# A Bird's-Eye View of Complex Streaming Data Analytics

Minos Garofalakis
Athena Research and Innovation Center
Athens, Greece
minos@athenarc.gr

## Abstract

Massive continuous data streams arise naturally in several dynamic big data analytics applications, such as enabling observability for complex distributed systems, network-operations monitoring in large ISPs, or incremental federated learning over dynamic distributed data. In such settings, usage information from numerous devices needs to be continuously collected and analyzed for interesting trends and real-time reaction to different conditions (e.g., anomalies/hotspots, DDoS attacks, or concept drifts). Streaming data raises important memory-, time-, and communication-efficiency issues, making it critical to carefully optimize the use of available computation and communication resources. We give a (biased) overview of some key algorithmic tools in the space of streaming data analytics, along with relevant applications and challenges.

## CCS Concepts

• **Information systems → Stream management**.

## Keywords

Data streams, Data analytics, Sketches, Randomized algorithms, Distributed streaming

**ACM Reference Format:**
Minos Garofalakis. 2025. A Bird's-Eye View of Complex Streaming Data Analytics. In *The 19th ACM International Conference on Distributed and Event-based Systems (DEBS '25), June 10–13, 2025, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3701717.3733848

## Overview

Filters and aggregate statistics constitute the workhorse of data analytics, and are implemented in all database systems and big data platforms. Accordingly, indexing, storage, and processing techniques have been implemented to answer such queries efficiently, even over large data volumes. Still, analytics on *streaming* data raises difficult challenges, which call for novel approaches. In particular, due to the massive (potentially, unbounded) size of contemporary streams, storing the full stream such that it can be processed later for answering ad-hoc queries becomes impractical. Typical streaming applications, such as monitoring of network traffic or application observability data, require *extremely fast query response times* (in the order of milliseconds) to support reactive applications. For example, in network monitoring for early detection of DDoS attacks, a delay of a couple of seconds when answering a network-statistics query may provide sufficient time for an attacker to compromise a corporate network. Furthermore, the *high stream update rates* (millions of updates per second) that are typical of contemporary streams render traditional data structures for speeding up analytics queries essentially useless in such settings. Finally, the majority of large-scale stream processing applications rely on continuous tracking of measurements and events in *distributed* environments, with several remote monitor sites observing their local data streams and exchanging information through a communication network. This distribution implies critical communication constraints that prohibit centralizing all the streaming data, due to either technical (e.g., bandwidth, power) or administrative restrictions.

**Approximate Data Stream Processing.** Abstractly, a relational data stream can be modeled as a massive, dynamic data-distribution vector $A$ that is continuously rendered through a continuous stream of updates to the underlying relation. This general model can naturally capture general, *turnstile* streams (where updates can insert or delete data), as well as special cases such as *cash-register* (insert-only) and *time-series* data streams [14, 23] The generic goal of streaming data analytics is to compute queries (or, functions) on the vector $A$ at any point during the lifetime of the stream (continuous or ad-hoc), while only using space and time that is (significantly) *sublinear* in the size of $A$ (i.e., without storing/accessing the entire stream). The go-to techniques for complex streaming analytics typically rely on *sketches*: compact data structures that can effectively summarize streaming data using small space and can be updated and queried in small time, where "small" usually means logarithmic or poly-logarithmic in the size of the stream [4]. Sketches are randomized structures that typically employ random hash functions to map elements of the stream into a small number of random subsets, maintaining an appropriate counter for each subset. As a result, sketches can often be thought of as collections of *random linear projections* of the streaming vector $A$ — this linearity implies not only fast incremental maintenance but also straightforward sketch merging/composition operations. Furthermore, sketches are often backed by rigorous theoretical analysis, allowing the user to control the space/accuracy trade-off with strong probabilistic accuracy guarantees. Owing to their simplicity and high performance, sketches have been widely adopted across various domains (in both research and industrial settings) to meet diverse application requirements. For instance, frequency-based sketches (e.g., *AMS* [2] and *Count-Min* [7]) have been used for tracking complex analytics such as top-$k$/heavy-hitter elements [3, 6], join/multi-join aggregates [1, 9, 13], and Haar wavelets [5]; similarly, set-based sketches (e.g., *Flajolet-Martin (FM)* [11] and *HyperLogLog* [10]) have been deployed for tracking distinct element counts and set-expression cardinalities [12, 19, 20].


*DEBS '25, Gothenburg, Sweden*

ACM ISBN 979-8-4007-1332-3/25/06
https://doi.org/10.1145/3701717.3733848

The scope and versatility of sketches have been significantly extended with the emergence of *composite sketching* (aka"sketch-in-sketch") techniques, where the key idea is to replace the simple counters used in conventional sketch structures with embedded sketches/summaries to enable additional query functionality. Early examples include the *ECM sketch* [25] that embeds approximate sliding-window counters (e.g., exponential histograms [8]) in Count-Min sketches to effectively track frequency-based analytics in the sliding-window model [14]; and, the *Many-Distinct-Count sketch* of Ting [29] where Count-Min counters are replaced by HyperLogLog sketches to enable space-efficient distinct-count estimation over potentially billions of data sets. The recently-proposed *OmniSketch* [26] addresses a key limitation of existing sketch structures when dealing with multi-dimensional data, namely that they are typically purpose-built for a specific query type. By employing per-dimension Count-Min structures that embed fixed-size samples (maintained via minwise hashing), OmniSketch enables the effective approximation of *any counting query* with dynamically-specified predicates (at query time); thus, it makes a substantial step in the direction of "general-purpose" sketch structures.

**Distributed Data Streaming.** The naturally distributed nature of large-scale stream-monitoring applications implies one additional level of complexity, in the sense that there is no centralized observation point for the dynamic stream vector $A$; instead, $A$ is fragmented across several sites. More specifically, we consider a distributed computing environment, comprising a collection of *remote sites* and a designated *coordinator site*. Local streams are continuously updated at remote sites and, at any point in time, the global stream vector $A$ is defined as an aggregation (e.g., weighted average) of the local stream vectors. As earlier, our focus is on the problem of effectively answering user queries over the global stream vector $A$ at the coordinator. Rather than one-time query evaluation, we assume a continuous-querying environment which implies that the coordinator needs to continuously track the answers to queries as the local streams evolve at individual remote sites. The distributed nature and large data stream volumes imply that the naive solution of continuously shipping all streaming data to the coordinator (turning this into a conventional, centralized streaming problem) is clearly impractical. Thus, novel distributed monitoring protocols that can effectively trade-off space/time/communication efficiency and query-approximation accuracy are required. Furthermore, while simple protocols based on allocating local slacks to remote sites are possible for the special case of linear aggregate queries (e.g., sum, count) [21, 24], such approaches fail for complex *non-linear queries* which raise difficult technical challenges.

To address the general problem, Sharfman et al. [28] consider the fundamental primitive of *distributed threshold monitoring*: determine whether $q(A) > \tau$, for a given *general aggregate query* $q$ over the global stream vector and a fixed threshold $\tau$. Their key idea is that, since it is generally impossible to connect the locally-observed values of the query to the global value $q(A)$, one can employ geometric arguments to monitor the *domain* (rather than the range) of the monitored query. Their proposed *Geometric Method (GM)* offers the first communication-efficient protocol for efficiently tracking general queries over distributed streams. In a nutshell, GM relies on locally monitoring (at remote sites) geometric regions of the stream vector domain defined through the local data streams. Since its inception, GM has been extended in numerous ways, including the incorporation of prediction models [18], sketches [15], and sampling [17]; other work [16, 22] has explored interesting generalizations of GM ideas through the lens of convex analysis, leading to much more efficient distributed stream monitoring protocols, like *Functional Geometric Monitoring (FGM)* [27].

## References


[1] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. 1999. "Tracking Join and Self-Join Sizes in Limited Storage". In *ACM PODS*

[2] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. "The Space Complexity of Approximating the Frequency Moments". In *ACM STOC*.

[3] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. "Finding Frequent Items in Data Streams". In *ICALP*.

[4] Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. 2012. "Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches". *Foundations and Trends in Databases* 4, 1-3 (2012).

[5] Graham Cormode, Minos Garofalakis, and Dimitris Sacharidis. 2006. "Fast Approximate Wavelet Tracking on Streams". In *EDBT*.

[6] Graham Cormode and S. Muthukrishnan. 2003. "What's Hot and What's Not: Tracking Most Frequent Items Dynamically". In *ACM PODS*.

[7] G. Cormode and S. Muthukrishnan. 2004. "An Improved Data Stream Summary: The Count-Min Sketch and its Applications". In *LATIN*.

[8] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 2002. "Maintaining Stream Statistics over Sliding Windows". In *ACM-SIAM SODA*.

[9] Alin Dobra, Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2002. "Processing Complex Aggregate Queries over Data Streams". In *ACM SIGMOD*.

[10] Marianne Durand and Philippe Flajolet. 2003. "Loglog Counting of Large Cardinalities (Extended Abstract)". In *ESA*.

[11] Philippe Flajolet and G. Nigel Martin. 1985. "Probabilistic Counting Algorithms for Data Base Applications". *Journal of Computer and Systems Sciences*.

[12] Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. 2003. "Processing Set Expressions over Continuous Update Streams". In *ACM SIGMOD*.

[13] Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. 2004. "Processing Data-Stream Join Aggregates Using Skimmed Sketches". In *EDBT*.

[14] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2016. *"Data-Stream Management – Processing High-Speed Data Streams"*. Springer-Verlag.

[15] Minos Garofalakis, Daniel Keren, and Vasilis Samoladas. 2013. "Sketch-based Geometric Monitoring of Distributed Stream Queries". In *VLDB*.

[16] Minos Garofalakis and Vasilis Samoladas. 2017. "Distributed Query Monitoring via Convex Analysis: Towards Composable Safe Zones". In *ICDT*.

[17] N. Giatrakos, A. Deligiannakis, and M. Garofalakis. 2016. "Scalable Approximate Query Tracking over Highly Distributed Data Streams". In *ACM SIGMOD*.

[18] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. 2014. "Distributed Geometric Query Monitoring using Prediction Models". *ACM TODS*.

[19] Phillip B. Gibbons. 2001. "Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports". In *VLDB*.

[20] S. Heule, M. Nunkesser, and A. Hall. 2013. "HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm". In *EDBT*.

[21] Ram Keralapura, Graham Cormode, and J. Ramamirtham. 2006. "Communication-efficient distributed monitoring of thresholded counts". In *ACM SIGMOD*.

[22] A. Lazerson, I. Sharfman, D. Keren, A. Schuster, M. Garofalakis, and V. Samoladas. 2015. "Monitoring Distributed Streams using Convex Decompositions". In *VLDB*.

[23] S. Muthukrishnan. 2005. "Data Streams: Algorithms and Applications". *Foundations and Trends in Theoretical Computer Science* 1, 2.

[24] Chris Olston, Jing Jiang, and Jennifer Widom. 2003. "Adaptive Filters for Continuous Queries over Distributed Data Streams". In *ACM SIGMOD*.

[25] Odysseas Papapetrou, Minos Garofalakis, and Antonios Deligiannakis. 2012. "Sketch-based Querying of Distributed Sliding-Window Data Streams". In *VLDB*.

[26] W. Punter, O. Papapetrou, and M. Garofalakis. 2024. "OmniSketch: Efficient Multi-Dimensional High-Velocity Stream Analytics with Arbitrary Predicates". *VLDB*.

[27] Vasilis Samoladas and Minos Garofalakis. 2019. "Functional Geometric Monitoring for Distributed Streams". In *EDBT*.

[28] I. Sharfman, A. Schuster, and D. Keren. 2006. "A geometric approach to monitoring threshold functions over distributed data streams". In *ACM SIGMOD*.

[29] Daniel Ting. 2019. "Approximate Distinct Counts for Billions of Datasets". In *ACM SIGMOD*.