# Continuous and Concurrent Data Summarization

Vinh Quang Ngo
Supervised by: Marina Papatriantafilou
Chalmers Un. of Technology and Gothenburg Un.
vinhq@chalmers.se

## ABSTRACT

In today's data-driven world, the deluge of information from digitized systems poses significant processing challenges. Synopses of data offer a compelling solution by providing effective ways to process, structure, and analyze high-volume data streams without the computational burden of logging every event. These compact representations serve not only as analysis results themselves, but also as "distilled" input for downstream data mining and machine learning applications in high-rate data pipelines. Our research addresses the challenge of maintaining these synopses efficiently in continuous/parallel environments. Specifically, we target algorithmic constructs and data structures that support concurrent updates and queries across varying window frames while enabling mergeable summaries through state partitioning. By bridging the gap between data summarization techniques and parallel processing, we aim to overcome the limitations of traditional sequential approaches that struggle with modern data velocities.

## KEYWORDS

Stream Processing, Parallel Algorithms, Approximation Algorithms, High-performance Algorithms

**ACM Reference Format:**

Vinh Quang Ngo and Supervised by: Marina Papatriantafilou. 2025. Continuous and Concurrent Data Summarization. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 PROBLEM STATEMENT

With the growing interest in digitalization and its associated ecosystems, the need to store, process, index, and gain valuable insights from data is imperative. However, given the increasing volume, velocity, and variety of data that must be generated and processed, two approaches have been proposed to address this issue: data summarization (or synopsis) and concurrent/parallel Processing.

The first approach, data summarization, is based on the observation that big data is often very large but also often ephemeral, with the value brought by different pieces of data being uneven. Instead of needing to store, process, and index the entire amount of data, we can extract useful information from massive data sets into synopses data structures, typically requiring much less space and computation. Examples include simple functions such as min, max, average, or median, and more complex ones such as statistics about the frequencies of encountered elements and heavy hitters. During this process, some information may be lost compared to the original amount of data. Therefore, depending on the problem, the processed data may exist in the form of probabilistic data structures. This means the calculation results from these compact probabilistic data structures can return an approximate result with a bounded difference from the accurate answer and can allow a small probability of deviation from the bounded guarantee.

In another approach, researchers believe that we need to make fuller use of the resources we have, especially parallel hardware that has become popular with today's commodity devices. This is necessary in an era when Moore's law is gradually ending, as we can no longer rely on the processing speed of hardware increasing exponentially every couple of years. However, the speed-up due to concurrency and parallelism will not come in straight-forward, formal definitions and reasons are described rigorously in Amdahl's law. There are many things that need to be carefully designed for a concurrent system. These include work partitioning, concurrent/parallel access control, resource partitioning and replication, interacting with hardware, languages and environments, relaxed semantics, etc. Proving an effective concurrent system is not just about increased throughput or reduced latency experimentally; it also needs to ensure its safety and liveness properties, often referred to as correctness and progress.

Although the above two approaches are different, they are not mutually exclusive. This means that we can apply both methods simultaneously. Thus, the effective combination of both methods, known as concurrent data summarization, is expected to bring a new perspective, possibilities to big data processing e.g., allowing concurrent insertions and queries to happen which is essential to reduce latency in many critical systems or achieve good scalability.

## 2 RELATED WORK

### 2.1 On Data Summarization (Synopses)

Comprehensive overviews on synopses over data sets and data streams can be found in [6, 9, 18], where known methods for summarizing streams using sampling, histograms, wavelets, and sketches are explained in depth, while the needs and technical challenges associated with parallelization and distributed settings are emphasized as demanding research directions. In the following paragraphs, we give a brief outline of highlights on data summarization and associated trends.

*Sampling:* Sampling is considered one of the most fundamental and widely used techniques for data summarization, with a long


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

history. The core concept involves random selection of items of a stream (or a set), followed possibly by statistical modeling. Sampling is indeed a powerful tool for data analytics, but in some cases can perform not as intended, depending on the purposes of the task or the data distribution, especially when the data is highly skewed [6]. As a result, alternative methods for data summarization, such as histograms, wavelets, and sketches, continue to develop concurrently with sampling.

*Histograms:* Histograms, first introduced by Karl Pearson [24], represent the distribution of data sets, displaying the frequency or probability of different values within predefined intervals or "bins". They give a rough sense of the density of the underlying distribution of the data and are often used for density estimation. Such data structures are used widely, especially for data visualization, as they are intuitive means for identifying patterns and trends in the data. Histograms have also been studied in the database literature for many years as a query-answering technique [5, 10, 25].

*Wavelets:* A wavelet transformation decomposes the data set into a coarse overall approximation together with detailed coefficients hierarchically. Furthermore, wavelet transforms are generally simple linear transformations, thus allowing for efficient synopsis-construction algorithms. The space for research related to wavelets applied in databases and data streams attracts significant interest [7, 8, 20, 21] given the promising application potential, as well as the scientific challenges.

*Sketches:* A sketch is a type of synopsis that is a linear transformation of input; as such, it can be updated flexibly and efficiently to capture certain properties of a data set or data stream, such as frequencies of elements observed, or estimation of joints or self joins and support of range queries. It features highly desirable properties for high-rate stream processing such as speed, space-efficiency, delete-compliance, and especially mergeability. The latter also makes it easy to optimize, parallelize sketch updates, and prove correctness. Compared to sampling or histograms, the history of sketches is relatively young, but because of their useful properties, they are applied extensively to various fields and use-cases.

From 2010, with the fast development of multi-core computing devices and distributed systems, alongside the emergence of interest and progress in ML, new research directions and applications of sketches in those fields have been in focus, motivated by the fast, compact, and mergeable nature of sketches. These will be discussed in subsequent parts of this section.

## 2.2 On Concurrent Data Summarization

As data volumes and rates continue to grow while applications evolve, there is an increasing need for solutions that go beyond sequential processing capabilities, posing *challenges on both scalable parallel performance and support for concurrent queries and updates* [9]. While there is some recent work realizing such needs [27, 29, 33], parallelizing summaries has focused mainly on parallelizing inserts (cf. [19, 37]), with only a few recent work [15, 28, 32], to the best of our knowledge, on concurrent queries and inserts.

Concurrency implies the need to deal with consistency and progress properties when accessing shared data, i.e. when considering shared data objects. *Sequential consistency* [17], and *linearizability* [14] are safety properties to characterize consistency of concurrent object accesses, equivalent to sequential executions. They guarantee that each operation appears as if it takes place in an atomic instance of time, with effects equivalent to the object's sequential specification. Linearizability requires additionally that each such atomic time instance (linearization point) falls in the duration of the actual operation it corresponds to. This implies that linearizable objects' executions can combine and the resulting execution is still linearizable [14]. There exist other consistency property formulations that are concurrency-aware and allow for relaxed conditions, e.g., *regularity* and *quiescence consistency*, which distinguish their requirements, depending on whether operations act concurrently on an object or not [13, 22].

Recent works, including [3, 12, 22, 30] and references therein, propose relaxing linearizability and sequential consistency requirements by, roughly speaking, allowing equivalence to controlled perturbations of sequential executions instead of strictly sequential executions with the same operations, trading off timeliness with consistency guarantees. In the context of leveraging the aforementioned consistency notions for summaries, [26] proposed *Intermediate Value Linearizability* (IVL) as a correctness criterion, that relaxes linearizability to allow a query to return intermediate values of concurrent updates of a (monotonically increasing) summary data structure, showing that concurrent IVL implementations of $(\epsilon, \delta)$-bounded approximations (i.e. with an error of at most $\epsilon$ with probability at least $1 - \delta$), such as the Count-Min sketch, are themselves $(\epsilon, \delta)$-bounded. In a similar spirit, the Delegation sketch, proposed in [34], is constructed to satisfy quiescent consistency while keeping competitiveness regarding the $(\epsilon, \delta)$-bound compared to baseline parallelization schemes, due to its state partitioning.

These consistency models relate closely to the notion of determinism in stream processing, which targets equivalence to sequential processing of streaming data ordered by event-time cf [4, 11] and references therein). Relaxations of strict determinism allow bounded deviations (cf [2, 16, 35, 36] and references therein), which proves particularly useful when data arrives in arbitrary order (aka cash-register model).

For distributed environments, the concept of mergeable summaries becomes essential, extending beyond linear sketches to enable combining partial results from different processing nodes [1, 23, 31]. This mergeability facilitates both distribution and concurrency, opening interesting research questions about various synopsis types that permit merging in some form.

# 3 APPROACH

We approach the problem of concurrent data summarization from multiple complementary directions:

*Summaries for Massive Data Streams:* Data-stream processing is all the more important in the evolving landscape of digitized, cyber-physical systems, in order to generate continuous streams of information useful for improving the underlying physical systems, e.g., electricity grids, vehicular systems, production systems, and even data networks. It is often desirable to transform the data into forms that facilitate further processing, e.g., for the detection of anomalies, the identification of similarities for optimization, and various other data mining and ML types of analysis. Data summaries of the type that we mentioned in the previous subsections have

been shown to be useful for such purposes. Furthermore, different transformations are useful at the same time, e.g., for supporting range queries, combining different streaming operators, such as aggregation and filtering. Hence, *multiple summaries* may need to be maintained in parallel. How to design *efficient data structures* that can be *updated and queried in multiple ways, concurrently* is an active research problem. Also, how to maintain synopses in the *sliding window model* (i.e., over bounded subsets instead of the whole stream) represents a relatively unexplored yet promising research direction, with limited existing work.

*Parallelism and Concurrency for Summaries:* As mentioned in an earlier paragraph, common correctness criteria for concurrent object constructions are *linearizability* and *sequential consistency*, which guarantee that all operations' effects are observed in total order, consistent with the target's sequential specification, while for concurrency-aware semantics notions such as *regularity, quiescent consistency, eventual consistency* describe possible guarantees in presence of concurrent operations on the data [13]. Since data summaries (synopses, sketches, as described earlier in this document) provide results with approximation guarantees (semantic relaxation), it is possible to consider similarly regarding consistency requirements in the presence of concurrent updates and queries. Our research will explore how the inherent approximation properties of data synopses can be leveraged to design more efficient concurrent operations. Beyond correctness properties, we will develop techniques for effective work partitioning and state sharding that minimize contention in high-throughput scenarios while preserving summary accuracy. This includes investigating lock-free and wait-free data structure implementations tailored specifically for summary data types, as well as exploring hardware-aware optimizations such as cache-aware data layouts and SIMD instruction utilization. Importantly, our research targets parallelization approaches that can be applied across entire algorithm categories (e.g., heavy hitter detection, quantile estimation, wavelet transforms), allowing existing algorithms and systems to benefit from concurrency and parallelism without requiring complete redesigns or specialized implementations.

*ML for Summaries and Summaries for ML:* We will investigate the bidirectional relationship between machine learning (ML) and data summarization. In one direction, data summarization techniques can transform high-dimensional data into lower dimensions, reducing communication costs and making training and inference processes faster. Another example is clustering algorithms on large datasets do not need to work on the full dataset. They can operate on data summaries or micro-clusters which represent groups of points that are extremely similar to each other and would fall into the same cluster regardless of the selected configuration and distance metric. In the other direction, ML can enhance data summarization by exploiting patterns found in the input data to improve the accuracy of results. Our research will focus on developing techniques that integrate these two approaches, particularly in concurrent environments where traditional sequential approaches face limitations.

## 4 EVALUATION PLAN

Our evaluation strategy is to assess the effectiveness of our concurrent data summarization techniques across multiple dimensions, including diverse hardware configurations, shared-memory multicore systems, distributed clusters, and heterogeneous computing platforms, to demonstrate adaptability and scalability characteristics. To promote adoption and practical utility, we plan to integrate our implementations into established open-source frameworks, such as Apache Flink, Apache Spark Streaming, and the Apache DataSketch project. Many performance metrics will be taken into account, such as throughput, latency, accuracy, and resource utilization (memory footprint, cache behavior). We will benchmark against both synthetic workloads with varying skewness and real-world datasets from public repositories. Importantly, we will validate our approaches through collaborations with industrial partners. This industrial validation will help guide our research priorities toward the most impactful aspects of concurrent data summarization and demonstrate the real-world utility of our proposed methods.

## 5 CONCLUSIONS AND REFLECTIONS

In this paper, we have outlined a comprehensive approach to concurrent data summarization, targeting the growing challenges of processing high-velocity, high-volume data streams. By bridging techniques from data summarization and parallel processing, we aim to develop algorithmic constructs and data structures that support efficient concurrent updates and queries across varying window frames while maintaining accuracy guarantees.

Our initial research has focused on the fundamental problem of heavy hitter detection in data streams, resulting in both an adaptive sequential algorithm with strong generalization properties and a framework for parallelizing existing heavy hitter algorithms. These contributions provide a solid foundation for our ongoing work on broader synopsis types including wavelets and quantiles, as well as advanced operational models such as sliding windows and out-of-order data processing.

Despite the clear benefits, adoption of advanced data summarization techniques remains primarily limited to large technology companies with specialized knowledge. We believe this represents both a challenge and an opportunity. By contributing accessible implementations, integrating with popular open-source frameworks, and demonstrating concrete performance improvements in real-world scenarios, we can help bridge the gap between theoretical advances and practical applications.

Looking forward, we will continue to explore the interplay between consistency models, accuracy guarantees, and performance optimizations in concurrent environments. As digitized systems become increasingly ubiquitous, we expect the demand for efficient, continuous data processing to grow, making data summarization, especially its concurrent/parallel variants essential components of future data infrastructure.

## ACKNOWLEDGMENTS

Work supported by Marie Skłodowska-Curie Doctoral Network RELAX-DN, funded by EU under Horizon Europe 2021-2027 FP Grant Agreement nr. 101072456 (www.relax-dn.eu/); Swedish Research Council prj."EPITOME" 2021-05424; prj TANDEM (Swedish Energy Agency SESBC); Chalmers AoA Energy and Production, WPs "Scalability, Big Data and AI" and INDEED.

# REFERENCES


[1] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. 2012. Mergeable summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '12)*. Association for Computing Machinery, 23–34.

[2] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, et al. 2015. The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *VLDB* (2015).

[3] Dan Alistarh, Trevor Brown, Justin Kopinsky, Jerry Z Li, and Giorgi Nadiradze. 2018. Distributionally linearizable data structures. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. 133–142.

[4] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache Flink™: Stream and Batch Processing in a Single Engine. In *Proceedings of the 14th ACM International Conference on Extending Database Systems*. ACM, 281–294.

[5] D. Chamberlin. 1998. *A Complete Guide to DB2 Universal Database*. Morgan Kaufmann.

[6] 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), 1–294.

[7] Minos Garofalakis. 2018. *Discrete Wavelet Transform and Wavelet Synopses*. Springer New York, New York, NY, 4588–4595. https://doi.org/10.1007/978-1-4614-8265-9_453

[8] Minos Garofalakis. 2018. *Wavelets on Streams*. Springer New York, New York, NY.

[9] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2016. *Data stream management: processing high-speed data streams*. Springer.

[10] P. B. Gibbons, Y. Matias, and V. Poosala. 1997. *Aqua Project White Paper*. Technical Report. Bell Laboratories, Murray Hill, NJ.

[11] Vincenzo Gulisano, Yiannis Nikolakopoulos, Daniel Cederman, Marina Papatriantafilou, and Philippas Tsigas. 2017. Efficient data streaming multiway aggregation through concurrent algorithmic designs and new abstract data types. *ACM TOPC* (2017).

[12] Thomas A Henzinger, Christoph M Kirsch, Hannes Payer, Ali Sezgin, and Ana Sokolova. 2013. Quantitative relaxation of concurrent data structures. In *Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 317–328.

[13] Maurice Herlihy, Nir Shavit, Victor Luchangco, and Michael Spear. 2020. *The art of multiprocessor programming*. Newnes.

[14] Maurice P. Herlihy and Jeannette M. Wing. 1990. Linearizability: A Correctness Condition for Concurrent Objects. *ACM Transactions on Programming Languages and Systems* 12, 3 (1990), 463–492.

[15] Victor Jarlow, Charalampos Stylianopoulos, and Marina Papatriantafilou. 2024. QPOPSS: Query and Parallelism Optimized Space-Saving for Finding Frequent Stream Elements. arXiv:2409.01749 [cs.DC]

[16] Yuanzhen Ji, Jun Sun, Anisoara Nica, Zbigniew Jerzak, Gregor Hackenbroich, and Christof Fetzer. 2016. Quality-driven disorder handling for m-way sliding window stream joins. In *ICDE, Helsinki, Finland*. IEEE, 493–504.

[17] Leslie Lamport. 1979. How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs. *IEEE Trans. Comput.* C-28, 9 (September 1979), 690–691.

[18] Jure Leskovec, Anand Rajaraman, and Jeff Ullman. 2014. *Mining of Massive Datasets 3/e*. Cambridge University Press.

[19] Ankush Mandal, He Jiang, Anshumali Shrivastava, and Vivek Sarkar. 2018. Topkapi: parallel and fast sketches for finding top-K frequent elements. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., 10921–10931.

[20] Ioannis Mytilinis, Dimitrios Tsoumakos, and Nectarios Koziris. 2019. Maintaining wavelet synopses for sliding-window aggregates. In *Proceedings of the 31st International Conference on Scientific and Statistical Database Management*. 73–84.

[21] Ioannis Mytilinis, Dimitrios Tsoumakos, and Nectarios Koziris. 2020. Workload-aware wavelet synopses for sliding window aggregates. *Distributed and Parallel Databases* (2020), 1–38.

[22] Ioannis Nikolakopoulos, Anders Gidenstam, and Marina Papatriantafilou et al. 2015. A Consistency Framework for Iteration Operations in Concurrent Data Structures. In *29th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2015*. Hyderabad, India, 239–248.

[23] Odysseas Papapetrou, Minos Garofalakis, and Antonios Deligiannakis. 2015. Sketching distributed sliding-window data streams. *The VLDB Journal* 24, 3 (2015), 345–368.

[24] K. Pearson. 1895. Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 186 (1895), 343–414. https://doi.org/10.1098/rsta.1895.0010

[25] G. Piatetsky-Shapiro and C. Connell. 1984. Accurate Estimation of the Number of Tuples Satisfying a Condition. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 256–276.

[26] Arik Rinberg and Idit Keidar. 2020. Intermediate Value Linearizability: A Quantitative Correctness Criterion. In *34th International Symposium on Distributed Computing (DISC 2020)*.

[27] Arik Rinberg and Idit Keidar. 2023. Intermediate Value Linearizability: A Quantitative Correctness Criterion. *J. ACM* 70, 2, Article 17 (2023).

[28] Arik Rinberg, Alexander Spiegelman, Edward Bortnikov, Eshcar Hillel, Idit Keidar, Lee Rhodes, and Hadar Serviansky. 2020. Fast Concurrent Data Sketches. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (San Diego, California) *(PPoPP '20)*. Association for Computing Machinery, New York, NY, USA, 117–129. https://doi.org/10.1145/3332466.3374512

[29] Arik Rinberg, Alexander Spiegelman, Edward Bortnikov, Eshcar Hillel, Idit Keidar, Lee Rhodes, and Hadar Serviansky. 2022. Fast Concurrent Data Sketches. *ACM Trans. Parallel Comput.* 9, 2, Article 6 (2022).

[30] Adones Rukundo, Aras Atalar, and Philippas Tsigas. 2019. Monotonically Relaxing Concurrent Data-Structure Semantics for Increasing Performance: An Efficient 2D Design Framework. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[31] Zubair Shah, Abdun Naser Mahmood, Zahir Tari, and Albert Y Zomaya. 2017. A technique for efficient query estimation over distributed data streams. *IEEE Transactions on Parallel and Distributed Systems* 28, 10 (2017), 2770–2783.

[32] Charalampos Stylianopoulos. 2020. Delegation Sketch. https://github.com/mpastyl/DelegationSketch.

[33] Charalampos Stylianopoulos, Ivan Walulya, Magnus Almgren, Olaf Landsiedel, and Marina Papatriantafilou. 2020. Delegation sketch: a parallel design with support for fast and accurate concurrent operations. In *Proceedings of the 15th European Conference on Computer Systems (EuroSys '20)*. Association for Computing Machinery, Article 4.

[34] Charalampos Stylianopoulos, Ivan Walulya, Magnus Almgren, Olaf Landsiedel, and Marina Papatriantafilou. 2020. Delegation sketch: a parallel design with support for fast and accurate concurrent operations. In *Proceedings of the 15th ACM European Conference on Computer Systems (EuroSys)*. 1–16.

[35] Joris van Rooij, Vincenzo Gulisano, and Marina Papatriantafilou. 2020. TinTiN: Travelling in time (if necessary) to deal with out-of-order data in streaming aggregation. In *Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems*. 141–152.

[36] Nikos Zacheilas, Vana Kalogeraki, Yiannis Nikolakopoulos, Vincenzo Gulisano, Marina Papatriantafilou, and Philippas Tsigas. 2017. Maximizing determinism in stream processing under latency constraints. In *ACM DEBS*.

[37] Yu Zhang, Yue Sun, Jianzhong Zhang, Jingdong Xu, and Ying Wu. 2014. An efficient framework for parallel and continuous frequent item monitoring. *Concurrency and Computation: Practice and Experience* 26, 18 (2014), 2856–2879.