UDDSketch: Accurate Tracking of Quantiles in Data Streams

ITALO EPICOCO®, CATIUSCIA MELLE, MASSIMO CAFARO®, (Senior Member, IEEE), MARCO PULIMENO®, AND GIUSEPPE MORLEO

Department of Engineering for Innovation, University of Salento, 73100 Lecce, Italy

Corresponding author: Massimo Cafaro (massimo.cafaro@unisalento.it)

ABSTRACT We present UDDSketch (Uniform DDSketch), a novel sketch for fast and accurate tracking of quantiles in data streams. This sketch is heavily inspired by the recently introduced DDSketch, and is based on a novel bucket collapsing procedure that allows overcoming the intrinsic limits of the corresponding DDSketch procedures. Indeed, the DDSketch bucket collapsing procedure does not allow the derivation of formal guarantees on the accuracy of quantile estimation for data which does not follow a sub-exponential distribution. On the contrary, UDDSketch is designed so that accuracy guarantees can be given over the full range of quantiles and for arbitrary distribution in input. Moreover, our algorithm fully exploits the budgeted memory adaptively in order to guarantee the best possible accuracy over the full range of quantiles. Extensive experimental results on both synthetic and real datasets confirm the validity of our approach.

INDEX TERMS Sketches, quantiles, streaming algorithms.

I. INTRODUCTION

A data stream $\sigma$ can be thought as a sequence of $n$ items drawn from a universe $\mathcal{U}$. In particular, the items need not be distinct, so that an item may appear multiple times in the stream. Data streams are ubiquitous, and, depending on the specific context, items may be IP addresses, graph edges, points, geographical coordinates, numbers etc.

Since the items in the input data stream come at a very high rate, and the stream may be of potentially infinite length (in which case $n$ refers to the number of items seen so far), it is hard for an algorithm in charge of processing its items to compute an expensive function of a large piece of the input. Finally, long term archival of the stream is usually unfeasible. A detailed presentation of data streams and streaming algorithms, discussing the underlying reasons motivating the research in this area is available to the interested reader in [1].

In this article we are concerned with the problem of accurately tracking quantiles in data streams. The difficulty is strictly related to the underlying nature of the input data stream, since it is a well-known fact that computing exact quantiles is impossible without storing all of the data [2]. Therefore, approximate solutions such as those provided by sketches are the only viable possibility.

Quantiles can be used to characterize the distributions of real datasets much better than simpler alternatives, i.e., by using the mean and the variance. Therefore quantiles are important from a practical perspective to both database implementers and users: as an example, they are necessary for query optimization, to split the data in parallel database systems, and are fundamental for statistical data analysis.

Among the possible applications, we recall here Internet-scale network monitoring. For instance, given a client-server application, the overall performance of the service (web site, database etc) is characterized by the latency encountered by the users’ requests. Since the distribution of the latency values is usually skewed, tracking some specific quantiles (e.g., the 95th percentile) is commonly done.

The Gigascope streaming database [3] relies on quantiles for monitoring network applications and systems. Moreover, as already discussed, quantiles are extensively used in database query optimizers in order to estimate how large intermediate results are; the estimates are then used to determine the overall best execution plan [4].

Formally, given a multi-set $S$ of size $n$ over $\mathbb{R}$, let $R(x)$ be the rank of the element $x$, i.e., the number of elements in
S smaller than or equal to x. Then, the lower (respectively upper) q-quantile item \( x_q \in S \) is the item x whose rank \( R(x) \) in the sorted multi-set \( S \) is \( \lfloor q(n - 1) \rfloor \) (respectively \( \lceil 1 + q(n - 1) \rceil \)) for \( 0 \leq q \leq 1 \).

Frequency distributions and histograms are concepts closely related to quantiles. The cumulative frequency distribution \( F(\cdot) \) summarizes the distribution of a totally ordered multi-set \( S \) consisting of \( n \) elements: given an arbitrary value \( x \), \( F(x) \) is defined as the number of values less than \( x \). The q-quantile item \( x_q \in S \) is the inverse of the function \( F(\cdot) \), i.e., the element \( x \) is such that \( Q(F(x)/n) = x \). By definition then, \( x_0 \) and \( x_1 \) respectively the minimum and maximum element of \( S \), and \( x_{0.5} \) is the median.

Regarding histograms, these allow summarizing data by using a small number of buckets. The key idea is to let the buckets retain the information related to how many elements fall between two consecutive bucket boundaries, but not their identities. A sequence of quantiles can be thought as being very similar to a histogram of the input dataset, and may provide an excellent and complete summary of the entire input distribution if the accuracy can be controlled by using a user’s defined parameter. Regarding tracking accuracy, it can be defined in two different ways, as follows.

**Definition 1 (Rank Accuracy):** \( \forall \) item \( v \) and a given tolerance \( \epsilon \), return an estimated rank \( \tilde{R} \) such that \( |R(v) - \tilde{R}(v)| \leq \epsilon n \).

**Definition 2 (Relative Accuracy):** \( \tilde{x}_q \) is an \( \alpha \)-accurate q-quantile if \( |\tilde{x}_q - x_q| \leq \alpha x_q \) for a given q-quantile item \( x_q \in S \). A sketch data structure is an \( \alpha \)-accurate \( (q_0, q_1) \)-sketch if it can output \( \alpha \)-accurate q-quantiles for \( q_0 \leq q \leq q_1 \).

Even though for long time research efforts have been focused on data structures providing rank accuracy, data sets with heavy tails are such that rank-error guarantees can return values with large relative errors. In particular, rank accuracy is not viable for tracking higher order quantiles of heavy-tailed distributions.

DDSketch (Distributed Distribution Sketch) [5] is a recent sketch data structure providing relative accuracy for tracking quantiles in data streams whose underlying distribution is heavy-tailed. This sketch is conceptually very simple and can be implemented either using an unlimited number of buckets or fixing a desired maximum number of buckets to be used. In the former case, the space used may grow unbounded, whilst in the latter case when the current number of buckets in the sketch exceeds the predefined maximum a bucket collapsing procedure must be executed in order to guarantee that the number of buckets is always bounded from above. Unfortunately, the authors of DDSketch do not provide formal guarantees on the estimation’s accuracy for a collapsed sketch when the input data is not drawn from sub-exponential distributions.

In this article we introduce and discuss a novel collapsing strategy for DDSketch. The main contributions of this article are the following ones: (i) we provide a novel collapsing procedure for the DDSketch algorithm; (ii) we formally provide an error bound, modeling the relationship between accuracy and space occupied by the sketch for arbitrary input distributions; (iii) we show that our algorithm fully exploits the budgeted memory adaptively in order to guarantee the best possible accuracy over the full range of quantiles; (iv) we extensively compare the behaviour and the accuracy of DDSketch and UDDSketch over a huge set of both synthetic and real input datasets; (v) we provide freely available C implementations of both DDSketch and UDDSketch for full reproducibility of results.

The rest of this article is organized as follows. We recall related work in Section II, and discuss DDSketch in Section III. We introduce our UDDSketch algorithm and formally prove its correctness and error bound in Section IV. Next, we thoroughly compare DDSketch and UDDSketch through an extensive set of experiments in Section V. Finally, we draw our conclusions in Section VI.

**II. RELATED WORK**

The problem of quantile computation has been extensively studied in the scientific literature; there are indeed several publications about it, with algorithms characterized by very different approaches. The common goal is to provide the most accurate result possible with the minimum use of resources.

The first works for the determination of a quantile sketch date back to the 80’s when Munro and Paterson [2] demonstrated the first quantile sketching algorithm with formal guarantees. They proved the relationship between the amount of space needed related to the number of steps required to select the highest order statistical \( k \)-th on a dataset of \( N \) elements.

Munro and Paterson designed a probabilistic algorithm to estimate the median by keeping \( s \) samples out of the \( N \). If the data are presented in random order and \( s = \Theta(N^{1/3}) \), then the algorithm has a high probability of storing samples containing the median. This algorithm can be adapted to find a specific quantile. The main result obtained is the proof that the amount of memory required by a deterministic \( p \)-pass selection algorithm is \( \Omega(N^{p/3}) \). For a data stream, where only one-pass is allowed, i.e. \( p = 1 \), the computation of the exact value of any quantile requires \( \Omega(N) \) memory space. This result led subsequent work to focus on algorithms providing approximate quantile values.

A common technique used in practice for the selectivity estimation problem, is to maintain histograms of frequency, that is buckets containing groups of values that approximate the true value and its frequency according to the statistics maintained by each bucket. Gibbons et al. [6] presented two fast and efficient procedures for maintaining two classes of histograms: equi-depth histograms and compressed histograms. In the equi-depth histogram, the elements are grouped into buckets so as to ensure the same number of elements for each of them (same height). In the compressed histogram, the \( n \) highest frequencies are stored into \( n \) separate buckets, the rest of the elements are partitioned according to the equi-depth histogram. An equi-depth histogram
approximates the exact histogram by relaxing the requirements on the number of elements in the bucket and counting accuracy. Its distance from the real histogram can be measured by the following error metric. Consider an approximate equi-depth histogram with $\beta$ buckets for $N$ elements, the error metric $\mu_{eq}$ is the standard deviation of the buckets sizes from their average, normalized with respect to the average of the buckets sizes. The variant with compressed histograms is also treated in a similar way, obviously with the necessary modifications to adapt the algorithm to that class of histograms.

These algorithms provide a summary of the data using histograms and can be used to estimate quantiles according to a different error metric, however, they need to perform multiple passes on the whole input dataset.

Manku et al. [7] designed an algorithm whose accuracy bound is independent from the input distribution and the approximation error is uniformly distributed over all quantiles. The algorithm uses $b$ buffers which store $k$ elements each. Each buffer $B$ is associated with a weight $w_B$ which represents the occurrences of the input items fallen in the buffer. When the algorithm starts, all buffers are empty and they are populated with the elements from the input dataset; when all of the buffers are full, the collapsing procedure applies, modifying the weight of the collapsed buffer accordingly. The authors proved that the error $\epsilon$ committed on the estimation of the $q$-quantile is bounded and the space required to guarantee the error bound is $O(\frac{1}{\epsilon^2} \log^2 N \epsilon N)$. The algorithm is efficient and offers opportunities for parallelism, however, it requires to know in advance the size $N$ of the dataset which makes the algorithm not suitable for data streams processing.

Summarizing large datasets is important because of limited memory resources. The GK sketch algorithm by Greenwald and Khanna [8] addressed the problem of designing a space-efficient algorithm based on quantile summaries. A summary consists of a small number of items sampled from the input sequence. These items are then used to respond to any quantile request. The algorithm provides an $\epsilon$-approximate estimate $r_q'$ of the $q$-quantile. A summary is $\epsilon$-approximate if the estimate of a $q$-quantile differs from the exact value $r_q$ by $|r_q' - r_q| \leq \epsilon N$. The algorithm requires memory space $O(\frac{1}{\epsilon^2} \log(\epsilon N))$ and it is independent by the input distribution. GK Sketch offers excellent results in terms of approximation and space used, however it is not fully mergeable, which makes it impossible to use it in a distributed setting, moreover the memory required depends on the size of the input dataset which is not known when processing data streams.

Summarizing distributions which have high skew using uniform quantiles is not informative because having a uniformly spread out summary of a stretched distribution does not describe the interesting tail region adequately. Motivated by this, Cormode et al. [9] designed an algorithm to efficiently estimate the high-biased quantiles. The high-biased quantiles are defined as: $1 - \phi, 1 - \phi^2, \cdots, 1 - \phi^k$ with $0 < \phi < 1$. The algorithm keeps information about particular items from the input, and also stores some additional tracking information. The intuition for this algorithm is as follows: suppose we have kept enough information so that the median of a dataset with $N$ elements can be estimated with an absolute error of $\epsilon N$ in rank. Now suppose that there are $N$ more insertions of items above the median, so that this item is pushed up to being the first quantile. If the same absolute uncertainty of $\epsilon N$ is maintained, then this corresponds to a relative error of size $\epsilon/2$, considering that the number of items is doubled. Inspired by the GK algorithm, Cormode et al. provided an algorithm which is able to support greater accuracy for the high-biased quantiles.

The Moments Sketch algorithm of Gan et al. [10] is based on a data structure defined as moments sketch. The sketch requires a minimal amount of space and it is mergeable and computationally efficient. The authors use the moments methods to build the $f(x)$ distribution function which can be used to describe the input dataset. Letting $k$ be the highest power used for the moments, the moments sketch of a dataset $D$ includes: the minimum value $x_{min}$; the maximum value $x_{max}$; the number of items $n$; the sample of moments $\mu_i = \frac{1}{n} \sum_{x \in D} x^i$ for $i = 1, \cdots, k$; the logarithmic moments $v_i = \frac{1}{n} \sum x \log(x)$ for $i = 1, \cdots, k$. To estimate a quantile from a Moments Sketch, the moments method is applied to build the PDF $f(x)$ whose moments match those stored in the sketch and that maximizes the entropy defined as $H[f] = - \int f(x) \log f(x) dx$. $f(x)$ is then used to estimate the quantiles of the dataset. The Moments Sketch proves to be very fast, with an average error of less than 0.01 using about 200 bytes of space. However, there could be pathological situations with certain distributions for which it is not possible to compute finite moments. Moreover, the error is guaranteed in the average case, but not in the worst case, and errors caused by floating point multiplications can occur.

The work done by Dunning and Ertl [11] has introduced a new data structure known as $t$-digest, formed by clustering real value samples. This structure differs from the previous ones in several ways: the data are grouped and summarized in the $t$-digest structure, however the range of data included in different clusters may overlap; the buckets are represented by a centroid value and a weight value that represents the number of samples contributing to the bucket, instead of the classic lower and upper limits; the samples are accumulated in such a way that only a few of them help determining extreme quantiles, so that relative error is bounded instead of maintaining constant the absolute error. The accuracy of the $q$-quantile estimate is near to $q(1 - q)$. In this algorithm the accuracy depends on the quantile and is more accurate for computing quantiles close to 0 and 1.

Given a set of elements $x_1, \cdots, x_n$, the quantile $x$ is the fraction of elements in the stream such that $x_i \leq x$, i.e., the rank of $x$ denoted by $R(x)$. A data structure is accurate for all quantiles if for each $x$, letting $\tilde{R}(x)$ be the estimated rank of $x$, with probability $1 - \delta$ it holds that $|\tilde{R}(x) - R(x)| \leq \epsilon n$. 

147606
VOLUME 8, 2020
Karlin et al. [12] designed their algorithm as a reinterpretation of the work of [13] and [14] from a different point of view. The algorithm is based on the concept of a compactor, a data structure that can store $k$ elements all with the same weight $w$, and if necessary can compact its $k$ elements into $k/2$ elements of weight $2w$ in the following way: items are sorted, then odd (respectively even) items are selected and the non-selected even (respectively odd) items are discarded, and the weight $w$ of each selected item is doubled. Each compactor will eliminate odd or even items with equal probability. The rank estimation after this process depends at most on $w$. The output elements of a compactor are put into another one and so on, and since each compactor has half of the elements in the sequence there will be at most $H \leq \lceil \log(n/k) \rceil$ compactors chained together creating a hierarchy with variable capacity. Considering an algorithm run ending with $H$ different compactors the theorem proved by the authors states that there is a streaming algorithm that calculates an $\epsilon$ approximation for the rank of a single item with probability $1 - \delta$ whose space complexity is $O((1/\epsilon)\sqrt{\log(1/\delta)})$. Moreover, there is another streaming algorithm that produces mergeable summaries and computes an $\epsilon$ approximation for the rank of a single item with probability $1 - \delta$ whose space complexity is $O((1/\epsilon)\sqrt{\log(1/\delta)}+\log(\epsilon n))$. An additional optimization guarantees the rank computation of a single element with $1 - \delta$ probability and with a space complexity of only $O((1/\epsilon)\log(\log(1/\delta)))$ for the non-mergeable version and $O((1/\epsilon)\log^2(\log(1/\delta)))$ for the mergeable version. The algorithm provides a randomized solution to the problem of computing quantiles on data streams with a probability of error of $1 - \delta$ and a minimum amount of space used, ensuring the property of full mergeability. However, the algorithm provides estimates with a greater relative error for the high quantiles on heavy-tailed data.

III. DDSKETCH

A basic version of DDSketch, described in [5], can provide $\alpha$-accurate $q$-quantiles for any $0 \leq q \leq 1$. This version of the algorithm is both simple to understand and implement, and provides support for item insertion/deletion and merging of two compatible sketches (i.e., sketches characterized by the same $\alpha$ value). The main drawback of this algorithm is that the accuracy is obtained by trading off the space required: the number of buckets in a sketch can grow without bound. Owing to this limitation, the authors of DDSketch introduced in [5] an advanced version of DDSketch that can deliver $\alpha$-accurate $q$-quantiles for $q_0 \leq q \leq 1$ with a bounded number of buckets. In this manuscript we will only deal with the second improved version of DDSketch.

DDSketch works by dividing $\mathbb{R}_{>0}$ into indexed buckets. Let $B_i$ be the bucket with index $i$ and $m$ the maximum number of buckets. It is worth noting here that $m$ is a small constant, typically in the range [500, 2000]. The algorithm works reactively, by invoking a collapsing procedure if inserting a value causes the number of buckets to grow beyond $m$.

Denoting by $\gamma$ the quantity $\frac{1+\alpha}{1-\alpha}$ where $\alpha$ represents the user’s defined accuracy, the bucket $B_i$ is a counter holding the occurrences of values $x$ falling between the interval given by $y^{i-1} < x \leq y^i$. Algorithm 1 shows the pseudo-code related to the insertion procedure of an item $x$. We assume that the number $b$ of buckets stored in the sketch at any time is $0 \leq b \leq m$, i.e., the number of buckets maintained is dynamic, depending on the sequence of insertions and deletions operations. Of course, the bucket indexes are dynamic as well. A bucket always holds a positive count. This is certainly true for insertion-only streams. However, DDSketch also allows deletions, in which case a bucket count may be zero. When this happens, a bucket is discarded and thrown away.

As shown in Algorithm 1, to insert a value $x$ the index $i$ of the bucket in which $x$ falls is computed as $i = \lceil \log_{y^i} x \rceil$. If the bucket $B_i$ has been already inserted into the sketch, then the bucket’s counter is incremented by one. Otherwise, $B_i$ is added to the sketch with a count initialized to one. Then, if the number of buckets exceeds $m$ after inserting $x$, a bucket collapsing procedure is executed, by collapsing the initial two buckets.

Note that, in general, the first two buckets are not $B_1$ and $B_2$, since the indexes of these buckets depend on the actual insertions done. Therefore, we denote in the pseudo-code these buckets as $B_0$ and $B_1$. In particular, it holds that $y < z$ but it is not necessarily true that $z = y + 1$, i.e. the indexes need not be consecutive. The buckets $B_i$ and $B_{i+1}$ are updated so that the count stored by $B_i$ is added to $B_{i+1}$, and $B_i$ is removed from the sketch. Alternatively, the collapsing procedure can be applied to the last two buckets.

The computational complexity of Algorithm 1 in the worst case is $O(1)$, i.e. constant time. Indeed, inserting a value simply requires incrementing a bucket’s counter if the bucket already exists or creating a new bucket with value equal to one. Then, if the size of the sketch exceeds $m$, the first two buckets are collapsed, and again this operation requires $O(1)$ time in the worst case.

\begin{algorithm}
\caption{DDSketch-Insert($x$, $S$)}
\textbf{Require:} $x \in \mathbb{R}_{>0}$
\begin{algorithmic}
\State $i = \lceil \log_{y^i} x \rceil$
\If{$B_i \in S$} \Comment{Bucket already exists}
\State $B_i \leftarrow B_i + 1$
\Else \Comment{Bucket does not exist, create it}
\State $B_i \leftarrow 1$
\State $S \leftarrow S \cup B_i$
\EndIf
\If{$|S| > m$} \Comment{Number of buckets exceeds maximum}
\State let $B_y$ and $B_{y+1}$ be the first two buckets
\State $B_y \leftarrow B_y + B_{y+1}$
\State $S \leftarrow S \setminus B_y$
\EndIf
\end{algorithmic}
\end{algorithm}

The authors of DDSketch show that $m$ buckets suffice to $\alpha$-accurately answer a given $q$-quantile query if: $x_1 \leq x_q y^{m-1}$,
Theorem 1: Let $X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$ be the order statistics for i.i.d. random variables $X_i$ distributed according to a sub-exponential distribution $F$ with parameters $(\sigma, b)$. Then with probability at least $1 - \delta_1 - \delta_2$, DDSketch is an $\alpha$-accurate $(q, 1)$-sketch with size at most $(\log X_{(n)} - \log X_{(\alpha n)})/\log(\gamma) + 1$, which is bounded from above by:

$$\frac{\log(2b \log(n/\delta_2) + EX) - \log(F^{-1}(q - 1))}{\log(\gamma)} + 1 \quad (2)$$

for:

$$\gamma = \frac{(1 + \alpha)/(1 - \alpha)}{t^2} = \sqrt{\log(1/\delta_1)/2n}, \text{ and } t < q < 1/2.$$  

DDSketch, as described by its authors, only deals with i.i.d. random variables $X_i$ distributed according to a sub-exponential distribution $F$ with parameters $(\alpha, \gamma)$. Moreover, an item $x$ falling in bucket with index $i$ of a collapsing sketch, will fall in bucket with index $\lceil i/2 \rceil$ of the collapsed sketch.

Proof: Let $B_i$ and $B_{i+1}$ be two adjacent buckets of the sketch to be collapsed. The collapsing procedure sums them up and replaces them with a new bucket, which we denote by $B'_i$. Let $U_i$, $U_{i+1}$ and $U'_i = U_i \cup U_{i+1}$ denote the intervals of values which refer respectively to buckets $B_i$, $B_{i+1}$ and $B'_i$. Let $\gamma = \frac{1 + \alpha}{1 - \alpha}$ and $\gamma' = \frac{\gamma^{i+1} - \gamma^i}{\gamma^i - \gamma^{i+1}}$.

We have that:

$$U_i = (\gamma^i, \gamma'^i], U_{i+1} = (\gamma'^i, \gamma^{i+1}],$$

$$U'_i = U_i \cup U_{i+1} = (\gamma^i, \gamma^{i+1}],$$

from which we derive that:

$$\gamma' = \gamma'^{i+1} = \frac{\gamma^{i+1} - \gamma^i}{\gamma^i - \gamma^{i+1}} = \gamma^2,$$  

and, as a consequence of the relation between $\alpha'$ and $\gamma'$, and $\alpha$ and $\gamma$:

$$\alpha' = \frac{\gamma' - 1}{\gamma' + 1} = \frac{\gamma^2 - 1}{\gamma^2 + 1} = \frac{2\alpha}{1 + \alpha^2}. \quad (4)$$

Furthermore, we have that, if $B_i$ and $B'_i$ are the buckets in which a value $x$ falls, respectively, before and after the collapse, then it holds that:

$$j = \lceil \log_{\gamma'} x \rceil = \lceil \log_{\gamma^2} x \rceil = \left\lceil \frac{\log x}{\log r} \right\rceil = \left\lceil \frac{i}{2} \right\rceil \quad (5)$$

that proves the relation between the bucket keys of the collapsing sketch and those ones of the collapsed sketch. \[ \square \]

After collapsing the buckets, $\alpha'$ represents the new theoretical error bound for the sketch. Each time we perform a collapse, $\alpha$ increases, i.e. we lose accuracy. However, we do not expect executing the collapsing procedure repeatedly up to the point where the loss in accuracy adversely impacts on the data structure precluding its use. The reason is that each

Algorithm 2 UniformCollapse($S$)

**Require:** sketch $S = \{B_i\}$

**for each** $\{i : B_i > 0\}$ **do**

$$j \leftarrow \lceil \frac{i}{2} \rceil$$

$$B'_i \leftarrow B'_i + B_i$$

**end for**

**return** $S \leftarrow \{B'_i\}$

constant time. However, $m$ is a constant which impacts on the algorithm’s accuracy and it does not depend on the problem size $n$; the overall computational complexity in the worst case therefore is $O(1)$.

The following lemma formally shows and justifies how uniform collapse modifies the sketch and its accuracy.

Lemma 2: The collapsing procedure applied to an $\alpha$-accurate $(0, 1)$-quantile sketch produces an $\alpha'$-accurate $(0, 1)$-quantile sketch on the same input data with $\alpha' = \frac{2\alpha}{1 + \alpha^2}$. Moreover, an item $x$ falling in bucket with index $i$ of a collapsing sketch, will fall in bucket with index $\lceil i/2 \rceil$ of the collapsed sketch.
time a collapsing is done, the input interval covered by the \( m \) available buckets increases as well, so that a few collapsing are enough to process input data streams with very large range of values.

According to the collapsing algorithm, we can formulate the following theorem which provides an upper bound on the accuracy of the results, i.e., on the error committed approximating the quantile computations when a limited number of buckets are at disposal.

**Theorem 3:** Given an input whose data domain is an interval \([x_{\text{min}}, x_{\text{max}}]\) \( \in \mathbb{R}_{>0} \) and an UDDSketch data structure using at most \( m \) buckets to process the input, the approximation error committed by UDDSketch using the uniform collapse procedure is bounded by

\[
\tilde{\gamma} = \frac{\tilde{\gamma}^{k+1} - \tilde{\gamma}^{k}}{\tilde{\gamma}^{k+1} + \tilde{\gamma}^{k}},
\]

with

\[
\tilde{\gamma} = \sqrt[m]{\frac{x_{\text{max}}}{x_{\text{min}}}}.
\]

**Proof:** In order to provide an upper bound on the accuracy achieved by the UDDSketch data structure, we analyze the worst case, i.e., the situation in which the \( m \) buckets must uniformly cover the interval \([x_{\text{min}}, x_{\text{max}}]\). In such a case, the corresponding indexes are consecutive numbers denoted by \( i_1, i_2, \ldots, i_m \). Let the covered interval be \((\tilde{\gamma}^{i_1-1}, \tilde{\gamma}^{i_m}]\). Choosing \( i = \lceil \log_\tilde{\gamma} x_{\text{min}} \rceil \), it holds that \( x_{\text{min}} \) falls into the first bucket \( B_{i_1} \). Therefore, it holds that

\[
\tilde{\gamma}^{i_1-1} < x_{\text{min}} \leq \tilde{\gamma}^{i_1}.
\]  

We now show that \( x_{\text{max}} \) falls into the last bucket \( B_{i_m} \), i.e.,

\[
\tilde{\gamma}^{i_m} \geq x_{\text{max}}.
\]  

It holds that \( \tilde{\gamma}^{i_m} = \tilde{\gamma}^{i_1} \tilde{\gamma}^{i_m} \) since the buckets indexes are consecutive and the buckets uniformly cover the whole interval. As a consequence, taking into account the definition of \( \tilde{\gamma} \), equation (7) is equivalent to

\[
\tilde{\gamma}^{i_1} \frac{x_{\text{max}}}{x_{\text{min}}} \geq x_{\text{max}}.
\]  

Equation (8) holds, since \( \tilde{\gamma}^{i_1} \frac{x_{\text{max}}}{x_{\text{min}}} \geq 1 \), owing to equation (6).

Now consider an initial \( \alpha \) value and a corresponding initial \( \gamma \) such that an integer number of collapses which brings \( \gamma \) to \( \tilde{\gamma} \) does not exist, but it holds that \( \gamma^{2k} < \tilde{\gamma} < \gamma^{2k+1} \), for a \( k \in \mathbb{N} \). In this case, we may need a number \( k+1 \) of collapses to accommodate all of the input values and end up with a final value of \( \gamma > \tilde{\gamma} \). However, even in this eventuality, the value of \( \gamma \) can not grow beyond \( \tilde{\gamma}^2 \) and this is the reason why the upper bound of the accuracy is set to \( \frac{\tilde{\gamma}^2 - 1}{\tilde{\gamma}^2 + 1} \).

In Theorem 3, we assume that the values \( x_{\text{min}} \) and \( x_{\text{max}} \) of the input data are known. This is not always true, but we can always estimate these values with a probability \( \delta \) of failing our prediction. In that case, the bound showed by Theorem 3 holds with probability \( 1 - \delta \).

We now discuss how to, given a user desired level \( \alpha \) of accuracy and a number of buckets sufficient to satisfy that accuracy based on Theorem 3, choose the initial value of \( \alpha_0 \) to start our algorithm. By construction, the sequence of \( \alpha_k \) values corresponding to the \( \gamma_k \) values changed upon a collapsing procedure follows the recurrence equation:

\[
\alpha_k = \begin{cases} 
\alpha_0 & \text{for } k = 0 \\
\frac{2\alpha_{k-1}}{1 + \alpha_{k-1}^2} & \text{for } k > 0.
\end{cases}
\]  

where \( k \) denotes the number of collapses performed.
The solution to Eq. 9 is \( \alpha_k = \tanh (2^k - 1 \arctanh(\alpha_0)) \). Similarly, the equation

\[
\alpha_0 = \tanh \left( \frac{\arctanh(\alpha_k)}{2^k - 1} \right),
\]

allows to compute \( \alpha_0 \) given a final accuracy \( \alpha_k \) corresponding to \( k \) collapses.

We can use Eq. 10 to compute the initial value of the accuracy parameter for our algorithm by setting \( \alpha_k \) to the value of user desired accuracy and \( k \) to the number of collapses that we are willing to accept. There is a trade-off to take into consideration: if we go backward too far, i.e., we set \( k \) too large, we could end up with too many collapses and a decrease of performance, but with a favourable input
distribution, we can obtain a better accuracy. On the contrary, if we compute the initial accuracy with a few collapses or no collapses at all ($\alpha_0$ and user $\alpha$ coincide), we improve the performance and we may require less space, but we can not do better in terms of accuracy than guaranteeing the desired alpha. We have seen that a good empirical value for $\kappa$ is 10.

V. EXPERIMENTAL RESULTS

In this Section, we present and discuss the experimental investigation carried out in order to compare UDDSketch against DDSketch.

Both UDDSketch and DDSketch algorithms have been implemented in C and compiled using the GCC compiler.
FIGURE 4. Relative errors on quantiles $q_0, q_{0.1}, q_{0.2} \ldots q_1$, for the power dataset varying the collapsing strategy.

FIGURE 5. Relative errors on quantiles (boxplots), varying the number of buckets.
v4.8.5 on linux CentOS 7 with optimization level O3. The tests have been executed on a workstation equipped with 64 GB of RAM and two 2.0 GHz exa-core Intel Xeon CPU E5-2620 with 15 MB of cache level 3. The source code is freely available for inspection and reproducibility of results.\footnote{https://github.com/cafaro/UDDSketch}

The tests have been performed on 16 various synthetic datasets, including the pareto distribution which was also used by the authors of DDSketch [5] and a dataset drawn from a Uniform distribution with a time varying range that increases each $2 \times 10^7$ items. The properties of the synthetic datasets are summarized in Table 1. Each dataset consists of 10 million real values except for the unifTv which includes 100 million real values. Figure 1 shows the statistical

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.png}
\caption{Relative errors on quantiles (boxplots), varying the number of buckets.}
\end{figure}
distributions from which the datasets are drawn. In addition, the same tests have been performed on a real dataset, named *power*, which is one of the two real datasets used in [5].

DDSketch is executed using in each experiment all of the possible collapsing strategies: collapses of buckets with higher IDs (DDSketch H), collapses of buckets with lower IDs (DDSketch L) and a third variant (DDSketch D) where the available buckets have been equally partitioned between two sketches, one DDSketch H and one DDSketch L. In that case, each quantile is estimated through the most accurate sketch, i.e. the sketch whose estimation comes from a non collapsed bucket. If both answers come from the collapsed bucket, the estimation from the sketch with less overall collapses is chosen.

The three variants of DDSketch and UDDSketch have been executed on each dataset in Table 1 and on the *power* dataset.
varying the value of $\alpha$ and the maximum number of buckets available to the algorithm. The sets of values used are shown in Table 2; however, we report here only the results obtained with 1024 buckets and with $\alpha$ set to 0.001 since in the other cases the behaviour is very similar to the results reported here.

In each run, the throughput, i.e., the number of values processed in a unit of time (updates per millisecond), and the accuracy, i.e., the relative errors committed on estimation of quantiles $q_0$, $q_1$, $q_2$, ..., $q_1$, are measured for all of the collapsing strategies under investigation.
Figures 2, 3 and 4 show the estimation errors committed by DDSketch L, DDSketch H and DDSketch D, compared with UDDSketch. Figure 2 refers to the beta, chisquare and exponential datasets, whilst Figure 3 is relative to the normal, pareto and uniform datasets. Figure 4 refers to the real dataset power. The number of buckets is set to 1024 and $\alpha$ is set to 0.001. The results obtained when processing the other datasets in Table 1 are not reported here, since they exhibit similar behaviours.

The plots show a major robustness of UDDSketch with reference to the distribution of the values in input. Even when the number of buckets granted to the algorithm is not enough to reach the desired $\alpha$ (dotted line), nonetheless UDDSketch guarantees an overall better accuracy, regardless of the input distribution. Even when we are only interested to specific quantiles, DDSketch does not always succeed in guaranteeing a bounded relative error, as UDDSketch does, independently of the chosen collapsing strategy. Furthermore, it is not possible for DDSketch to choose the best collapsing strategy a priori, when the input distribution is unknown. Particularly critical are the quantiles around the median, which rarely DDSketch can report with sufficient accuracy.

Figures 5 and 6 show how the median and interquartile range of the relative errors on quantiles change varying the number of buckets, when $\alpha$ is fixed to 0.001. As in Figures 2, 3 and 4, the plots in each column refer to the same collapsing strategy, and the plots in each row are relative to the same dataset. The datasets examined are the same as in the previous figures.

The observations made by inspecting Figures 2, 3 and 4 are confirmed by Figures 5 and 6. UDDSketch returns quantile estimations that are overall more accurate than DDSketch, also in terms of lower medians and shorter interquartile ranges of relative errors. Moreover, the experiments show that our solution puts to better use the number of buckets at disposal: in fact, UDDSketch keeps improving the estimate when the space granted grows, whilst DDSketch stops when the required $\alpha$ is reached and makes no use of the extra buckets at disposal.

### Table 1. Synthetic datasets.

| Dataset       | Min value | Max value | Distribution |
|---------------|-----------|-----------|--------------|
| betaL         | 3.04 x 10^{-4} | 0.99 | Beta(5, 1.5) |
| betaR         | 8.34 x 10^{-7} | 0.97 | Beta(1.5, 5) |
| chisquare     | 5.67 x 10^{-3} | 42.9 | $\chi^2(5)$  |
| exponential   | 1.19 x 10^{-7} | 34.9 | Exp(0.5)     |
| extremevalue  | 14.5      | 54.2     | Extravalue(20, 2) |
| gamma         | 2.99 x 10^{-3} | 81.8 | $\Gamma(2, 4)$ |
| gumbel        | 31.0      | 1.11 x 10^{2} | Gumbel(100, 4) |
| halfnormal    | 4.01 x 10^{-7} | 13.7 | Halfnormal(0.5) |
| inversegaussian | 0.17   | 5.12 x 10^{2} | IG(10, 5)   |
| laplace       | 26.4      | 3.67 x 10^{3} | Laplace(200, 10) |
| logistic      | 32.8      | 3.68 x 10^{3} | Logistic(200, 10) |
| lognormal     | 1.08 x 10^{-3} | 7.91 x 10^{3} | Lognormal(1, 1.5) |
| normal        | 39.7      | 60.5     | N(50, 2)     |
| pareto        | 1.0       | 3.04 x 10^{7} | Pareto(1, 1) |
| uniform       | 2.18 x 10^{-3} | 2.49 x 10^{4} | Unif(0, 2.5 x 10^{4}) |
| univ time varying | 1.00   | 10^{25} | Unif(1, 10^{6}...10^{25}) |

### Table 2. Synthetic datasets.

| Parameter          | Set of values |
|--------------------|---------------|
| User alpha         | {0.00001, 0.0001, 0.001, 0.01, 0.1} |
| Number of buckets  | {128, 256, 512, 1024, 2048} |

Figure 7 shows how UDDSketch performs versus DDSketch L when the range of the uniform distribution changes over time. As expected, the relative errors committed by UDDSketch and DDSketch L tend to increase, but overall UDDSketch is the most robust and accurate also in this case. It is worth noting here that the relative error of DDSketch L is many order of magnitude higher than that of UDDSketch in almost all of the cases. In this test we compared UDDSketch versus DDSketch L; of course, testing versus DDSketch H shall produce symmetrical results, with DDSketch H being consistently inaccurate except for the initial quantiles; this is the reason we do not report the corresponding results here.

At last, Figure 8 shows the performance of the different DDSketch collapsing strategies compared with UDDSketch, when varying the number of buckets and with reference to the datasets beta, exponential, uniform. The other datasets lead to similar behaviours and are not reported.

The performance of the algorithms under test are in general comparable. UDDSketch is better when the number of buckets is low, DDSketch L and DDSketch H are more performant when the space grows, for they tends to not make use of the extra space. DDSketch D is always less performant due to the use of two sketches that must be updated at the same time.

### VI. CONCLUSION

We have introduced UDDSketch (Uniform DDSketch), a novel sketch for fast and accurate tracking of quantiles in data streams. Our sketch was heavily inspired by the recently introduced DDSketch, and is based on a novel bucket collapsing procedure that allows overcoming the intrinsic limits of the corresponding DDSketch procedures. UDDSketch has been designed so that accuracy guarantees can be given over the full range of quantiles and for arbitrary distribution in input. Moreover, our algorithm fully exploits the budgeted memory adaptively in order to guarantee the best possible accuracy over the full range of quantiles. Extensive experimental results on both synthetic and real datasets have confirmed the validity of our approach.

### REFERENCES

[1] S. Muthukrishnan, “Data streams: Algorithms and applications,” *Found. Trends Theor. Comput. Sci.*, vol. 1, no. 2, pp. 117–236, 2005, doi: 10.1561/0400000002.

[2] J. I. Munro and M. S. Paterson, “Selection and sorting with limited storage,” *Theor. Comput. Sci.*, vol. 12, no. 3, pp. 315–323, Nov. 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397580900064

[3] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk, “Gigascope: A stream database for network applications,” in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA: Association for Computing Machinery, 2003, pp. 647–651, doi: 10.1145/872757.872838.
[4] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, “Approximate medians and other quantiles in one pass and with limited memory,” in Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD). New York, NY, USA: Association for Computing Machinery, 1998, pp. 426–435, doi: 10.1145/276304.276342.

[5] C. Masson, J. E. Rim, and H. K. Lee, “DDSketch: A fast and fully-mergeable quantile sketch with relative-error guarantees,” Proc. VLDB Endowment, vol. 12, no. 12, pp. 2195–2205, Aug. 2019, doi: 10.14778/3352063.3352135.

[6] P. B. Gibbons, Y. Matias, and V. Poosala, “Fast incremental maintenance of approximate histograms,” ACM Trans. Database Syst., vol. 27, no. 3, pp. 261–298, Sep. 2002, doi: 10.1145/581751.581753.

[7] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, “Approximate medians and other quantiles in one pass and with limited memory,” ACM SIGMOD Rec., vol. 27, no. 2, pp. 426–435, Jun. 1998, doi: 10.1145/276305.276342.

[8] M. Greenwald and S. Khanna, “Space-efficient online computation of quantile summaries,” in Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD), 2001, pp. 58–66.

[9] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava, “Effective computation of biased quantiles over data streams,” in Proc. 21st Int. Conf. Data Eng. (ICDE). Washington, DC, USA: IEEE Computer Society, 2005, pp. 20–31, doi: 10.1109/ICDE.2005.55.

[10] E. Gan, J. Ding, K. S. Tai, V. Sharan, and P. Bailis, “Moment-based quantile sketches for efficient high cardinality aggregation queries,” Proc. VLDB Endowment, vol. 11, no. 11, pp. 1647–1660, Jul. 2018, doi: 10.14778/3236187.3236212.

[11] T. Dunning and O. Ertl, “Computing extremely accurate quantiles using t-digests,” 2019, arXiv:1902.04023. [Online]. Available: https://arxiv.org/abs/1902.04023

[12] Z. Karnin, K. Lang, and E. Liberty, “Optimal quantile approximation in streams,” in Proc. IEEE 57th Annu. Symp. Found. Comput. Sci. (FOCS), Oct. 2016, pp. 71–78.

[13] P. K. Agarwal, G. Cormode, Z. Huang, J. Phillips, Z. Wei, and K. Yi, “Mergeable summaries,” in Proc. 31st Symp. Princ. Database Syst. (PODS). New York, NY, USA: ACM, 2012, pp. 23–34, doi: 10.1145/2213556.2213562.

[14] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, “Random sampling techniques for space efficient online computation of order statistics of large datasets,” ACM SIGMOD Rec., vol. 28, no. 2, pp. 251–262, Jun. 1999, doi: 10.1145/304181.304204.

**ITALO EPICOCO** received the Ph.D. degree in computational engineering from the University of Lecce, Italy. He is currently an Assistant Professor with the University of Salento. He is also an Affiliate Researcher with the Euro-Mediterranean Center on Climate Change, CMCC. His research interests include high performance, distributed, grid, and cloud computing with particular emphasis on parallel data mining. During his past research activities, he addressed issues related to the optimization of numerical methods for solving PDEs applied to Earth System Models and to fluid dynamics models on high-end parallel architectures including heterogeneous architectures made of accelerators (NVIDIA GPU and Intel MIC). Relevant activities also included optimized management of a huge amount of data produced by the climate models. He published more than 40 papers in refereed books, journals and conference proceedings.

**CATIUSCIA MELLE** received the M.Sc. degree (magna cum laude) in computer engineering from the University of Salento, Italy, where she is currently pursuing the Ph.D. degree in data mining. Her past research interests include networking and content distribution applications.

**MASSIMO CAFARO** (Senior Member, IEEE) received the Laurea (M.Sc.) degree in computer science from the University of Salerno and the Ph.D. degree in computer science from the University of Bari. He is currently an Associate Professor with the Department of Innovation Engineering, University of Salento. His researches include parallel and distributed computing, cloud and grid computing, data mining, big data, and security. He is the author of more than 100 refereed articles on parallel, distributed, cloud/grid computing and data mining. He holds a patent on distributed database technologies. He focuses his research on high performance and distributed computing on both theoretical and practical aspects, in particular the design and analysis of sequential, parallel, and distributed algorithms. He is a Senior Member of the IEEE Computer Society, a Senior Member of the ACM, the Vice Chair of Regional Centers, and a Coordinator of the Technical Area on Data Intensive Computing for the IEEE Technical Committee on Scalable Computing. He serves as an Associate Editor for IEEE Access and as a moderator for the IEEE TechRxiv preprint repository (Computing and Processing category).

**MARCO PULIMENO** received the Ph.D. degree in mathematics and computer science from the University of Salento, Italy. He is currently a Postdoctoral Researcher with the University of Salento. His research interests include high performance computing and distributed computing, in particular parallel data mining. He has published on the topic of frequent items in several refereed journals and conference proceedings.

**GIUSEPPE MORLEO** received the M.Sc. degree in computer engineering from the University of Salento, Italy. His research interests include data mining and analysis of algorithms.