A TTL-based Approach for Data Aggregation in Geo-distributed Streaming Analytics

Dhruv Kumar[1], Jian Li[2], Abhishek Chandra[1] and Ramesh Sitaraman[2]
[1]{dhruv, chandra}@umn.edu, [2]{jianli, ramesh}@cs.umass.edu
[1]University of Minnesota, Twin Cities, [2]University of Massachusetts, Amherst

ACM Reference Format:
Dhruv Kumar[1], Jian Li[2], Abhishek Chandra[1] and Ramesh Sitaraman[2]. 2019. A TTL-based Approach for Data Aggregation in Geo-distributed Streaming Analytics. In *ACM SIGMETRICS / International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '19 Abstracts), June 24–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3309697.3331491

Streaming data analytics has been an important topic of research in recent years. Large quantities of data are generated continuously over time across a variety of application domains such as web and social analytics, scientific computing and energy analytics. One of the key requirements in modern data analytics services is the real-time analysis of these data streams to extract useful and timely information for the analyst. Several distributed data analytics platforms have been developed in recent times to meet this growing requirement of real-time streaming analytics. Nowadays, a large amount of data is generated continuously by geographically distributed sources (e.g., agents, sensors, mobile devices, edge nodes, etc.) in many streaming applications. For instance, services like Facebook, Twitter and Netflix continuously gather data from the end users for a variety of analytical purposes such as finding the popular web content amongst their users or monitoring the QoS metrics. Large content delivery networks (CDNs) like Akamai that serve a significant fraction of content on the Internet continuously collect data from their edge servers and clients from around the globe to understand what, where and how content is accessed for the purpose of providing content analytics insights to businesses.

Hub-and-spoke model for analytics processing. A typical analytics infrastructure for processing such geo-distributed streams follows a hub-and-spoke model, which conceptually comprises a single centralized "hub" connected to multiple edges by a wide-area network (WAN). The data is either generated at the edge or collected at the edge from clients such as sensors, mobile devices etc. In the latter case, clients report data to the edge that is "closest" to them. Each edge has a cluster of servers to collect and process its data streams and then send the processed data to a central hub for further processing. Analysts directly query the central server for retrieving the relevant analyzed data. In this paper, we limit ourselves to aggregation-based processing which we explain next.

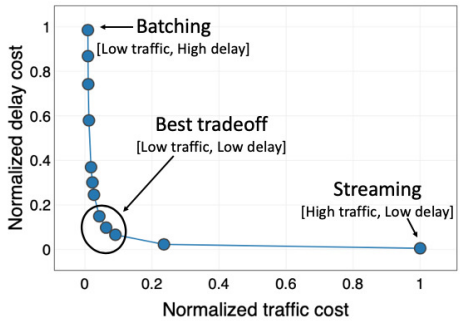

Figure 1: *Delay-traffic tradeoff*. This shows empirically computed delay and traffic for varying amounts of edge-based aggregation for Akamai trace on AWS EC2 testbed.

Data aggregation. We focus on optimizing a common and widely-used operation that is performed within any analytics system – the operation of computing aggregates, such as the Reduce operation in MapReduce, GroupBy in SQL and LINQ etc. We consider data streams in which every record is of the type (k, v), where k is the key and v is its corresponding value, e.g., in the Akamai content analytics context the key could be a combination of content id (url) and geographic location and the value could be the number of clients accessing the url from that location. Aggregation is performed over such a key-value stream by grouping all records $(k, v_i), 1 \leq i \leq n$ that have the same key value k, to produce an aggregate record $(k, v_1 \oplus v_2 \oplus \cdots \oplus v_n)$, where $v_1, v_2, \cdots, v_n$ are the values received for key k up to time T and $\oplus$ is an application-defined associative binary operator. Such operators can be as simple as *sum* or *max*, or more sophisticated, including filters (such as Bloom filters), transforms and sketches (such as HyperLogLog), and user-defined functions. In this paper, we focus on continuous aggregation at the center where the newly arrived data record (k, v) is immediately aggregated into its key k's aggregated value to provide the most updated aggregated result for any key k.

Delay-traffic tradeoff. The data transfer from edges to the center happens over a WAN link which is generally scarce or expensive. To save WAN bandwidth, the computing resources on the edge could be utilized to perform (partial) aggregation on the input data stream before sending intermediate results to the center for a full aggregation. Such edge-based aggregation leads to a fundamental tradeoff between two key metrics: *delay* and *WAN traffic*, as illustrated in Figure 1. Here, delay corresponds to the edge-induced aggregation delay in computing the results, while WAN traffic is the amount of data sent out over wide-area links. Figure 1 shows

[3]For a full journal version of this paper see: Dhruv Kumar, Jian Li, Abhishek Chandra and Ramesh K. Sitaraman. 2019. A TTL-based Approach for Data Aggregation in Geo-distributed Streaming Analytics. Proc. ACM Meas. Anal. Comput. Syst. Vol. 3, No. 2, Article 29 (June 2019). DOI: https://doi.org/10.1145/3326144.

[4]This research was sponsored by the NSF under Grant CNS-1717834, 1717179, and by the U.S. ARL and the U.K. MoD under Agreement Number W911NF-16-3-0001.

SIGMETRICS '19 Abstracts, June 24–28, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6678-6/19/06.
https://doi.org/10.1145/3309697.3331491

the delay-traffic tradeoff, with delay decreasing as traffic increases and vice versa. In particular, the two end-points correspond to two extreme approaches to edge aggregation: *streaming* and *batching*. Streaming refers to sending all the data from edges to the center without any processing at edges. This approach is able to achieve the desired low delay but results in high WAN traffic. On the other hand, batching refers to aggregating the data at the edge for a long time (typically, a few hours) and then sending the summarized data to the center for further processing. This approach is able to achieve the desired low traffic but results in high delay. However, different analytics systems, or even different applications using the same system, require different tradeoffs between delay and traffic.

Real-world examples. We discuss examples of analytics services from industry to further support the different delay-traffic tradeoff requirements explained above.

Akamai Media Analytics: Akamai Media Analytics provides insights into online video performance, quality of experience, and audience behavior by monitoring crucial metrics that drive business critical decisions. Media Analytics consists of two main services: a delay-sensitive Quality of Service (QoS) Monitor and a delay-tolerant Audience Analytics. The customers of both the services are video providers who use Akamai's video delivery services. Both analytics services use a hub-and-spoke model where individual clients (video players) send information as a stream of packets (called "beacons") in real-time to the widely-distributed Akamai edge servers. Each beacon has key-value pairs that are aggregated by the edge servers and then forwarded to a central hub for more aggregation and processing. Users visualize the fully aggregated data by accessing the central hub.

QoS Monitor analyzes the quality of video streams using metrics such as startup time, rebuffer rates, audience size, bitrates, and availability in near real-time. The goal is for video providers using the service to get a real-time view of how their end-users are experiencing their video streams and take immediate diagnostic action if there is a noticeable quality degradation. A common use case is when a large live streaming event such as the FIFA soccer world cup is being delivered by Akamai. The analytics service is used to quickly identify and solve video quality degradations, even as the event is in progress. Given the performance diagnostic goals of the service, reducing delay is extremely important, even at a greater traffic cost.

Audience Analytics provides analytics around the behavior of the audience engaging with video content using metrics such as time spent per video, geographic location of the user and so on. This is not a service used for performance diagnostics, but rather for the video provider to gain analytic insights useful for the business. So, it is more crucial to reduce the traffic cost, even at the expense of greater delay.

Twitter Analytics: A large number of businesses use Twitter for their marketing and advertisement campaigns. In such cases, the businesses want to understand how their audience engages with their brands and campaigns. Twitter Analytics such as trending hashtags, trending topics, audience statistics etc are very common for making business decisions. The need for real-time updates versus reducing traffic cost may vary depending on the specific use. For instance, real-time advertisement campaigns are delay sensitive. But, brand awareness campaigns have longer-term goals and less delay sensitive, making traffic cost reduction important. Consequently, our model and algorithms allow for adjusting the relative weights of delay and traffic at a granular level, even allowing per-key weights.

In summary, between the extremes of streaming and batching, there are different operating points for a wide-area data analytics system that represent different tradeoffs between delay and traffic. *A goal of our work is to devise edge aggregation mechanisms that can provably achieve the desired delay-traffic tradeoffs.*

Research contributions. In this paper, we propose a novel *Time-to-Live (TTL)-based aggregation* mechanism for online stream aggregation that provably optimizes delay and traffic jointly. In this approach, records corresponding to each key are aggregated at the edge for a certain time period dictated by its TTL, before the aggregates are sent over the WAN to the center. The proposed approach is able to achieve the desired delay-traffic tradeoff, and is also able to satisfy the *low delay - low traffic* requirement where needed. To the best of our knowledge, we are the first to provide a theoretical basis for understanding the delay-traffic tradeoff that is fundamental to streaming analytics. In doing so, we provide analytical answers to how much aggregation should be performed at the edge versus the center, how much delay can be incurred at the edges, and how the edge-to-center bandwidth must be apportioned across different applications with different delay requirements. We study the tradeoff between delay and traffic by presenting a family of optimal TTL-based algorithms for jointly minimizing both delay and traffic. In addition, we showcase the generalizability of our proposed model by solving two complementary problems: (i) minimizing delay under a traffic constraint and (ii) minimizing traffic under a delay constraint. This paper makes the following research contributions:

- To the best of our knowledge, the proposed TTL-based aggregation model is the first to provide a theoretical basis for understanding the delay-traffic tradeoff that is fundamental to geo-distributed streaming analytics. Our model also characterizes the storage requirements at the edges to achieve such a tradeoff.
- Using this model, we show how to optimize delay and traffic jointly, achieving a user-desired delay-traffic tradeoff, while also characterizing a stable "sweet spot" operating region that achieves the "best" tradeoff where both delay and traffic are relatively small.
- We have implemented the TTL-based aggregation mechanism in Apache Flink, a popular stream analytics framework. As part of this implementation, we provide a simple, expressible API for users to easily leverage the proposed optimization framework. In addition, our optimization dynamically adapts to changing workloads.
- We evaluate our approach through experiments running our Flink implementation on geo-distributed Amazon EC2 data centers, as well as a local cluster emulating WAN characteristics. The experiments are driven by real-world Akamai and Twitter traces. Our empirical results are in close agreement with our theoretical model predictions. Further, we show that by deriving the optimal TTLs using our model, our system can achieve a "sweet spot" stable operating point where both delay and traffic are minimized, in comparison to traditional aggregation schemes such as batching and streaming.