated by the framework. We encountered the same issue for Druid and for dataset sizes larger than 60GB. However, even for a small input, the querying latency of Spark SQL is 2 orders of magnitude larger than that of HYDRA. Indeed the small memory footprint of HYDRA-sketch which often enables for a cache-resident data structure offers the desired low estimation latency.

### 3.5.3 Detailed Analysis of HYDRA-sketch

We evaluate HYDRA-sketch across three axis. First, we compare HYDRA-sketch's memory footprint to that of our baselines. Second, we show that our configuration strategies converge to a near optimal configuration for HYDRA-sketch with respect to memory and runtime. Last, we show that our performance optimizations reduce HYDRA's runtime by 27%.

**Memory footprint vs. subpopulations** Figure 3.14 shows HYDRA's memory footprint (and that of baselines) as a function of the number of subpopulations monitored for the CAIDA dataset. HYDRA is consistent with the theoretical sub-linear memory scaling as both the dataset size and data subpopulations increase. Indeed, while we observe that for smaller datasets, a Spark-KV implementation might be preferable in terms of memory footprint (as the size of the sketch instances might even exceed that of the input), this trend is very quickly reversed, turning HYDRA into the better approach. Indeed, this is an observation that is confirmed in the case of the video-QoE dataset where the optimal memory footprint of HYDRA-sketch is approximately the same to Spark-KV.

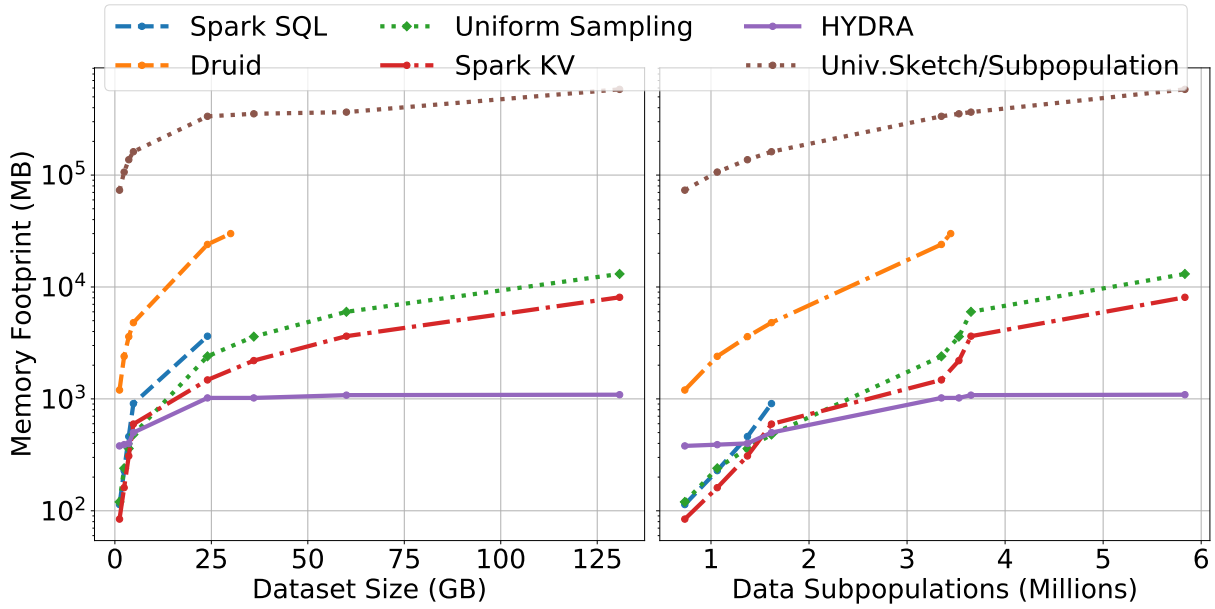



Figure 3.14: Memory footprint given dataset size and subpopulations.

**Configuration heuristics** Figure 3.15 depicts the relationship between the memory footprint of HYDRA-sketch and the resulting estimation error for different configurations. The estimation error of the figure corresponds to the estimation of the L1-Norm of the CAIDA dataset. Naturally, the optimal configurations are those that simultaneously minimize the estimation error and HYDRA-sketch memory footprint (encircled in graph and marked with red stars). The orange diamond configuration is the suggested configuration based on the

configuration strategies discussed in §3.3. We can see that our configuration strategies result in a sketch configuration that lies within the set of optimal configurations. This observation holds across all summary statistics and datasets.

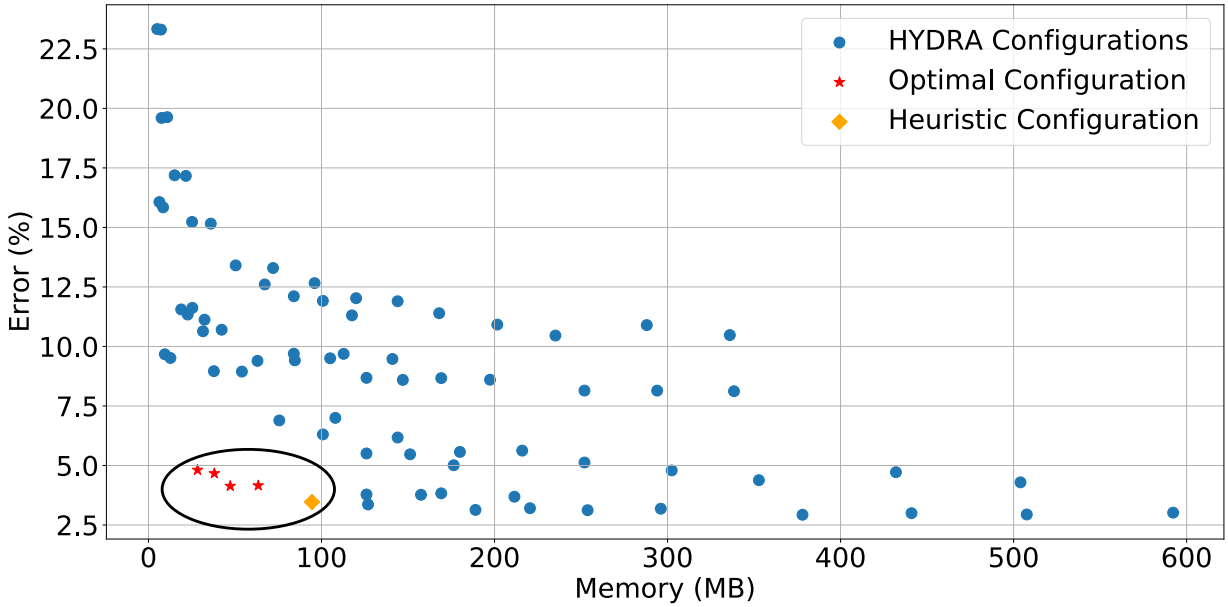



Figure 3.15: HYDRA's configuration strategies ensure a configuration within the set of optimal configurations

In addition, we investigate how sensitive HYDRA-sketch is to changes in the properties of the data stream. Specifically, we check whether HYDRA-sketch needs to be frequently reconfigured in order to adapt to the changing input data stream. For our experiment, we initially configure HYDRA-sketch optimally and, every 10 epochs, we estimate its error gap, *i.e.,* the difference between the latest estimation error and that of the initial, optimal, configuration. Figure 3.16 shows the error gap as a function of the data epoch. We observe that for the CAIDA dataset, the maximum absolute error gap across statistics is ∼3% and argue that it is ultimately up to the operator to determine the frequency at which HYDRA-sketch needs to be reconfigured based on the accuracy objectives they set.

**Analysis of performance optimizations** Figure 3.17 depicts the cumulative improvement in HYDRA's performance after applying the performance optimizations discussed in §3.4.

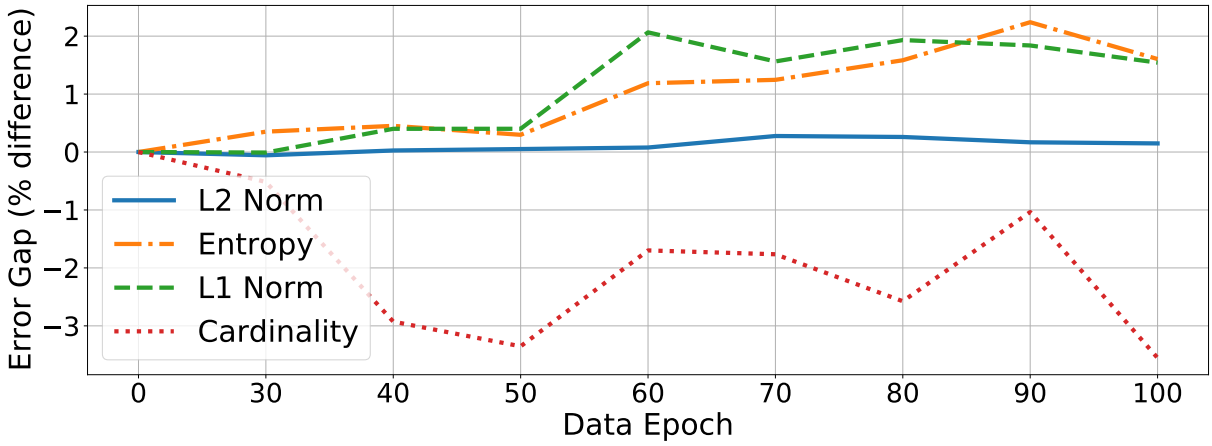



Figure 3.16: Sensitivity of HYDRA-sketch configuration to changing data epochs for CAIDA dataset.

Each datapoint corresponds to a different HYDRA-sketch configuration (the Pareto frontier of Figure 3.15) and we run each configuration twice, once for the basic HYDRA-sketch design and once with the performance optimizations. We can clearly see that the performance optimizations further reduce the memory footprint of HYDRA-sketch and also the total system runtime *i.e.,* the sum of ingestion and query time.

Table 3.4 captures HYDRA's runtime reduction after each performance optimization. The baseline is HYDRA without optimizations and, overall, we see a total performance improvement of 27%.

| Baseline | Heap-only Merge | One Hash | One Layer Update |
|---|---|---|---|
| 100% | 92% | 81% | 73% |

Table 3.4: Runtime improvements with performance optimizations

**Skewness of dataset** Figure 3.18 highlights the difference in estimation accuracy for two synthetic datasets generated with a zipfian distribution. The subpopulations are samples from a zipfian distribution with parameters $\alpha = 0.7$ and $\alpha = 0.99$ respectively (a value of $\alpha = 0$ indicates a perfectly uniform distribution). Our experiment confirms our intuition

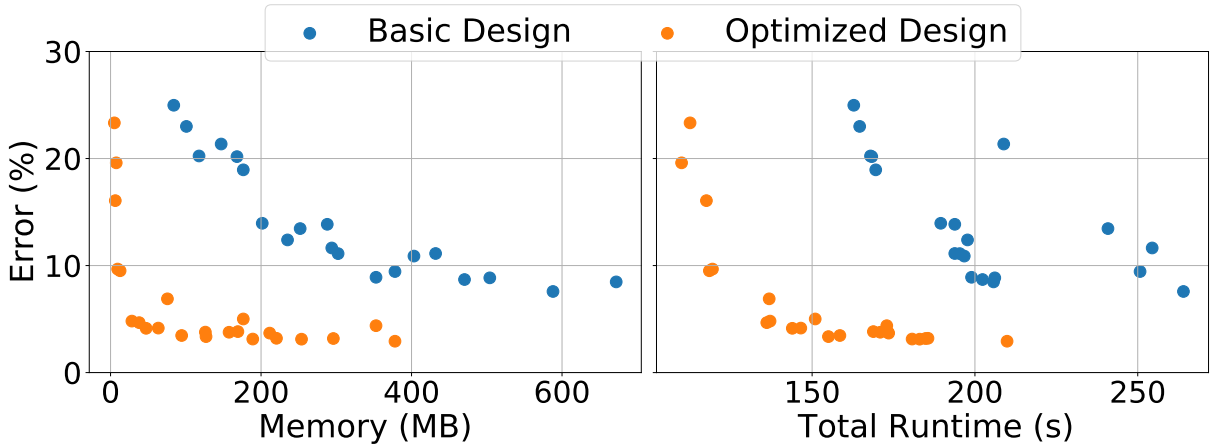



Figure 3.17: Comparison of the Pareto frontiers of basic and the optimized Hydra-sketch implementation for the same configurations.

that the more skewed dataset ensures a better (memory, error) tradeoff. In practice, many real-world datasets are skewed and thus can benefit from being analyzed by Hydra.

## 3.6 Related Work

**MapReduce-based analytics frameworks** Starting with Apache Hadoop, there are various frameworks for large-scale data analysis that are based on the MapReduce paradigm [87, 141]. Dryad [100] introduced the concept of user-defined functions in general DAG-based workflows and SCOPE [61] provided a language and an SQL-query optimizer. Apache Drill and Impala [111] in a similar fashion limit their operations to SQL variants. Apache Spark [166] is a data-processing framework that leverages a DAG-based execution engine, provides SQL optimizers and treats unbounded computation as micro-batches. Apache Flink [60] builds on these ideas to enable pipelined streaming execution for batched and streaming data, offers exactly-one semantics through checkpointing and supports out-of-order processing. Similar to Spark, Apache Flink could be used as an alternative base framework for Hydra.

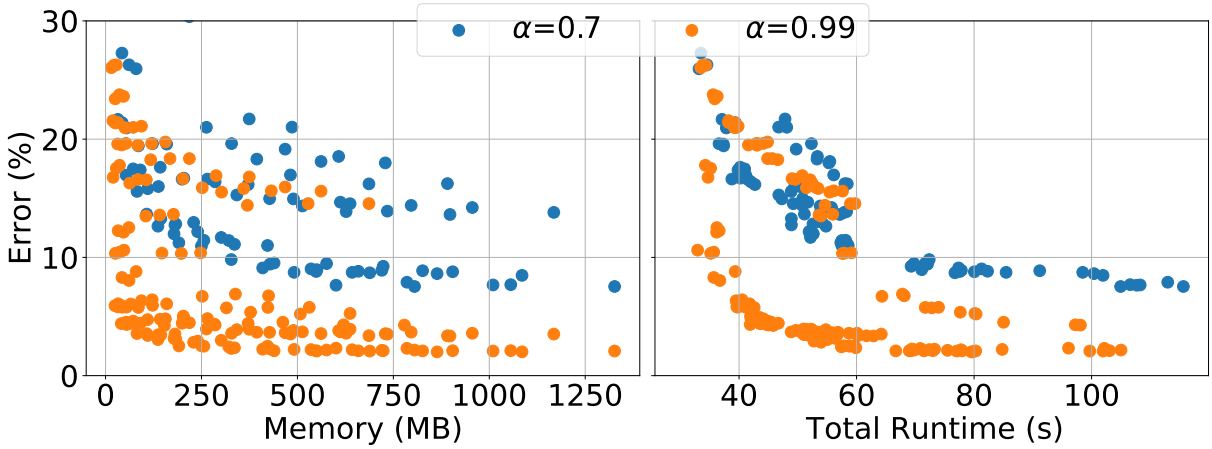



Figure 3.18: Impact of data skewness on HYDRA's memory footprint and runtime. We use a synthetic dataset where subpopulation sizes are sampled from a Zipfian distribution with parameter $\alpha$.

**Stream processing frameworks** There are multiple stream processing systems both commercial and academic. This line of research focuses on the architecture of stream processing systems, attempting to answer questions about out-of-order management of data in streams, fault tolerance, high-availability, load management, elasticity and more [13, 14, 28, 38, 42, 48, 60, 98, 108, 124]. Many of these systems do not scale computation horizontally on clusters of commodity servers. Apache Storm [99] enables horizontal scalability and compositional workflow but with weaker state consistency guarantees *i.e.*, at-least-once processing. Fragkoulis *et al.* in their comprehensive survey outline past research findings analyze the state of the art of stream processing engines [82]. The work on stream processing engines is complementary to ours, as ideally such engines could serve as a engine HYDRA can be deployed on top of.

**Data aggregations** Aggregation-based queries appear in multiple existing streaming data systems [35, 46, 56, 73, 91, 137, 161]. These systems can estimate statistics across various data subpopulations and motivate HYDRA. Many of the above frameworks enable approximate analytics but do not fully satisfy operators' requirements. HYDRA improves the state of the art in multidimensional telemetry.

**Sampling-based approaches** There are multiple analytics frameworks that use sampling to provide approximate estimations [30, 62, 129, 152]. BlinkDB [35] builds a couple of stratified samples on the original data and executes the queries on the samples to reduce query execution time. The number and sizes of the stratified samples are limited by the storage budget specified when importing the data. STRAT [65] also uses stratified sampling but instead builds a single sample. SciBORQ [142] builds biased samples based on past query results and cannot provide any guarantees on its error margin.

**Online aggregation** Online Aggregation frameworks [93, 116, 133] proposed the idea of continuously refining approximate answers at runtime. In these frameworks, it is up to the user to determine when the acceptable level of accuracy is reached and to terminate estimation. Naturally, this approach is unsuitable for multidimensional telemetry that needs to estimate multiple statistics simultaneously across data subpopulations.

**Data summaries** Data "synopses" (*e.g.,* wavelets, histograms, sketches, *etc.*) have been extensively used for data analytics [34, 59, 71, 88, 101, 118, 151, 154]. These data summaries can either be lossless or lossy and they aim at providing efficiency, especially for multidimensional analytics. A key drawback of these approaches is that they are heavily tailored to a very narrow set of estimation tasks and are not general. Gan *et al.* develop a compact and efficiently mergeable quantile sketch for multidimensional data [84]. Ting *et al.* with a similar motivation focus on cardinality estimation [148]. Also, few prior works have also considered the idea of nested sketches to account for the combinatorial explosion of data subpopulations [70, 147, 148]. However, these approaches do not offer neither high fidelity estimations nor generality to statistics.

## 3.7 Conclusions

Today's large-scale services and infrastructures require real-time estimations of a diverse set of summary statistics across multiple subpopulations of their multidimensional datasets.

However, the combinatorial explosion of data subpopulations due to increases in data volumes and dimensionality makes it hard to offer multidimensional analytics at a reasonable cost to the operator. HYDRA is a sketch-based analytics framework that leverages HYDRA-sketch, a sketch-of-universal sketches that summarizes data streams in sub-linear memory to the number of subpopulations. We show that HYDRA is an order of magnitude more efficient in than existing analytics engines while ensuring interactive estimation times. While HYDRA is a general-purpose analytics framework, since viewership counts is a summary statistic that HYDRA can estimate, its applicability in video viewership analytics is straightforward.

# Chapter 4

# PROTEAS: A Real-Time Alerting Framework for Video Viewership Anomalies

As discussed previously, to sustain viewer engagement and maintain ad- and subscription-driven revenue streams [75, 113, 117], Internet video providers monitor their infrastructure to detect and diagnose incidents (*e.g.,* ISP outages, buggy players) [6, 26], and to inform mitigation efforts [102, 104].

Our conversations with analysts working in this domain highlight the need for *viewership-based analytics*, to complement existing monitoring workflows and proactively identify anomalies that would be missed otherwise (*e.g.,* based on monitoring QoE metrics like buffering time or startup latency [102, 104]). To see why, consider the following real-world incidents: AV encoding errors can cause pixelation or missing audio, resulting in viewers dropping out. Similarly, a platform failure, such as a provider's authentication system crash may result in viewers getting disconnected and viewership dropping. However, in both cases delivery will not be impacted, available QoE metrics will continue to look normal and the incident will go undetected. In addition, only considering QoE-based monitoring may not always provide actionable insights to operators. For example, increases in buffering time during live events can either be due to technical issues (*e.g.,* a buggy player device)

or due to a flash crowd that stresses the available CDN resources. However, it is hard to disambiguate these two scenarios just by looking at QoE metrics. In this case, flagging an unexpected viewership surge could alert operators for upcoming QoE issues.

To this end, we envision a real-time alerting framework that *detects* and *diagnoses* such viewership anomalies. However, as we noted in Chapter 1, realizing such a framework is challenging given (i) the contextuality of anomalies and the non-stationarity of viewership and (ii) the need for compact and actionable alerts. In this chapter, we discuss how we address these challenges and present PROTEAS, an alerting framework for video viewership anomalies. Our design builds on the following *structural insights*:

1. **Shape persistence for anomaly detection:** Despite the non-stationarity of viewership, we find that its *underlying shape*[1] remains invariant over longer periods of time, and can thus be leveraged as a basis for detection. By modeling this key structural invariant using custom Gaussian Processes [139], we enable accurate, timely and robust anomaly detection of viewership anomalies across multiple viewership groups.
2. **Hierarchical group dependencies and spatiotemporal anomaly signatures for diagnosis:** A single logical event *propagates* predictably across viewership groups. Using practical heuristics, we extract the set of candidate groups that best explain an anomalous incident. Moreover, while viewership anomalies might be the result of many different root causes, we observe that anomalies resulting from similar incidents share common spatiotemporal features, allowing us to compile a library of *anomaly signatures*. This enables associating each viewership incident to a small set of possible root causes for further investigation.

We evaluate PROTEAS using real viewership data from 3 major content providers operating in Europe and USA. Our dataset spans a 13-week period (January-March 2020) and also includes the onset of the COVID-19 outbreak in these continents. Obtaining ground truth at this scale is a fundamental challenge and to this end, we corroborate our analysis

[1] By shape (§4.2), we refer to the structural form of the viewership timeseries after eliminating magnitude variations.

through (1) manual verification of anomalies by experts of a large video analytics firm, (2) sentiment analysis of viewer behavior from Twitter, (3) cross-evaluation of detected viewership anomalies with alerts on video-QoE metrics, (4) public holiday calendars and (5) public databases providing information on content providers' uptime.

The precision/recall analysis of our pilot study (§3.5) shows that under normal operating conditions PROTEAS issues a practical (for the analyst) number of alerts with low false positives (precision >86%) and practically no false negatives. PROTEAS outperforms the closest state-of-the art alternatives by Twitter [95] and Netflix [25] both quantitatively in terms of precision and qualitatively in terms of trust shown by expert analysts to PROTEAS's alerts over those issued by prior work (>95%). Our synthetic analysis shows that the summarization component prunes out more than 99% of redundant anomalies and accurately identifies the root-cause viewership group by classifying anomalies in 4 broad but representative classes of events.

PROTEAS identified 3 large-scale technical outages that affected up to 50% of the content provider's viewers and were not caught by the existing alerting workflows of a major video analytics provider. PROTEAS tracked flash crowds during popular live events (*e.g.,* sports games) before the corresponding QoE alerts were raised, thus exposing the need for resource re-provisioning. Last, PROTEAS uncovered qualitative insights into the pervasive changes in viewership caused by COVID-19.

## 4.1 Challenges of Viewership Analytics

While anomaly detection and diagnosis are not new problems, video viewership raises several challenges that make it difficult to directly apply techniques from prior work.

**Dataset** To highlight these challenges, we use a real-world dataset consisting of 13 weeks of viewership data from 3 major content providers $CP_{\{1,2,3\}}$, two being US-based and one national TV provider in northern Europe. For each content provider, our dataset contains

| Attributes | Description |
|---|---|
| ISP | ISP viewers receive their traffic from. |
| Site | Content provider of requested video content. |
| City | City where the viewer is located. |
| CDN | CDN viewers receive their content from. |
| Player | App/Web portal to access content. |
| Device | *e.g.,* iOS, AppleTv, RokuTV, HTML5 *etc.* |
| Protocol | Bitrate streaming protocol (*e.g.,* DASH). |
| Live | Binary indicator of Live vs. VoD content. |
| Channel | Channel in the content provider's network. |

Table 4.1: Video session attributes.

∼350 million individual video sessions which are grouped into viewership groups based on the session attributes shown in Table 4.1. Overall, the dataset spans 6 large CDNs, 180 ISPs, 60 different cities, and 170 TV channels. §4.6 provides more details on our dataset and discusses ethical considerations.

### Detection challenges

Viewership exhibits two key properties that raise important detection challenges and complicate the use of well-known timeseries anomaly detection techniques:

**Contextual anomalies** Viewership anomalies need to be seen in *context*. That is, in addition to detecting a statistically unexpected rise/fall in the signal, the analyst also needs to consider when that change appears in time *i.e.,* its temporal context. To illustrate this, Figure 4.1 shows two 30-min episodes of the same popular weekly TV show. At the start and end of each episode, viewership exhibits sharp change points (marked in the top figure). As such, these changes are expected events and should not be flagged as anomalous. On the other hand, any other sharp drop/rise (as the one marked in the lower figure) should likely

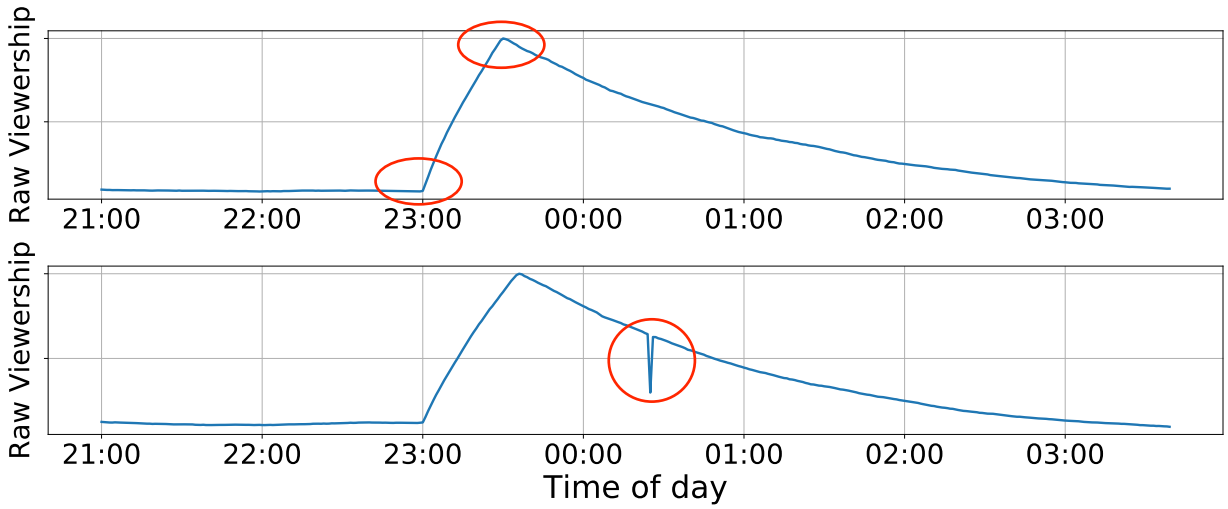



Figure 4.1: Example of viewership anomaly. Real viewership values not shown.

be flagged as an anomaly. More generally, the contextual nature of viewership anomalies renders *model-free* anomaly detection techniques that only rely on short history windows for detection ineffective.

**Lack of stationarity** The contextual nature of anomalies hints toward *model-based* anomaly detection techniques that model what expected viewership looks like and use that knowledge as a detection baseline. However, many model-based approaches rely on a *stationarity* assumption [126]. That is, they assume that the statistical properties of the signal do not change over time and, thus, historical data can be used as predictors of the future.

Indeed, our experiments using stationarity tests *e.g.,* the Augmented Dickey Fuller test [83] showed that viewership is non-stationary due to its large temporal magnitude variability. For the same weekday, viewership values fluctuate greatly from week to week as the result of exogenous factors, such as content popularity, weather, competing shows *etc.* with differences in its magnitude being as high as 30%. Model-based techniques that assume stationarity in the underlying signal would naturally try to capture this variability thus resulting in inaccurate viewership baselines and false positives.

### Diagnosis challenges

**Anomaly summarization** The same anomalous incident can redundantly manifest itself across overlapping viewership groups. To see why, consider the following real-world example where a major content provider rolled out a buggy firmware update for its RokuTV player which resulted in viewers on RokuTV (irrespective of geography) not being able to access any content. While the event should only raise an alert for the ⟨RokuTV⟩ viewership group, anomalies were detected at multiple other viewership groups including (i) finer partitions of the root-cause group (*e.g.,* ⟨RokuTV, NYC⟩, ⟨RokuTV, Comcast⟩, *etc.*) and (ii) broader viewership groups such as ⟨ALL_viewers⟩ (given that RokuTV is a major player among streaming devices). Naively alerting on all such detected anomalies for all viewership groups is clearly impractical as it unnecessarily increases analysts' efforts to diagnose the incident. Therefore, how do we efficiently *summarize* these alerts and identify the viewership group that best describes the incident? In §4.4.1, we discuss why seemingly intuitive approached proposed in prior work for other domains (*e.g.,* ad-systems [53]) are not a natural fit for viewership.

**Root-cause attribution** Ideally, an insightful alert should contain pointers to the incident's potential root causes to help analysts investigate the event. However, associating a viewership alert to candidate underlying events is not straightforward, *e.g.,* how can we disambiguate between a viewership drop that was the result of an outage or an encoding error? The complexity of the streaming ecosystem, the possible external factors that can trigger an anomaly and the lack of direct feedback from viewers as to why they stopped streaming complicates the task of root-cause attribution.

## 4.2 System Overview

To fill this missing piece in the video analytics toolkit, PROTEAS' design focuses on detecting and diagnosing viewership anomalies at scale. Figure 4.2 depicts a high-level overview

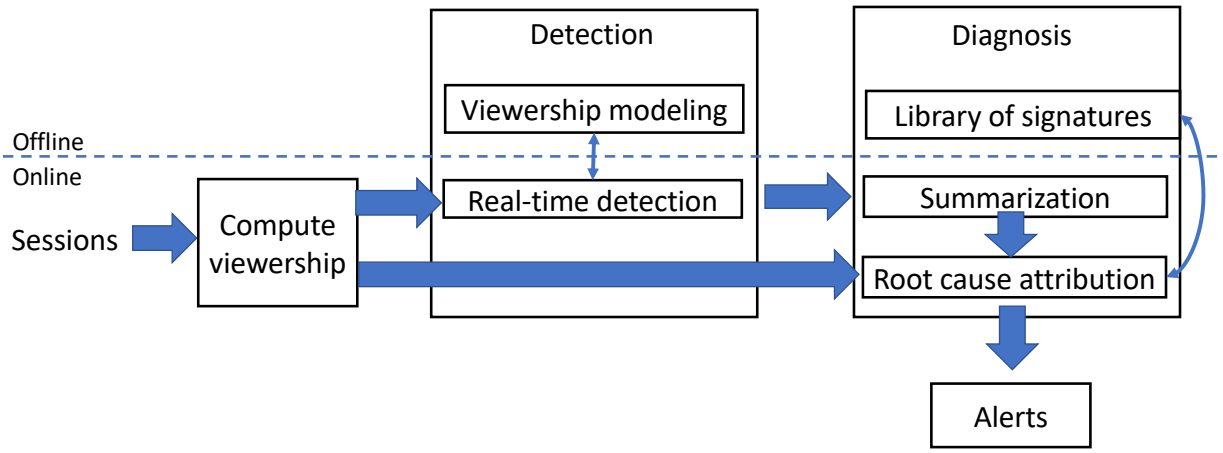



Figure 4.2: A conceptual view of PROTEAS and its components.

of PROTEAS and its two key components. The detection component leverages structural insights about viewership to build models for real-time detection at a viewership group granularity. The diagnosis component collects group-specific anomalies, summarizes them, and produces candidate diagnoses to inform further investigation.

**Inputs** PROTEAS every minute receives batches of raw *video sessions*. Consistent with the definitions of Chapter 2, a video session represents a user consuming one piece of content, describes the content across a number of attributes (Table 4.1) and contains real-time measurements of video QoE. as input. It groups video sessions by all combinations of session attributes (Table 4.1) and for every resulting group it computes viewership as the count of the group's sessions. For each group, PROTEAS maintains a short timeseries of recent viewership history (∼1 hour) that the new observation gets appended to.

**Detection component (§4.3)** The detection component tracks viewership across groups and consists of two submodules. The offline submodule leverages a key structural invariant regarding the shape of the viewership curve to learn a shape-based detection model per viewership group. Each model is trained using summaries of the group's historical viewership data and is fed into the real-time module that determines in near real-time (<10 minutes) whether a new observation is anomalous. To ensure that expected changes in

viewership patterns are accounted for (*e.g.,* due to content changes) and that viewership baselines are always up-to-date, the offline component re-trains its models weekly. This module outputs a list of potentially anomalous viewership groups per minute.

**Diagnosis component (§4.4)** The diagnosis component consists of two sub-modules, summarization and root-cause attribution. Their goal is to provide the human analyst with actionable anomaly alerts consisting of the most informative summary of detection's output. Given the list of anomalous groups from the detection step, PROTEAS's summarization uses a set of heuristics that model the propagation paths of a single anomalous incident across groups to prune out redundant anomalies. The group that best explains the event is read by root-cause attribution whose goal is to identify likely root-causes. To that end, root-cause attribution relies on a library of anomaly signatures, (that is periodically updated as more signatures are identified) and matches the observed new event to one or more of these signatures.

## 4.3 Shape-Based Anomaly Detection

We now introduce the domain-specific observations about viewership that drive PROTEAS's detection mechanism.

### 4.3.1 High-level Insight: Shape Invariance

Our insight to tackle the contextual and non-stationary nature of viewership modeling is that the *shape* of the viewership curve (formally defined below) is consistent over multiple weeks. Figures 4.1 and 4.3 visually suggest that despite the variability in magnitude for two viewership groups, the *shape* of each group is consistent across weeks.

While this visual result is appealing, a natural question is whether this observation holds more generally across groups, weekdays, and content providers. To this end, we compute the pairwise structural similarity of the viewership timeseries for all (group, weekday) tuples

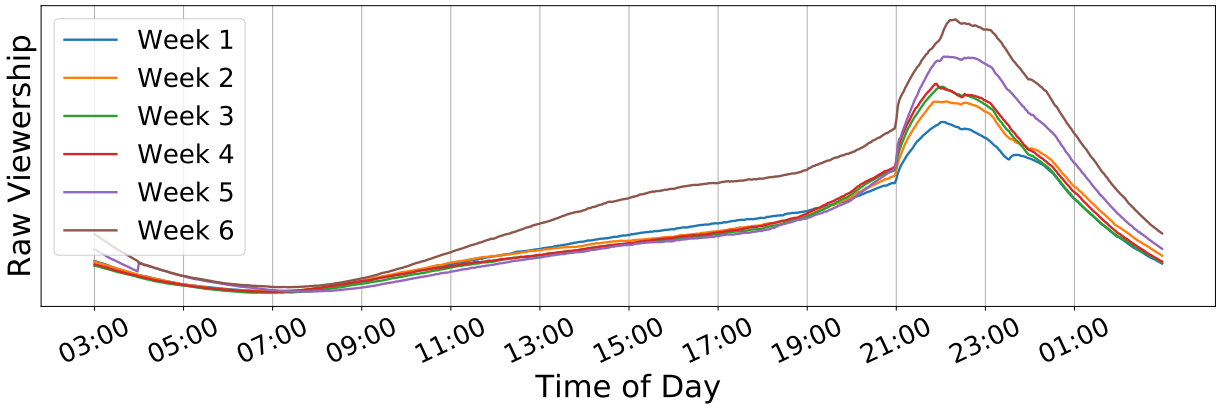



Figure 4.3: Intuition on the invariability of shape of viewership.

through their *cosine distance*. The cosine distance of two vectors quantifies structural similarity through their angular difference. A value of 0 indicates a perfect match [9]. Figure 4.4 shows these results for two content providers and a testing period of 3 months. Indeed, we see that (i) when the two curves are from the same group and weekday, the cosine distance is both low and minimal and (ii) this observation does not necessarily hold for any random choice of groups.

**Problem formulation** The above observation leads to a shape-based detection workflow described below (Figure 4.5).

Let $g$ indicate a viewership group and $\mathrm{v}^{\mathrm{d}}_{\mathrm{g,w}}$ be its timeseries of viewership on weekday w (*e.g.,* Monday) and date d (*e.g.,* March 23). Let T be the set of all timestamps in one day and $D$ the set of all dates. $\mathrm{s}^{\mathrm{d}}_{\mathrm{g,w}}$ is a timeseries of the *shape* of viewership derived from $\mathrm{v}^{\mathrm{d}}_{\mathrm{g,w}}$ after eliminating magnitude variability for that day:

$$\mathrm{s}^{\mathrm{d}}_{\mathrm{g,w}}(\mathrm{t}) = \mathcal{Z}^{\mathrm{d}}(\mathrm{v}^{\mathrm{d}}_{\mathrm{g,w}}(\mathrm{t})) = \frac{\mathrm{v}^{\mathrm{d}}_{\mathrm{g,w}}(\mathrm{t}) - \mu^{\mathrm{d}}_{\mathrm{g,w}}}{\sigma^{\mathrm{d}}_{\mathrm{g,w}}} \tag{4.1}$$

Let $\mathcal{M}_{\mathrm{g,w}}$ denote our prediction model trained using *summaries* of historical observations of $\mathrm{s}^{\mathrm{d}}_{\mathrm{g,w}}$, $\forall\ \mathrm{d} \in D$ (Figure 4.5). $\mathcal{M}_{\mathrm{g,w}}$ takes as input a timestamp t and outputs the corresponding estimate $\hat{\mathrm{s}}_{\mathrm{g,w}}(\mathrm{t})$ along with a confidence bound $\hat{\mathrm{u}}_{\mathrm{g,w}}(\mathrm{t})$ as shown in (4.2):

$$\mathcal{M}_{\mathrm{g,w}}(\mathrm{t}) = \langle \hat{\mathrm{s}}_{\mathrm{g,w}}(\mathrm{t}),\ \hat{\mathrm{u}}_{\mathrm{g,w}}(\mathrm{t}) \rangle, \quad \forall t \in \mathrm{T}\,. \tag{4.2}$$

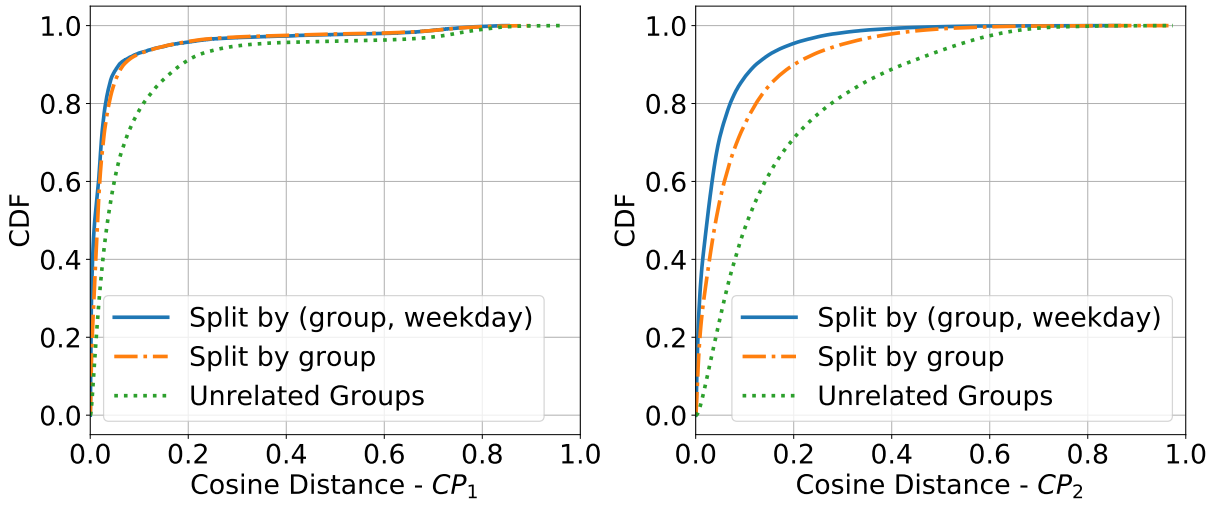



Figure 4.4: Viewership curves of the same group and weekday exhibit high structural similarity, unlike unrelated pairs of viewership curves.

Having $\mathcal{M}_{g,w}$, we can determine whether viewership $v^d_{g,w}(t')$ is anomalous. First, we compute $s^d_{g,w}(t')$ using (4.1). In an offline setup where $v^d_{g,w}$ is apriori known, computing $s^d_{g,w}(t')$ is straightforward because $\mu^d_{g,w}$ and $\sigma^d_{g,w}$ are known. However, in the online scenario where $\mu^d_{g,w}$ and $\sigma^d_{g,w}$ are unknown, they need to be approximated (§4.3.4). Ultimately, $s^d_{g,w}(t')$ is marked as anomalous if it doesn't lie within the interval defined by $\hat{s}_{g,w}(t') \pm c \cdot \hat{u}_{g,w}(t')$, where $c$ is a parameter determining the width of the confidence bound [5].

Given this formulation, ideally $\mathcal{M}_{g,w}$ should be: (1) *General* to model a large (and a priori unknown) number of shapes; (2) *Automated* and require minimal manual tuning to handle 100k+ viewership groups; (3) *Robust* to outliers in historical data and (4) *Compact* in terms of space/time efficiency of storing and using models.

### 4.3.2 A Case for Using Gaussian Processes

Unfortunately, meeting all of these requirements is challenging (Table 4.2). To see why, consider two candidate solutions. One class of work uses basis functions (*e.g.,* polynomials [54], wavelets [39], redundant dictionaries [37]) or PCA (as done by Netflix's Surus [25]) to decompose and compactly model traffic timeseries using a few coefficients. However,

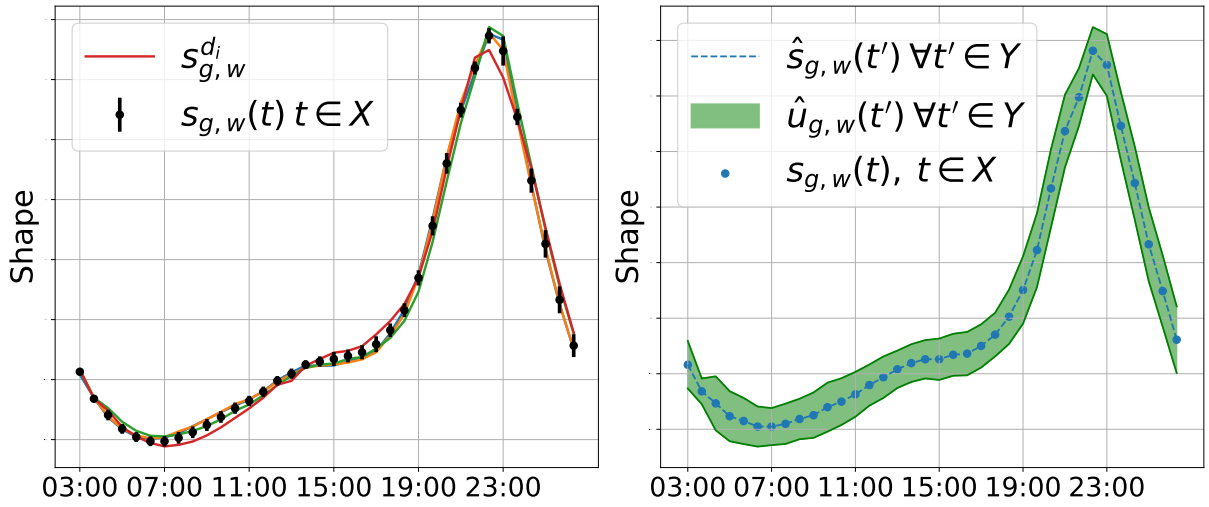



Figure 4.5: Shape, training observations and $\mathcal{M}_{\text{g,w}}$.

| | General | Automated | Robust | Compact |
|---|---|---|---|---|
| Decomposition | **N** | **N** | **Y** | **Y** |
| Pointwise Estimation | **Y** | **Y** | **N** | **N** |
| Vanilla GP | **?** | **Y** | **?** | **Y** |
| PROTEAS GP | **Y** | **Y** | **Y** | **Y** |

Table 4.2: Strawman solutions vs. PROTEAS's detection approach.

it is intractable to manually identify and tune the basis set for a large number of diverse groups. On the other hand, we can use historical traces to compute an empirical point-wise mean-variance for each (d, w, t) combination. Unfortunately, this is not resilient to outliers as a single surge or missing data observation can skew our model. Furthermore, maintaining these point estimates for a large number of groups is not compact. For instance, ARIMA models are not robust to outliers and are not compact as they require long history windows [167]. The same compactness argument applies to LSTM neural networks and other deep learning-based approaches [86, 159].

**Why Gaussian Processes** Rather than look for a universal basis function, we revisit the idea of *stochastic Gaussian Processes* [89, 139]. Conceptually, a Gaussian Process (GP) is a probability distribution over all possible shape functions. Due to this non-parametric nature, GPs can model *any shape with minimal apriori structural assumptions*. GPs enable simple, robust, and automated learning procedures that eliminate the need for manual tuning [139].

**Primer on GP** Figure 4.5 visualizes a GP-based detection model to learn $\mathcal{M}_{g,w}$ from history. The GP models both the underlying shape and the uncertainty such that, for any unknown timestamp $t'$, we can produce (through interpolation) an $\hat{s}_{g,w}(t)$ and appropriate error bounds. We refer readers to [89, 139] for more details beyond this high-level primer.

The key assumption behind GPs is that each observation $s^{d}_{g,w}(t)$ is a *sample* drawn from a normally distributed random variable $S_{g,w}(t)$. Making a normality assumption is not unreasonable. Recent work suggests using a student distribution but the performance benefits are marginal [160]. Then, the entire timeseries $s^{d}_{g,w}$ then can be seen as a sample drawn from the joint (Gaussian) distribution of the set of random variables $S_{g,w}(T) \coloneqq \{S_{g,w}(t)\}_{t,\ \forall\ t \in T}$.

Let $X \subseteq T$ be the set of all timestamps where viewership is observed and $Y$ the set of timestamps where viewership is unknown. We let $S_{g,w}(X)$ and $S_{g,w}(Y)$ be the vectors of all random variables that correspond to timestamps in $X$ and $Y$ respectively. The joint distribution $S_{g,w}(T)$ then becomes:

$$S_{g,w}(T) \coloneqq \begin{bmatrix} S_{g,w}(X) \\ S_{g,w}(Y) \end{bmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \text{where} \tag{4.3}$$

$\boldsymbol{\mu}$ is the mean vector of the distribution and $\boldsymbol{\Sigma}$ its covariance matrix. For simplicity, $\boldsymbol{\mu}$ is commonly set to 0.

Given (4.3), making point estimates about $\hat{s}_{g,w}(t'),\ \ t' \in Y$ reduces to computing the conditional expectation $\boldsymbol{E}[S_{g,w}(Y) \mid S_{g,w}(X),\ X,\ Y]$ at $t' \in Y$. Then, $\hat{u}_{g,w}(t')$ becomes the variance of the conditional distribution at $t'$.

### 4.3.3 Using Gaussian Processes in PROTEAS

Given this background on GP, two questions remain:

1. **Modeling $S_{g,w}$ with time-dependent noise:** Traditional GP assumes that their output is either noise-free or the noise is time-independent. In viewership, however, noise levels depend on the time of day, which needs to be considered when modeling $S_{g,w}$, without breaking the theoretical guarantees of GPs.
2. **Modeling covariance $\mathbf{\Sigma}$:** The covariance matrix $\mathbf{\Sigma}$ is at the heart of a GP as it encodes the impact that neighboring observations have on viewership estimates at time $t'$ [139]. To learn $\mathbf{\Sigma}$, GPs commonly use appropriate *kernels* $\kappa(t, t', \theta)$ *i.e.,* covariance functions that take as input a set of hyperparameters $\boldsymbol{\theta}$, timestamps t, $t'$ and return the corresponding covariance between $S_{g,w}(t)$, $S_{g,w}(t')$. Thus, modeling $\mathcal{M}_{g,w}$ boils down to a suitable choice of kernel(s) and optimizing their hyperparameters $\boldsymbol{\theta}$.

We discuss how we address these two key issues in PROTEAS.

**Extending GP to support time-dependent variance** We incorporate noise in our model using the intuitive definition of the shape of viewership. We define $S_{g,w}$ as follows:

$$S_{g,w}(t) \coloneqq \bar{S}_{g,w}(t) + \bar{\epsilon}_{g,w}(t) \quad \forall\, t \in T, \quad \text{where} \tag{4.4}$$

$$\bar{S}_{g,w}(t) \sim \mathcal{N}\Big(\frac{1}{|D|}\sum_i s^{d_i}_{g,w}(t),\ 0\Big) \quad \text{and} \quad \bar{\epsilon}_{g,w}(t) \sim \mathcal{N}(0,\ \sigma_t^2)$$

Specifically, each latent random variable $S_{g,w}(t)$, $t \in X$ has a normal distribution whose mean is the mean of historical shape observations at time t and a noise component $\sigma_t^2$ equal to the standard deviation of $s^d_{g,w}(t)$, $\forall\, d \in D$.

To extract the desired objectives $\hat{s}_{g,w}(t')$, $t' \in Y$ and $\hat{u}_{g,w}(t')$, we first model the joint Gaussian distribution of $S_{g,w}(X)$ and $S_{g,w}(Y)$. We use the notation $K_{XY}$ to refer to the covariance matrix of $S_{g,w}(X)$ and $S_{g,w}(X)$ (and apply the same notational logic to all four possible combinations of $S_{g,w}$ sets). Formally:

$$\begin{bmatrix} \mathrm{S_{g,w}(X)} \\ \mathrm{S_{g,w}(Y)} \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \quad \text{where} \tag{4.5}$$

$$\mathbf{\Sigma} \coloneqq \begin{bmatrix} \mathrm{K}_X + \sigma^2_{\mathrm{g,w}}(\mathrm{X})\mathrm{I} & \mathrm{K}_{XY} \\ \mathrm{K}_{XY} & \mathrm{K}_{YY} \end{bmatrix}.$$

Note that according to GP theory, the mean of the joint distribution can be safely set for notational simplicity to 0. Using (4.3), the formulas for $\hat{s}_{g,w}(t')$ and $\hat{u}_{g,w}(t')$ become:

$$\begin{aligned} \hat{\mathrm{s}}_{\mathrm{g,w}}(t') &= \boldsymbol{E}[\mathrm{S_{g,w}(Y)} \mid \mathrm{S_{g,w}(X)},\ \mathrm{X},\ \mathrm{Y}] \\ &= \mathrm{K}_{XY}^T[\mathrm{K}_X + \sigma^2_{\mathrm{g,w}}(\mathrm{X})\mathrm{I}]^{-1}\boldsymbol{E}[\mathrm{S_{g,w}(X)}] \end{aligned} \tag{4.6}$$

$$\begin{aligned} \hat{\mathrm{u}}_{\mathrm{g,w}}(t') &= \boldsymbol{V}[\mathrm{S_{g,w}(Y)} \mid \mathrm{S_{g,w}(X)},\ \mathrm{X},\ \mathrm{Y}] \\ &= \mathrm{K}_Y - \mathrm{K}_{XY}^T[\mathrm{K}_X + \sigma^2_{\mathrm{g,w}}(\mathrm{X})\mathrm{I}]^{-1}\,\mathrm{K}_{XY} \end{aligned} \tag{4.7}$$

**Choice of covariance kernel** Kernels intuitively represent families of functions (*e.g.,* periodic, differentiable *etc.*) and allow us to incorporate prior knowledge on the shape of viewership. Note that the choice of kernel is a much less restrictive process in terms of prior assumptions compared to strawman solutions [77]. We identify the following structural characteristics that $\kappa(\mathrm{t}, \mathrm{t}', \boldsymbol{\theta})$ should account for; (1) *Decaying periodicity:* Viewership often exhibits a periodic structure with values dropping in early morning in order to rise again during prime time hours; (2) *Lack of smoothness:* Viewership exhibits sharp change points (*e.g.,* at the beginning or end of a show) that need to be included; and (3) *Time-dependent noise.* To address (1), we combine a periodic and a squared exponential kernel [18, 23]. To capture (2), we add non-smoothness with a Matérn kernel [16]. Finally, for (3) we use a white-noise kernel, tweaked to account for time-dependent noise [17]. Our choice of kernels is as follows:

$$\kappa_{\text{periodic}}(\mathrm{t},\mathrm{t}') = \alpha^2 \exp\left(-\frac{2\sin^2(\pi|\,\mathrm{t}-\mathrm{t}'\,|/p)}{\ell^2}\right) \tag{4.8}$$

$$\kappa_{\text{sqExp}}(\mathrm{t},\mathrm{t}') = \gamma^2 \exp\left(-\frac{(\mathrm{t},\mathrm{t}')^2}{2\ell^2}\right) \tag{4.9}$$

$$\kappa_{\text{Matern}}(\mathrm{t},\mathrm{t}') = \frac{\beta^2}{\Gamma(\nu)2^{\nu-1}}(\frac{\sqrt{2\nu}(\mathrm{t},\mathrm{t}')}{l})^{\nu} B_{\nu}(\frac{\sqrt{2\nu}(\mathrm{t},\mathrm{t}')}{l}) \tag{4.10}$$

$$\kappa_{\text{wNoise}}(\mathrm{t},\mathrm{t}') = \sigma^2_{\text{g,w}}(\mathrm{X})\mathrm{I} \tag{4.11}$$

**Proof of validity for noise kernel** Typically, when defining a valid noise kernel the resulting covariance matrix needs to be positive semidefinite. Below we prove that the modified noise kernel above (where $\sigma^2_{\text{g,w}}(\mathrm{X})\mathrm{I}$ is a function of the training timestamp) is positive semidefinite. We now argue that the matrix inverse above is legitimate. First, it is well-known that a covariance matrix is always a positive semidefinite matrix [139]. Second, since $\sigma > 0$, $\sigma^2 I$ is a positive definite matrix. As the sum of a positive semidefinite matrix with a positive definite matrix is positive definite, XYZ is positive definite and therefore invertible.

**Final kernel version** Kernels can be combined together through addition or multiplication [139] and the final version of our kernel is as shown below. After a choice of kernel has bee made by the analyst, a fully automated fitting process is carried out to optimize its hyperparameters $\boldsymbol{\theta}$ [139].

$$\kappa_{\mathcal{M}_{\text{g,w}}} = \kappa_{periodic} * \kappa_{sqExp} + \kappa_{Matérn} + \kappa_{wNoise} \tag{4.12}$$

### 4.3.4 Online Detection

Having learnt $\mathcal{M}_{g,w}$, the next step is to use (4.1) to detect in real-time whether a newly observed viewership value is anomalous or not. However, recall that the standardization operator $\mathcal{Z}^{\text{d}}$ is date-dependent, *i.e.,* it requires knowledge of the (unknown) mean $\mu^d_{g,w}$ and the standard deviation $\sigma^d_{g,w}$ for the current date $d$, thus raising our next challenge.

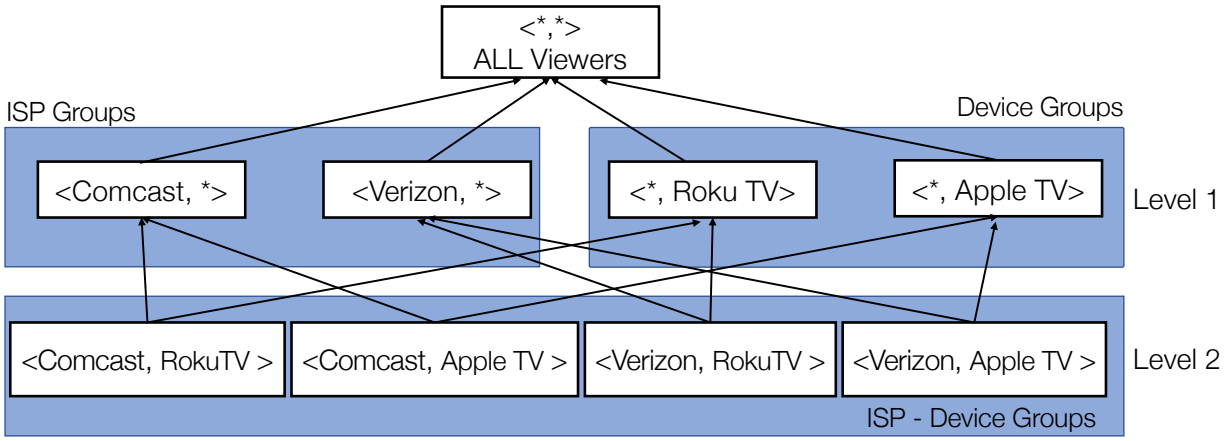



Figure 4.6: Lattice of session groups

To address this issue, we resort to a simple iterative algorithm for approximating $Z^d_{g,w}$ for the current date. Our approach relies on the intuition that we can begin with an approximation of $\mu^d_{g,w}$ and $\sigma^d_{g,w}$ estimated with historical data of the most recent past observation for that weekday (*e.g.,* data from the past week) and refine these estimates as new observations of the current date arrive. More specifically, we maintain a very sparse representation of historical observations of raw viewership (*e.g.,* sampled every 30 minutes for a total of 48 samples). Every 30 minutes we replace one stale observation with the corresponding most recent viewership observation of the current date. This heuristic enables us to compute an estimate of $Z^d$ that quickly converges to the optimal value.

## 4.4 Generating Actionable Alerts

Next, we present the PROTEAS's diagnosis module. This module takes as inputs the set of group-specific viewership anomalies and provides: (1) a summarized alert consisting of the top-k viewership groups that best explain the incident and (2) a small set of likely root causes.

### 4.4.1 Summarizing Viewership Anomalies

An anomalous incident often manifests itself redundantly across multiple viewership groups. Naively issuing an alert for every such anomaly increases the analysts' workload for diagnosis and degrades the overall usability of the framework. For instance, in the case of the buggy firmware update for RokuTV discussed in Chapter 2, the resulting outage (that was only caught though viewership) manifested at more than 100 different viewership groups. Naively raising 100 redundant alerts would compromise PROTEAS's usability. Thus, we need a mechanism that identifies the *critical* viewership group(s) that can best summarize the detection anomalies. To see why this is challenging, consider two strategies from prior work [53, 102, 104].

1. **Summarize by group succinctness** [102] Jiang *et al.* in their work on video QoE suggest that an anomaly is best explained by the most *succinct* (*i.e.,* in terms of number of attributes) session group whose anomalous sessions, when removed, result in the remaining groups becoming "healthy" (*i.e.,* fraction of anomalous sessions drops below a threshold). Quantifying a group's "health" through its fraction of anomalous sessions means we can freely remove subsets of sessions and measure the resulting changes in that group's health. In PROTEAS, however, such an approach would alter the baseline shape of a group and would introduce additional complexity.
2. **Summarize by "surprise"** [53] Prior work by Bhagwan *et al.* in summarizing anomalies in ad-systems argues that succinctness alone is insufficient and reports "surprising" data partitions that exhibit the greatest relative change with respect to its expected value. This is built upon a domain-specific observation about ad systems, that anomalies typically appear in coarse one-dimensional groups (*e.g.,* at ISP or CDN level). However, this does not hold in our scenario as anomalies can originate both in coarser (*e.g.,* viewership in SF drops) and in finer groups (*e.g.,* NYC-based viewers on Comcast using RokuTV have an outage).

While these strawman solutions do not directly apply in our context, we can learn from them. From Jiang *et al.* we observe that video incidents manifest in a natural *hierar-*

*chical structure* across the space of session attributes, which also applies to viewership (Figure 4.6). From Bhagwan *et al.* we can build on the insight of finding a small/sparse set of root causes that have most *explanatory power*. Yet, it is not clear how to define the explanatory power of a group in our context. To that end, we build on the following two observations: First, it is rare for multiple independent anomalous incidents to occur simultaneously. That is, the set of underlying root-causes is *sparse*. This intuitively suggests that most (if not all) anomalous events can be traced back to a few critical groups. Second, anomaly *propagation* follows two natural structural rules:

1. **Propagation to descending nodes.** For each group $g$, each of its descendants accounts for a fixed share of $g$'s viewership. When an anomalous incident originates at $g$, each descendant will experience a viewership change proportional to their share of $g$'s viewership. By the same token, the change propagates recursively to all descendants in the hierarchy. In the ⟨RokuTV⟩ case, the anomaly also appeared at ⟨RokuTV, Comcast⟩, ⟨RokuTV, Verizon⟩, and all members of the finer ⟨Player, ISP⟩ set.
2. **Propagation to parent nodes.** The viewership of $g$'s parent is the sum of $g$'s viewership plus that of the parent's other descendants with whom $g$ shares the same attributes[2]. As a result, when an anomalous incident originates at $g$, the parent node experiences a change in viewership nominally equal to $g$'s. That changer, however, will be less pronounced as the relative change of viewership is smaller in the larger group. By the same token, the change will propagate recursively up until the root.

Using the above rules, our strategy for summarization boils down to quantifying for each anomalous group $g$ its *Explanatory Power ($EP_g$)*, a score that quantifies how well group $g$ explains the manifestation of the incident in the remaining anomalous groups. We use the following heuristics to estimate $EP_g$;

[2]If we partition the direct descendants of each node by session attributes, the sum of viewership within each resulting partition equals the viewership of the node.

1. **Range of impact.** If $g$ explains the anomalous incident, the incident should (i) simultaneously manifest in $g$'s direct descendants and (ii) the cumulative change in traffic of the anomalous descendants, grouped by combination of attributes $AC$, is approximately equal to the change in $g$'s viewership. Intuitively, given the first propagation rule, the fewer anomalous descendants $g$ has, the lower its explanatory power should be. We quantify this quantity as follows:

$$H_{g,1} = \min\{\frac{\#anomDescendants}{\#allDescendants}\}_{AC} \tag{4.13}$$

2. **Surprise.** If $g$ explains the anomalous incident, its change in viewership *i.e.,* the difference between the predicted viewership value and the observed one, relative to other anomalous groups should be large. This is inspired by the notion of "surprise" introduced in prior work [53]. We quantify this measure of change as:

$$H_{g,2} = \frac{|\widehat{V_g} - V_g^{obs}|}{max\{|\widehat{V_{g'}} - V_{g'}^{obs}|\}} \tag{4.14}$$

Given these heuristics, we formally define the $EP_g$ as $EP_g = H_{g,1} \times H_{g,2}$, where a higher value of $EP_g$ indicates a better ability to explain the anomaly. Our heuristic computes this for all groups and outputs the top-$k$ groups. Since our goal is to identify a suitable starting point for further investigation, we believe that releasing the top-$k$ groups is a more realistic approach than releasing only the top group. As we will see in §4.6, in practice, the true answer lies within a small $k$ ($< 3$).

### 4.4.2 Root Cause Suggestions

The final step is to annotate the summarized incident with candidate root causes. Since many factors are invisible to Proteas (*e.g.,* ISP upgrades), our goal in practice is not exact root-cause attribution but to reduce the analyst's overhead by providing a set of candidates for further investigation.

Our high-level intuition is that each anomalous event can be associated with a set of *anomaly signatures*. Consequently, we can match a new anomaly against a library of

signatures learned from history to determine (in real time) a set of candidate root-causes. For instance, in the RokuTV example, the intrinsic properties of the anomaly (*e.g.,* its shape, duration, start time, the critical viewership group *etc.*) potentially indicate that the underlying root-cause is an outage. Based on our data-driven insights, we see that this signature has two facets: (1) a *spatial* dimension capturing the groups that have been impacted by the anomaly and their relationships in the group hierarchy; and (ii) a *temporal* dimension such as its duration or its views-per-time pattern.

**Spatial dimension of anomaly signatures** An intuitive way to capture the spatial component is to to enumerate the anomalous viewership groups. For instance, for our RokuTV outage, we identify all the related anomalous groups. However, this approach cannot generalize to other occurrences of similar events; *e.g.,* if FireTV manifested the same anomaly, we cannot match the signature. Thus, we extend this basic idea and represent the spatial characteristics of an anomaly signature at an *attribute-level* granularity (*i.e.,* Device and descendants), rather than specific groups (*i.e.,* RokuTV). To this end, we create a hierarchy of attribute groups and mark the nodes that contain anomalous groups. Matching a new anomaly to an anomaly signature is a simple comparison of the fraction of shared nodes.

**Temporal dimension of signatures** In practice, we find that the shape of the anomaly remains largely consistent across affected groups. This has two implications with respect to the anomaly's temporal signature. First, to isolate a signature's temporal component, it suffices to characterize one of the affected groups. Second, matching an anomaly to our library, requires a timeseries similarity metric.

However, there are two challenges. The detection time of an anomaly does not necessarily coincide with the incident's start time as viewership might have been anomalous for a while before we flag it. Thus, to extract the temporal pattern of the anomaly, we also include a short window of viewership history (empirically set to 15 mins) to ensure the onset of he incident is not missed. Second, two similar events with the same temporal manifestation

may still differ in duration, suggesting the use of Dynamic Time Warping (DTW) [52] for comparisons.

**Building a library of signatures** Given this approach, we explain the workflow for bootstrapping and maintaining a library for associating anomalous events with their respective signatures. To bootstrap the library, we first generate "raw" signatures as described above corresponding to different anomalies. Then, we systematically group together possibly similar signatures, and ultimately determine whether there is a representative pattern that can be associated with an anomalous event. To this end, we apply DTW across all pairs of "raw" signatures and then apply a clustering algorithm on the resulting distances. Updating the library, however, requires some analyst intervention. For every new event that does not map to a known signature from our library, we defer to the analyst's forensics investigation to diagnose the event and add its signature to the library.

Our signature library consists of 4 broad categories of incidents; *(1) Technical issues* for alerts warranting technical intervention; *(2) Flash crowds*, encompassing short-lasting viewership surges, scoped by the duration of the piece of content they manifest in; *(3) Model Drift* for changes in viewership being the result of change in offered content or viewing habits; *(4) Measurement errors*, which included short lasting drops due to the potential loss of session batches or system updates *etc.*

## 4.5 Implementation

We implement PROTEAS on Apache Spark [166]. PROTEAS receives batches of raw video sessions every minute which are then grouped by combinations of session attributes.

**Detection** There are two practical issues. First, with detection running every minute for tens of thousands of groups per cluster node, we need compact models to ensure minimal storing and fetching cost. Second, to capture model drift, PROTEAS needs to periodically retrain models.

Given our GP-based approach (§4.3), there is natural tradeoff between model fidelity and the temporal granularity (*i.e.,* size of the set of observed timestamps $X$). Intuitively if $X$ incorporates more prior observations, the higher the model fidelity. However, larger $X$ implies higher overhead to store necessary parameters. Empirically, we achieve a reasonable tradeoff when timestamps in $X$ are sampled at 5-minute intervals ($\sim$ 5KB per group).

Any learning approach in practice has to tackle *model drift*; *e.g.,* content popularity, or viewership habits change over time. To that end, PROTEAS performs an offline retraining of viewership models daily (of the subset of models corresponding to that weekday). Specifically, for each timestamp in $X$, we recompute its summary statistics to incorporate the most recent observations of the shape of viewership. To prioritize recent observations over older ones, we estimate summary statistics using exponentially weighted moving averages and standard deviations and observe that re-training is lightweight (<20 sec per group).

**Diagnosis** The diagnosis process begins the moment an anomaly is first detected. We iteratively compare the current anomaly with entries in the library. For some events, our confidence increases as the anomaly progresses over time. For other abrupt events (*e.g.,* ISP outage), the onset of the anomaly is often sufficient in order for PROTEAS to reach an accurate diagnosis soon (*e.g.,* within 10 minutes of the onset). When we do not find a reasonable match in the library, PROTEAS informs the operator of the need to start a manual root-cause attribution analysis.

## 4.6 Evaluation

In this section, we show that PROTEAS is: (1) *accurate* with a mean True Positive Rate (TPR) >86% in the common case; (2) *adds value* to operators, with ≈50% of its alerts being distinct from alerts by other QoE-tracking tools; (3) provides *useful alerts*, within minutes after the incident's onset; and (4) is *usable* as it produces a manageable volume of daily alerts (up to 3 alerts per incident), thus preventing alert fatigue. We compare PROTEAS with the two closest state-of-the-art alerting frameworks; (1) Twitter's offline statistical

anomaly detector [95] and (2) PCA-based Surus by Netflix [25] and show that PROTEAS is more accurate than prior work and that analysts in a blind survey trust PROTEAS's alerts over those of prior work ≈93% of the time.

### 4.6.1 Methodology

We begin with a real-world study to assess the system's overall value (§4.6.2) and complement it with synthetic, trace-driven sensitivity analysis of PROTEAS and its components (§4.6.4).

**Setup** Table 4.3 summarizes key features of our real-world dataset. Our dataset comes from three content providers offering different types of service; $CP_1$ mostly offers VoD content across three channels. $CP_2$ is a Linear TV provider *i.e.,* it combines scheduled content and scheduled live transmissions under one channel. Finally, $CP_3$ combines VoD, scheduled content and live transmissions under 164 (regional and US-wide) channels.

| **Feature** | **$CP_1$** | **$CP_2$** | **$CP_3$** |
|---|---|---|---|
| Dataset | January - March 2020 | | |
| Duration of Pilot | March 2020 (4 weeks) | | |
| Geography | USA | EU | USA |
| Session Attributes | 7 | 7 | 9 |
| Viewership Groups (Average) | 3000 | 650 | 5500 |
| Channels | 3 | 1 | 164 |
| Live/VoD ratio (Average) | 0 | 0.2 | 10 |

Table 4.3: PROTEAS's real-world dataset

**Ethical considerations** To the extent that the dataset tracks actual viewer behavior, our dataset does concern real Internet users. All viewership data used in this work are covered by NDAs prohibiting any re-sharing with 3rd parties even for research purposes. Further,

raw viewership data have been reviewed and validated by the operator with respect to GPDR compliance (*e.g.,* no identifier can be associated to person), and data processing only looks at viewership counts per combination of session dimension values. No personal and/or contact information was available in the viewership data used for this study. The data for the ground truth collection (*e.g.,* Twitter) also entail user data but they are in the public domain. We do not use user/poster names, only the contents relevant to the incidents under evaluation.

**Ground truth** A key challenge in this domain is the lack of ground truth for real-world anomalies. This challenge is not unique to our work and prior works attempt to address this issue through manual data labelling [53, 95, 160]. For PROTEAS, in addition to expert analysis, we leverage complementary sources of ground truth to augment this standard approach and to reduce potential biases. Specifically:

1. *Twitter-based sentiment analysis:* During an outage, viewers often resort to Twitter to express annoyance or to reach out to operators. We run a simple sentiment analysis to expose viewer reactions that allow us to correlate a viewership alert with its ground truth [132]. To reduce the risk of diluting our analysis with unrelated tweets (*e.g.,* personal opinions about quality of show *etc.*), we, limit this analysis to providers' public help-desk Twitter profiles. We do acknowledge the potential bias toward users who are more active on Twitter.
2. *Public downtime records:* We collect uptime reports for Internet services from public databases [10]. Given that video outages can result from many other related services (ISPs, CDNs, content provider portals *etc.*), we collect downtime reports for all of these services.
3. *Video QoE alerts:* Our third source of ground truth consists of video QoE alerts that the analytics provider runs already. Given that some QoE issues can lead to viewership drops (as viewers quit their sessions due to poor experience), this can help validate some viewership alerts.

4. *Calendar of holidays and major events:* Days like Thanksgiving are singular events and tracking these enables interpreting PROTEAS's alerts accordingly.
5. *Blind review by expert:* Finally, we leverage the expertise of 4 video analysts to confirm the presence of an anomaly. To minimize biases in the labeling process, we set up a 2-phase survey. First, the analyst is presented with the unlabeled timeseries of the date of interest as well as recent timeseries corresponding to the same weekday and is asked to identify anomalous segments (if any). Then, the analyst is presented with the alerts issued by PROTEAS and the prior works we compare against (§4.6.2) on that timeseries and is asked to mark the outputs they agree with (again, if any). The events of the second phase of the survey were chosen and presented randomly to ensure there was no association with the data of the first phase.

**Labelling strategy** We compile the ground truth as the union set of incidents from the above sources and check whether a detected anomaly matches with any of the anomalies in this set. If yes, we mark it as a True Positive (TP). Otherwise, we conservatively mark it as a False Positive (FP). For anomalies in the ground truth set that do not match with any of PROTEAS's anomalies, we manually confirm the presence of a False Negative (FN). This is because an anomaly in a different context (*e.g.,* QoE metrics) might not necessarily translate to a viewership anomaly. Naturally, we acknowledge that our labelling strategy does not provide us with the perfect ground truth, but only with a subset of it.

**Evaluation with synthetic trace** To complement the pilot study, we also evaluate PROTEAS with synthetic viewership data. Specifically, we leverage our understanding of viewership anomalies from the real-world dataset and build an anomaly generator that enables injecting custom anomaly signatures on healthy segments of the original viewership trace. This allows us to run a synthetic analysis with perfect knowledge of ground truth.

**Accuracy metrics** To evaluate the accuracy of PROTEAS (and of the baselines), we measure its TP, FP and FN counts. In addition, we also compute the frameworks'

precision (TP/(TP+FP)). For our synthetic analysis in §4.6.4, we also measure recall (TP/(TP+FN)) and F-score [12, 19].

### 4.6.2 Pilot Study: End-to-End Evaluation

| | Metric | W1 | W2 | W3 | W4 |
|---|---|---|---|---|---|
| $\mathbf{CP}_1$ | Alerts | 8 | 10 | 136 | 138 |
| | False Positives | 1 | 1 | 22 | 50 |
| | Precision (%) | 87.5% | 90.9% | 85.5% | 73.4% |
| $\mathbf{CP}_2$ | Alerts | 25 | 23 | 63 | 59 |
| | False Positives | 3 | 4 | 18 | 13 |
| | Precision (%) | 89.2% | 85.1% | 77.9% | 81.9% |
| $\mathbf{CP}_3$ | Alerts | 85 | 92 | 165 | 211 |
| | False Positives | 17 | 15 | 42 | 85 |
| | Precision | 83.3% | 85.9% | 79.7% | 71.2% |

Table 4.4: PROTEAS's key accuracy metrics.

Table 4.4 summarizes PROTEAS's key accuracy metrics per testing week ($W_x$) and content provider ($CP_y$). $W_1$ and $W_2$ reflect PROTEAS's expected operational scenario. $W_3$ and $W_4$ correspond to onset of the shelter-in-place period due to COVID-19 in USA, thus a period of fundamentally altered viewership patterns. Table 4.4 shows that in the common case, the number of *weekly* alerts issued by PROTEAS is low. $CP_1$ is a big VoD provider with persistent viewership patterns and incurs the fewest alerts (<10 per week). $CP_2$ and $CP_3$ raise higher numbers of alerts in the common case. This is due to their higher percentage of live content offered (which can be more unpredictable in terms of viewership). In addition, in the case of $CP_3$, alerts are often traced back to the provider's many channels which are not grouped together during summarization. Nevertheless, across all content providers, we can say that the number of alerts raised results in a manageable workload for human analysts.

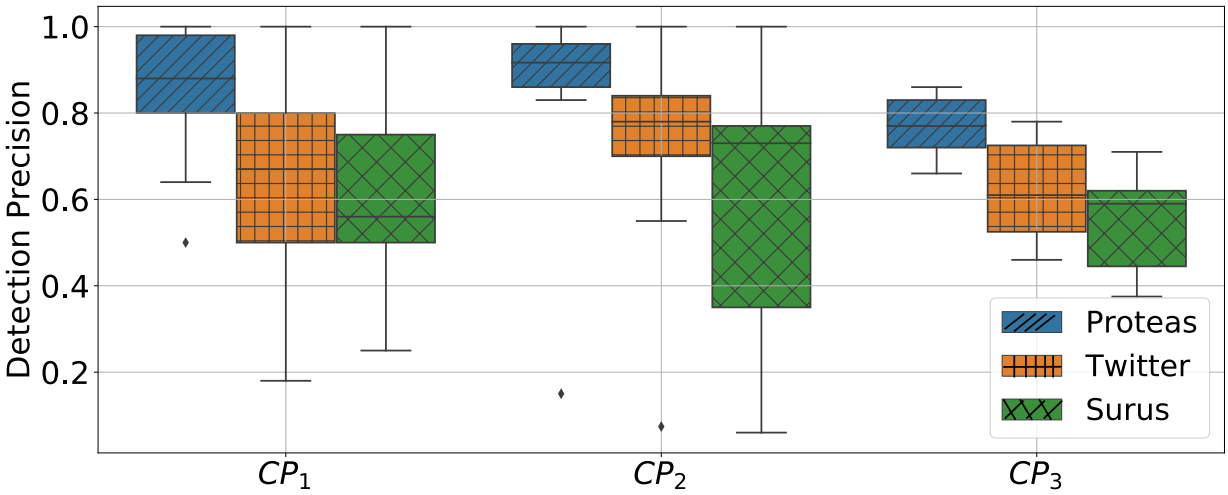



Figure 4.7: Comparison of end-to-end precision analysis.

We also observe that PROTEAS's precision is consistently >85% in the expected operational region indicating that PROTEAS incurs few false positives. During the pandemic period, however, we see precision dropping slightly as a result of pervasive changes in viewership behaviors. Regarding FN, we find that PROTEAS does not miss any major incidents.

**Comparison with prior work** We compare PROTEAS with (1) Twitter's offline statistical anomaly detector [95] and (2) PCA-based Surus by Netflix [25]. These prior works only perform anomaly detection, so we combine their outputs with PROTEAS's summarization routine. Figure 4.7 shows the distribution of their daily precision values and shows that PROTEAS consistently outperforms prior work. In the cases of $CP_1$ and $CP_2$ PROTEAS's median precision is 88% and 90% respectively whereas for $CP_3$ it is 79%. On the other hand, Twitter's median precision for the three providers equals to 68.5%, 78% and 65% respectively, and for Netflix's Surus these values become 61%, 71% and 59%. Further investigation of these precision gaps indicates that while prior work exhibits similar true positives as PROTEAS, it produces higher numbers of false positives (all frameworks have the same nominal sensitivity).

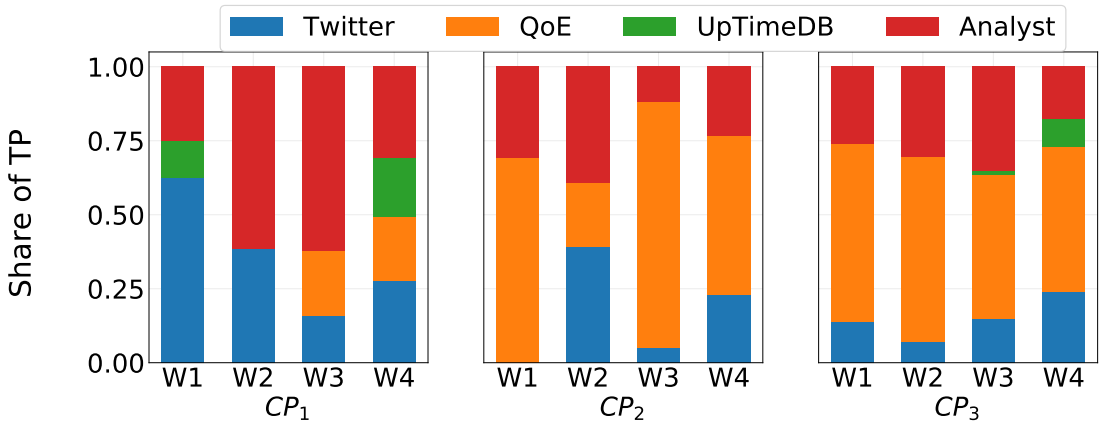



Figure 4.8: Breakdown of true positives by source of ground truth.

**Trust in PROTEAS's alerts** In addition to precision, we measure the "trust" our 4 experts show to each framework's alerts. Recall that as part of ground truth extraction (§4.6.1), experts were asked to choose between three different, anonymized outputs per incident (one per framework examined), the ones that subjectively capture best each incident. Among a sample of 30 different incidents, experts chose PROTEAS's output ≈93% of the time and also indicated that they would also agree with a competing framework for ≈45% of the samples.

**Value to content providers** To quantify PROTEAS's added value to an operator's toolbox, we measure what percentage of its true positives coincide with alerts issued by existing QoE-tracking tools [102, 104]. Figure 4.8 illustrates the weekly breakdown of PROTEAS's true positives by source of ground truth. Indeed, we observe that in $CP_1$'s case, over 50% of viewership alerts were not caught by QoE tracking tools, whereas for $CP_2$ and $CP_3$ this percentage was approximately equal to 35%. This shows that PROTEAS substantially complements QoE-based alerting workflows.

### 4.6.3 Anatomy of Anomalies

Table 4.5 analyzes PROTEAS's True Positives. We look at viewership surges and drops and examine the mean duration of each type of alert. Overall, we do not observe strong

patterns in favor of surges or drops across the three content providers. In the case of $CP_3$, however, we note that the number of drops is generally comparable to the number of surges. We attribute that to the fact that $CP_3$ is the largest content provider of the three with 164 different channels.

| | Dimension | W1 | W2 | W3 | W4 |
|---|---|---|---|---|---|
| $CP_1$ | Surges | 72% | 10% | 94% | 60% |
| | Drops | 28% | 90% | 6% | 40% |
| | Drop Duration (min) | 58 | 20 | 23 | 38 |
| | Surge Duration (min) | 36 | 10 | 50 | 88 |
| $CP_2$ | Surges | 81% | 11% | 51% | 71% |
| | Drops | 19% | 89% | 49% | 29% |
| | Drop Duration (min) | 38 | 54 | 40 | 55 |
| | Surge Duration (min) | 21 | 24 | 26 | 30 |
| $CP_3$ | Surges | 53% | 52% | 65% | 71% |
| | Drops | 47% | 48% | 35% | 29% |
| | Drop Duration (min) | 38 | 54 | 40 | 55 |
| | Surge Duration (min) | 21 | 24 | 26 | 30 |

Table 4.5: Characteristics of true positives

Figure 4.9 shows a breakdown by group dimensions of PROTEAS's issued alerts and Figure 4.10 depicts a breakdown by anomaly signature. For $CP_1$, $W_1$ was marked by a number of surge alerts during the airing of a popular show's season finale. These alerts were raised as geographically-localized flash crowds. Also, during $W_1$, $CP_1$ experienced a major device failure which resulted in device-related drop alerts. In $W_2$, $CP_1$'s viewership patterns changed due to changes in offered content. This manifested as drops that were correctly classified as model drifts. During $W_3$ and $W_4$, the newly-imposed, geographically-diverse shelter-in-place orders in the US triggered large surges and pervasive changes in viewership patterns (*e.g.,* viewership surges during business hours and in devices generally used in the

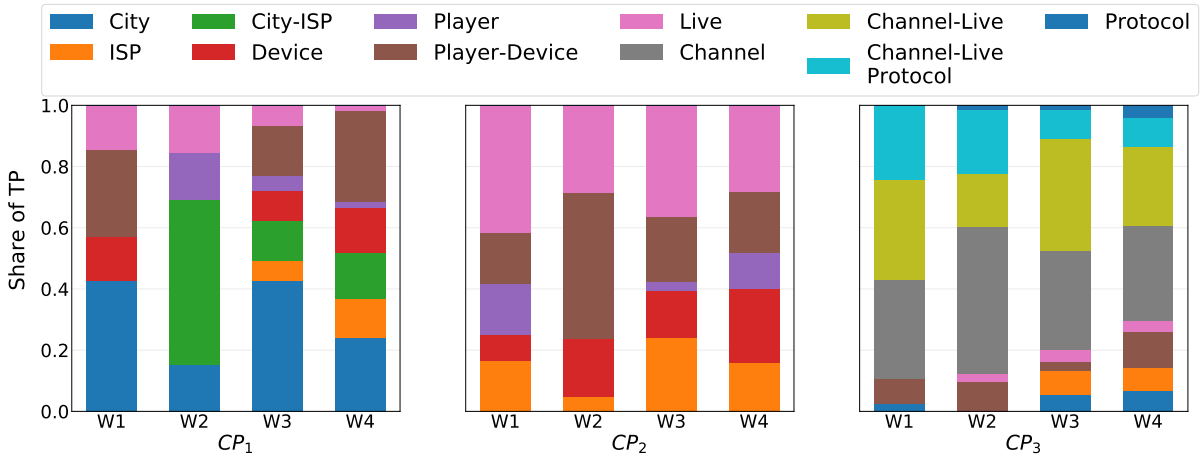



Figure 4.9: Breakdown of true positives by session attributes.

home). PROTEAS, categorized these surges both as model drift and flash crowds.

The alerts of $CP_2$ and $CP_3$ tell us a different story given their frequent live content transmissions. $CP_2$ airs live shows at fixed time-slots during the day (*e.g.,* daily news, sports). During $W_1$, PROTEAS raised multiple flash crowd alerts during the transmission of Champion's League soccer games. On the other hand, the high numbers of drop alerts during $W_2$, were attributed to the cancellations of multiple live transmissions due to COVID-19. Note thet $CP_2$ is based in Europe and measures were taken earlier than in USA. Similar to $CP_1$, we also observe shifts in the use of devices, generally found in the home (*e.g.,* AppleTV, Chromecast *etc.*). $W_3$ and $W_4$ are marked by sudden flash crowds when $CP_2$ aired its evening news. Similarly, the key observation about $CP_3$ is the rise in surges during $W_3$ and $W_4$ due to flash crowds in specific channels dedicated to daily news, stock market *etc.*

**Analysis of anomaly signatures** In the case of $CP_1$, PROTEAS correctly classified surges during $W_1$ as flash crowds. In addition, PROTEAS correctly diagnosed all technical failures $CP_1$ experienced during the pilot (one device failure and one high-impact platform failure). During the pandemic, PROTEAS classified smooth, persistent viewership surges as cases of model drift whereas sharper surges (especially during the prime time zone) were classified as flash crowds. In either case, analysts agreed that either diagnosis would be sufficient for

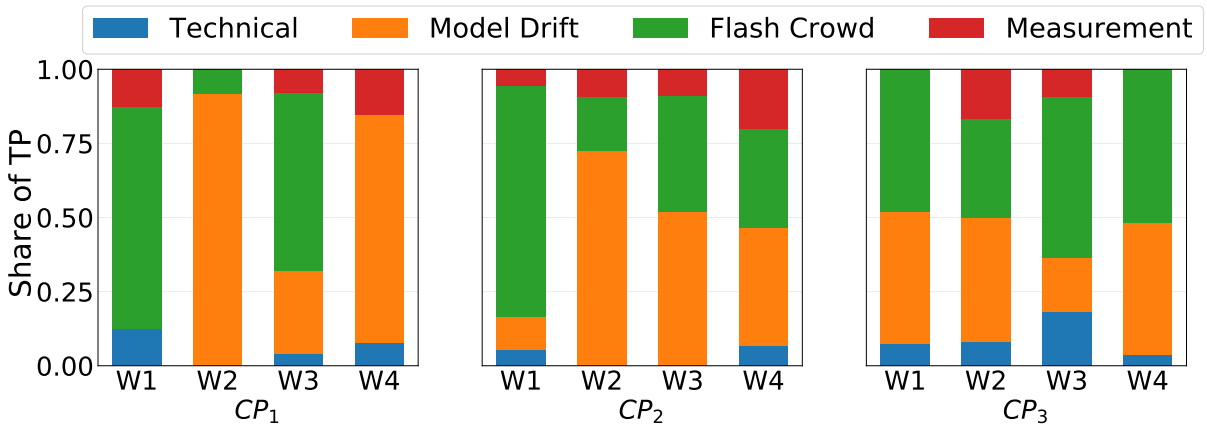



Figure 4.10: Breakdown of true positives by anomaly signature.

content providers to re-provision resources as and when needed. Regarding $CP_2$, >90% of the surges of $W_1$ were correctly associated with flash crowds during sports events whereas in $W_2$, drops were almost exclusively marked as model drift due to the cancellation of live transmissions. Overall, our manual verification of PROTEAS's root-cause attribution showed that PROTEAS gave an acceptable diagnosis for ≈90% of all true positive alerts and mis-classified the remaining 10%.

**Analysis of false positives** Perhaps unsurprisingly, PROTEAS's false positives rose significantly during $W_3$ and $W_4$ across all three content providers. Manual investigation of these alerts showed that >85% of false positives could be traced back to the same cause. Recall that during online detection (§4.3.4), we estimate the shape of viewership at a new timestamp t′ using summary statistics of past weeks. In the common operating scenario, this provides sufficiently accurate estimates but that hypothesis did not always hold during $W_3$ and $W_4$. Due to the shelter-in-place order, total viewership showed a sharp increase by 40% which degraded the quality of online estimates for the shape of viewership.

**Impact of COVID-19 on viewership** The COVID-19 pandemic caused massive changes in viewership [55, 80, 120]. PROTEAS detected large viewership surges during business hours. These surges rolled out gradually across the US, first in Seattle, California, NY

and finally elsewhere, thus tracking not only the progression of the pandemic but also the political response. Second, we observed swift changes in content preferences, with multiple flash crowd and model-drift alerts being traced back to news channels. Unsurprisingly, we also saw a shift to home-centric players and devices (*i.e.,* AppleTV, FireTV *etc.*), unlike other types of players (mobile devices or custom-players for hotel entertainment systems [11]).

### 4.6.4 Component-wise Sensitivity Analysis

To complement the pilot, we manually injected anomalies onto viewership. Specifically, we generate a synthetic dataset by means of a custom anomaly generator that reproduces anomaly signatures observed in the wild and superimposes them to healthy segments of the original trace as follows:

**Synthetic anomaly generation** The synthetic anomaly generator that we use for our evaluation configures both the temporal and the spatial dimension of each generated anomaly.

**Temporal dimension:** With respect to the temporal dimension our generator has the following configurable parameters:

1. *Start time:* The start time of the anomaly.
2. *Duration:* To configure the duration of the anomaly, we use a Markovian model where we define the probability of remaining in an anomalous state for the next timestamp.
3. *Directionality:* This parameter determines whether the anomaly is going to be a surge, drop or oscillate over time.
4. *Change in magnitude:* This parameter determines how far the viewership deviates from the expected normalized w.r.t. the standard deviation.
5. *Gradient:* to simulate how sharp or smooth is the onset/recovery of the anomaly.

**Spatial dimension:** Once we generate the temporal signature, we leverage the propagation mechanism discussed in §4.4.1 to spatially propagate the change to other viewership groups

in the hierarchy.

**Detection accuracy** Figure 4.11 compares PROTEAS with several additional baselines: Holt-Winters [64], ARIMA [167], regression using Fourier, Wavelet basis functions as well as combinations thereof [37]. For Holt-Winters and ARIMA we used Python's StatsModels package [24]. Regarding Fourier and Wavelet (Debauchies and Haar) transforms we chose the smallest set of coefficients that collectively accounted for 98% of the original signal's energy [39, 49]. We set the confidence bounds to 3 standard deviations from the mean value. Figure 4.11 shows the resulting F-score (combining precision/recall [12]) and also the precision and recall of the simulation.

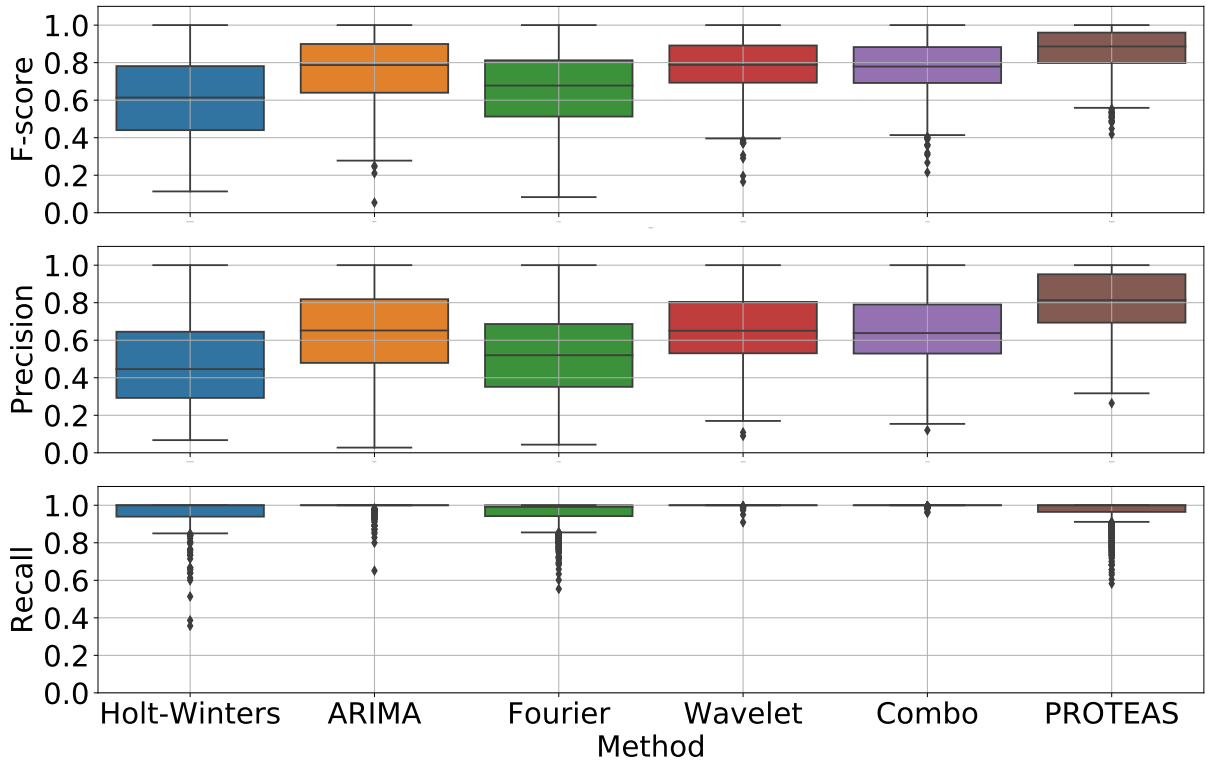



Figure 4.11: Detection accuracy using synthetic traces

**Model accuracy** *vs.* **compactness** As discussed in §4.3 there is a trade-off between the size of the detection model and accuracy. Indeed, our experiments (shown in Table 4.6)

indicate that accuracy drops sharply when the distance between two observations exceeds 5 minutes, making that the optimal sampling candidate for timestamps in $X$ in terms of accuracy and compactness.

| **Sampling period (min)** | **1** | **5** | **10** | **30** | **60** |
|---|---|---|---|---|---|
| F-score | 0.88 | 0.86 | 0.67 | 0.39 | 0.38 |

Table 4.6: Accuracy vs. compactness tradeoff

**Summarization accuracy** Table 4.7 shows accuracy of the summarization heuristics as a function of $k$, the number of critical groups PROTEAS issues alerts for. These results represent 500 randomized tests across providers where the critical group was chosen at random and an anomaly was injected and propagated as discussed in §4.4.1. In Table 4.7, we compare our results with two competing approaches: (1) output the group that exhibits the highest nominal change in viewership value (Naive Summarization) and (2) Adtributor [53].

| **Number of Top-$k$ Groups** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| PROTEAS | 0.5 | 0.92 | 0.98 | 1.0 | 1.0 |
| Adtributor [53] | 0.23 | 0.67 | 0.8 | 0.91 | 0.93 |
| Naive Summarization | 0.22 | 0.29 | 0.41 | 0.66 | 0.71 |

Table 4.7: Comparison of summarization accuracy results

Indeed, when $k = 3$, PROTEAS is almost guaranteed to identify the critical group. We attribute the jump from $k = 1$ to $k = 2$ to the fact that the generator often picks a group with only one descendant. That descendant (whose viewership is equal to that of its critical parent) was the top choice as our heuristics would favor it over the parent. In these cases, the critical parent node had the second highest $EP$ score.

## 4.7 Related Work

To the best of our knowledge, there is no prior work on real-time alerting workflows specifically for anomalies on the time series of viewership. Nevertheless, below, we provide a high-level overview of mechanisms for time-series anomaly detection and diagnosis.

**Time-Series anomaly detection** Anomaly detection, especially in timeseries is a classical problem that has been extensively studied in many diverse domains and there are several detailed surveys on this topic (e.g., [36, 62, 92, 96, 130]). Techniques like spectral processing or wavelets, have been used in the field of signal processing and for internet traffic patterns [39, 49, 97]. Other techniques *e.g.,* Kalman filtering and PCA have also been proposed as candidates for anomaly detection [25, 127]. Twitter, Facebook, Snapchat recently released their own statistical technique for timeseries anomaly detection [21, 95, 153].

**Anomaly summarization and root-cause analysis** Previous research has also focused on extensively analyzing root-causing in systems and networks [33, 45, 50, 106, 110, 131]. The types of data and problems these focus on is orthogonal to our focus. More closely related to our work is prior work on root-cause attribution and anomalies summarization in the context of ad platforms [53]. Our notion of anomaly signatures and generating compact summaries builds from this literature but customizes it for video viewership.

**Internet video alerting workflows** In the context of video streaming, other efforts have focused on QoE. CFA leverages domain specific insights that enable accurate prediction of video quality [104]. Other prior work has shown correlations between video quality metrics and user engagement [47, 144]. There is also large literature in measuring video quality in the wild [136, 163], quality issues [102], server selection [149] as well as techniques to improve user experience [103, 164].

## 4.8 Conclusions

Accurate detection and diagnosis of viewership anomalies can complement existing management tasks for Internet video providers, as it can provide unique insights that QoE measurements cannot capture. However, the complex spatiotemporal relationships in video viewership make this problem challenging, preventing the use of many conventional techniques. PROTEAS identifies key structural insights about viewership to enable accurate and real-time detection and diagnosis. Our evaluation and pilot studies with operators suggest that PROTEAS outperforms many existing solutions.

# Chapter 5

# End-to-End System Integration

In the previous chapters, we looked at each component of a multidimensional analytics framework individually: In Chapter 3, we introduced Hydra, a sketch-based design for provably accurate, multidimensional analytics. However, in designing Hydra, we did not explore how well its approximate analytics combine with downstream tasks, such as anomaly detection. In Chapter 4, we presented Proteas, an alerting workflow for video viewership anomalies. Yet, since Proteas's inputs were already pre-aggregated, precisely estimated viewership counts, we did not investigate the impact of multidimensionality in estimating viewership counts.

In this chapter, we are interested in determining to what extent the use of approximate, albeit provably accurate, analytics can negatively impact the accuracy of downstream anomaly detection. To that end, we prototype Integr, an end-to-end multidimensional telemetry framework for timeseries anomaly detection that combines key design insights of Hydra and Proteas. We demonstrate the feasibility of implementing accurate and efficient end-to-end anomaly detection workflows using approximate analytics estimations.

## 5.1 System Overview

Integr implements approximate analytics estimation and anomaly detection, as shown in

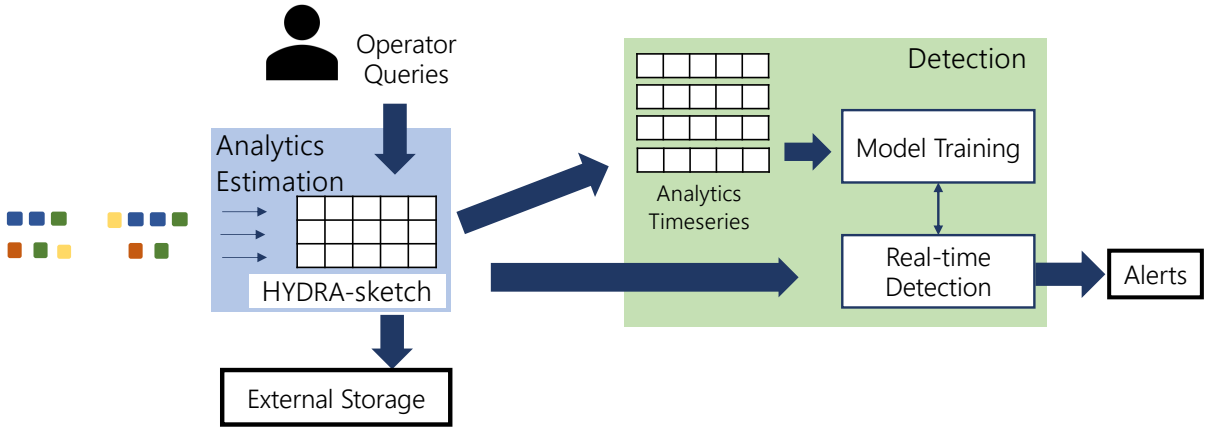



Figure 5.1: INTEGR design overview

Figure 5.1.

**Inputs and outputs** INTEGR ingests batches of multidimensional data streams on a per-epoch basis. Every epoch, INTEGR implements analytics estimation as well as timeseries anomaly detection workflows and outputs any data subpopulations that might be exhibiting anomalous behavior.

**Analytics estimation** We implement INTEGR's analytics component using HYDRA. For every data epoch, the component ingests a batch of multidimensional incoming data which are then summarized across instances of HYDRA-sketch. Then, given a set of analytical queries (that it also receives as input from the operator), it estimates the corresponding summary statistics of interest per subpopulation. An example of such a statistic can be viewership counts. In addition to HYDRA's workflow, the estimation component implements two additional functions at the end of every epoch. First, it stores the merged HYDRA-sketch instance in case the user needs to retrieve data from that epoch in the future. Second, it filters subpopulations and outputs estimates only for the ones it can offer robust accuracy guarantees (*i.e.,* the ones whose $G-$sum is greater than a user-defined threshold).

**Anomaly detection** This components implements various timeseries anomaly detection algorithms using the system design principles as discussed in PROTEAS. In particular, it processes the approximate estimations of the various per-subpopulation statistics from the estimation component and uses model-based anomaly detection to detect the presence of anomalies in each data subpopulation.

## 5.2 Implementation

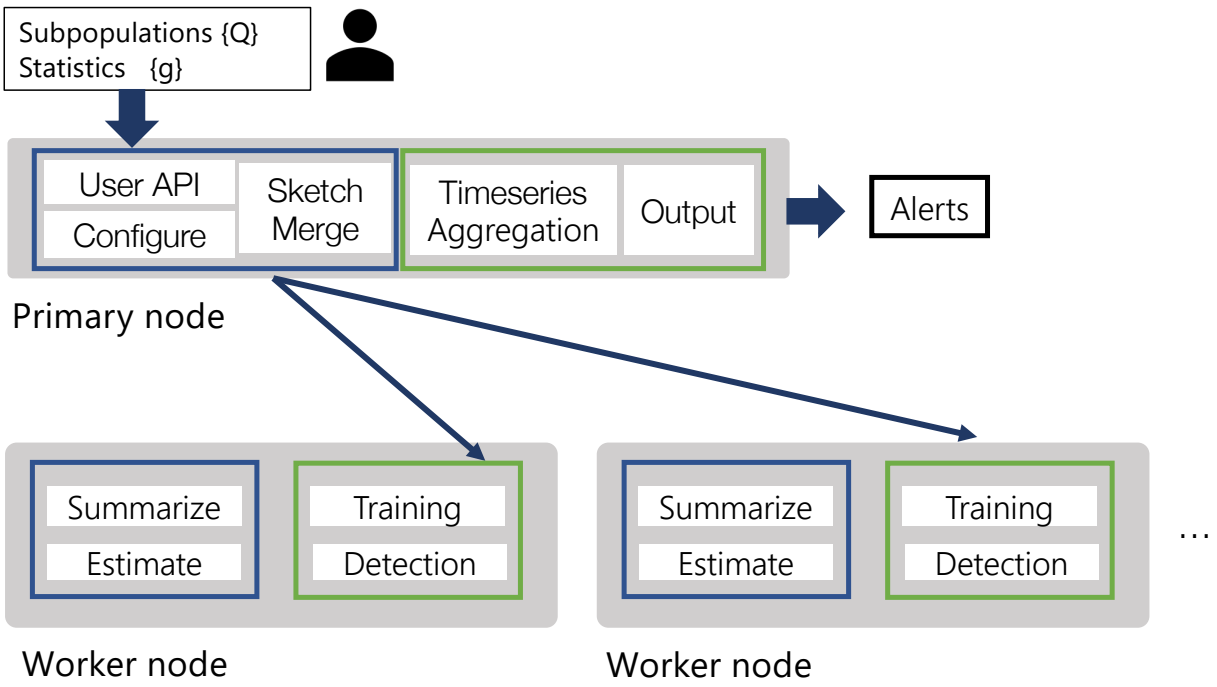



Figure 5.2: Implementation of INTEGR

**Analytics estimation** In our implementation of the analytics estimation component, similar to HYDRA's implementation, worker nodes implement data ingestion and query estimation, whereas the frontend node is responsible for sketch merging as well as for estimating per-subpopulation statistics. At the end of each epoch, after estimation results are returned to the frontend node, INTEGR persists the corresponding merged HYDRA-sketch instance to storage in Amazon S3. This is to facilitate cases where the framework might

need to re-create timeseries of the various per-subpopulation estimations, run historical data analysis *etc.* Second, the frontend node filters the subpopulations for which it can offer robust accuracy guarantees. To achieve that, the analytics component maintains one additional instance of universal sketch that estimates the $G$-sum of the entire datastream. Then, the frontend node compares each subpopulation's $G$-sum against the datastream's in order to filter out subpopulations for which the system cannot offer robust accuracy guarantees. Last, the frontend node maintains a dataframe that contains a window of recent estimations per data subpopulation of interest. This dataframe is used by anomaly detection module for model training.

**Anomaly detection** Given anomaly detection is a highly parallelizable task, we implement it at the worker nodes. Building on PROTEAS' implementation, we focus on building compact, per-subpopulation models to ensure minimal storing and fetching cost. To train the anomaly detection models, our prototype uses the dataframe of per-subpopulation timeseries data, that is populated by the estimation component. Then for each epoch, the estimated values are checked against the anomaly detection model to detect the presence of an anomaly. In our prototype, unlike PROTEAS, we focused on the generic implementation of multiple different state-of-the-art timeseries anomaly detection workflows, instead of the shape-based alerting workflow for video viewership.

## 5.3 Evaluation

We evaluate INTEGR's end-to-end performance using two real-world datasets. Our key findings are:

1. INTEGR offers similar resource and cost benefits as HYDRA. In particular, INTEGR's memory usage per epoch is approximately an order of magnitude smaller than traditional Spark-based approaches and query latency is 8× smaller.
2. The use of provably accurate, approximate analytics does not result in noticeable de-

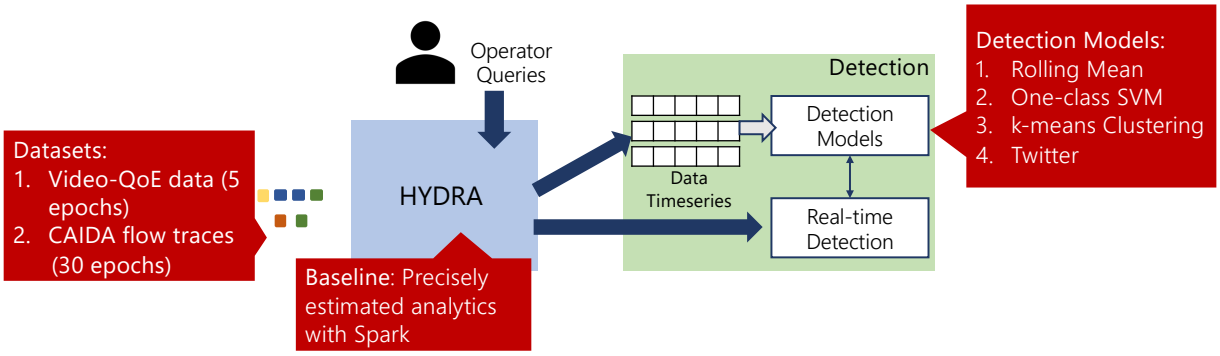



Figure 5.3: INTEGR evaluation plan

terioration in INTEGR's anomaly detection accuracy. Indeed, compared to a baseline telemetry system that runs anomaly detection on precise analytics, an instance of INTEGR that is configured to ensure 95% accuracy offers negligible deterioration in anomaly detection accuracy (both precision and recall >0.97).

### 5.3.1 Experimental Methodology

**INTEGR testbed** Similar to HYDRA, we run INTEGR on a 20-node cluster on m5.xlarge AWS servers [20]. For the end-to-end performance evaluation, we deploy optimally configured HYDRA-sketch instances to ensure 95% estimation accuracy with at least 90% probability for data subpopulations for which $G(Q_j)/G(\mathcal{S}) >= 0.001$. The input data consists of CSV files that are ingested from AWS S3, once per data epoch.

**Datasets** We evaluate INTEGR using two large real-world datasets. First, we use CAIDA flow traces [3] collected at a backbone link of a Tier1 US-based ISP. The total trace is 130GB in size and flow data can be clustered in up to approximately 5.6M subpopulations. We break down this dataset into 30 epochs of ∼4.5GB per epoch. Second, we use a real-world trace of video session summaries corresponding to one major US-based streaming-video provider. The size of the video-QoE trace is approximate 5GB, with data that we cluster in up to 700k subpopulations and break down into 5 epochs of 1GB each.

**Experimental setup** For our evaluation, we use as baseline a two-step telemetry workflow that uses a traditional Spark-based workflow for analytics estimation. This component ingests the same multidimensional stream, pre-aggregates them across subpopulations and estimates the same per-subpopulation summary statistics. For the anomaly detection component, we implement the following anomaly detection workflows:[1]

1. *Rolling window anomaly detection.* For this method, we are using a rolling window of length 5 epochs to estimate the rolling mean and standard deviation of each subpopulation's timeseries. We set an anomaly threshold at 3 standard deviations from the rolling mean.
2. *One-Class SVM.* One-class SVM is a variation of the regular SVM that can be used in an unsupervised setting for anomaly detection. This technique first maps input data into a higher-dimensional feature space, then obtains the optimal separating hyperplane in the feature space. The decision boundary is determined by support vectors rather than the whole training sample, and thus is extremely robust to outliers.
3. *K-means clustering.* This method looks at the data points of the timeseries and groups them into a predefined number of clusters $K$. Using Euclidean distances, we add a threshold value to detect anomalies: if the distance between a data point and its nearest centroid is greater than the threshold value, then it is an anomaly.
4. *Twitter anomaly detection.* This is an automatic timeseries anomaly detection algorithm developed by Twitter that employs statistical learning to detect anomalies in timeseries that contain inherent seasonal and trend components [95].

**Evaluation metrics** For each of the above techniques, we measure the precision (False positives) and recall (False Negatives) of INTEGR when compared to the Spark-based baseline. More specifically, we classify an anomaly as a false positive when it was marked as anomalous by INTEGR but not by the baseline. Similarly, we classify an anomaly of the baseline that was not captured by INTEGR is a false negative.

[1]Due to limitations with respect to reusing PROTEAS's source code and data, we do not re-use PROTEAS' workflow and scope our exploration to more generic timeseries anomaly detection.

### 5.3.2 End-to-End Evaluation

**Accuracy** Figure 5.4 shows how the two frameworks compare across precision and recall when running anomaly detection on the timeseries of L1-norm and entropy respectively. Note that INTEGR's analytics estimation has been configured to ensure at least 95% accuracy. Our key observation is that for both datasets, accuracy values for both precision and recall are on average >90%. This is a strong indicator that approximate estimations of the estimated statistics have a negligible impact on the accuracy of anomaly detection, despite their (bounded) estimation error.

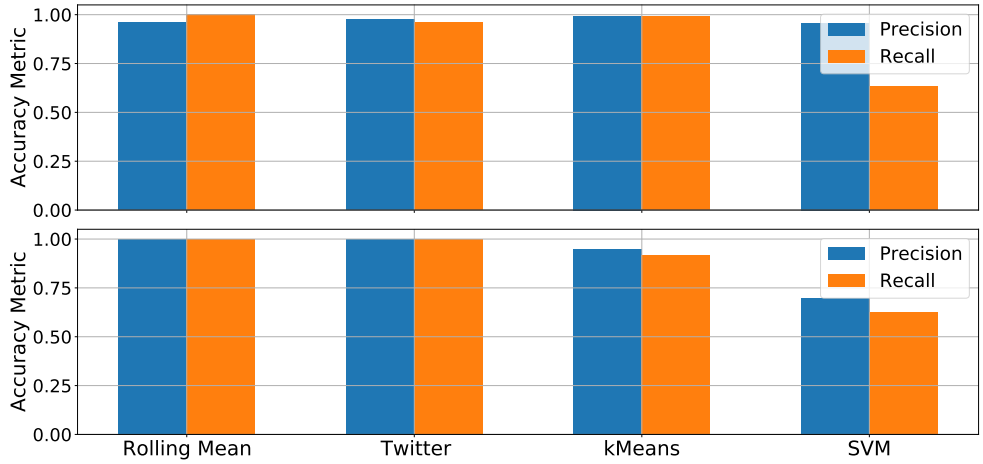
**Resource utilization** Table 5.1 illustrates key performance and resource utilization numbers for INTEGR's run with the CAIDA dataset. We observe that INTEGR's memory utilization is approximately 14% of the baseline whereas both the ingestion and the query time are approximately 1/3 of the baseline. Note that given the relatively small size of each data batch (5GB), HYDRA's benefits are not fully illustrated. Nevertheless, the observed results are fully consistent with the resource utilization results of Chapter 3.

| | **Baseline** | INTEGR |
|---|---|---|
| Memory Utilization | ~ 1.5GB | 200MB |
| Ingestion Time | ~30 sec | ~11 sec |
| Query Time | ~12 sec | ~3 sec |

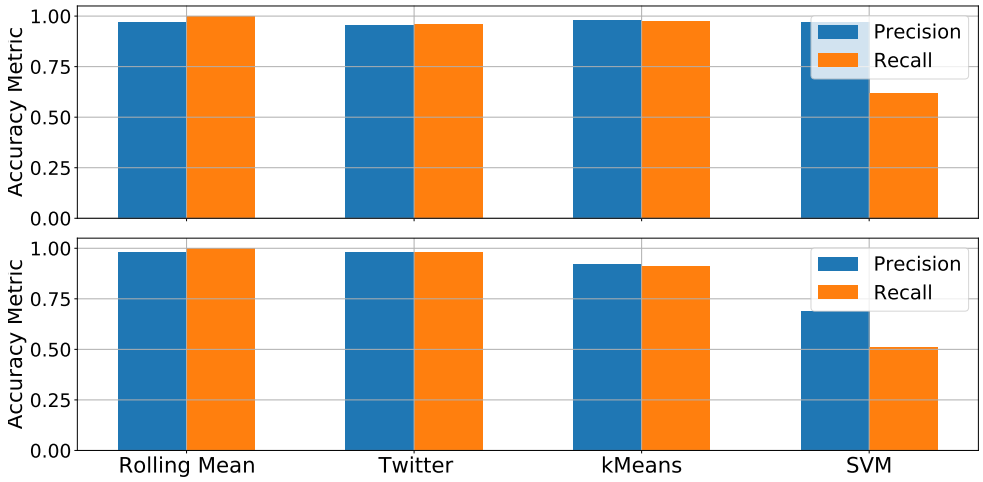
Table 5.1: Comparison of resource utilization estimates (per data epoch).

## 5.4 Summary

In prior chapters, we focused on the design and development of each component of an analytics framework separately. INTEGR showcases the benefits and shortcomings of an end-to-end telemetry framework that builds anomaly detection workflows using approximate estimations. There are two key takeaways from this prototype. The first, is that




(a) L1-Norm.




(b) Entropy.

Figure 5.4: Precision and recall compared to anomaly detection with precise estimations

the use of a HYDRA-inspired analytics estimation component ensure clear resource, performance, and cost benefits to the end-to-end system. Second, the use of approximate analytics does not translate to reduced anomaly detection accuracy. Indeed, the error of HYDRA's estimations has a marginal impact on the accuracy of the downstream anomaly detection task.

# Chapter 6

# Reflections, Limitations and Future Work

In this chapter, we first summarize the lessons learned from the design and implementation of the thesis contributions (Section 6.1) and then discuss limitations of our solutions (Section 6.2). Last, we conclude this dissertation by identifying future research directions (Section 6.3).

## 6.1 Lessons Learnt

**Reflection #1: The increasing dimensionality of data streams dictates the need for new telemetry paradigms.**

The key motivation of this dissertation was the observation that existing telemetry frameworks cannot cope with the combinatorial explosion of data subpopulations that result from increases in data dimensionality. Existing telemetry either offers linear scaling capabilities to data subpopulations, which is unsustainable from a cost perspective or, when it resorts to approximations, cannot offer robust accuracy guarantees and/or generality across statistics. This raises the need for newer designs for telemetry systems.

**Reflection #2: Sketch-based telemetry enables scalable designs with a good tradeoff between accuracy, generality and efficiency**

While sketching has been a known approach for approximate telemetry, it was not clear how a sketch-based design could enable scalability in a multidimensional context. Our sketch-of-sketches design that shows that we can achieve provable accuracy with sub-linear (to the number of subpopulations) sketching primitives can be a starting point for further improvements in sketch-based analytics.

**Reflection #3: Full coverage across data subpopulations may not always be necessary**

Operators see multidimensional analytics as a golden opportunity to ensure fine-grained visibility in their data, ideally across all subpopulations thereof. In hindsight, we believe it is worth investigating further whether full coverage is a strong requirement of telemetry. In this thesis (and through the design of Hydra), we paraphrase a common systems design principle "Optimize for the common case" and design a system that focuses on the most important data subpopulations. In other words, we argue in favor of sacrificing accuracy guarantees for a (non-deterministic) subset of subpopulations to maximize scalability and efficiency benefits. In Hydra, these subpopulations are those exhibiting small $G$-sum values.

**Reflection #4: Effective alerting workflows require multiple complementary views of the data**

The motivation behind designing Proteas was the need to address the blind spots of existing video-QoE workflows. Our analysis showed that viewership is indeed a valuable indicator of anomalies. That is because it captures the reaction of viewers to an anomalous incident and also because it quantifies the change in viewer engagement—one of the most important performance indicators for content providers.

**Reflection #5: Shape-based detection workflows may be transferable to other domains**

A key challenge in detecting viewership anomalies is their contextual nature and the signal's lack of stationarity. Ultimately, the shape persistence of the timeseries of viewership enabled us to extract "blueprints" of expected viewership behavior and compare them with new observations in order to detect anomalies. We speculate that such a property is not limited to viewership but can be found in other domains as well *e.g.,* anomaly detection in electric power systems.

## 6.2 Limitations of this thesis

**Limitation #1: The resource benefits of sketch-based analytics can depend on data properties**

While HYDRA-sketch's accuracy guarantees are data-agnostic, in practice we observe that properties of the incoming data stream can have a direct impact on HYDRA-sketch's resource footprint. The best-case scenario for HYDRA is when the distribution of subpopulation sizes is skewed. In the case of a dataset with more uniformly distributed subpopulation sizes, HYDRA would deliver modest gains because it would require a substantially larger amount of resources. Fortunately, we believe that most real-world datasets exhibit this desirable skewness.

**Limitation #2: The complex configuration space of HYDRA-sketch.**

HYDRA-sketch is a data structure whose empirical performance depends critically on 6 configuration parameters. As a result, despite the configuration strategies that we provide and evaluate, we acknowledge that configuring HYDRA-sketch can potentially be a complicated and error-prone process, especially for cases where the framework might need periodic re-calibration.

### Limitation #3: Operators show limited trust in approximate telemetry

While approximate analytics engines are not a newly introduced concept, research has shown that user trust issues have stymied broader adoption [107]. While HYDRA offers robust accuracy guarantees in terms of analytics estimation, our end-to-end prototype highlighted that sacrifices in accuracy cascade to downstream tasks and have the potential to result in false positives or negatives in terms of service alerts. Ultimately, a practical question to be answered is whether the cost of a false positive to the operator can erase the benefits in terms of operating cost of an approximate analytics framework.

### Limitation #4: The need to maintain one anomaly detection model per viewership group

A key observation of PROTEAS was that the shape of the viewership was an intrinsic property of each viewership group. Having established our GP-based anomaly detection technique [139], a design decision behind the system was to train one model per subpopulation. While PROTEAS' detection models are compact, the number of models the system needs to maintain scales linearly to the number of viewership groups. This can be problematic given the scalability challenge induced by multidimensionality. We hypothesize that there is room to reduce the number of models needed, since viewership groups that are close in the group hierarchy might exhibit similar viewership patterns.

### Limitation #5: Root-Cause diagnosis

In PROTEAS, we observe that it is unlikely for more than one logical event to occur simultaneously. Based on this observation, we developed a classification module that associates an anomaly signature with potential root causes. While we believe that such a process can speed up and the diagnosis process and hint at possible root causes, root-cause attribution is still an elusive and complicated task.

**Limitation #6: Cascading errors**

In the end-to-end implementation of the telemetry framework, we observe that the approximate analytics estimations, while bounded in error, might cascade in the downstream detection workflows. As discussed below, a possible direction for future work would involve designing detection techniques that take the possibility of estimation error into account in their workflow.

## 6.3 Suggestions for Future Work

Our overarching goal for multidimensional telemetry is to provide operators with accurate, actionable, and affordable insights about their infrastructures. Our work in this dissertation has laid some steps towards achieving that goal. To this end, we now identify the following broad research directions:

1. Given the importance of efficient and provably accurate telemetry workflows, we need to *enhance analytics estimation techniques* to further enable these capabilities (Section 6.3.1).
2. The video streaming ecosystem is a complex one. While this dissertations introduces a suite of tools and design proposals for video viewership analytics, we argue that we still need to *enhance our understanding of anomalies and information sources* in order to unlock more monitoring capabilities and facilitate diagnosis (Section 6.3.2).

### 6.3.1 Enhancing Analytics Estimation Techniques

**Online adaptation of sketching primitives** As we saw in Chapter 3, the empirical accuracy of HYDRA-sketch, depends on its configuration which, in turn, depends on properties of the data. As properties of the incoming data streams change over time, the configuration of the respective sketching primitives needs to change accordingly. As future work, we envision an adaptive sketch-based telemetry framework that tracks changes in properties

of the input data stream and reconfigures sketching primitives on the fly.

Recent prior work by Aamand *et al.* [27] and Eden *et al.* [78] explore, at a theoretical level, learning-based approaches and algorithms for automatic sketch configuration and adaptation. These works show that it is possible to enable online adaptation of a sketching primitive while maintaining formal estimation bounds. We believe that these theoretical findings have the potential of being leveraged in a HYDRA-like systems context in order to enable the framework's ability to adapt to stream changes.

**Measure the impact of approximate analytics on downstream tasks** While sketch-based primitives offer robust accuracy guarantees on their estimations, the impact of the approximation on downstream tasks (*e.g.,* anomaly detection) remains an open question. As future work, we believe it would be interesting to quantify and/or bound more formally the resulting end-to-end accuracy of a telemetry framework that consists of different modules when one or more of these modules leverage approximate analytics. In other words, in the presence of analytics modules that can introduce cascading errors, can we formally reason about the end-to-end accuracy of the system?

**Applicability of sketch-based telemetry in different domains** In our work, we touted that HYDRA is a general-purpose framework for multidimensional telemetry. Given the abundance of domains that can benefit of multidimensional telemetry (*e.g.,* in-band network monitoring [118, 119], low-power sensor deployments [162], edge computing *etc.*), we believe it would be an exciting future direction to explore the potential benefits and challenges of deploying HYDRA in these domains. We provide some detailed examples below:

1. **In-band network telemetry** With network elements becoming increasingly programmable and efficient in-network computation becoming a pragmatic and feasible target, we see an important opportunity in exploring the possibility of deploying distributed HYDRA-like deployments for in-band, in-network telemetry. The goal of such deployments would be to provide the necessary data to enable smarter routing, traffic

engineering, network security or forensics. Yet, as prior work has shown [32, 125], this can be a challenging vision given the heterogeneity and the different capabilities in terms of programmability of network elements.

2. **Low-power sensor deployments** Wireless sensors have enabled several key applications. Due to their energy constraints, today's sensor deployments communicate occasional short samples or pre-determined summary statistics of the data they collect. This means that computing every additional statistic at high fidelity incurs additional communication, energy, and resource overhead. Given that sensor deployments could benefit from multidimensional telemetry, we believe it would be exciting to explore how HYDRA-sketch or similar primitives would behave in an environment that is constrained in terms of available resources and energy but, still, needs to produce efficient and general multidimensional analytics.

### 6.3.2 Enhancing Our Understanding of the Video Streaming Ecosystem

**In-depth classification of session attributes that cause anomalies** The video streaming ecosystem is a large and complex one [102, 104]. We argue that to improve our ability to diagnose viewership anomalies (and more), it is worth further investigating anomaly signatures to identify most common and/or important anomalous events and how they manifest in terms of session attributes.

**Automate the process of labeling anomalies and extracting ground-truth extraction** One of the most time consuming and arduous tasks in the development of PROTEAS was identifying the ground truth to evaluate anomaly detection. Indeed, our process for identifying ground truth involved cross-checking many different metadata sources for every detected anomaly. We believe that a tool that automates the labelling process would be a practical and useful addition in an video operator's toolbox.

**Extend viewership analysis to different types of video streams** In PROTEAS, we investigated viewership anomalies from the perspective of content providers for TV. In particular, we focused on providers of Video-on-Demand content, as well as linear TV channels (where content is transmitted according to relatively fixed schedules). Nevertheless, streaming video is not only limited to streaming TV; instead, some of the larger streaming video providers are social media platforms as well as platforms like Twitch, YouTube *etc.* To that end, it would be interesting to explore viewership patterns for such platforms and whether our TV-related insights extend to these content outlets.

**Faster adaptation to model drift** While viewership patterns change slowly over time, PROTEAS needed to periodically retrain its detection models in order to account for model drift. Nevertheless, given in our implementation, detection models are trained with one sample per week, adapting to model drift can be a long process. A possible avenue for future work could involve exploring avenues for faster adaptation to model drift.

**Collaborative workflows to consolidate information from different metrics** Providing operators with actionable insights alerts requires telemetry infrastructures that offer visibility into the data from different angles. In video streaming, as we saw, this translates to estimating and looking for anomalies in parallel across many different metrics and types of data. Nevertheless, we observe that often these workflows operate independently of each other. This is not surprising; telemetry infrastructures are often built incrementally, and capabilities are being added based on what types of events or information are of interest to operators. We believe that a possible avenue for future work would be to develop collaborative workflows that combine information from existing, currently, independent, workflows and, thus, simplify the information that arrives to the operators [74, 112, 157, 158].

# Bibliography


[1] EBS Service Event in the Tokyo Region. https://aws.amazon.com/message/56489/, .

[2] EC2 DNS Resolution Issues in the Asia Pacific Region. https://aws.amazon.com/message/74876/, .

[3] CAIDA Network Flow Traces. https://www.caida.org/catalog/datasets/overview/, .

[4] CAIDA Trace. https://www.caida.org/catalog/datasets/monitors/passive-equinix-nyc/, .

[5] Confidence Interval Critical Values. https://bit.ly/3bRM9aK.

[6] COmcast outagel. https://www.miamiherald.com/news/nation-world/national/article214067869.html.

[7] Conviva - Real-time Streaming Video Intelligence. https://www.conviva.com/, .

[8] Conviva State of Streaming Report. https://pages.conviva.com/rs/138-XJA-134/images/RPT_Conviva_State_of_Streaming_Q2_2021.pdf, .

[9] Cosine Distance. https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.distance.cosine.html.

[10] Down Detector. https://www.downdetector.com/.

[11] ENSEO systems. https://www.enseo.com/.

[12] F-score. hhttps://deepai.org/machine-learning-glossary-and-terms/

f-score.

[13] IBM Streams. https://www.ibm.com/cloud/streaming-analytics.

[14] Kafka Streams. https://kafka.apache.org/documentation/streams/.

[15] Kafka tops 1 trillion messages per day at linkedin. https://www.datanami.com/2015/09/02/kafka-tops-1-trillion-messages-per-day-at-linkedin/.

[16] Matern Kernel. https://en.wikipedia.org/wiki/Mat%C3%A9rn_covariance_function.

[17] White noise Kernel. https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.WhiteKernel.html.

[18] Periodic Kernel. http://jhamrick.github.io/gaussian_processes/gp.kernels.PeriodicKernel.html.

[19] Precision Recall Analysis. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec.

[20] Amazon AWS EC2 pricing . https://aws.amazon.com/ec2/pricing/on-demand/.

[21] Prophet. https://facebook.github.io/prophet/.

[22] Cisco Annual Internet Report. https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/air-highlights.html.

[23] Squared Exponential Kernel. http://evelinag.com/Ariadne/covarianceFunctions.html.

[24] StatsModels. https://www.statsmodels.org/stable/index.html.

[25] SURUS - Anomaly detection at Netflix. https://netflixtechblog.com/rad-outlier-detection-on-big-data-d6b0494371cc.

[26] Verizon Outage. https://bgr.com/2019/06/24/internet-outage-2019-google-amazon-reddit-down/.

[27] Anders Aamand, Piotr Indyk, and Ali Vakilian. (learned) frequency estimation algorithms under zipfian distribution. *arXiv preprint arXiv:1908.05198*, 2019.

[28] Daniel J Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, et al. The design of the borealis stream processing engine. In *Cidr*, volume 5, pages 277–289, 2005.

[29] Lior Abraham, John Allen, Oleksandr Barykin, Vinayak Borkar, Bhuwan Chopra, Ciprian Gerea, Daniel Merl, Josh Metzler, David Reiss, Subbu Subramanian, et al. Scuba: Diving into data at facebook. *Proceedings of the VLDB Endowment*, 6(11): 1057–1067, 2013.

[30] Swarup Acharya, Phillip B Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. The aqua approximate query answering system. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 574–576, 1999.

[31] Swarup Acharya, Phillip B Gibbons, and Viswanath Poosala. Congressional samples for approximate answering of group-by queries. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 487–498, 2000.

[32] Anup Agarwal, Zaoxing Liu, and Srinivasan Seshan. Heterosketch: Coordinating network-wide monitoring in hetero-geneous and dynamic networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, Renton, WA, April 2022. USENIX Association.

[33] Bhavish Agarwal, Ranjita Bhagwan, Tathagata Das, Siddharth Eswaran, Venkata N Padmanabhan, and Geoffrey M Voelker. Netprints: Diagnosing home network misconfigurations using shared knowledge. In *NSDI*, volume 9, pages 349–364, 2009.

[34] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):1–28, 2013.

[35] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42, 2013.

[36] Charu C. Aggarwal. *Outlier Analysis*. Springer Publishing Company, Incorporated, 2013. ISBN 1461463955, 9781461463955.

[37] William Aiello, Anna Gilbert, Brian Rexroad, and Vyas Sekar. Sparse approximations for high fidelity compression of network traffic data. In *Proceedings of the 5th ACM SIGCOMM conference on Internet measurement*, pages 22–22. USENIX Association, 2005.

[38] Tyler Akidau, Alex Balikov, Kaya Bekiroğlu, Slava Chernyak, Josh Haberman, Reuven Lax, Sam McVeety, Daniel Mills, Paul Nordstrom, and Sam Whittle. Millwheel: Fault-tolerant stream processing at internet scale. *Proceedings of the VLDB Endowment*, 6(11):1033–1044, 2013.

[39] Vicente Alarcon-Aquino and Javier A. Barria. Anomaly detection in communication networks using wavelets. 2001.

[40] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proc. of ACM STOC*, 1996.

[41] Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowl. Inf. Syst.*, 51(2):339–367, May 2017. ISSN 0219-1377. doi: 10.1007/s10115-016-0987-z. URL https://doi.org/10.1007/s10115-016-0987-z.

[42] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. Stream: The stanford data stream management system. In *Data Stream Management*, pages 317–336. Springer, 2016.

[43] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 1383–1394, 2015.

[44] A Asta. Observability at twitter: technical overview, part i, 2016, 2016.

[45] Paramvir Bahl, Ranveer Chandra, Albert Greenberg, Srikanth Kandula, David A

Maltz, and Ming Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. *ACM SIGCOMM Computer Communication Review*, 37(4):13–24, 2007.

[46] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. Macrobase: Prioritizing attention in fast data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 541–556, 2017.

[47] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. *SIGCOMM Comput. Commun. Rev.*, 43(4):339–350, August 2013. ISSN 0146-4833. doi: 10.1145/2534169.2486025. URL http://doi.acm.org/10.1145/2534169.2486025.

[48] Magdalena Balazinska, Hari Balakrishnan, Samuel Madden, and Michael Stonebraker. Fault-tolerance in the borealis distributed stream processing system. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 13–24, 2005.

[49] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron. A signal analysis of network traffic anomalies. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 71–82, New York, NY, USA, 2002. ACM. ISBN 1-58113-603-X. doi: 10.1145/637201.637210. URL http://doi.acm.org/10.1145/637201.637210.

[50] Paul Barham, Austin Donnelly, Rebecca Isaacs, and Richard Mortier. Using magpie for request extraction and workload modelling. In *OSDI*, volume 4, pages 18–18, 2004.

[51] Christoph Bergmeir, Rob J Hyndman, and José M Benítez. Bagging exponential smoothing methods using stl decomposition and box–cox transformation. *International journal of forecasting*, 32(2):303–312, 2016.

[52] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[53] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. Adtributor: Revenue debugging in advertising systems. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 43–55, 2014.

[54] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[55] Timm Böttger, Ghida Ibrahim, and Ben Vallis. How the internet reacted to covid-19: A perspective from facebook's edge network. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 34–41, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381383. doi: 10.1145/3419394.3423621. URL https://doi.org/10.1145/3419394.3423621.

[56] Lucas Braun, Thomas Etter, Georgios Gasparis, Martin Kaufmann, Donald Kossmann, Daniel Widmer, Aharon Avitzur, Anthony Iliopoulos, Eliezer Levy, and Ning Liang. Analytics in motion: High performance event-processing and real-time analytics in the same database. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 251–264, 2015.

[57] Vladimir Braverman and Stephen R Chestnut. Universal sketches for the frequency negative moments and other decreasing streaming sums. *arXiv preprint arXiv:1408.5096*, 2014.

[58] Vladimir Braverman and Rafail Ostrovsky. Zero-one frequency laws. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 281–290, 2010.

[59] Chiranjeeb Buragohain and Subhash Suri. Quantiles on streams., 2009.

[60] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[61] Ronnie Chaiken, Bob Jenkins, Per-Åke Larson, Bill Ramsey, Darren Shakib, Simon Weaver, and Jingren Zhou. Scope: easy and efficient parallel processing of massive data sets. *Proceedings of the VLDB Endowment*, 1(2):1265–1276, 2008.

[62] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL http://doi.acm.org/10.1145/1541880.1541882.

[63] Badrish Chandramouli, Jonathan Goldstein, and Abdul Quamar. Scalable progressive analytics on big data in the cloud. *Proceedings of the VLDB Endowment*, 6(14): 1726–1737, 2013.

[64] Chris Chatfield. The holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 27(3):264–279, 1978.

[65] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems (TODS)*, 32(2):9–es, 2007.

[66] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 511–519, 2017.

[67] Xiaoqi Chen, Shir Landau-Feibish, Mark Braverman, and Jennifer Rexford. Beaucoup: Answering many network traffic queries, one memory update at a time. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 226–239, 2020.

[68] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *Journal of official statistics*, 6(1):3–73, 1990.

[69] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M Hellerstein, Khaled Elmeleegy, and Russell Sears. Mapreduce online. In *Nsdi*, volume 10, page 20, 2010.

[70] Jeffrey Considine, Marios Hadjieleftheriou, Feifei Li, John Byers, and George Kollios. Robust approximate aggregation in sensor data management systems. *ACM Transactions on Database Systems (TODS)*, 34(1):1–35, 2009.

[71] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[72] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[73] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. Gigascope: A stream database for network applications. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 647–651, 2003.

[74] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8(12):2024–2027, 2015.

[75] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.

[76] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *European Symposium on Algorithms*, pages 605–617. Springer, 2003.

[77] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[78] Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. Learning-based support estimation in sublinear time. *arXiv preprint arXiv:2106.08396*, 2021.

[79] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational ip networks: Methodology and experience. *IEEE/ACM Transactions On Networking*, 9(3):265–279, 2001.

[80] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, and Georgios Smaragdakis. The lockdown effect: Implications of the covid-19 pandemic on internet traffic. In *Proceedings of*

*the ACM Internet Measurement Conference*, IMC '20, page 1–18, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381383. doi: 10.1145/3419394.3423658. URL https://doi.org/10.1145/3419394.3423658.

[81] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[82] Marios Fragkoulis, Paris Carbone, Vasiliki Kalavri, and Asterios Katsifodimos. A survey on the evolution of stream processing systems. *arXiv preprint arXiv:2008.00842*, 2020.

[83] Wayne A Fuller. *Introduction to statistical time series*, volume 428. John Wiley & Sons, 2009.

[84] Edward Gan, Jialin Ding, Kai Sheng Tai, Vatsal Sharan, and Peter Bailis. Moment-based quantile sketches for efficient high cardinality aggregation queries. *arXiv preprint arXiv:1803.01969*, 2018.

[85] Edward Gan, Peter Bailis, and Moses Charikar. Coopstore: Optimizing precomputed summaries for aggregation. *Proceedings of the VLDB Endowment*, 13(12):2174–2187, 2020.

[86] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[87] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 29–43, 2003.

[88] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, 2001.

[89] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 2019. doi: 10.23915/distill.00017. https://distill.pub/2019/visual-exploration-gaussian-processes.

[90] Peter J Haas and Joseph M Hellerstein. Ripple joins for online aggregation. *ACM*

*SIGMOD Record*, 28(2):287–298, 1999.

[91] Alex Hall, Alexandru Tudorica, Filip Buruiana, Reimar Hofmann, Silviu-Ionut Ganceanu, and Thomas Hofmann. Trading off accuracy for speed in powerdrill. 2016.

[92] Douglas M Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[93] Joseph M Hellerstein, Peter J Haas, and Helen J Wang. Online aggregation. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 171–182, 1997.

[94] Daniel N Hill, Houssam Nassif, Yi Liu, Anand Iyer, and SVN Vishwanathan. An efficient bandit algorithm for realtime multivariate optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1813–1821, 2017.

[95] Jordan Hochenbaum, Owen S Vallis, and Arun Kejariwal. Automatic anomaly detection in the cloud via statistical learning. *arXiv preprint arXiv:1704.07706*, 2017.

[96] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, Oct 2004. ISSN 1573-7462. doi: 10.1007/s10462-004-4304-y. URL https://doi.org/10.1007/s10462-004-4304-y.

[97] Alefiya Hussain, John Heidemann, John Heidemann, and Christos Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 99–110, New York, NY, USA, 2003. ACM. ISBN 1-58113-735-4. doi: 10.1145/863955.863968. URL http://doi.acm.org/10.1145/863955.863968.

[98] J-H Hwang, Magdalena Balazinska, Alex Rasin, Ugur Cetintemel, Michael Stonebraker, and Stan Zdonik. High-availability algorithms for distributed stream processing. In *21st International Conference on Data Engineering (ICDE'05)*, pages 779–790. IEEE, 2005.

[99] Muhammad Hussain Iqbal, Tariq Rahim Soomro, et al. Big data analysis: Apache

storm perspective. *International journal of computer trends and technology*, 19(1): 9–14, 2015.

[100] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 59–72, 2007.

[101] Jeffrey Jestes, Ke Yi, and Feifei Li. Building wavelet histograms on large data in mapreduce. *arXiv preprint arXiv:1110.6649*, 2011.

[102] Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. Shedding light on the structure of internet video quality problems in the wild. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 357–368. ACM, 2013.

[103] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *IEEE/ACM Transactions on Networking (ToN)*, 22(1):326–340, 2014.

[104] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. Cfa: A practical prediction system for video qoe optimization. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 137–150, Berkeley, CA, USA, 2016. USENIX Association. ISBN 978-1-931971-29-4. URL http://dl.acm.org/citation.cfm?id=2930611.2930621.

[105] Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. Peeking at a/b tests: Why it matters, and what to do about it. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1517–1525, 2017.

[106] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. Detailed diagnosis in enterprise networks. *ACM SIGCOMM Computer Communication Review*, 39(4):243–254, 2009.

[107] Srikanth Kandula, Kukjin Lee, Surajit Chaudhuri, and Marc Friedman. Experiences

with approximating queries in microsoft's production big-data clusters. *Proceedings of the VLDB Endowment*, 12(12):2131–2142, 2019.

[108] Seyed Jalal Kazemitabar, Ugur Demiryurek, Mohamed Ali, Afsin Akdogan, and Cyrus Shahabi. Geospatial stream query processing using microsoft sql server streaminsight. *Proceedings of the VLDB Endowment*, 3(1-2):1537–1540, 2010.

[109] Adam Kirsch and Michael Mitzenmacher. Less hashing, same performance: building a better bloom filter. In *European Symposium on Algorithms*, pages 456–467. Springer, 2006.

[110] Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C Snoeren. Fault localization via risk modeling. *IEEE Transactions on Dependable and Secure Computing*, 7(4):396–409, 2009.

[111] Marcel Kornacker, Alexander Behm, Victor Bittorf, Taras Bobrovytsky, Casey Ching, Alan Choi, Justin Erickson, Martin Grund, Daniel Hecht, Matthew Jacobs, et al. Impala: A modern, open-source sql engine for hadoop. In *Cidr*, volume 1, page 9, 2015.

[112] Tim Kraska. Northstar: An interactive data science system. 2021.

[113] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 211–224, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1705-4. doi: 10.1145/2398776.2398799. URL http://doi.acm.org/10.1145/2398776.2398799.

[114] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1939–1947, 2015.

[115] Aleksandar Lazarevic, Arindam Banerjee, Varun Chandola, Vipin Kumar, and Jaideep Srivastava. Data mining for anomaly detection. In *Tutorial at the European Conference on Principles and Practice of Knowledge Discovery in Databases*,

2008.

[116] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. Wander join: Online aggregation via random walks. In *Proceedings of the 2016 International Conference on Management of Data*, pages 615–629, 2016.

[117] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 359–370, 2012.

[118] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 101–114, 2016.

[119] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. Nitrosketch: Robust and general sketch-based monitoring in software switches. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 334–350. 2019.

[120] Andra Lutu, Diego Perino, Marcelo Bagnulo, Enrique Frias-Martinez, and Javad Khangosstar. A characterization of the covid-19 pandemic impact on a mobile network operator traffic. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 19–33, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450381383. doi: 10.1145/3419394.3423655. URL https://doi.org/10.1145/3419394.3423655.

[121] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2):330–339, 2010.

[122] Gregory T Minton and Eric Price. Improved concentration bounds for count-sketch. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 669–686. SIAM, 2014.

[123] VM Morgenstern, BR Upadhyaya, and M Benedetti. Signal anomaly detection using modified cusum method. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 2340–2341. IEEE, 1988.

[124] Derek G Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 439–455, 2013.

[125] Hun Namkung, Zaoxing Liu, Daehyeok Kim, Vyas Sekar, Peter Steenkiste, Guyue Liu, Ao Li, Christopher Canel, Adithya Abraham Philip, Ranysha Ware, et al. Sketchlib: Enabling efficient sketch-based monitoring on programmable switches. NSDI.

[126] GP Nason. *Stationary and non-stationary time series*, pages 129 – 142. Geological Society of London, United Kingdom, 2006. ISBN 1862392080.

[127] Joseph Ndong and Kave Salamatian. Signal processing-based anomaly detection techniques: A comparative analysis. 06 2011.

[128] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110, 2008.

[129] Christopher Olston, Edward Bortnikov, Khaled Elmeleegy, Flavio Junqueira, and Benjamin Reed. Interactive analysis of web-scale data. In *CIDR*. Citeseer, 2009.

[130] Keith Ord. Outliers in statistical data : V. Barnett and T. Lewis, 1994, 3rd edition, (John Wiley & Sons, Chichester), 584 pp., [UK pound]55.00, ISBN 0-471-93094-6. *International Journal of Forecasting*, 12(1):175–176, March 1996. URL https://ideas.repec.org/a/eee/intfor/v12y1996i1p175-176.html.

[131] Xinghao Pan, Jiaqi Tan, Soila Kavulya, Rajeev Gandhi, and Priya Narasimhan. Ganesha: Blackbox diagnosis of mapreduce systems. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):8–13, 2010.

[132] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations*

*and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[133] Niketan Pansare, Vinayak Borkar, Chris Jermaine, and Tyson Condie. Online aggregation for large mapreduce jobs. *Proceedings of the VLDB Endowment*, 4(11): 1135–1145, 2011.

[134] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment*, 8(12):1816–1827, 2015.

[135] Duc-Son Pham, Svetha Venkatesh, Mihai Lazarescu, and Saha Budhaditya. Anomaly detection in large-scale data stream networks. *Data Mining and Knowledge Discovery*, 28(1):145–189, 2014.

[136] Louis Plissonneau and Ernst Biersack. A longitudinal view of http video streaming performance. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 203–214. ACM, 2012.

[137] Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S Pai, and Michael J Freedman. Aggregation and degradation in jetstream: Streaming analytics in the wide area. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 275–288, 2014.

[138] Anirudh Ramachandran, Srinivasan Seetharaman, Nick Feamster, and Vijay Vazirani. Fast monitoring of traffic subpopulations. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 257–270, 2008.

[139] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[140] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3009–3017, 2019.

[141] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage*

*systems and technologies (MSST)*, pages 1–10. Ieee, 2010.

[142] Lefteris Sidirourgos, Martin L Kersten, Peter A Boncz, et al. Sciborq: scientific data management with bounds on runtime and quality. In *CIDR*, volume 11, pages 296–301, 2011.

[143] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1067–1075, 2017.

[144] Han Hee Song, Zihui Ge, Ajay Mahimkar, Jia Wang, Jennifer Yates, Yin Zhang, Andrea Basso, and Min Chen. Q-score: Proactive service quality assessment in a large iptv system. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 195–208. ACM, 2011.

[145] Marina Theodosiou. Forecasting monthly and quarterly time series using stl decomposition. *International Journal of Forecasting*, 27(4):1178–1195, 2011.

[146] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2): 1626–1629, 2009.

[147] Daniel Ting. Count-min: optimal estimation and tight error bounds using empirical error distributions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2319–2328, 2018.

[148] Daniel Ting. Approximate distinct counts for billions of datasets. In *Proceedings of the 2019 International Conference on Management of Data*, pages 69–86, 2019.

[149] Ruben Torres, Alessandro Finamore, Jin Ryong Kim, Marco Mellia, Maurizio M Munafo, and Sanjay Rao. Dissecting video server selection strategies in the youtube cdn. In *2011 31st International Conference on Distributed Computing Systems*, pages 248–257. IEEE, 2011.

[150] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and

Neoklis Polyzotis. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, volume 8, page 2182. NIH Public Access, 2015.

[151] Jeffrey Scott Vitter and Min Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. *Acm Sigmod Record*, 28(2):193–204, 1999.

[152] Lu Wang, Robert Christensen, Feifei Li, and Ke Yi. Spatial online sampling and aggregation. *Proceedings of the VLDB Endowment*, 9(3):84–95, 2015.

[153] Weinan Wang, Zhengyi Liu, Xiaolin Shi, and Lucas Pierce. Online fdr controlled anomaly detection for streaming time series. 2019.

[154] Zhewei Wei, Ge Luo, Ke Yi, Xiaoyong Du, and Ji-Rong Wen. Persistent data sketching. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data*, pages 795–810, 2015.

[155] Weng-Keen Wong and Daniel B Neill. Tutorial on event detection kdd 2009.

[156] Yinglian Xie, Vyas Sekar, David A Maltz, Michael K Reiter, and Hui Zhang. Worm origin identification using random moonwalks. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 242–256. IEEE, 2005.

[157] Doris Xin, Hui Miao, Aditya Parameswaran, and Neoklis Polyzotis. Production machine learning pipelines: Empirical analysis and optimization opportunities, 2021.

[158] Doris Xin, Eva Yiwei Wu, Doris Jung-Lin Lee, Niloufar Salehi, and Aditya Parameswaran. Whither automl? understanding the role of automation in machine learning workflows. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2021.

[159] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference*, pages 187–196, 2018.

[160] Zhao Xu, Kristian Kersting, and Lorenzo Von Ritter. Stochastic online anomaly analysis for streaming time series. In *IJCAI*, pages 3189–3195, 2017.

[161] Fangjin Yang, Eric Tschetter, Xavier Léauté, Nelson Ray, Gian Merlino, and Deep Ganguli. Druid: A real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 157–168, 2014.

[162] Mingran Yang, Junbo Zhang, Akshay Gadre, Zaoxing Liu, Swarun Kumar, and Vyas Sekar. Joltik: enabling energy-efficient" future-proof" analytics on low-power wide-area networks. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[163] Hao Yin, Xuening Liu, Feng Qiu, Ning Xia, Chuang Lin, Hui Zhang, Vyas Sekar, and Geyong Min. Inside the bird's nest: Measurements of large-scale live vod from the 2008 olympics. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, IMC '09, pages 442–455, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-771-4. doi: 10.1145/1644893.1644946. URL http://doi.acm.org/10.1145/1644893.1644946.

[164] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 325–338. ACM, 2015.

[165] Minlan Yu, Lavanya Jose, and Rui Miao. Software defined traffic measurement with opensketch. In *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 29–42, 2013.

[166] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[167] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[168] Zeng-Guang Zhou and Ping Tang. Improving time series anomaly detection based

on exponentially weighted moving average (ewma) of season-trend model residuals. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3414–3417. IEEE, 2016.