Brief Announcement: Fast Concurrent Data Sketches

Arik Rinberg
arikrinberg@campus.technion.ac.il
Technion

Alexander Spiegelman
spiegelmans@vmware.com
VMware Research

Edward Bortnikov
ebortnik@verizonmedia.com
Yahoo Research

Eshcar Hillel
eshcar@verizonmedia.com
Yahoo Research

Idit Keidar
idish@ee.technion.ac.il
Technion

Hadar Serviansky
hadar.serviansky@weizmann.ac.il
Weizmann Institute

ABSTRACT

Data sketches are approximate succinct summaries of long data streams. They are widely used for processing massive amounts of data and answering statistical queries about it. Existing libraries producing sketches are very fast, but do not allow parallelism for creating sketches using multiple threads or querying them while they are being built. We present a generic approach to parallelising data sketches efficiently and allowing them to be queried in real time, while bounding the error that such parallelism introduces. Utilising relaxed semantics and the notion of strong linearisability we prove our algorithm's correctness and analyse the error it induces in two specific sketches. Our implementation achieves high scalability while keeping the error small. We have contributed one of our concurrent sketches to the open-source data sketches library.

CCS CONCEPTS

• **Theory of computation** → *Parallel algorithms*; • **Computer systems organization** → *Parallel architectures*; Real-time systems.

KEYWORDS

concurrency, synchronization, persistence, design, analysis of distributed algorithms

ACM Reference Format:
Arik Rinberg, Alexander Spiegelman, Edward Bortnikov, Eshcar Hillel, Idit Keidar, and Hadar Serviansky. 2019. Brief Announcement: Fast Concurrent Data Sketches. In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29–August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3293611.3331567

1 INTRODUCTION

Data sketching algorithms, or *sketches* for short [6], have become an indispensable tool for high-speed computations over massive datasets in recent years. Their applications include a variety of analytics and machine learning use cases, e.g., data aggregation [2, 4], graph mining [5], anomaly (e.g., intrusion) detection [20], real-time data analytics [8], and online classification [16].

Sketches are designed for *stream* settings in which each data item is only processed once. A sketch data structure is essentially a succinct (sublinear) summary of a stream that approximates a specific query (unique element count, quantile values, etc.). The approximation is typically very accurate – the error drops fast with the number of processed elements [6].

Practical implementations of sketch algorithms have recently emerged in toolkits [19] and data analytics platforms (e.g., PowerDrill [12], Druid [8], Hillview [17], and Presto [1]). However, these implementations are not thread-safe, allowing neither parallel data ingestion nor concurrent queries and updates; concurrent use is prone to exceptions and gross estimation errors. Applications using these libraries are therefore required to explicitly protect all sketch API calls by locks [9, 13].

In our full paper [15], we present a generic approach to parallelising data sketches efficiently, while bounding the error that such a parallelisation might introduce. Our goal is to enable simultaneous queries and updates to a sketch from an arbitrary number of threads. Our solution is carefully designed to do so without slowing down operations as a result of synchronisation. This is particularly challenging because sketch libraries are extremely fast, often processing tens of millions of updates per second.

We capitalise on the well-known sketch *mergeability* property [6], which enables computing a sketch over a stream by merging sketches over substreams. Previous works have exploited this property for distributed stream processing (e.g., [7, 12]), devising solutions with a sequential bottleneck at the merge phase and where queries cannot be served before all updates complete. In contrast, our method is based on shared memory, with parallel updates of small thread-local sketches, and continuous background propagation of local results to a common, queryable sketch.

We instantiate our generic algorithm with two popular sketches from the open-source Java DataSketches library [19]: (1) a KMV Θ sketch [4], which estimates the number of unique elements in a stream; and (2) a Quantiles sketch [2] estimating the stream element with a given rank. Our design is generic and applicable to additional sketches. Figure 1 compares the ingestion throughput of our concurrent Θ sketch to that of a lock-protected sequential sketch, on multi-core hardware. As expected, the trivial solution does not scale whereas our algorithm scales linearly.

Concurrency induces an error, and one of the main challenges we address is analysing this additional error. To begin with, we need to specify a correctness criterion for the concurrent sketch. We do so using a flavour of *relaxed consistency* due to Henzinger et al. [11] that allows operations to "overtake" some other operations. Thus, a query may return a result that reflects all but a bounded number of the updates that precede it. While relaxed semantics were previously used for deterministic data structures like stacks [11]

PODC '19, July 29–August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07.
https://doi.org/10.1145/3293611.3331567

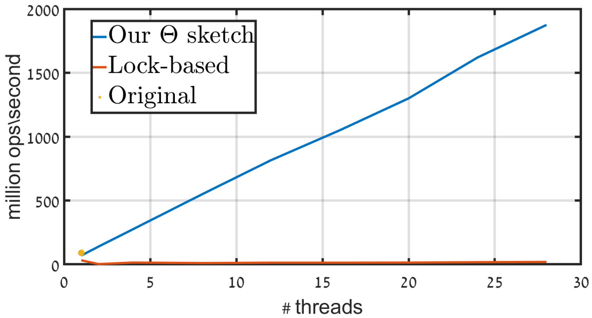

Figure 1: Scalability of DataSketches' Θ sketch protected by a lock vs. our concurrent implementation.

and priority queues [3, 14], we believe that they are a natural fit for data sketches. This is because sketches are typically used to summarise streams that arise from multiple real-world sources and are collected over a network with variable delays, and so even if the sketch ensures strict semantics, queries might miss some real-world events that occur before them. Additionally, sketches are inherently approximate. Relaxing their semantics therefore "makes sense", as long as it does not excessively increase the expected error.

But this raises a new difficulty: relaxed consistency is defined wrt a deterministic specification, whereas sketches are randomised. We therefore first de-randomise the sketch's behaviour by delegating the random coin flips to an oracle. We can then relax the resulting sequential specification. Next, because our concurrent sketch is used within randomised algorithms, it is not enough to prove its linearisability. Rather, we prove that our generic concurrent algorithm instantiated with sequential sketch S satisfies *strong linearisability* [10] wrt a relaxed sequential specification of the de-randomised S.

We then analyse the error of the two relaxed sketches under random coin flips, with an adversarial scheduler that may delay operations in a way that maximises the error. We show that our concurrent Θ sketch's error is coarsely bounded by twice that of the corresponding sequential sketch. The error of the concurrent Quantiles sketch approaches that of the sequential one as the stream size tends to infinity.

Main contribution. In summary, our full paper [15] this paper tackles an important practical problem, offers a general efficient solution for it, and rigorously analyses this solution. While the paper makes use of many known techniques, it combines them in a novel way; we are not aware of any previous application of relaxed consistency to randomised statistical algorithms. The main technical challenges we address are (1) proving the relaxed consistency of a high-performance generic algorithm that supports real-time queries concurrently with updates; and (2) analysing the error induced by this relaxation. We have contributed our parallel Θ sketch implementation to the DataSketches library [18].

REFERENCES

[1] 2018. HyperLogLog in Presto: A significantly faster way to handle cardinality estimation. https://code.fb.com/data-infrastructure/hyperloglog/.

[2] 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)*. ACM, New York, NY, USA, 23–34. https://doi.org/10.1145/2213556.2213562

[3] Dan Alistarh, Justin Kopinsky, Jerry Li, and Nir Shavit. 2015. The SprayList: A Scalable Relaxed Priority Queue. *SIGPLAN Not.* 50, 8 (Jan. 2015), 11–20. https://doi.org/10.1145/2858788.2688523

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2002. Counting Distinct Elements in a Data Stream. In *Randomization and Approximation Techniques in Computer Science*, Jos'e D. P. Rolim and Salil Vadhan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–10.

[5] Edith Cohen. 2014. All-distances Sketches, Revisited: HIP Estimators for Massive Graphs Analysis. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '14)*. ACM, New York, NY, USA, 88–99. https://doi.org/10.1145/2594538.2594546

[6] Graham Cormode. 2017. Data Sketching. *Queue* 15, 2, Article 60 (April 2017), 19 pages. https://doi.org/10.1145/3084693.3104030

[7] Graham Cormode, S Muthukrishnan, and Ke Yi. 2011. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms (TALG)* 7, 2 (2011), 21.

[8] Druid. [n. d.]. How We Scaled HyperLogLog: Three Real-World Optimizations. http://druid.io/blog/2014/02/18/hyperloglog-optimizations-for-real-world-systems.html.

[9] Github. [n. d.]. ArrayIndexOutOfBoundsException during serialization. https://github.com/DataSketches/sketches-core/issues/178#issuecomment-365673204..

[10] Wojciech Golab, Lisa Higham, and Philipp Woelfel. 2011. Linearizable implementations do not suffice for randomized distributed computation. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 373–382.

[11] Thomas A Henzinger, Christoph M Kirsch, Hannes Payer, Ali Sezgin, and Ana Sokolova. 2013. Quantitative relaxation of concurrent data structures. In *ACM SIGPLAN Notices*, Vol. 48. ACM, 317–328.

[12] Stefan Heule, Marc Nunkesser, and Alex Hall. 2013. HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm. In *Proceedings of the EDBT 2013 Conference*. Genoa, Italy.

[13] Lee Rhodes. [n. d.]. SketchesArgumentException: Key not found and no empty slot in table. https://groups.google.com/d/msg/sketches-user/S1PEAneLmhk/dI8RbN6iBAAJ..

[14] Hamza Rihani, Peter Sanders, and Roman Dementiev. 2014. Multiqueues: Simpler, faster, and better relaxed concurrent priority queues. *arXiv preprint arXiv:1411.1209* (2014).

[15] Arik Rinberg, Alexander Spiegelman, Edward Bortnikov, Eshcar Hillel, Idit Keidar, and Hadar Serviansky. 2019. Fast Concurrent Data Sketches. *arXiv preprint arXiv:1902.10995* (2019).

[16] Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. 2018. Sketching Linear Classifiers over Data Streams. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18)*. ACM, New York, NY, USA, 757–772. https://doi.org/10.1145/3183713.3196930

[17] VMware. [n. d.]. Hillview: A Big Data Spreadsheet. https://github.com/vmware/hillview.

[18] Yahoo. [n. d.]. DataSketches: Concurrent Theta Sketch Implementation. https://github.com/DataSketches/sketches-core/blob/master/src/main/java/com/yahoo/sketches/theta/ConcurrentDirectQuickSelectSketch.java.

[19] Yahoo! [n. d.]. DataSketches: sketches library from Yahoo! https://datasketches.github.io/.

[20] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic Sketch: Adaptive and Fast Network-wide Measurements. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. ACM, New York, NY, USA, 561–575. https://doi.org/10.1145/3230543.3230544