Real Time Analytics: Algorithms and Systems

Arun Kejariwal, Sanjeev Kulkarni, Karthik Ramasamy
Twitter Inc.
{akejariwal, skulkarni, kramasamy}@twitter.com

ABSTRACT

Velocity is one of the **4 V**s commonly used to characterize Big Data [27]. In this regard, Forrester remarked the following in Q3 2014 [94]: *"The high velocity, white-water flow of data from innumerable real-time data sources such as market data, Internet of Things, mobile, sensors, clickstream, and even transactions remain largely unnavigated by most firms. The opportunity to leverage streaming analytics has never been greater."* Example use cases of streaming analytics include, but not limited to: (a) visualization of business metrics in real-time (b) facilitating highly personalized experiences (c) expediting response during emergencies. Streaming analytics is extensively used in a wide variety of domains such as healthcare, e-commerce, financial services, telecommunications, energy and utilities, manufacturing, government and transportation.

In this tutorial, we shall present an in-depth overview of streaming analytics – applications, algorithms and platforms – landscape. We shall walk through how the field has evolved over the last decade and then discuss the current challenges – the impact of the other three **V**s, viz., **V**olume, **V**ariety and **V**eracity, on Big Data streaming analytics. The tutorial is intended for both researchers and practitioners in the industry. We shall also present state-of-the-affairs of streaming analytics at Twitter.

1. INTRODUCTION

Big Data is characterized by the increasing volume (of the order of zetabytes), and the velocity of data generation [31, 126]. It is projected that the market size of Big Data will climb up from the current market size of $5.1 billion to $53.7 billion by 2017 [12]. In recent years, Big Data analytics has been transitioning from being predominantly offline (or batch) to primarily online (or streaming). The trend is expected to become mainstream owing to the various facets, exemplified below, of the emerging data-driven society [136].

- Social media: Over 500M tweets are created everyday. A key challenge in this regard is how to surface the most personalized content in real time.
- Internet of Things (IoT): By 2020, the number of connected devices is expected to grow by 50% to 30 billion [26]. Data from embedded systems - the sensors and systems that monitor the physical universe - is expected to rise to 10% (from the current 2%) of the digital universe by 2020.
- Health Care: Increasingly Big Data is being leveraged in health care to, for example, improve both quality and efficiency in health care areas such as readmissions, adverse events, treatment optimization, and early identification of worsening health states or highest-need populations [147]. The volume of healthcare data is expected to swell to 2,314 exabytes by 2020, from 153 exabytes in 2013 [64].

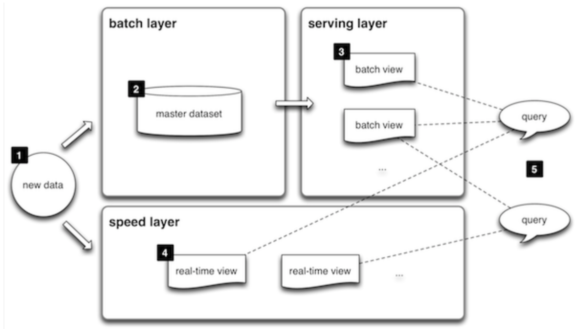

Figure 1: Overview of Lambda Architecture (source: [18])

- Machine data: With cloud computing becoming ubiquitous, machine generated data is expected to grow to 40% of the digital universe by 2020 [20].
- Connected vehicles: New telematics systems and the installation of ever greater numbers of computer chips, applications, electronic components and many other components provide data on vehicle usage, wear and tear, or defects [13]. The volume of data transferred per vehicle per month is expected to grow from around 4 MB to 5 GB. Further, by 2016 as many as 80% of all vehicles sold worldwide are expected to be "connected".

Over the years, several streaming platforms have been developed. Examples include, S4 [131], Samza [8], Sonora [167], Millwheel [37], Photon [40], Storm [158], Flink [4], Spark [9], Pulsar [130] and Heron [118]. Some of these platforms have been open sourced. The evolution of the streaming platforms is discussed in detail in Section 3.

In order to be satisfy both, batch and streaming analytics, Lambda Architecture (LA) has been proposed as a robust, distributed platform to serve a variety of workloads, including low-latency high-reliability queries [18] (refer to Figure 1). The various stages of LA are explained below:

1. Input **data** is dispatched to both the batch layer and the speed layer for processing.
2. The **batch layer** manages the master dataset (an immutable, append-only set of raw data) and pre-computes the batch views.
3. The **serving layer** indexes the batch views so that they can be queried in a low-latency, ad-hoc way.
4. The **speed layer** handles recent data only to compensate for the high latency of updates to the serving layer.
5. Incoming **queries** are answered by merging results from batch views and real-time views.

Several platforms have been built based on the Lambda Architecture. Examples include Summingbird [24] and Lambdoop [19]. Commercial platforms such as TellApart [25] are also based on the Lambda Architecture.

Problem	**Description**	**Application**
Sampling [45, 169, 68, 33, 156, 88, 90, 51, 69, 70]	Obtain a representative set of the stream	A/B Testing
Filtering [49, 62, 133, 76, 50, 102, 116, 143, 138, 129, 73, 171, 144, 82]	Extract elements which meet a certain criterion	Set membership
Correlation [163, 146, 134, 165, 99]	Find data subsets (subgraphs) in (graph) data stream which are highly correlated to a given data set	Fraud detection
Estimating Cardinality [86, 46, 78, 92, 85, 89, 54, 112, 103, 59, 157]	Estimate the number of distinct elements	Site audience analysis
Estimating Quantiles [93, 42, 170, 97, 107, 123, 148]	Estimate quantiles of a data stream with small amount of memory	Network analysis
Estimating Moments [39, 63, 109, 48, 96]	Estimating distribution of frequencies of different elements	Databases
Finding Frequent Elements [125, 75, 114, 66, 110, 57, 67, 128, 65, 154, 155, 124, 84, 106, 104, 166, 145, 52, 137]	Identify items in a multiset with frequency more than a threshold θ	Trending Hashtags
Counting Inversions [36]	Estimate number of inversions	Measure sortedness of data
Finding Subsequences [122, 152, 87, 159]	Find Longest Increasing Subsequences (LIS), Longest Common Subsequence (LCS), subsequences similar to a given query sequence	Traffic analysis
Path Analysis [79]	Determine whether there exists a path of length $\leq \ell$ between two nodes in a dynamic graph	Web graph analysis
Anomaly Detection [135, 151, 150, 115, 71, 77, 47, 153, 43]	Detect anomalies in a data stream	Sensor networks
Temporal Pattern Analysis [60, 168, 38]	Detect patterns in a data stream	Traffic analysis
Data Prediction [111, 162, 100, 164, 142, 160]	Predict missing values in a data stream	Sensor data analysis
Clustering [98, 132, 105]	Cluster a data stream	Medical imaging
Graph analysis [83, 101, 35, 113, 127, 61, 80]	Extract unweighted and weighted matching, vertex cover, independent sets, spanners, subgraphs (sparsification) and random walks, computing min-cut	Web graph analysis
Basic Counting [72]	Estimate $\hat{m}$ of the number m of 1-bits in the sliding window (of size n) such that $\|\hat{m} - m\| \leq \epsilon m$	Popularity Analysis
Significant One Counting [119]	Estimate $\hat{m}$ of the number m of 1-bits in the sliding window (of size n) such that if $m \geq \theta n$, then $\|\hat{m} - m\| \leq \epsilon m$	Traffic accounting [81]

Table 1: Streaming algorithms and their applications

The rest of the proposal is organized as follows: Section 2 overviews the various problems addressed previously in the context of streaming analytics and their real-world applications. Section 3 walks through the evolution of streaming platforms over the last decade. Finally, we conclude in Section 4.

2. STREAMING ALGORITHMS

Elements of a data stream need to be processed in real time, else one may lose the opportunity to process them at all. Thus, it is critical that the data footprint of the algorithm fits in the main memory. Also, in light of the real-time constraint, it may be preferable to compute an approximate solution than an exact solution. Research in approximation algorithms for problems defined over data streams has led to some general techniques for data reduction and synopsis construction, including:

- ***Sampling***: Techniques such as reservoir sampling [161], weighted sampling [58] have been proposed to capture the essential characteristics of a data stream.
- ***Sliding windows***: Use of sliding windows prevents stale data from influencing analysis and statistics and also serve as a tool for approximation, given bounded memory. The following problems for sliding windows are being actively researched: clustering, maintaining statistics like variance, and computing correlated aggregates.
- ***Clustering***: Algorithms for problems such as the k-median problem – wherein the objective is to choose k representative points, such that the sum of the errors over the n data points is minimized – have been proposed based on clustering [98, 95, 149, 34].
- ***Sketches***: Randomized sketching, introduced by Alon et al. [39], summarizes a data stream using a small amount of memory. The sketch if used to estimate the answer to certain queries (typically, "distance" queries) over a data set.
- ***Histograms***: V-Optimal histogram approximates the distribution of a set of values $v_1, \ldots v_n$ by a piecewise-constant function $\hat{v}(i)$, so as to minimize the sum of squared error. Equi-width histograms partition the domain into buckets such that the number of v_i values falling into each bucket is uniform across all buckets. End-biased histograms maintain exact counts of items that occur with frequency above a threshold, and approximate the other counts by a uniform distribution.
- ***Wavelets***: Wavelets coefficients are projections of the given signal (set of data values) onto an orthogonal set of basis vectors. The coefficients have the desirable property that the signal reconstructed from the top few wavelet coefficients best approximates the original signal in terms of the L_2 norm [91]. The choice of basis vectors determines the type of wavelets.

Further, to be able to support Web scale and high velocity data, the algorithms should intrinsically distribute computation across multiple nodes and, if required, across data centers. In other words, the algorithms should be able to scale out.

In light of the dynamic nature of streaming data, a field of incremental machine learning has emerged to cater to Big Data streaming analytics. The techniques being developed are designed to work with incomplete data, to identify

Platform	Description
S4 [131]	Real-time analytics with a key-value based programming model and support for scheduling/message passing and fault tolerance
Storm [158]	The most popular and widely adopted real-time analytics platform developed at Twitter
Millwheel [37]	Google's proprietary realtime analytics framework thats provides exact once semantics
Samza [8]	Framework for topology-less real-time analytics that emphasizes sharing between groups
Akka [1]	Toolkit for writing distributed, concurrent and fault tolerant applications
Spark [9]	Does both offline and online analysis using the same code and same system
Flink [4]	Fuses offline and online analysis using traditional RDBMS techniques
Pulsar [130]	Does real-time analytics using SQL
Heron [118]	Storm re-imagined with emphasis on higher scalability and better debuggability

Table 2: Open source streaming platforms

hidden variables to help steer future data collection and to quantify the change between one or more states of the model.

Common streaming operators include, but not limited to: filtering, time windows, aggregation/correlation, temporal patterns, location/motion, enrichment, query and action interfaces. Table 1 lists some of the most common problems addressed in prior research in the domain of streaming analytics and their example applications in the real world. Examples of use of streaming analytics include, but not limited to, (a) sequence mining [139, 121, 117] for, say, credit card fraud detection, motion capture sequences and chlorine levels in drinking water (b) discovering human activity, which often exhibit discontinuity (interruption) or varying frequencies, from sensor streams [140] (c) determing top-K traversal sequences in streaming clicks (d) finding closed structures in music melody streams [120]. In December 2015, yahoo! open source a library called *DataSketches* for approximate analysis of Big Data [141].

We shall walk through some of the problems and the recent algorithms in the tutorial. Further, we shall throw light on the scalability of the existing approaches at Web scale.

3. STREAMING PLATFORMS

In late 1990s and early 2000s, main memory DataBase Management Systems (DBMSs) and rule engines[1], were re-purposed and remarketed to cater to stream processing. However, these systems did not scale with high volume data streams (models and issues in data stream systems are discussed in detail in [44]). Later on, Stream Processing Engines (SPEs) such as Aurora [53], STREAM [41], TelegraphCQ [55] and Borealis [32] were proposed. Even these systems did not scale with the increasing velocity and volume of the data streams characteristic of modern systems. To this end, several streaming platforms have been developed in the industry. Table 2 summarizes the various streaming platforms developed over the years.

Some of the common requirements of streaming systems are itemized below:

- Provide resiliency against stream "imperfections", including missing and out-of-order data, which are commonly present in data streams in production.
- Must guarantee predictable and repeatable outcomes.
- Ensure that the applications are up and available, and the integrity of the data is maintained at all times despite failures (which can happen due to, for example, node failures, network failures, software bugs and resource limitations [108]).
- Distribute processing across multiple processors and machines to achieve incremental scalability.
- Should be easy to integrate with a batch processing data pipeline (ala the Lambda architecture described in Section 1). This is key for a wide variety of applications, such as online fraud detection, electronic trading based on historical patterns.

In the rest of this section, we briefly overview the platforms listed in Table 2. In addition, we also overview low-latency platforms built on top of Hadoop. In the tutorial, we shall walk the audience through the different design choices of the various platforms and the challenges which still remain.

S4 [131] is one of earliest distributed streaming system developed by Yahoo! It is near real-time, scalable and event-driven platform that allows easy implementation of applications for processing unbounded streams of data. At a high level, it allows for easy assembly of small applications into larger ones, flexible and easy deploy, provides fault tolerance for high availability, checkpointing and a recovery mechanism for minimizing state loss. The platform handles communication, scheduling and distribution. S4 streaming applications are modeled as a graph, with vertices representing computation (called processing elements) and the edges representing streams of data. The applications are deployed on S4 clusters that run several distributed containers called S4 nodes. Processing elements communicate asynchronously by sending events on streams. These events are routed to the appropriate nodes according to their key.

Apache Storm [158] is the next generation system that is widely popular and open sourced by Twitter. Storm applications, referred to as topologies, is a DAG where the vertices can either represent a data source (spouts) and a computation (bolts). These topologies are run on a Storm cluster. Storm provides guarantees about data processing with support for *at least once* and *almost once* semantics. It is horizontally scalable thereby allowing the cluster to expand and supports robust fault tolerance for process and machine failures. Storm data model allows users to express their analytics concisely. A Storm cluster consists of Nimbus that acts as a master node and is responsible for scheduling and distribution of topologies. Other nodes in the cluster, called Slave Nodes, run Storm Supervisor that spawns workers which actually run the user logic code.

MillWheel [37] is a key-value based streaming system developed at Google. A MillWheel application is a directed

[1]A rule engine typically accepts condition/action pairs, usually expressed using "if-then" notation. As streaming data enters the system, it is immediately matched against the existing rules. When the condition of a rule is matched, the rule is said to "fire". The corresponding action(s) taken may then produce alerts/outputs to external applications or may simply modify the state of internal variables, which may in turn lead to further rule firings.

graph where each node is a computational unit and the vertices are the messages passed between them. MillWheel distributes the computational nodes across the cluster and repairs them in case units/machines go down. Mill wheel also provides *exactly once* semantics by checkpointing state every time. To checkpoint reliably, MillWheel uses BigTable [56]. MillWheel's programming model provides a notion of logical time, making it simple to write time-based aggregations. MillWheel is closed-source.

Apache Samza [8] is a realtime, asynchronous computational framework for stream processing developed at LinkedIn. Unlike Storm or MillWheel, where you stitch together a bunch of computations in a topology, a Samza application is single computational task scaled across several partitions. Each Samza application reads one or more input streams and can output zero or more output streams. One can then stitch together several such applications to form a Storm like topology doing a given higher level function. Samza uses Kafka [7] to manage the input and output streams. One side-effect of using Kafka for stream management is that Samza inherits all the persistence and fault-tolerance of Kafka. As all streams exist on Kafka, one does not need external systems/brokers for inter-application communication. However this comes at the cost of increased latency as even the intermediate stages have to be persisted to disk.

Akka [1] is a toolkit for building distributed, concurrent and fault-tolerant applications. One can use Akka to build general data processing applications – batch or streaming. An Akka application consists of a set of Akka Actors and messages passed between those Actors. An Akka Actor is very similar to a Storm Bolt, except that it is very lightweight. Thus, it is commonplace to see millions of Akka Actors in a single Akka application. Actors process messages asynchronously and each actor instance is guaranteed to be run using at most one thread at a time, making concurrency much easier. Akka provides out-of-the-box primitives to distribute actors across the cluster, do load balancing of messages and repair lost actors. A unique feature of Akka is that actors can reply to incoming messages thereby giving it a request-response capability thats usually not present in systems.

Apache Spark [9] is an effort that came out of AMPLabs Berkeley to replace Hadoop's two stage disk-based MapReduce paradigm. Spark provides in-memory primitives which allow intermediate data to be kept in memory. Spark distributes Resilient Distributed Datasets (RDDs) throughout the cluster and can even store them to disk for persistence. For a class of iterative machine learning algorithms, this in-memory approach provides 100× more throughput than traditional MapReduce based implementations. As a result, the community has built a large set of ML and Graph processing libraries on top of Spark. APIs are provided in Java/Python/Scala languages. Spark also provides streaming primitives so that streaming applications can run in the same cluster as batch applications. This consolidation of infrastructure for running disparate classes of applications drives down the opex cost significantly. Spark Streaming provides a high-level abstraction called discretized stream or DStream, which represents a continuous stream of data. DStreams can be created either from input data streams from sources such as Kafka, Flume [5], Twitter, ZeroMQ [30], Kinesis [2] or TCP sockets or by applying high-level operations on other DStreams (internally, a DStream is represented as a sequence of RDDs). The data can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and live dashboards. Spark streaming supports stateful *exactly once* semantics out-of-the-box.

Apache Flink [4] takes a different approach to achieve the same goal as Spark. Flink borrows concepts from the traditional RDBMS world like byte-buffer based data serialization and binary representation of data (instead of Java/Scala object representation). Flink has a cost-based optimizer, akin to relational platforms that selects execution strategies and avoids expensive partitioning and sorting steps. Moreover, Flink features a special kind of iterations called delta-iterations that can significantly reduce the amount of computations as iterations go on. Like Spark, Flink also unifies stream and batch processing.

Pulsar [130] is a realtime analytics engine open sourced by eBay. A unique feature of Pulsar is its SQL interface. Thus, instead of writing code in, say, Java, one can just write SQL queries to run on a Pulsar cluster. This eases the use of the analytics pipeline by non-technical business folks who tend to know SQL pretty well. Pulsar transforms each query into a directed acyclic graph (DAG) of processing nodes and distributes them across the cluster. Pulsar achieves low latencies by keeping all intermediate data in memory. However if the downstream components are either down or not able to consume fast enough, it stores the messages into Kafka for later replay. Another neat feature of Pulsar is the ability to dynamically resize queries on the fly while the query is still running. In this way, one can add/remove machines into a Pulsar cluster without affecting any running queries.

After years of experience with Storm, as the scale of data being processed in real-time increased, several issues such as debugability, manageability, scalability and performance became apparent. Most of these issues were a result of the underlying architectural issues such as multiplexing of disparate tasks running user logic code in a single worker process. As a consequence, a worker has a complex set of queues through which the data passes making the performance worse. Heron [118] addresses these issues by running each task in a process of its own thereby making it easy to debug, tune and improved performance.

While Apache Hive [6] opened up HDFS to SQL, its architecture, centered around MapReduce [74], made it unsuitable for interactive querying. Quite a few efforts have been initiated by different companies to solve this problem. The most prominent ones are Drill [3] from MapR, Presto [21] from Facebook, Impala [15] from Cloudera and Tez [10] from Hortonworks. While differing in details, they all generally have the same architecture. All of them prefer to be co-located with the HDFS for best performance. An incoming SQL query is parsed and a physical plan is generated which is then optimized. A query co-ordinator sends pieces of the query to all relevant data nodes where servers execute that part of query, reading from the local data node if needed. This is done to minimize network traffic. The query co-ordinator then merges all the results and returns the combined result back to the user. The systems differ in the flavor of supported SQL (while Presto and Drill support ANSI SQL, Impala supports HiveQL), language of implementation (Impala is written in C++, while others are all Java) and levels of maturity/adoption.

In addition to the platforms discussed above, several commercial stream processing products are available on the market [17, 16, 22, 11, 28, 23, 29]. In [94], Gualtieri and Curran reviewed some of the widely used and emerging commercial streaming analytics platforms. Numenta has developed a tool, called *Grok* [14], for anomaly detection in data streams.

Lastly, we shall walk the audience through the various use cases of Heron for streaming analytics – such as, but not limited to, real-time targeting, content discovery, online machine learning – at Twitter.

4. CONCLUSIONS

In the proposed tutorial, we shall present an in-depth overview of streaming analytics – applications, algorithms and platforms – landscape. We shall walk through how the field has evolved over the last decade and then discuss the current challenges.

5. REFERENCES

[1] Akka. http://akka.io/.
[2] Amazon Kinesis. http://aws.amazon.com/kinesis/.
[3] Apache Drill. http://drill.apache.org/.
[4] Apache Flink. https://flink.apache.org/.
[5] Apache Flume. http://flume.apache.org/.
[6] Apache Hive. https://hive.apache.org/.
[7] Apache Kafka: A high-throughput distributed messaging system. http://kafka.apache.org/.
[8] Apache Samza. https://samza.apache.org/.
[9] Apache Spark. https://spark.apache.org/.
[10] Apache Tez. https://tez.apache.org/.
[11] Apama Streaming Analytics. http://www.softwareag.com/corporate/products/apama_webmethods/analytics/overview/default.asp.
[12] Big Data Market Size and Vendor Revenues. http://wikibon.org/wiki/v/Big_Data_Market_Size_and_Vendor_Revenues.
[13] Connected cars get big data rolling. http://www.telekom.com/media/media-kits/179806.
[14] Grok. http://numenta.com/grok/.
[15] Impala. http://impala.io/.
[16] Informatica Vibe Data Stream. https://www.informatica.com/products/data-integration/real-time-integration/vibe-data-stream.html#fbid=Z8rhqt-b1nd.
[17] InfoSphere Streams: Capture and analyze data in motion. http://www-03.ibm.com/software/products/en/infosphere-streams.
[18] Lambda Architecture. http://lambda-architecture.net/.
[19] Lambdoop. http://lambdoop.com/.
[20] New Digital Universe Study Reveals Big Data Gap: Less Than 1% of Worlds Data is Analyzed; Less Than 20% is Protected. http://www.emc.com/about/news/press/2012/20121211-01.htm.
[21] Presto. https://prestodb.io/.
[22] SAP Event Stream Processor. http://www.sap.com/pc/tech/database/software/sybase-complex-event-processing/index.html.
[23] SQLstream Blaze. http://www.sqlstream.com/blaze/.
[24] Summingbird. https://github.com/twitter/summingbird.
[25] TellApart. http://www.tellapart.com.
[26] The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. http://www.emc.com/leadership/digital-universe/2014iview/internet-of-things.htm.
[27] The Four V's of Big Data. http://www.ibmbigdatahub.com/infographic/four-vs-big-data.
[28] TIBCO StreamBase. http://www.streambase.com/.
[29] Vitria OI For Streaming Big Data Analytics. http://www.vitria.com/solutions/streaming-big-data-analytics/benefits/.
[30] ZeroMQ. http://zeromq.org/.
[31] Federal Government Big Data Rollout. http://www.nsf.gov/news/news_videos.jsp?cntn_id=123607&media_id=72174&org=NSF, 2012.
[32] D. J. Abadi, Y. Ahmad, M. Balazinska, M. Cherniack, J. hyon Hwang, W. Lindner, A. S. Maskey, E. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik. The design of the Borealis stream processing engine. In *Proceedings of the Conference on Innovative Data Systems Research*, pages 277–289, 2005.
[33] C. C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 607–618, Seoul, Korea, 2006.
[34] C. C. Aggarwal. A survey of stream clustering algorithms. In C. Aggarwal and C. Reddy, editors, *Data Clustering: Algorithms and Applications, CRC Press*. 2013.
[35] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: Sparsification, spanners, and subgraphs. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pages 5–14, Scottsdale, AZ, 2012.
[36] M. Ajtai, T. S. Jayram, R. Kumar, and D. Sivakumar. Approximate counting of inversions in a data stream. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, pages 370–379, Montreal, Quebec, Canada, 2002.
[37] T. Akidau, A. Balikov, K. Bekiroğlu, S. Chernyak, J. Haberman, R. Lax, S. McVeety, D. Mills, P. Nordstrom, and S. Whittle. Millwheel: Fault-tolerant stream processing at internet scale. *Proceedings of the VLDB Endowment*, 6(11):1033–1044, Aug. 2013.
[38] F. Alavi and S. Hashemi. Mining jumping emerging patterns by streaming feature selection. In V. N. Huynh, T. Denoeux, D. H. Tran, A. C. Le, and S. B. Pham, editors, *Knowledge and Systems Engineering*, volume 245 of *Advances in Intelligent Systems and Computing*, pages 337–349. 2014.
[39] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 20–29, Philadelphia, Pennsylvania, USA, 1996.
[40] R. Ananthanarayanan, V. Basker, S. Das, A. Gupta, H. Jiang, T. Qiu, A. Reznichenko, D. Ryabkov, M. Singh, and S. Venkataraman. Photon: Fault-tolerant and scalable joining of continuous data streams. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 577–588, 2013.
[41] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. STREAM: The stanford stream data manager (demonstration description). In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 665–665, 2003.
[42] A. Arasu and G. S. Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of the Twenty-third ACM Symposium on Principles of Database Systems*, pages 286–296, Paris, France, 2004.
[43] I. Assent, P. Kranen, C. Baldauf, and T. Seidl. AnyOut: Anytime outlier detection on streaming data. In *Proceedings of the 17th International Conference on Database Systems for Advanced Applications - Volume Part I*, pages 228–242, 2012.
[44] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the Symposium on Principles of Database Systems*, pages 1–16, Madison, Wisconsin, 2002.
[45] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 633–634, San Francisco, California, 2002.
[46] Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In D. P. Rolim and S. Vadhan, editors, *Randomization and Approximation Techniques in Computer Science*, volume 2483 of *Lecture Notes in Computer Science*, pages 1–10. 2002.
[47] M. S. Beigi, S.-F. Chang, S. Ebadollahi, and D. C. Verma. Anomaly detection in information streams without prior domain knowledge. *IBM Journal of Research and Development*, 55(5):550–560, Sept. 2011.
[48] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithm for estimating frequency moments of data streams. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 708–713, Miami, Florida, 2006.
[49] B. H. Bloom. Space/Time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, July 1970.
[50] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese. An improved construction for counting bloom filters. In Y. Azar and T. Erlebach, editors, *Algorithms ESA 2006*, volume 4168 of *Lecture Notes in Computer Science*, pages 684–695. 2006.
[51] V. Braverman, R. Ostrovsky, and C. Zaniolo. Optimal sampling from sliding windows. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 147–156, Providence, Rhode Island, USA, 2009.
[52] T. Calders, N. Dexters, J. J. M. Gillis, and B. Goethals. Mining frequent itemsets in a stream. *Information Systems*, 39:233–255, Jan. 2014.
[53] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: A new class of data management applications. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 215–226, Hong Kong, China, 2002.
[54] Y. Chabchoub and G. Hebrail. Sliding HyperLogLog: Estimating cardinality in a data stream over a sliding window. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, pages 1297–1303, 2010.
[55] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 668–668, 2003.
[56] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, pages 205–218, 2006.
[57] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.
[58] S. Chaudhuri, R. Motwani, and V. Narasayya. On random sampling over joins. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 263–274, Philadelphia, PA, 1999.
[59] A. Chen and J. Cao. Distinct counting with a self-learning bitmap. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 1171–1174, March 2009.
[60] Y. Chen, M. Nascimento, B. C. Ooi, and A. Tung. Spade: On shape-based pattern detection in streaming time series. In *Proceedings of the IEEE 23rd International Conference on Data Engineering*, pages 786–795, April 2007.
[61] R. Chitnis, G. Cormode, M. Hajiaghayi, and M. Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1234–1251, 2015.
[62] S. Cohen and Y. Matias. Spectral bloom filters. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 241–252, 2003.
[63] D. Coppersmith and R. Kumar. An improved data stream algorithm for frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 151–156, New Orleans, Louisiana, 2004.
[64] K. Corbin. How CIOs Can Prepare for Healthcare 'Data Tsunami More like this. http://www.cio.com/article/2860072/healthcare/how-cios-can-prepare-for-healthcare-data-tsunami.html.
[65] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, 1(2):1530–1541, Aug. 2008.
[66] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, pages 464–475, Berlin, Germany, 2003.
[67] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, Apr. 2005.
[68] G. Cormode, S. Muthukrishnan, and I. Rozenbaum. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 25–36, Trondheim, Norway, 2005.
[69] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. Optimal sampling from distributed streams. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 77–86, Indianapolis, Indiana, USA, 2010.
[70] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. Continuous sampling from distributed streams. *Journal of the ACM*, 59(2):10:1–10:25, May 2012.
[71] T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, and K. Yi. Change (detection) you can believe in: Finding distributional shifts in data streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, pages 21–34, 2009.
[72] M. Datar and S. Muthukrishnan. Estimating rarity and similarity over data stream windows. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 323–334, 2002.
[73] J. L. Dautrich, Jr. and C. V. Ravishankar. Inferential time-decaying bloom filters. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 239–250, Genoa, Italy, 2013.
[74] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, pages 10–10, San Francisco, CA, 2004.
[75] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 348–360, 2002.
[76] B. Donnet, B. Baynat, and T. Friedman. Retouched bloom filters: Allowing networked applications to trade off selected false positives against false negatives. In *Proceedings of the 2006 ACM CoNEXT Conference*, pages 13:1–13:12, Lisboa, Portugal, 2006.

[77] P. H. dos Santos Teixeira and R. L. Milidiú. Data stream anomaly detection through principal subspace tracking. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1609–1616, Sierre, Switzerland, 2010.

[78] M. Durand and P. Flajolet. LogLog Counting of Large Cardinalities. In *Annual European Symposium on Algorithms, volume 2832 of LNCS*, pages 605–617, 2003.

[79] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – a technique for speeding up dynamic graph algorithms. *Journal of the ACM*, 44(5):669–696, Sept. 1997.

[80] H. Esfandiari, M. T. Hajiaghayi, V. Liaghat, M. Monemizadeh, and K. Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1217–1233, 2015.

[81] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 323–336, Pittsburgh, Pennsylvania, USA, 2002.

[82] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, pages 75–88, Sydney, Australia, 2014.

[83] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Journal of Theoretical Computer Science*, 348(2):207–216, Dec. 2005.

[84] G. Feigenblat, O. Itzhaki, and E. Porat. The frequent items problem, under polynomial decay, in the streaming model. *Theoretical Computer Science*, 411(34-36):3048–3054, July 2010.

[85] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *PROCEEDINGS OF THE 2007 INTERNATIONAL CONFERENCE ON ANALYSIS OF ALGORITHMS*, 2007.

[86] P. Flajolet and G. N. Martin. Probabilistic counting. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, pages 76–82, Washington, DC, USA, 1983.

[87] A. Gál and P. Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. *SIAM Journal on Computing*, 39(8):3463–3479, Aug. 2010.

[88] S. Gandhi, S. Suri, and E. Welzl. Catching elephants with mice: Sparse sampling for monitoring sensor networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 261–274, Sydney, Australia, 2007.

[89] S. Ganguly. Counting distinct items over update streams. *Theoretical Computer Science*, 378(3):211–222, June 2007.

[90] R. Gemulla and W. Lehner. Sampling time-based sliding windows in bounded space. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 379–392, Vancouver, Canada, 2008.

[91] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 389–398, Montreal, Quebec, Canada, 2002.

[92] F. Giroire. Order statistics and estimating cardinalities of massive data sets. In *Proceedings of the 2005 International Conference on Analysis of Algorithms*, pages 157–166, 2005.

[93] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 58–66, Santa Barbara, California, USA, 2001.

[94] M. Gualtieri and R. Curran. The Forrester Wave$^{\text{TM}}$: Big Data Streaming Analytics Platforms, Q3 2014. 2014.

[95] S. Guha. Tight results for clustering and summarizing data streams. In *Proceedings of the 12th International Conference on Database Theory*, pages 268–275, St. Petersburg, Russia, 2009.

[96] S. Guha, N. Koudas, and K. Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Transactions on Database Systems*, 31(1):396–438, Mar. 2006.

[97] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

[98] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 359–366, 2000.

[99] T. Guo, S. Sathe, and K. Aberer. Fast correlation discovery for large-scale streaming time-series data. Technical Report EPFL-REPORT-200473, EPFL, 2014.

[100] M. Halatchev and L. Gruenwald. Estimating missing values in related sensor data streams. In *Proceedings of the 11th International Conference on Management of Data*, pages 83–94, Hyderabad, India, 2005.

[101] B. V. Halldórsson, M. M. Halldórsson, E. Losievskaja, and M. Szegedy. Streaming algorithms for independent sets. In *Proceedings of the 37th International Colloquium Conference on Automata, Languages and Programming*, pages 641–652, Bordeaux, France, 2010.

[102] F. Hao, M. Kodialam, and T. V. Lakshman. Building high accuracy bloom filters using partitioned hashing. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 277–288, 2007.

[103] S. Heule, M. Nunkesser, and A. Hall. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 683–692, Genoa, Italy, 2013.

[104] N. Homem and J. P. Carvalho. Finding top-k elements in data streams. *Information Sciences*, 180(24):4958 – 4974, 2010.

[105] P. Hore, L. Hall, and D. Goldgof. Creating streaming iterative soft clustering algorithms. In *Annual Meeting of the North American Fuzzy Information Processing Society*, pages 484–488, June 2007.

[106] R. Y. S. Hung, L.-K. Lee, and H. F. Ting. Finding frequent items over sliding windows with constant update time. *Information Processing Letters*, 110(7):257–260, Mar. 2010.

[107] R. Y. S. Hung and H. F. Ting. An $\omega(1/\epsilon \log 1/\epsilon)$ space lower bound for finding ϵ-approximate quantiles in a data stream. In *Proceedings of the 4th International Conference on Frontiers in Algorithmics*, pages 89–100, Wuhan, China, 2010.

[108] J.-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. High-availability algorithms for distributed stream processing. In *Proceedings of the International Conference on Data Engineering*, pages 779–790, April 2005.

[109] P. Indyk and D. Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pages 202–208, Baltimore, MD, 2005.

[110] C. Jin, W. Qian, C. Sha, J. X. Yu, and A. Zhou. Dynamically maintaining frequent items over a data stream. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 287–294, New Orleans, LA, 2003.

[111] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[112] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 41–52, Indianapolis, Indiana, USA, 2010.

[113] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751, Portland, OR, 2014.

[114] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28(1):51–55, Mar. 2003.

[115] M. Khan. Anomaly detection in data streams using fuzzy logic. In *Proceedings of the International Conference on Information and Communication Technologies*, pages 167–174, Aug 2009.

[116] A. Kirsch and M. Mitzenmacher. Less hashing, same performance: Building a better bloom filter. *Random Struct. Algorithms*, 33(2):187–218, Sept. 2008.

[117] A. Koper and H. Nguyen. Sequential pattern mining from stream data. In J. Tang, I. King, L. Chen, and J. Wang, editors, *Advanced Data Mining and Applications*, volume 7121 of *Lecture Notes in Computer Science*, pages 278–291. 2011.

[118] S. Kulkarni, N. Bhagat, M. Fu, V. Kedigehalli, C. Kellogg, S. Mittal, J. M. Patel, K. Ramasamy, and S. Taneja. Twitter Heron: Streaming at scale. In *Proceedings of SIGMOD*, Melbourne, Australia, 2015.

[119] L. K. Lee and H. F. Ting. Maintaining significant stream statistics over sliding windows. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 724–732, Miami, FL, 2006.

[120] H.-F. Li, S.-Y. Lee, and M.-K. Shan. Mining frequent closed structures in streaming melody sequences. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 2031–2034, June 2004.

[121] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. DynaMMo: Mining and summarization of coevolving sequences with missing values. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 507–516, Paris, France, 2009.

[122] D. Liben-Nowell, E. Vee, and A. Zhu. Finding longest increasing and common subsequences in streaming data. In *Proceedings of the 11th Annual International Conference on Computing and Combinatorics*, pages 263–272, Kunming, China, 2005.

[123] Q. Ma, S. Muthukrishnan, and M. Sandler. Frugal streaming for estimating quantiles. In A. Brodnik, A. Lpez-Ortiz, V. Raman, and A. Viola, editors, *Space-Efficient Data Structures, Streams, and Algorithms*, volume 8066 of *Lecture Notes in Computer Science*, pages 77–96. 2013.

[124] N. Manerikar and T. Palpanas. Frequent items in streaming data: An experimental evaluation of the state-of-the-art. *Data & Knowledge Engineering*, 68(4):415–430, Apr. 2009.

[125] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 346–357, Hong Kong, China, 2002.

[126] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity. http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation, May 2011.

[127] A. McGregor. Graph stream algorithms: A survey. *SIGMOD Record*, 43(1):9–20, May 2014.

[128] A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In T. Eiter and L. Libkin, editors, *Database Theory - ICDT 2005*, volume 3363 of *Lecture Notes in Computer Science*, pages 398–412. 2005.

[129] I. Moraru and D. G. Andersen. Exact pattern matching with feed-forward bloom filters. *Journal on Experimental Algorithmics*, 17:3.4:3.1–3.4:3.18, Sept. 2012.

[130] S. Murthy and T. Ng. Announcing Pulsar: Real-time Analytics at Scale. http://www.ebaytechblog.com/2015/02/23/announcing-pulsar-real-time-analytics-at-scale, Feb. 2015.

[131] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, pages 170–177, 2010. http://incubator.apache.org/s4/.

[132] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings of the 18th International Conference on Data Engineering*, pages 685–694, 2002.

[133] A. Pagh, R. Pagh, and S. S. Rao. An optimal bloom filter replacement. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 823–829, Vancouver, BC, 2005.

[134] S. Pan and X. Zhu. Cgstream: Continuous correlated graph query for data streams. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1183–1192, Maui, Hawaii, USA, 2012.

[135] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 697–708, Trondheim, Norway, 2005.

[136] A. S. Pentland. The data-driven society. *Scientific American*, 309:78–83, 2013.

[137] K. Pripužić, I. P. Žarko, and K. Aberer. Time- and space-efficient sliding window top-k query processing. *ACM Trans. Database Syst.*, 40(1):1:1–1:44, Mar. 2015.

[138] F. Putze, P. Sanders, and J. Singler. Cache-, hash-, and space-efficient bloom filters. *Journal of Experimental Algorithmics*, 14:4:4.4–4:4.18, Jan. 2010.

[139] C. Raïssi and M. Plantevit. Mining multidimensional sequential patterns over data streams. In I.-Y. Song, J. Eder, and T. Nguyen, editors, *Data Warehousing and Knowledge Discovery*, volume 5182 of *Lecture Notes in Computer Science*, pages 263–272. 2008.

[140] P. Rashidi and D. Cook. Mining sensor streams for discovering human activity patterns over time. In *Proceedings of the IEEE 10th International Conference on Data Mining*, pages 431–440, Dec 2010.

[141] L. Rhodes. DataSketches: Fast, Approximate Analysis of Big Data. https://yahooeng.tumblr.com/post/135390948446/data-sketches, Dec. 2015.

[142] P. P. Rodrigues and J. Gama. Online prediction of streaming sensor data. In *Proceedings of the 3rd International Wokshop on Knowledge Discovery from Data Streams*, 2006.

[143] C. E. Rothenberg, C. A. B. Macapuna, F. L. Verdi, and M. F. Magalhães. The deletable bloom filter: A new member of the bloom family. *IEEE Communications Letters*, 14(6):557–559, June 2010.

[144] O. Rottenstreich, Y. Kanizo, and I. Keslassy. The variable-increment counting bloom filter. *IEEE/ACM Transactions on Networking*, 22(4):1092–1105, Aug. 2014.

[145] D. Sahpaski, A. S. Dimovski, G. Velinov, and M. Kon-Popovska. Efficient processing of Top-k Join queries by attribute domain refinement. In *Proceedings of the 16th East European Conference on Advances in Databases and Information Systems*, pages 318–331, Pozna&#324;, Poland, 2012.

[146] M. Sayal. Detecting time correlations in time–series data streams. Technical Report HPL-2004-103, 2004.

[147] N. D. Shah and J. Pathak. Why Health Care May Finally Be Ready for Big Data. https://hbr.org/2014/12/why-health-care-may-finally-be-ready-for-big-data.

[148] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, pages 239–249, Baltimore, MD, USA, 2004.

[149] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. d. Carvalho, and J. a. Gama. Data stream clustering: A survey. *ACM Computing Surveys*, 46(1):13:1–13:31, July 2013.

[150] L. Su, W. Han, S. Yang, P. Zou, and Y. Jia. Continuous adaptive outlier detection on distributed data streams. In *Proceedings of the Third International Conference on High Performance Computing and Communications*, pages 74–85, 2007.

[151] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, pages 187–198, Seoul, Korea, 2006.

[152] X. Sun and D. P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 336–345, New Orleans, Louisiana, 2007.

[153] S. C. Tan, K. M. Ting, and T. F. Liu. Fast anomaly detection for streaming data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1511–1516, Barcelona, Spain, 2011.

[154] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee. Efficient single-pass frequent pattern mining using a prefix-tree. *Information Sciences*, 179(5):559 – 583, 2009.

[155] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee. Sliding window-based frequent pattern mining over data streams. *Information Sciences*, 179(22):3843 – 3865, 2009.

[156] Y. Tao, X. Lian, D. Papadias, and M. Hadjieleftheriou. Random sampling for continuous streams with arbitrary updates. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):96–110, Jan. 2007.

[157] D. Ting. Streamed approximate counting of distinct elements: Beating optimal batch methods. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 442–451, 2014.

[158] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy. Storm @ twitter. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 147–156, Snowbird, UT, 2014. https://storm.apache.org/.

[159] M. Toyoda, Y. Sakurai, and Y. Ishikawa. Pattern discovery in data streams under the time warping distance. *The VLDB Journal*, 22(3):295–318, June 2013.

[160] N. N. Vijayakumar and B. Plale. Prediction of missing events in sensor data streams using kalman filters. In *Proceedings of the First International Workshop on Knowledge Discovery from Sensor Data*, Aug. 2007.

[161] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):37–57, Mar. 1985.

[162] E. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

[163] M. Wang and X. Wang. Efficient evaluation of composite correlations for streaming time series. In G. Dong, C. Tang, and W. Wang, editors, *Advances in Web-Age Information Management*, pages 369–380. 2003.

[164] Y.-l. Wang, H.-b. Xu, Y.-s. Dong, X.-j. Liu, and J.-b. Qian. Apforecast: An adaptive forecasting method for data streams. In R. Khosla, R. Howlett, and L. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3682 of *Lecture Notes in Computer Science*, pages 957–963. 2005.

[165] Q. Xie, S. Shang, B. Yuan, C. Pang, and X. Zhang. Local correlation detection with linearity enhancement in streaming data. In *Proceedings of the 22Nd ACM International Conference on Conference on Information &#38; Knowledge Management*, pages 309–318, 2013.

[166] D. Yang, A. Shastri, E. A. Rundensteiner, and M. O. Ward. An optimal strategy for monitoring top-k queries in streaming windows. In *Proceedings of the 14th International Conference on Extending Database Technology*, pages 57–68, Uppsala, Sweden, 2011.

[167] F. Yang, Z. Qian, X. Chen, I. Beschastnikh, L. Zhuang, L. Zhou, and G. Shen. Sonora: A platform for continuous mobile-cloud computing. Technical Report MSR-TR-2012-34, Microsoft Research Asia, 2012.

[168] K. Yu, W. Ding, D. A. Simovici, and X. Wu. Mining emerging patterns by streaming feature selection. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 60–68, Beijing, China, 2012.

[169] L. Zhang, Z. Li, M. Yu, Y. Wang, and Y. Jiang. Random sampling algorithms for sliding windows over data streams. In *Proceedings of the 11th Joint International Computer Conference*, pages 572–575, 2005.

[170] Q. Zhang and W. Wang. An efficient algorithm for approximate biased quantile computation in data streams. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, pages 1023–1026, Lisbon, Portugal, 2007.

[171] Y. Zhang and L. Liu. Distance-aware bloom filters: Enabling collaborative search for efficient resource discovery. *Journal on Future Generation Computer Systems*, 29(6):1621–1630, Aug. 2013.