Determining Exact Quantiles with Randomized Summaries

ZILING CHEN, Tsinghua University, China
HAOQUAN GUAN, Tsinghua University, China
SHAOXU SONG, Tsinghua University, China
XIANGDONG HUANG, Tsinghua University, China
CHEN WANG, Tsinghua University, China
JIANMIN WANG, Tsinghua University, China

Quantiles are fundamental statistics in various data science tasks, but costly to compute, e.g., by loading the entire data in memory for ranking. With limited memory space, prevalent in end devices or databases with heavy loads, it needs to scan the data in multiple passes. The idea is to gradually shrink the range of the queried quantile till it is small enough to fit in memory for ranking the result. Existing methods use deterministic sketches to determine the exact range of quantile, known as deterministic filter, which could be inefficient in range shrinking. In this study, we propose to shrink the ranges more aggressively, using randomized summaries such as KLL sketch. That is, with a high probability the quantile lies in a smaller range, namely probabilistic filter, determined by the randomized sketch. Specifically, we estimate the expected passes for determining the exact quantiles with probabilistic filters, and select a proper probability that can minimize the expected passes. Analyses show that our exact quantile determination method can terminate in P passes with $1-\delta$ confidence, storing $O(N^{1/P}\log^{\frac{P-1}{2P}}(\frac{1}{\delta}))$ items, close to the lower bound $\Omega(N^{1/P})$ for a fixed δ. The approach has been deployed as a function in an LSM-tree based time-series database Apache IoTDB. Remarkably, the randomized sketches can be pre-computed for the immutable SSTables in LSM-tree. Moreover, multiple quantile queries could share the data passes for probabilistic filters in range estimation. Extensive experiments on real and synthetic datasets demonstrate the superiority of our proposal compared to the existing methods with deterministic filters. On average, our method takes 0.48 fewer passes and 18% of the time compared with the state-of-the-art deterministic sketch (GK sketch).

CCS Concepts: • **Information systems** → **Online analytical processing**; *Data warehouses*.

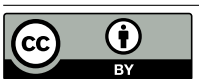
Additional Key Words and Phrases: Quantile, Data stream, Sketches

ACM Reference Format:
Ziling Chen, Haoquan Guan, Shaoxu Song, Xiangdong Huang, Chen Wang, and Jianmin Wang. 2024. Determining Exact Quantiles with Randomized Summaries. *Proc. ACM Manag. Data* 2, N1 (SIGMOD), Article 25 (February 2024), 26 pages. https://doi.org/10.1145/3639280

1 INTRODUCTION

Quantiles are frequently queried in various data science tasks, such as data profiling [17, 29, 44], outlier detection [3] and so on. Given a set of values $\{x_1, \ldots, x_N\}$ equipped with a total order, the rank of any value x, $R(x)$, is the number of values $\leq x$. The ϕ-quantile of the value set, denoted as

Authors' addresses: Ziling Chen, Tsinghua University, China, chen-zl22@mails.tsinghua.edu.cn; Haoquan Guan, Tsinghua University, China, ghq22@mails.tsinghua.edu.cn; Shaoxu Song, Tsinghua University, China, sxsong@tsinghua.edu.cn; Xiangdong Huang, Tsinghua University, China, huangxdong@tsinghua.edu.cn; Chen Wang, Tsinghua University, China, wang_chen@tsinghua.edu.cn; Jianmin Wang, Tsinghua University, China, jimwang@tsinghua.edu.cn.



ACM 2836-6573/2024/2-ART25
https://doi.org/10.1145/3639280

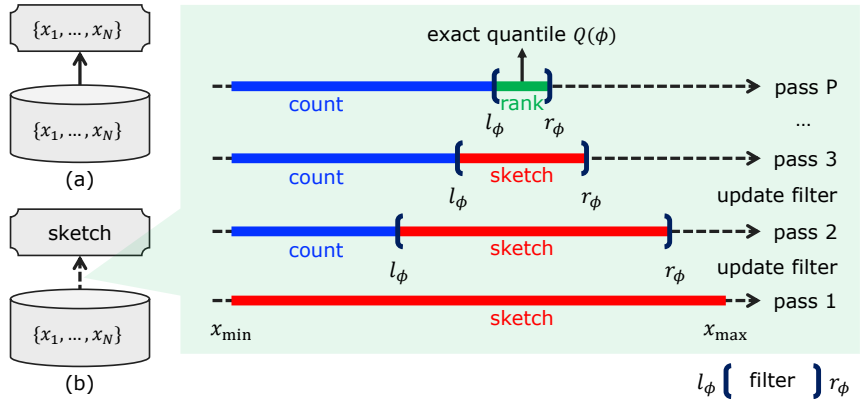


Fig. 1. Existing methods of (a) loading the entire data and (b) multi-pass selection with deterministic sketch

$Q(\phi)$, is the value x_i having $R(x_i) = \phi N$. For instance, the median is denoted as $Q(0.5)$. It is always desired to compute the quantiles with less computing resources.

1.1 Motivation

Unfortunately, computing exact quantiles, also known as the *selection* problem, is costly in either space or time. In databases, while data are sorted by keys, the quantiles queried on non-key values are difficult to rank. For example, the implementations in InfluxDB [38], PostgreSQL [39] and so on simply load the entire data and rank them for the quantiles. Although efficient algorithms exist for ranking, such as QuickSelect [18], as shown in Figure 1(a), the space cost of loading the entire data in memory is usually not affordable.

In practice, it is often the case with limited memory space, especially in end devices of IoT or databases with heavy loads. For example, there is only 320KB memory (SRAM) for a popular ARM Cortex-M7 microcontroller units (MCU) STM32F746 [30]. In the time-series database Apache IoTDB [23], the memory budget for a function defaults to 30MB and can be up to only 1.5% of physical memory. For the cases of high-concurrency scenarios or multi-quantile computation (like equal-frequency binning), the memory limit for each quantile is even tighter.

Existing methods such as [34] propose to scan the data in multiple passes. Sketches like GK sketch [16], DDSketch [33] or t-digest [14] are constructed in each pass to determine the range of values where the quantile must lie, known as *deterministic filter* with $l_\phi \le Q(\phi) \le r_\phi$. As illustrated in Figure 1(b), it gradually shrinks the range of the queried quantile till the range is small enough to fit in memory for ranking the result. Unfortunately, the determined ranges of quantiles could be large, i.e., shrink slowly with many passes. In this sense, we focus on practical solutions that may reduce the passes, by shrinking the ranges more aggressively.

1.2 Intuition

Rather than the exact ranges of quantiles, we propose *probabilistic filters*, as shown in Figure 2(a). By randomized summaries, such as KLL sketch [26], we can obtain that with a high probability $1 - \delta$ the quantile ϕ lies in a small range, i.e., $\Pr[l_{\phi,\delta} \le Q(\phi) \le r_{\phi,\delta}] \ge 1 - \delta$. Intuitively, the higher the probability $1 - \delta$ is, the larger the range would be, as illustrated in Figure 2(b). For $\delta = 0$, it is exactly the deterministic filter, guaranteeing $l_{\phi,0} \le Q(\phi) \le r_{\phi,0}$.

The probabilistic filter means that there is a probability δ the quantile ϕ not in the range $[l_{\phi,\delta}, r_{\phi,\delta}]$, namely *failure probability*. For instance, in pass 3 in Figure 2(a), after counting the values less than $l_{\phi,\delta}$, we find that the quantile is not in the range of $[l_{\phi,\delta}, r_{\phi,\delta}]$, but in $[l_{\phi,0}, l_{\phi,\delta}]$, i.e., a failure.

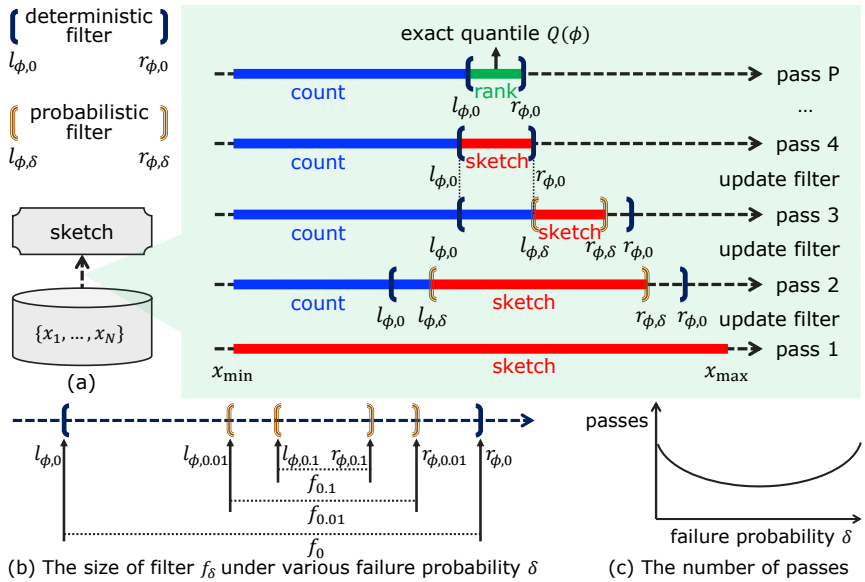


Fig. 2. Our proposal with more aggressive probabilistic filter determined by randomized sketch

Thereby, in the next pass, we build a sketch for the data in the new range $[l_{\phi,0}, r_{\phi,0} = l_{\phi,\delta}]$ and carry on the iteration. In other words, a smaller range of probabilistic filter may have a higher failure probability δ and incur more passes.

To our best knowledge, this is the first study on probabilistic filters for finding exact quantiles. The purpose of existing studies about randomized summaries [5, 26] is to obtain approximate ranks, whereas we aim to determine the ranges of exact quantiles. Therefore, probabilistic filters are proposed in this study for finding the exact quantile.

1.3 Challenge

In this paper, we use the existing KLL sketch [26] which can handle arbitrary ϕ. Similar to the existing deterministic filter [16], $l_\phi \le Q(\phi) \le r_\phi$, given any ϕ, the essential problem to solve is thus how to select a proper failure probability δ for the probabilistic filter $[l_{\phi,\delta}, r_{\phi,\delta}]$ such that the passes could be reduced. If the failure probability δ is too small, e.g., $\delta = 0$ as aforesaid, it is exactly the deterministic filter. On the other hand, if δ is too large, the probabilistic filter frequently fails and incurs extra passes as well. Figure 2(c) shows the relationship between δ and number of passes, which is also observed in Figure 14 of experiments on real data.

The problem is thus reduced to the estimation of passes needed to determine the exact quantile given any δ, so that we can find a proper failure probability δ with the minimum estimated passes. It is not surprising that the number of passes depends on the data size in each iteration (remaining after filtering), i.e., should be estimated recursively. Moreover, the failure case of probabilistic filter, incurring extra passes, further complicates the recursive estimation.

1.4 Contribution

Our major technique contributions in this study are as follows.

(1) We first propose the probabilistic filter, in Section 3.1, which has at least probability $1 - \delta$ with the queried quantile ϕ in the range of $l_{\phi,\delta}$ and $r_{\phi,\delta}$. The mean and distribution of filter size f_δ,

Table 1. Notations

Symbol	Description
N	Amount of values $\{x_1, ...x_N\}$ on disk
M	Memory limit
ϕ	Quantile in $[0, 1]$
$Q(\phi)$	Value of ϕ-quantile
$R(x)$	Rank of value x in data
$\hat{R}(x)$	Estimated rank of value x in a sketch
δ	Failure probability
$l_{\phi,\delta}, r_{\phi,\delta}$	Filter having $\Pr[l_{\phi,\delta} \leq Q(\phi) \leq r_{\phi,\delta}] \geq 1 - \delta$
f_δ	δ-filter size $\lvert\{x_i \mid l_{\phi,\delta} \leq x_i \leq r_{\phi,\delta}, 1 \leq i \leq N\}\rvert$ of any ϕ
$F_{M,\delta}(N)$	Estimated passes for values with size N
$G_{M,\delta}(X)$	Estimated passes for values with size following X

i.e., the number of values in the range, are analyzed in Proposition 3.3, which are essential to pass estimation.

(2) We estimate the number of passes for querying the exact quantile with probabilistic filters, in Section 3.2. It can be recursively estimated by using only data size N and memory limit M, without relying on specific features of the data (no need to traverse data).

(3) We compute the exact quantile by multiple passes of data with randomized summaries, in Section 3.3. Each pass uses a probabilistic filter with proper δ that can minimize the estimated remaining passes. When setting $\delta = 0$, it is exactly the existing method with deterministic filter, i.e., a special case of our proposal.

(4) We analyze the time performance of selection with probabilistic filters, in Section 4.1. Note that the required passes for determining the exact quantile with randomized summaries are bounded in the worst case, no worse than twice of the deterministic filter (Proposition 4.1). The extra time cost of pass estimation is negligible compared to value sketching (Proposition 4.4).

(5) We prove in Section 4.2 that our method can terminate in P passes with $1 - \delta$ confidence, storing $O(N^{1/P} \log^{\frac{P-1}{2P}}(\frac{1}{\delta}))$ items (Proposition 4.5). It is close to the lower bound $\Omega(N^{1/P})$ [34] for a fixed δ. The space bound is asymptotically better than that of selection with GK sketch [16], the optimal deterministic comparison-based sketch [7], known for $P = 2$ passes.

(6) We deploy the proposed method as the quantile function in a time-series database Apache IoTDB, in Section 5. Rather than single quantiles, the implementation supports query processing of multiple quantiles at a time, to share the passes of data. Moreover, pre-computation of the randomized summaries is also enabled over the immutable SSTables in the LSM-tree based storage [41].

Finally, we conduct extensive experiments on real and synthetic datasets to demonstrate the superiority of our proposal, in Section 6, compared to the existing methods with deterministic filters as well as the baseline with unlimited memory. Remarkably, our method takes 0.48 fewer passes and 18% of the time compared with the state-of-the-art deterministic sketch (GK sketch) on average.

The documentation of the deployed quantile function in Apache IoTDB is available in [20]. The code of our proposal is included in the GitHub repository of the system [21]. The experiment-related code and data are available (anonymously) for reproducibility [4]. Table 1 lists the frequently used notations in this paper. The full proofs are available in [40].

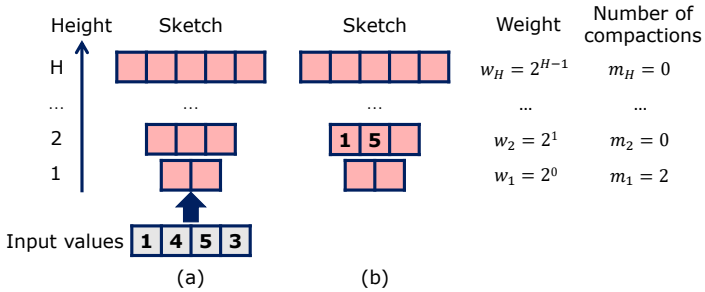


Fig. 3. A sketch with leveled compactors

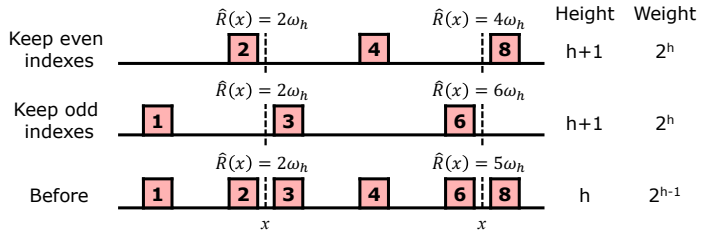


Fig. 4. Influence of randomized compaction

2 PRELIMINARIES

A family of sketches [2, 5, 24, 26, 32] use a buffer to store items, called compactor. It triggers a compaction operation when full. A compaction will sort items, output even-indexed or odd-indexed items to the higher level, and clear the current one. Figure 3 shows an example of adding 4 values to a KLL sketch [26] with H leveled compactors.

Randomized compaction, first introduced in [2], is to randomly keep even-indexed items or odd-indexed items. As shown in Figure 4, the i-th randomized compaction at height h additively contributes $w_h \cdot X_{i,h}(x)$ to the estimated rank $\hat{R}(x)$ of x, where $X_{i,h}(x)$ are independent random variables with $\mathbb{E}[X_{i,h}(x)] = 0, X_{i,h}(x) \in \{0, -1, 1\}$.

In Figure 3(a), when the first two values (1 and 4) are ingested into the sketch, a compaction is triggered and the odd-indexed value 1 in the sorted 1, 4 is propagated. When the next two values (5 and 3) are ingested, another compaction is triggered and the even-indexed value 5 in sorted 3, 5 is propagated. Thus, the result of propagating values 1 and 5 in Figure 3(b) comes from two independent randomized compactions. In each compaction, whether keeping even-indexed or odd-indexed values is randomly decided.

Quantiles can be estimated from the sketch in Figure 3 with rank error. The estimated rank of 1 is $\hat{R}(1) = 1 \cdot w_2 = 2$, while its actual rank is $R(1) = 1$. The estimated quantile of $\phi = 0.5$ is $\hat{Q}(0.5) = 1$ while $Q(0.5) = 3$. The error of rank estimation for any value x, is $\hat{R}(x) - R(x) = \sum_{h=1}^{H-1} \sum_{i=1}^{i=m_h} w_h \cdot X_{i,h}(x)$ and thus $\mathbb{E}[\hat{R}(x) - R(x)] = 0$, i.e., the sketch is unbiased.

The rank error is zero-mean sub-Gaussian as pointed out in [5, 26], since it is a weighted sum of independent $X_{i,h}(x)$. A zero-mean variable X with variance σ^2 is sub-Gaussian if $\mathbb{E}[\exp(sX)] \leq \exp(-\frac{1}{2}s^2\sigma^2)$ for any $s \in \mathbb{R}$. Its variance is simply given by $\sigma^2 = \sum_{h=1}^{H-1} \sum_{i=1}^{i=m_h} w_h^2 \cdot \mathrm{Var}[X_{i,h}(x)]$. Thus the standard (Chernoff) tail bound for sub-Gaussian variables can be applied as in [5, 26], $\Pr\left[|\hat{R}(x) - R(x)| > a\right] \leq 2\exp\left(-\frac{a^2}{2\sigma^2}\right)$, for any $a > 0$. In other words, the probability of a large rank error is small, which inspires our proposal of probabilistic filters.

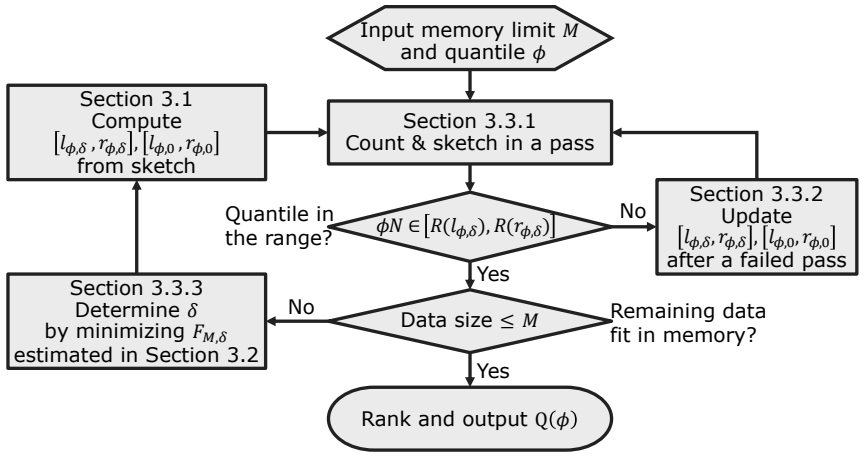


Fig. 5. Quantile query processing

Rather than estimation, our work is to determine $Q(\phi)$ from sketches with randomized compactions.

3 QUANTILE COMPUTATION

Our main goal is to adopt randomized quantile sketches in the multi-pass selection and reduce the number of passes on average. We compute probabilistic filters from sketches, investigate properties of filters and determine failure probability by well-evaluating the required passes. Figure 5 gives an overview of the solution, summarizing the technique details in the following sections.

3.1 Probabilistic Filter

As shown in Figure 2(a), probabilistic filters are the foundation of multi-pass selection. They are formally defined as follows.

Definition 3.1 (Probabilistic Filter). The probabilistic filter for ϕ-quantile with failure probability δ is denoted by $[l_{\phi,\delta}, r_{\phi,\delta}]$, having

$$\Pr\left[l_{\phi,\delta} \leq Q(\phi) \leq r_{\phi,\delta}\right] \geq 1 - \delta.$$

In the following, we present the computation of probabilistic filters based on sketches, and the properties of filter size (the number of input values lying between filters).

3.1.1 Filter with Sketch. Here we propose to compute probabilistic filters based on any randomized sketch and failure probability δ.

As described in Section 2, the rank error $\hat{R}(x) - R(x)$ for any value x in a randomized sketch is zero-mean sub-Gaussian with variance $\sigma^2 = \sum_{h=1}^{H-1} \sum_{i=1}^{i=m_h} w_h^2 \cdot \mathrm{Var}[X_{i,h}(x)]$, where $X_{i,h}(x) \in \{0, -1, 1\}$, $\mathbb{E}[X_{i,h}(x)] = 0$, $|X_{i,h}(x)| \leq 1$, $\Pr[X_{i,h}(x) = 0] \geq 0.5$ [24]. We maintain the number of compactions m_h for the sketch, estimate $\mathrm{Var}[X_{i,h}(x)]$ with 0.5 and thus estimate the variance of error with

$$\sigma^2 = \frac{1}{2} \sum_{h=1}^{H-1} m_h w_h^2. \tag{1}$$

Now we analyze the rank error under any δ, i.e., the t where $\Pr\left[|\hat{R}(x) - R(x)| \geq t\right] \leq \delta$. For $\delta > 0$, we simply compute t by solving $\Pr[|X| \geq t] \leq \delta$ where $X \sim \mathcal{N}(0, \sigma^2)$ and $\sigma^2 = \frac{1}{2} \sum_{h=1}^{H-1} m_h w_h^2$. That is, we use a Gaussian distribution to represent the sub-Gaussian error distribution, which

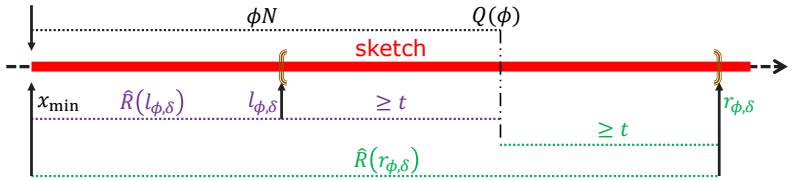


Fig. 6. Probabilistic filter computed in Formula 3

is different from the previous works [5, 26] applying the Chernoff tail bound for sub-Gaussian distribution.

The rank error is the sum of errors introduced by compaction operations, each of which is zero-mean and independent. According to the central limit theorem, a Gaussian distribution works when there are many compactions. The actual ranking error will be in the Chernoff bound but exceed the Gaussian distribution at the tail (where δ approaches 0) when there are few compactions. It is also the reason why the actual failure probability is larger than the estimation for small δ in Figure 13(a) of experiments below.

As for $\delta = 0$, we report the error bound as that in the deterministic sketch [32]. In summary, we have

$$t = \begin{cases} \mathcal{F}_X^{-1}(1 - \frac{1}{2}\delta), & \delta > 0 \\ \sum_{h=1}^{H-1} m_h w_h, & \delta = 0 \end{cases} \tag{2}$$

where $\mathcal{F}_X^{-1}$ is the inverse cumulative distribution function of $\mathcal{N}(0, \sigma^2)$ and σ^2 is computed in Formula 1.

Proposition 3.2 (Filter with Sketch). *Given δ and a sketch guaranteeing* $\Pr\left[|\hat{R}(x) - R(x)| \geq t\right] \leq \delta$ *for any value x, where t could be computed in Formula 2, the filter $[l_{\phi,\delta}, r_{\phi,\delta}]$ having*

$$\begin{aligned} l_{\phi,\delta} &= \max\{x \mid \hat{R}(x) + t \leq \phi N\} \\ r_{\phi,\delta} &= \min\{x \mid \hat{R}(x) - t \geq \phi N\} \end{aligned} \tag{3}$$

is a probabilistic filter with failure probability δ for ϕ-quantile.

Proof Sketch. Intuitively, since the sketch only provides estimated rank $\hat{R}(x)$, the idea is to let filter $l_{\phi,\delta}, r_{\phi,\delta}$ have estimated ranks $\hat{R}(l_{\phi,\delta}), \hat{R}(r_{\phi,\delta})$ containing the quantile ϕ. As shown in Figure 6, there is $\hat{R}(l_{\phi,\delta}) + t \leq \phi N$, $\hat{R}(r_{\phi,\delta}) - t \geq \phi N$ for filter computed by Formula 3. The full proof is in the online supplementary [40]. □

3.1.2 Filter Size. To evaluate filters, the filter size $f_\delta = R(r_{\phi,\delta}) - R(l_{\phi,\delta})$, i.e., how many data lie between the filter, must be known. Intuitively, the filter size is also a random variable as influenced by randomized compactions. The f_δ can be represented with rank error, $R(r_{\phi,\delta}) - R(l_{\phi,\delta}) = \hat{R}(r_{\phi,\delta}) - \hat{R}(l_{\phi,\delta}) - \left(\hat{R}(r_{\phi,\delta}) - R(r_{\phi,\delta})\right) + \left(\hat{R}(l_{\phi,\delta}) - R(l_{\phi,\delta})\right)$.

Proposition 3.3. *Given a sketch with σ^2-sub-Gaussian error guaranteeing* $\Pr\left[|\hat{R}(x) - R(x)| \geq t\right] \leq \delta$, *where σ^2 and t could be computed in Formulas 1 and 2, it holds that $\mathbb{E}[f_\delta] = 2t$, and $f_\delta - \mathbb{E}[f_\delta]$ is zero-mean σ^2-sub-Gaussian.*

Proof Sketch. Recall that we have $\mathbb{E}\left[\hat{R}(x) - R(x)\right]$=0 in Section 2, thus $\mathbb{E}[f_\delta] = 2t - 0 + 0 = 2t$. Then we represent $f_\delta - \mathbb{E}[f_\delta]$ with a weighted sum of $X_{i,h}(l_{\phi,\delta}) - X_{i,h}(r_{\phi,\delta})$, which is zero-mean sub-Gaussian based on the properties of $X_{i,h}(x)$ in Section 2. The full proof is in the online supplementary [40]. □

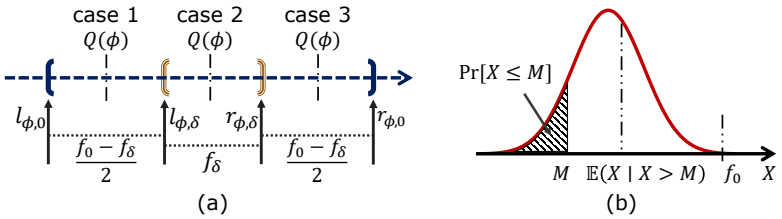


Fig. 7. Pass estimation with (a) various situations considered in computing F of Formula 4 and (b) filter size distribution considered in computing G of Formula 5

3.2 Pass Estimation

As shown in Proposition 3.3, the average size of probabilistic filters $\mathbb{E}[f_\delta]$ varies with δ. In this section, we provide the estimated number of passes under any δ and the relevant analysis.

The goal of multi-pass selection is to shrink the number of values in ranges and may succeed or fail in each pass. We propose to recursively estimate the number of passes. Our target is to estimate the number of passes given any N, M, δ, which is defined as follows.

Definition 3.4. Given memory budget M and δ, the estimated number of passes for selection in N values is denoted by $F_{M,\delta}(N)$.

Note that the filter size, i.e., the number of values in the range in the next pass, is sub-Gaussian as discussed in Proposition 3.3. Thereby, we should estimate the number of passes for values with size following a distribution, defined as follows.

Definition 3.5. Given memory budget M and failure probability δ, the estimated number of passes for selection in values with size following a sub-Gaussian distribution X is denoted by $G_{M,\delta}(X)$.

We denote δ-filter size after sketching N values as $f_\delta(N)$ and the recursive estimation of $F_{M,\delta}(N)$ is as follows:

$$\begin{aligned} F_{M,\delta}(N) = (1-\delta) \cdot \left(1 + G_{M,\delta}\left(f_\delta(N)\right)\right) \\ + \delta \cdot \left(2 + G_{M,\delta}\left(\frac{f_0(N) - f_\delta(N)}{2}\right)\right). \end{aligned} \tag{4}$$

As shown in Figure 7(a), when computing $F_{M,\delta}(N)$, we consider both success and failure. If the next pass succeeds, i.e., $Q(\phi) \in \left[l_{\phi,\delta}, r_{\phi,\delta}\right]$, the number of values in the next pass simply follows $f_\delta(N)$. There are $1 + G_{M,\delta}\left(f_\delta(N)\right)$ passes, where the number 1 is for the next pass. If the next pass fails, i.e., $Q(\phi) \notin \left[l_{\phi,\delta}, r_{\phi,\delta}\right]$, the number of values follows $\frac{f_0(N) - f_\delta(N)}{2}$. There are $2 + G_{M,\delta}\left(\frac{f_0(N) - f_\delta(N)}{2}\right)$ passes because of an extra pass incurred by failure.

We propose to estimate $G_{M,\delta}(X)$ as follows:

$$\begin{aligned} G_{M,\delta}(X) = &\Pr[X \le M] \cdot 1 \\ &+ \Pr[X > M] \cdot F_{M,\delta}\left(\mathbb{E}[X \mid X > M]\right). \end{aligned} \tag{5}$$

Figure 7(b) shows the cases considered in computing $G_{M,\delta}(X)$. There is a chance of $\Pr[X \le M]$ that all values can be stored in memory with size M and the recursion terminates. For the case of $X > M$, we compute filter sizes with the mean value $\mathbb{E}[X \mid X > M]$, instead of the truncated distribution $X \mid X > M$ itself, to make the computation feasible.

For Formula 4 about pass estimation, we illustrate the situations of multi-pass selection in Figure 7, based on the 2nd and 3rd passes in Figure 2(a). For the case of success, after the 2nd pass, the δ-filter (computed after the 1st pass) is found to be successful and now the sketch represents data in the δ-filter with a size following f_δ, as the case 2 in Figure 7(a). For the case of failure, after the

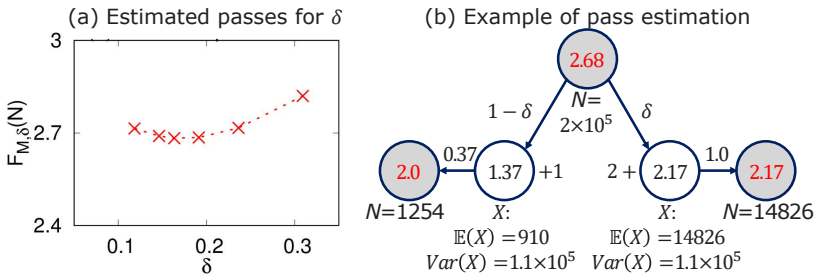


Fig. 8. (a) Estimated passes of some evaluated δ and (b) Recursive pass estimation for δ=0.174 in Example 3.6

3rd pass, the δ-filter (computed after the 2nd pass) is found to be a failure. In the 4th pass, we use the correct filter $[l_{\phi,0}, l_{\phi,\delta}]$ with a size following $\frac{f_0 - f_\delta}{2}$, as the case 1 in Figure 7(a).

3.3 Selection Algorithm

In this section, we introduce the selection algorithm with randomized sketches. The process is shown in Figure 5 and the corresponding pseudo-code is given in Algorithm 1.

3.3.1 Counting and Sketching in a Pass of Data. This part is about how to deal with data values. Recall that in Figure 2(a), a randomized sketch summarizing all values is built in the first pass. N is known after the first pass (Line 10). In the following passes, sketching and counting are performed based on the probabilistic filter $l_{\phi,\delta}$, $r_{\phi,\delta}$ (Lines 5-8). Values in $[l_{\phi,\delta}, r_{\phi,\delta}]$ are inserted into the sketch (Line 6). In Line 6, when the size limit of the sketch is exceeded, a compaction in KLL sketch at height h is triggered and the corresponding compaction number m_h increases by 1.

3.3.2 Handling Failure. At the end of each pass, whether $Q(\phi)$ lies between the filter is checked by comparing ϕN with the counting results (Lines 11-12). In the case of failure, the new filter is $[l_{\phi,0}, l_{\phi,\delta}]$ or $[r_{\phi,\delta}, r_{\phi,0}]$, depending on the result of comparison.

We can know that the quantile ϕ is on the right side of the filter if there is $\phi \cdot N > R(l_{\phi,\delta}) + S.getN()$, where $R(l_{\phi,\delta})$ is number of values on the left side of the filter, and $S.getN()$ is the number of values inserted in the sketch S.

To count the data on the left of the filter, i.e., to obtain $R(l_{\phi,\delta})$, we increase a counter by 1 if the scanned value is smaller than $l_{\phi,\delta}$ in the pass of scanning the data, as in Line 7 of Algorithm 1. Thus, we do not have to sort the data for the counting.

3.3.3 Determining Failure Probability. After a successful pass, either the quantile is computed (Line 13), or δ along with δ-filter for the next pass should be determined. Based on pass estimation for any δ in Section 3.2, we determine the best δ, i.e., finding $\delta^* = \arg\min_{\delta} F_{M,\delta}(N)$. A naive solution is to try $\delta = 0.01, 0.05, \ldots$ and report the best. Based on the observation of the single-peak curve in δ-pass figure, our solution is to use the Golden Section Search method [27] to find δ^* (Line 15). The $F_{M,\delta}(N)$ will be called $O(\log \epsilon_\delta^{-1})$ times, where ϵ_δ is the precision of δ. In our implementation, we set $\epsilon_\delta = 5 \times 10^{-4}$ and search among $\delta \in [5 \times 10^{-4}, 0.5]$. After determining δ, the new filters $[l_{\phi,0}, r_{\phi,0}]$ and $[l_{\phi,\delta}, r_{\phi,\delta}]$ for the next pass are computed (Line 16) by Formula 3.

Example 3.6. Consider $M = 1024, \phi = 0.5$ and $\{x_1, \ldots, x_N\}$ being a permutation of size N=2×10^5. After the first pass, we have a randomized sketch S summarizing all values, N=2×10^5 and ϕN=1×10^5 (Lines 5-10). Based on N, M, the failure probability δ is determined to be 0.174 (Line 15).

We provide details on how δ=0.174 is chosen and the corresponding pass estimations. Numbers of compactions triggered in the sketch S at height 1 to 9 are $[4771, 1745, 599, 198, 70, 24, 8, 3, 1]$

Algorithm 1: MultiPassSelection(M, $\{x_1, ..., x_N\}$, ϕ)

Input: Memory budget M, values $\{x_1, ..., x_N\}$, quantile ϕ
Output: Queried quantile $Q(\phi)$

```
1  P, δ ← 0; l_{φ,0}, l_{φ,δ} ← −∞; r_{φ,0}, r_{φ,δ} ← +∞;
2  while l_{φ,0} < r_{φ,0} do
3      S ← an empty randomized sketch with size limit M;
4      R(l_{φ,δ}) ← 1;
5      foreach value x do
6          if x ∈ [l_{φ,δ}, r_{φ,δ}] then insert x into S;
7          else if x < l_{φ,δ} then R(l_{φ,δ}) ← R(l_{φ,δ}) + 1 ;
8      end
9      P ← P + 1;
10     if P = 1 then N←S.getN();
11     if φN < R(l_{φ,δ}) then r_{φ,δ}, r_{φ,0}←l_{φ,δ}, l_{φ,δ}←l_{φ,0} ;
12     else if φN > R(l_{φ,δ}) + S.getN() then l_{φ,δ}, l_{φ,0}←r_{φ,δ}, r_{φ,δ}←r_{φ,0} ;
13     else if S.getN() ≤ M then return S.getV(φN − R(l_{φ,δ}));
14     else
15         δ←arg min_δ F_{M,δ}(N) with Golden Section Search;
16         Update l_{φ,δ}, r_{φ,δ}, l_{φ,0}, r_{φ,0} with Formula 3;
17     end
18 end
```

respectively, and the variance of rank error is computed as σ^2=1.1×10^5 according to Formula 1. Figure 8(a) shows the first six evaluated δ and their estimated passes $F_{M,\delta}(N)$ as in Section 3.3.3. After the six estimations, the range of the optimal δ is limited to $[0.146, 0.191]$, and the δ=0.174 will be determined after ten more estimations. Figure 8(b) shows the recursive pass estimation for δ=0.174, i.e., how $F_{M,0.174}(N)$=2.68 is calculated with Formula 4.

Though $\delta = 0.174$ is deterministic given N, M, the specific values of updated filters (Line 16) are uncertain due to the randomness in the sketching process, i.e., S will be different in each run:

- In one case, $[l_{\phi,\delta}, r_{\phi,\delta}]$=[99726, 100565], and $Q(0.5)$ is determined after the 2nd pass (Line 13);
- In another case, $[l_{\phi,\delta}, r_{\phi,\delta}]$ = [99132, 99879], $[l_{\phi,0}, r_{\phi,0}]$ = [84336, 114927], and $Q(0.5)$ is found to be larger than $r_{\phi,\delta}$. The new filter is [99879, 114927] (Line 12), needing more passes.

4 PERFORMANCE ANALYSIS

It is not surprising that there is a trade-off between the pass number and the required space, as analyzed in Proposition 4.5 and observed in Figure 17 of experiments. With reasonable δ, the space bound of our method is asymptotically better than that of GK sketch [16] known for 2 passes, which is meaningful for limited memory.

4.1 Time Complexity

To understand the time cost, we first present that the number of passes is always bounded after introducing probabilistic filters. Moreover, in each pass, the extra time overhead for δ evaluation is negligible compared to the unavoidable cost of sketching values.

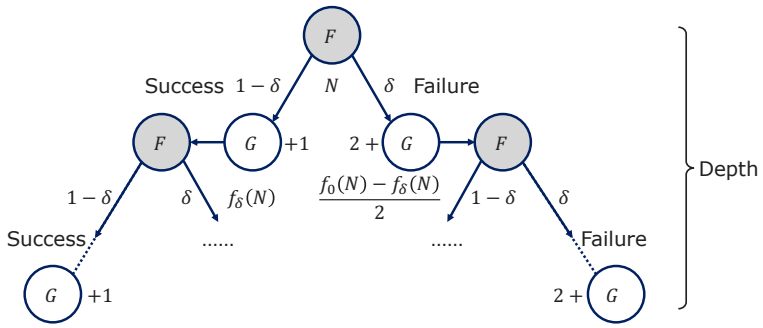


Fig. 9. Recursive tree of pass estimation

4.1.1 Upper Bound of Passes. We show that the number of passes is still bounded in the worst case of failure in all probabilistic filters.

Proposition 4.1. *If the quantile can be found in P-pass with deterministic summaries, the number of passes with randomized summaries is less than $2P$.*

Proof Sketch. The idea is that in the case of failure, the method takes 2 passes for a range smaller than that after 1 pass with deterministic summaries. If P passes (shrinking value ranges $P-1$ times) are enough for deterministic sketches, $(P-1)*2+1 = 2P-1$ passes are sufficient. The full proof is in the supplementary [40]. □

4.1.2 Estimation Time Complexity. To bound the time complexity of recursively computing $F_{M,\delta}(N)$, properties of the recursive tree shown in Figure 9 should be analyzed.

Lemma 4.2. *The number of nodes in the recursive tree, i.e., the number of recursive calls in computing $F_{M,\delta}(N)$, is $O\left(N^{1/\log(M/\log^2 N)}\right)$.*

Proof Sketch. The key idea is to bound the number of nodes with the depth of the recursive tree, which does not exceed the number of passes with deterministic filter, and apply results in [34]. The full proof is in the online supplementary [40]. □

Note that Formulas 1 and 2 about σ^2 and f_δ, used in Formula 4 of computing $F_{M,\delta}(N)$, rely on accurate m_h. The computation of m_h dominates the time complexity. In all nodes representing F, except the root node where m_h is maintained in the last pass, an additive complexity is introduced for computing m_h. Therefore, the total number of values should be considered.

Lemma 4.3. *The total number of values in all nodes representing F except the root node on the recursive tree is $O\left(N\log^2 N/M\right)$.*

Proof Sketch. The total number of values is $O(f_\delta(N) + \frac{f_0(N)-f_\delta(N)}{2} + \ldots) = O(f_0(N))$ and $O(f_0(N)) = O\left(N\log^2 N/M\right)$ is proven in [34]. □

Finally, we obtain the cost of pass estimation.

Proposition 4.4. *The time complexity of computing $F_{M,\delta}(N)$ is $O\left(N\log^4 N/M^2\right)$, which is $O(N)$ when $M = \Omega(\log^2 N)$.*

Proof Sketch. The idea is to analyze the cost given the number of values, and summing the cost in all nodes based on Lemma 4.3. The cost depends on the number of compactions, and results in [24] can be applied. The full proof is in the supplementary [40]. □

Table 2. Space complexity known for 2-pass selection

	Space
Munro-Paterson [34], MRL [32]	$O(N^{0.5}\log N)$
GK [16]	$O(N^{0.5}\log^{0.5} N)$
δ-KLL (our)	$O(N^{0.5}\log^{0.25}(\frac{1}{\delta}))$

Note that $M = \Omega(\log^2 N)$ is necessary for selection with deterministic sketches [34], i.e., $O\left(N\log^4 N/M^2\right)$ can be viewed as $O(N)$ in most cases. This complexity is negligible compared to $\Omega(N)$ of inserting N values into a sketch in the each pass.

4.2 Space Complexity

Asymptotic space cost for achieving certain effect is a common way to evaluate sketch-relevant works, as data sketches are often applied in the scenarios of limited memory. As in previous work about multi-pass selection [34], we analyze the space required for selection in certain passes when applying mergeable KLL sketch [24, 26] in our method, and compare it with existing methods.

4.2.1 P-pass Analysis. As other analyses for randomized sketches [7], we show the space bound for P-pass selection with confidence $1-\delta$.

Proposition 4.5. *The method terminates in P passes with $1-\delta$ confidence, storing $O(N^{1/P}\log^{\frac{P-1}{2P}}(\frac{1}{\delta}))$ items.*

Proof Sketch. The ideas are: (1) Applying $\frac{\delta}{2(P-1)}$-filter P-1 times will not meet any failure with high confidence; (2) When the filter size $f_{\frac{\delta}{2(P-1)}}$ is small, the answer can be found in the P-th pass with high confidence; (3) Recall Proposition 3.3 about filter size, the Chernoff bound for sub-Gaussian variables is applied for the space bound. The full proof is in the supplementary [40]. □

For a fixed δ, either manually specified or determined by our algorithm, the space cost is close to the lower bound $\Omega(N^{1/P})$ [34].

4.2.2 Two-pass Comparison. To our best knowledge, there is no space cost analysis of P-pass selection with GK sketch [16], the optimal deterministic comparison-based sketch [7]. Thereby, we compare the proposed method to the existing works with known result of $P = 2$. In practice, the space cost to achieve 2-pass selection is important, since the algorithms need at least 2 passes to determine the exact quantile, i.e., $P \geq 2$.

Substituting $P = 2$ in Proposition 4.5 gives the result in Table 2. For a better comparison, we give the corollary on certain δ below.

Corollary 4.6. *For $\delta = \Omega(e^{-\log^{\zeta} N})$, where $\zeta > 0$ is a constant, the space bound for 2-passes selection is $O(N^{0.5}\log^{\zeta/4} N)$.*

As shown, for $\delta = \Omega(e^{-\log^{\zeta} N})$ where $\zeta > 2$, the space bound is asymptotically better than that of selection with GK sketch [16]. Note that $\zeta = 1$ means $\delta = \Omega(1/N)$ and thus $\zeta < 2$ is a reasonable condition. Some popular sketches like t-digest [14] and DDSketch [33], which do not provide space bounds, are not shown in the table but compared in experiments. Given the more advanced space complexity, the GK sketch is also compared instead of MRL sketch. Experimental results in Section 6.3.3 show that our δ-KLL method needs the least space to achieve 2-pass selection.

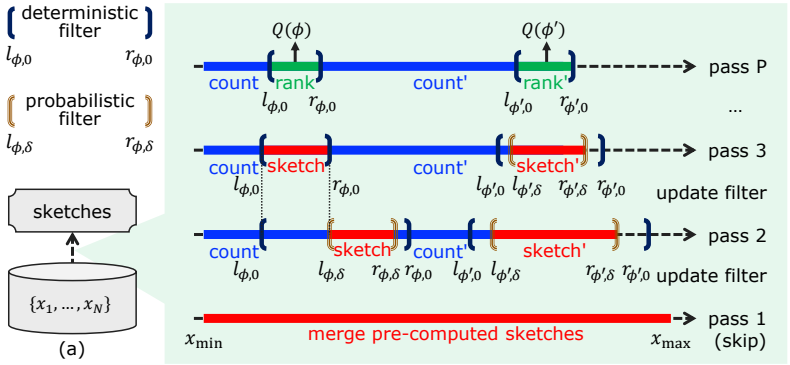


Fig. 10. Computing multiple quantiles

5 SYSTEM DEPLOYMENT

The proposed multi-pass selection with KLL sketch has been deployed as a function in an LSM-tree based database, Apache IoTDB [23]. In this section, we introduce how to deploy and speed up the multi-pass process with shared passes and pre-computation. The previous analyses and conclusions still hold in the implementation.

5.1 Algorithm Implementation

The SQL statement of querying a quantile in a series is as follows.

```
SELECT exact_quantile(s0,'Q'='0.5')
FROM root.sg0.device0
WHERE time >= 2020-01-01 00:00:05
```

5.1.1 Quantile as Multi-pass Aggregation. We introduce how to make the proposed Algorithm 1 work as an aggregate function scanning data in multiple passes. Before each pass, the query executor will make sure data files can be read and inform the quantile operator to prepare as described in Lines 3-4. During a pass, the values $x_1, \ldots, x_N$ are read from disk by I/O components and consumed by the quantile operator as described in Lines 5-8. When the pass ends, i.e., all data files have been fetched, the quantile operator will be informed, execute Lines 9-16 and report whether another pass is required. The query executor will return the query result when the quantile operator reports that the quantile has been computed.

5.1.2 Shared Passes among Multiple Quantiles. In addition to the concurrent queries, there are also statistics naturally rely on multiple exact quantiles, like the ϕ trimmed average and the equal depth binning. Intuitively, we can share the passes of data for K quantiles.

Indeed, all multi-pass selection methods, whether randomized or not, can be adapted to compute K quantiles at the same time. The common adaptions are as follows. After the first pass, K filters are computed. In the following passes, the memory is equally allocated to K' unfinished quantiles, i.e., there are K' sketches summarizing values with the same size limit M/K'.

Figure 10 shows the overview of computing multiple quantiles with randomized sketches. As shown, two quantiles ϕ and ϕ' are queried by maintaining $K = 2$ sketches and counting results after the first pass. The success and failure situations are checked and handled similarly as in computing one quantile (Figure 2(a)).

To locate the sketch or counting result to update for each value, an additive complexity of $O(\log K)$ is introduced. As a comparison, processing K quantiles one by one results in a multiplicative

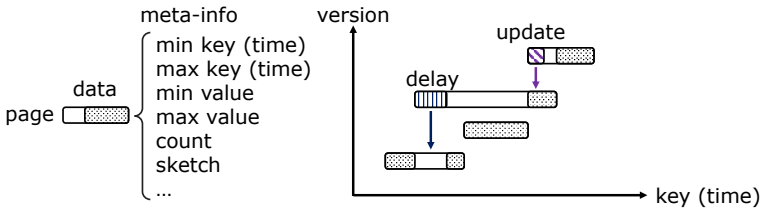


Fig. 11. Pages in LSM-tree store with pre-computed info

complexity of K. However, processing multiple quantiles at the same time may need more passes since memory is divided. Experiment results in Section 6.3 show that computing all quantiles together is more efficient than computing one by one and our proposed method still has superiority in this problem.

5.2 Acceleration with Statistics

Note that data flushed to SSTables on disk are immutable in the LSM-tree based storage [31, 36]. Intuitively, we may pre-compute some statistics or sketches for the immutable data. To improve the performance, we utilize the statistical information collected for every Page, which is the basic I/O unit. A Page in IoTDB is 64KB in default [19]. Figure 11 shows the pre-computed meta-information.

5.2.1 Basic Statistical Information. The query could be accelerated with basic meta-information, including min value, max value and count. As the ranges of filters gradually shrinks in the multi-pass process, some pages may entirely fall outside the filters. After the first pass, they can be found by checking min value and max value and skipped with utilizing the count information.

5.2.2 Pre-computed Quantile Summaries. For fully-mergeable [2] sketches, we can merge pre-computed sketches (also named synopses in [1]) to generate a sketch summarizing all values in the first pass in Figure 10 instead of reading all values. Specifically, we generate a synopsis for each Page when it is being flushed to the disk. This can be implemented in systems by storing index info or statistical info for Pages as in Figure 11.

Note that the out-of-place updates in the LSM-tree make utilizing synopses difficult. As shown in Figure 11, Pages may have overlapped key ranges and some overlaps come from updates, i.e., some values are updated but still on disk. We simply merge all synopses even with updates, since they are only used to estimate the ranges of quantiles. The estimation could still be accurate when updates are few. In the real-life scenario of storing time-series data in IoT [43], updates are rare and overlapped key (time) ranges usually come from unordered arrivals of data [25, 42]. Experiment results in Section 6.4 show that simply merging synopses can accelerate query when less than 0.5% of the data are updated.

6 EXPERIMENTAL EVALUATION

In the experiments, we verify the proposed method, compare it with baselines and show the performance in deployment. The related code and data are available for reproducibility [4].

6.1 Experimental Setup

The experimental evaluation is conducted on a machine with 3.2GHz CPU and 32GB memory. The methods are deployed in Apache IoTDB [23], an LSM-tree based time-series database. In the LSM-tree, the size of MemTable is 4MB and the size of Page is 64KB.

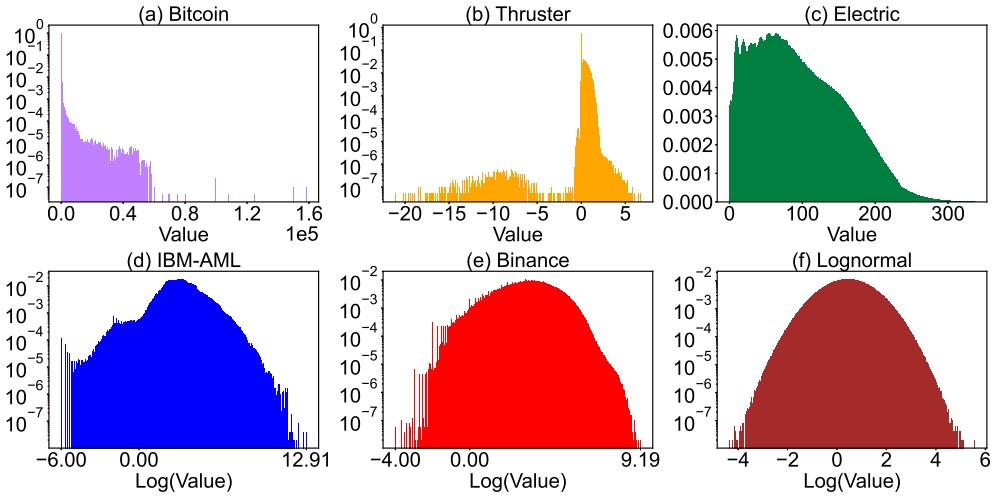


Fig. 12. Value distributions of different datasets

Table 3. Characteristics of datasets

Name	Records	Size	Skewness
Bitcoin	2×10^7	201MB	0.17
Thruster	2×10^7	384MB	1.93
Electric	2×10^7	188MB	0.39
IBM-AML	1.7×10^8	1.45GB	0.009
Binance	8.9×10^8	11.5GB	0.046
Lognormal	1×10^9	17.8GB	0.34

6.1.1 Baselines. Besides the proposed method denoted as δ-KLL, other implemented methods for comparison are as follows.

- QuickSelect is a one-pass method based on the classical algorithm QuickSelect [18, 37] storing all data in memory. We remove the memory limit on it and allow it to occupy all space for the server.
- GK sketch is a multi-pass method based on GK sketch [16], the best-known deterministic method while not fully-mergeable [2].
- DDSketch is a multi-pass method based on DDSketch [33]. It indexes values to buckets by taking logarithm and reports the value range of a bucket as deterministic filters. When values in filter are few enough, the method will terminate in the next pass. Pre-computed DDSketches with the same parameter α can be merged, which are $2^{-4}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-3}, 2^{-5}$ in 6 datasets, respectively.
- t-digest is a multi-pass method based on the merging t-digest with scale function k_0 [14]. The method uses pre-computation and stores extreme values of centroids to get deterministic filters.

6.1.2 Datasets. We employ several real-world or synthetic datasets with distributions as in Figure 12. Table 3 shows statistics of the datasets, including number of records, size on disk and Pearson's median skewness. Owing to the limited space, some similar results on different datasets may be omitted.

- Bitcoin [8] is a public dataset on Kaggle, which records the transaction of Bitcoin since 2009.
- Thruster [12] is a public dataset on Kaggle, which is based on the physics of monopropellant chemical thrusters.

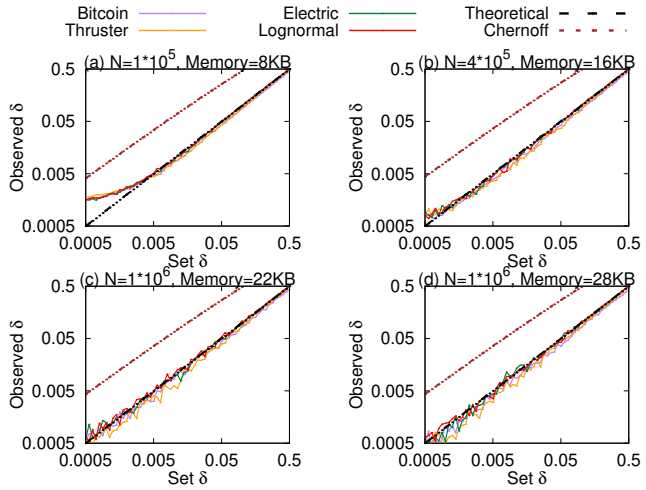


Fig. 13. Accurate failure probability in Formula 3

- Electric [10] is a public dataset on Kaggle, which records daily electrical consumption of blocks in London.
- IBM-AML [11] is a public dataset on Kaggle, recording monetary amounts of transactions for anti-money laundering.
- Binance [9] is a public dataset on Kaggle, recording trade volumes of historical trading pairs.
- Lognormal is a synthetic dataset with 1×10^9 i.i.d. values sampled from a log-normal distribution with a scale parameter of 1 and a shape parameter of 2.

6.1.3 Metrics. In default, to measure the performance, we perform several queries on different parts of data and quantiles ϕ uniformly chosen from $(0, 1)$, and report the average of passes and time cost.

The space costs of methods except QuickSelect are strictly limited for a fair comparison. For example, an item in KLL sketch occupies 8 bytes while a centroid in t-digest takes more space. The number of items M, centroids or buckets is controlled by the memory limit.

6.2 Verification of Techniques

In this section, we verify the proposed properties of probabilistic filters and effectiveness of techniques proposed in Section 3.

6.2.1 Verification of Probabilistic Filters. We verify whether the filter $[l_{\phi,\delta}, r_{\phi,\delta}]$ calculated by Formula 3 in Section 3.1.1 indeed has a failure probability δ. The proposed algorithm would be meaningless if the computed $[l_{\phi,\delta}, r_{\phi,\delta}]$ has a failure probability distinct from δ. Specifically, we compare the given δ and the real failure probability of $[l_{\phi,\delta}, r_{\phi,\delta}]$ observed in the next pass, under different parameters.

Figure 13 provides the results of modeling the error with the conservative (Chernoff) tail bound. As shown, using Chernoff tail bound is conservative and far from actual failure probability, especially for large δ, as it focuses on the tail of distribution.

Modeling error with Gaussian distribution works when there are many compactions, i.e., the central limit theorem can be applied. Figure 13(a) has the fewest compactions, given the smallest ratio of data size N to memory budget for the sketch, since compactions are barely triggered by ingesting less data to a sketch with large memory. Thereby, the sub-Gaussian rank error is

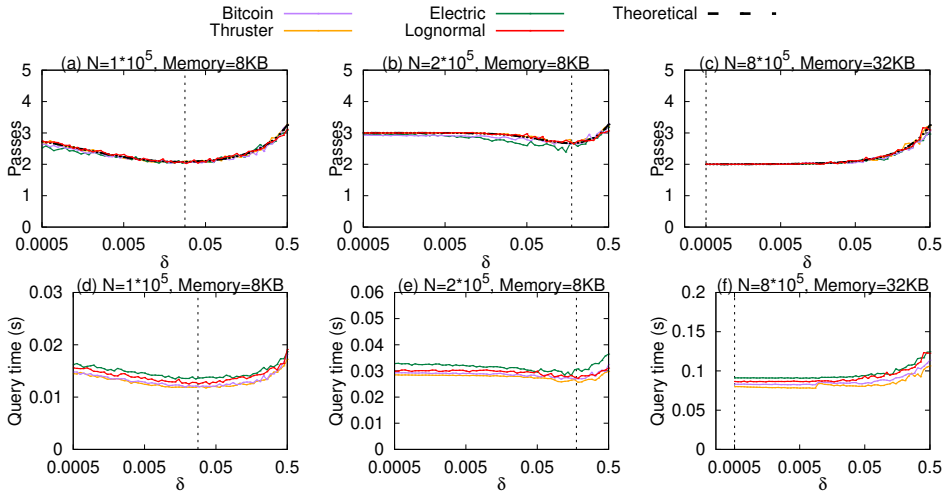


Fig. 14. Accurate pass estimation in Section 3.2

larger than the modeled Gaussian distribution at the tail (when δ approaches 0). Consequently, the computation is aggressive and results in larger actual failure probability. Note that the actual failure probability in Figure 13(a) is still below 0.005 and the method still works well as shown in Figure 14(a) below.

6.2.2 Verification of Pass Estimation. We evaluate the pass estimation for a failure probability δ in Section 3.2 and the determination of the best δ in Section 3.3.3. Accurate pass estimation and δ determination are essential to the performance of Algorithm 1.

Figure 14 shows the number of passes and time cost with varied δ. First, the passes estimated in Section 3.2 are very close to those observed in datasets, and the determined δ (vertical dot line) leads to the minimal passes and minimal query time. In Figure 14(c), the small δ with 5×10^{-4} is the best choice. There are cases for $\delta = 0$ being the best choice, e.g., when the memory limit is not tight and selection with a deterministic sketch requires only 2 passes. Compared with Figure 14(a), the case in Figure 14(b) has a stricter memory limit, leading to larger optimal δ and more passes.

6.3 Comparison with Baselines

In this section, our method is compared with baselines on the number of passes and time cost.

6.3.1 Varying Queried Data Size. Figure 15 shows the average number of passes and time cost with varying data size. QuickSelect works with unlimited memory but suffers from out-of-memory error for $N \geq 8 \times 10^8$. The memory budgets of 32KB and 1MB are chosen referring to the corresponding queried sizes of the small datasets (up to 4M records) and the large datasets (up to 1G records). Nevertheless, in Section 6.3.3, we report the results by varying the memory budget in Figure 17.

Regarding the non-integer passes, it is because we report the average of passes in various tests as introduced in Section 6.1.3. There is an increasing trend for all multi-pass methods. DDSketch needs more passes in dataset Thruster, whose value distribution has tails on both sides. It is consistent with the analysis of DDSketch [33]. The performance of t-digest also varies with different datasets as the effectiveness of averaging rely on value distribution. GK has a stable performance and always needs fewer passes than other deterministic methods. The proposed δ-KLL outperforms multi-pass baselines in all cases. On each case our method takes 0.48 fewer passes than the state-of-the-art

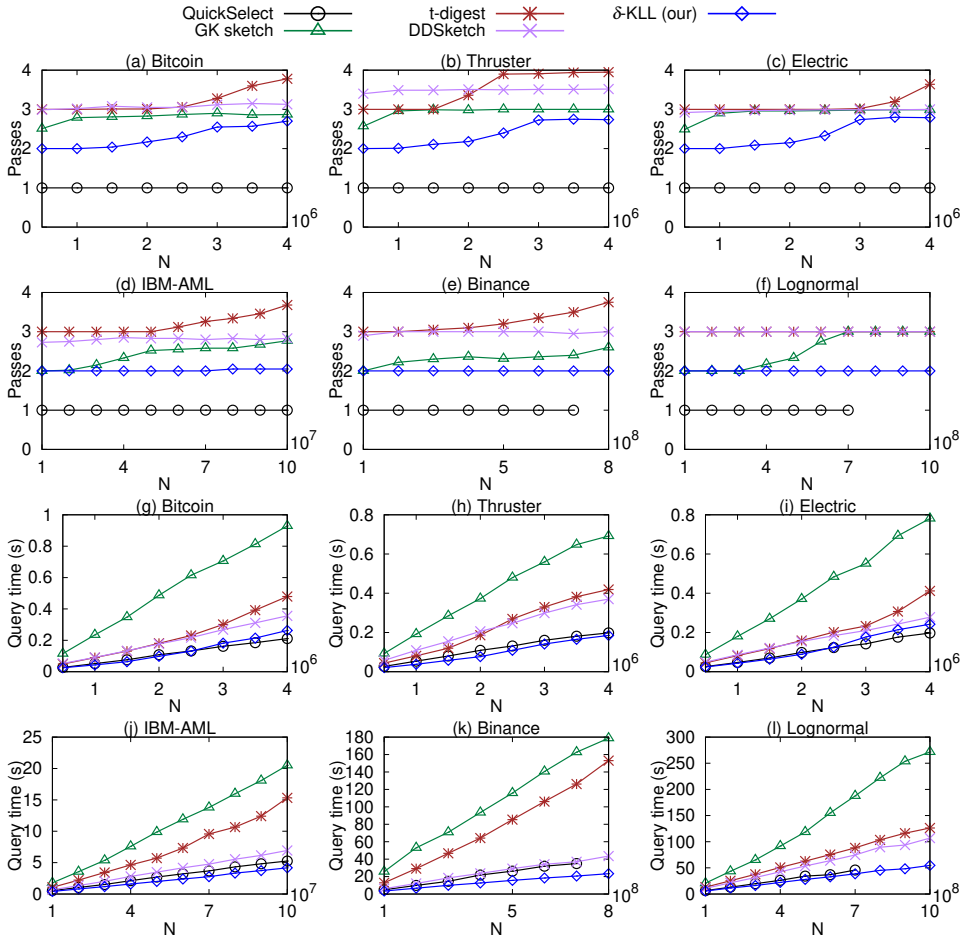


Fig. 15. Comparison by varying data size

deterministic sketch GK on average. Note that GK does not guarantee a result in 2 passes (more than 2 as observed in Figure 15). Contribution (5) and Section 4.2.2 mention that the space bound of GK is known when $P = 2$ passes. The results observed in Figure 15 mean that the memory budget is not sufficient to meet the space bound.

As for time cost, our method outperforms the deterministic ones and is comparable with QuickSelect in most cases. Our method takes 0.48 fewer passes and 18% of the time compared with the state-of-the-art deterministic sketch (GK sketch) on average. As shown in Figure 15, our method outperforms baselines more significantly in the two large-scale datasets IBM-AML and Binance, than that in the small dataset Electric. The large time cost of GK comes from the large update cost of GK sketch. It needs much more time to sketch data in each pass, compared with other methods. The phenomenon is consistent with the results in [33].

6.3.2 Performance without Statistics. We conduct an experiment in Figure 16 to compare all methods, when there is no pre-computation. Our method still takes the least passes, according to the space bound comparisons in Section 4.2.2. The corresponding time cost is therefore lower than

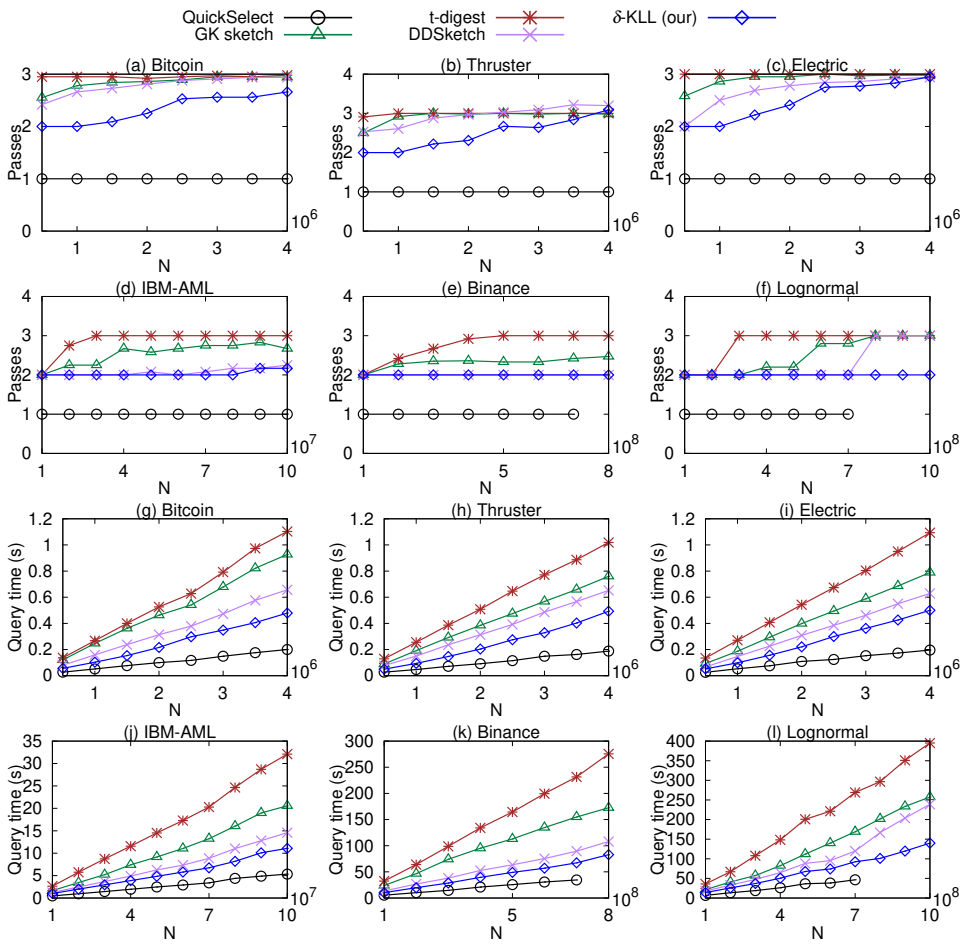


Fig. 16. Varying data size, without pre-computed statistics

the multi-pass baselines. Again, the baseline QuickSelect uses unlimited memory to load all data and suffers from out-of-memory error for $N \geq 8 \times 10^8$.

6.3.3 Varying Memory Limit. Figure 17 shows the performance under various memory limits. Since QuickSelect has unlimited memory, its performance is not affected and omitted.

Required passes of all methods decrease or remain unchanged as the memory limit increases. Our δ-KLL shows significant decrease in passes and fewer than other baselines. Recall the space complexity analysis in Proposition 4.5, it is not surprising to see our method achieves 2-pass selection with the least memory. Again, DDSketch has bad performance in dataset Thruster, and GK sketch outperforms other deterministic ones in passes but not in time cost. Figure 17(f) shows that when the memory limit is quite loose (128MB), DDSketch and t-digest can be efficient, too.

6.3.4 Varying Number of Quantiles. Figure 18 shows the performance in querying multiple quantiles. Memory budget is 1MB and 8MB for small and large datasets respectively. For passes, methods take more passes as more quantiles are queried. Our δ-KLL needs more passes than GK sketch for two reasons: (1) For small datasets, the memory budget (1MB) is larger than the total size of

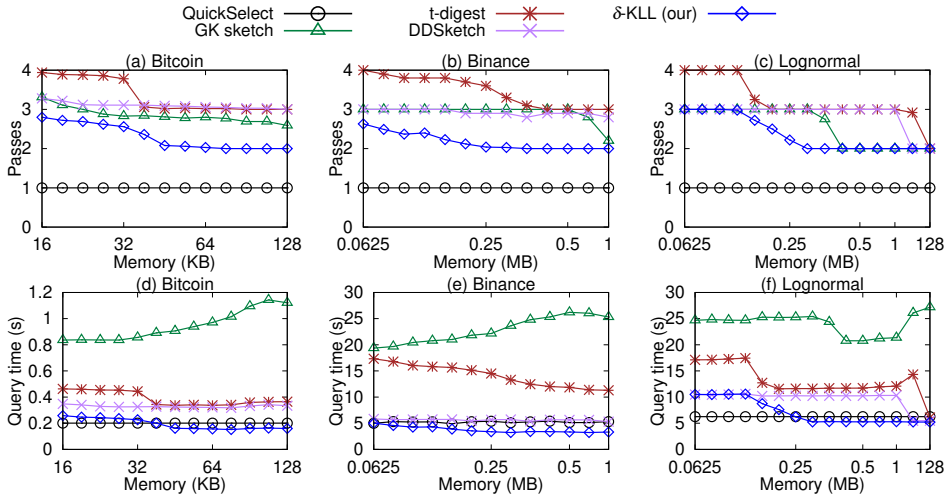


Fig. 17. Comparison by varying memory limit

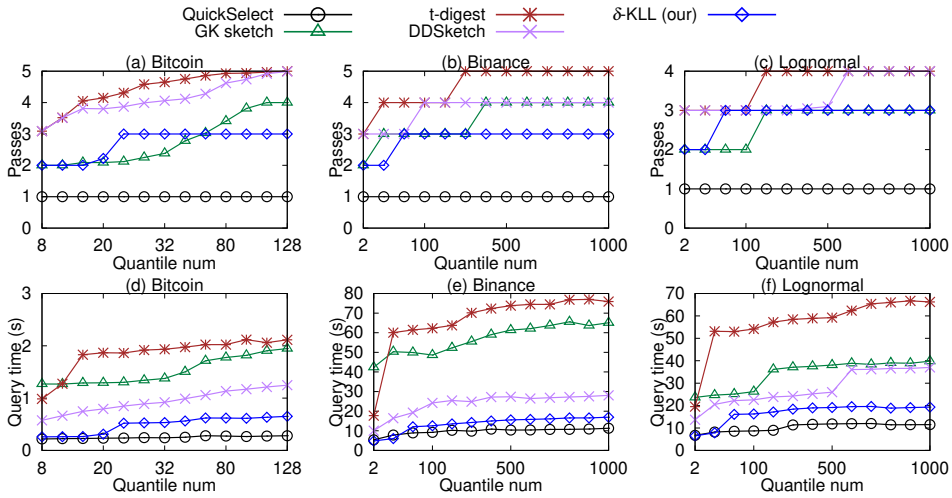


Fig. 18. Comparison by varying number of quantiles

pre-computed sketches (about 488KB), and thus mergeable methods do not utilize all memory in the first pass. (2) Failure is more likely to happen as more probabilistic filters are maintained. For the query time, our method outperforms multi-pass baselines and is comparable with multiple QuickSelect [37] when few quantiles are queried.

6.3.5 K Times Memory for Multiple Quantiles. In Apache IoTDB (and many other systems), the memory budget of a query is usually limited and thus equally distributed among multiple (K) quantiles. Nevertheless, we configure the system to allocate K times more memory for the query of K quantiles, and report the results in Figure 19. In this case, the memory budget for each quantile is not changed. Moreover, the shared pass (in Section 5.1.2) makes each quantile benefit from the total memory budget, and thus the required passes decrease as K increases.

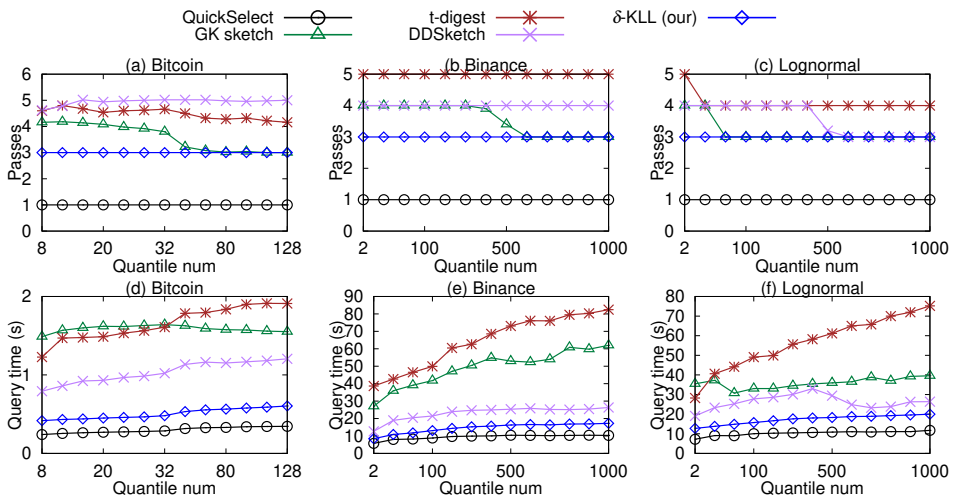


Fig. 19. Allocating K times memory for K quantiles

The time cost may increase and decrease. With more quantiles queried, the time cost may reduce as pass, while maintaining the larger shared sketch needs more time.

6.3.6 Extreme Quantiles. By varying the queried quantile ϕ in Figure 20, we provide the results of selecting certain extreme quantiles. Our method outperforms baselines when computing any quantile $\phi \in [0.01, 0.99]$ in all datasets. To compute quite extreme quantiles in some datasets, the baseline method DDSketch [33] may be better since it is specially designed for extreme quantiles of long-tailed distributions. Abnormal results, like 0.01-quantile in Figure 20(a) of Bitcoin, come from the specific nature of data. The answer 0.009 is a duplicated value taking 2% of the total data volume, which is good for methods except DDSketch.

6.4 Performance of System Deployment

6.4.1 Overhead of Determining Failure Probability. The overhead for δ evaluation is negligible, referring to the asymptotic time complexity analysis in Proposition 4.4. It is explained with more details at the end of Section 4.1.2. Nevertheless, we use an experiment in Figure 21 to evaluate the overhead for δ evaluation compared to that of sketching data by varying queried data size. It reports the relative time cost of determining the failure probability, compared to the total time cost of query. As shown, the overhead for evaluating and determining δ is below 2%. In particular, the percentage drops in both experiments. It is because more time is spent on sketching data as N grows. The cost of evaluating δ (analyzed in Section 4.1.2) decreases as memory (sketch size) grows.

6.4.2 Evaluation of Shared-pass. To evaluate the shared passes among quantiles, Figure 22 compares the results with/without sharing technique proposed in Section 5.1.2. The relative performance evaluates the selection of multiple quantiles with shared pass, compared to no shared pass. The runtime is improved more by the technique in determining numerous quantiles, since all quantiles benefit from the total memory budget in the shared pass.

6.5 Application of Exact Quantiles

In order to illustrate the scenarios of using exact quantiles rather than approximate estimation, we present some concrete applications in data partitioning, in Figure 23. Quantiles are often used

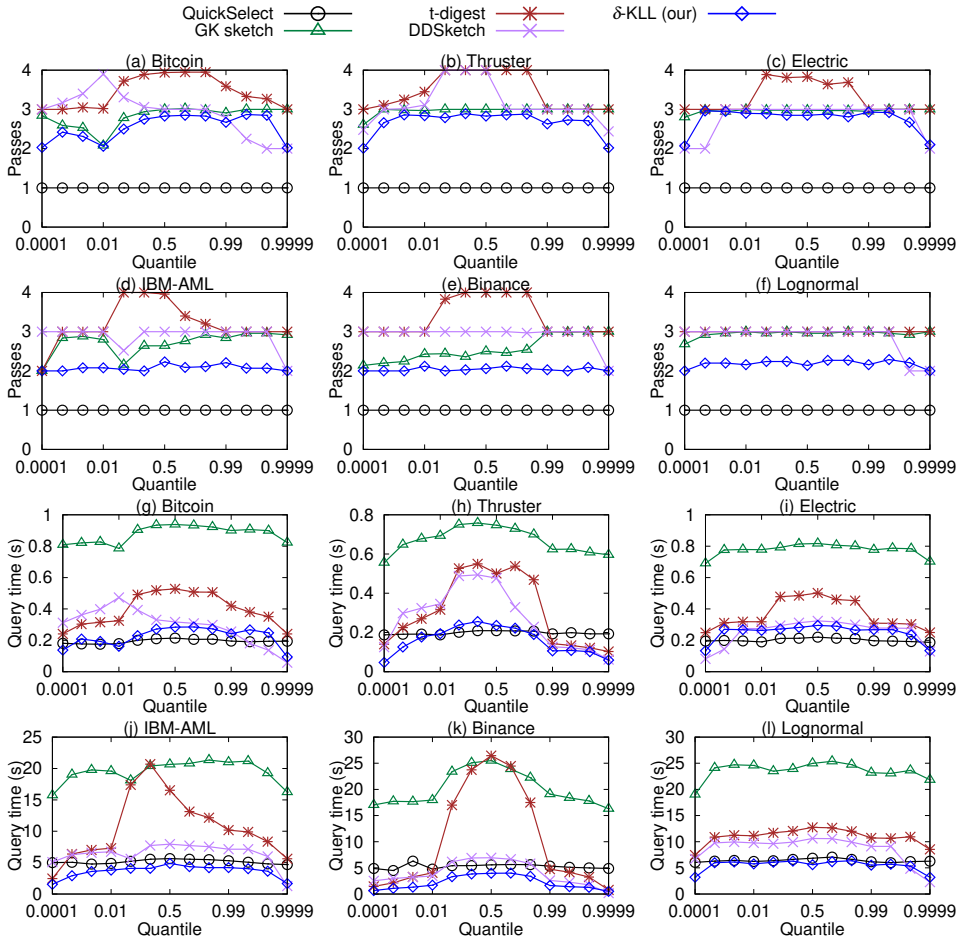


Fig. 20. Comparison by varying queried quantile

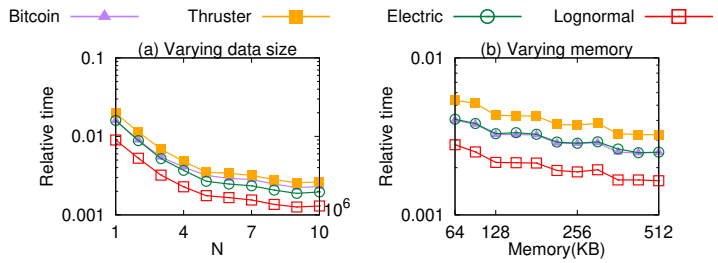


Fig. 21. Relative time cost of determining failure probability δ, compared to the total time cost of query

to partition data by value ranges, a.k.a. equi-depth histogram [22]. Exact equi-depth histograms (based on exact quantiles) can make sure all partitions having the same size, whereas approximate quantiles result in partitions varying in size.

In the application of parallel computing like MapReduce [13] tasks, the skewed partitioning results in skewed runtime of reducers and the slowest task dominates the overall job, as studied

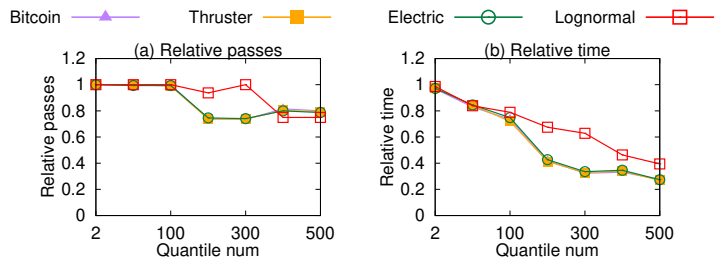


Fig. 22. Relative performance of selecting multiple quantiles with shared pass, compared to no shared pass

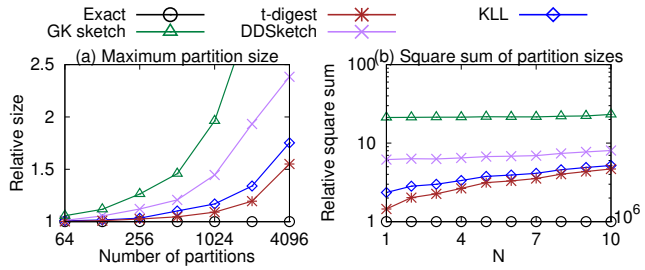


Fig. 23. Relative performances of approximate/exact quantiles in data partition applications for (a) parallel computing and (b) histogram sort

in [28]. We conduct a test in Figure 23(a) by varying the number of partitions for 1×10^8 values in the Lognormal dataset, and evaluate the performance by the maximum partition size, i.e., the bottleneck. As shown, the largest partition by approximate methods (t-digest [14], DDSketch [33], GKSketch [16] and KLL [26]) is over 1.5 times of the average size (the equal size returned by exact quantiles) when there are thousands of partitions. In this case, computing exact quantiles for data partitioning is worthwhile.

In the application of histogram sort, the sorting algorithm assigns n values to m partitions and sorts each partition. The implementation FlashSort [35] typically sets m to be $m = 0.1n$ and performs insertion sort on each partition. We conduct another test in Figure 23(b) by varying data size n in the non-uniform dataset IBM-AML [11], and evaluate the performance by the square sum of partition sizes (which represents the total number of comparisons in insertion sorting). As shown, the square sum by approximate quantiles relative to that by exact quantiles becomes worse as n grows. Thereby, exact equi-depth histogram is important for efficient FlashSort.

7 RELATED WORK

Here we introduce related quantile summaries and their potential in multi-pass selection.

7.1 Deterministic Sketches

The pioneering work about multi-pass selection by Munro et al. [34] implied a deterministic data sketch with formal guarantees. Manku et al. [32] developed the work of Munro et al. and proposed the MRL sketch for quantile estimation. The best-known deterministic quantile sketch is the GK sketch [16], which is not fully-mergeable [2] and proved to be optimal by matching the lower bound for streaming algorithms that can only compare items [7].

Other sketches perform arithmetic operations to provide good performance on specific data distributions. The representatives are DDSketch [33] and t-digest [6, 14, 15]. Both DDSketch and

the merging variant of t-digest do not apply any randomized operations, i.e., they are deterministic methods providing deterministic filters.

7.2 Randomized Sketches

Agarwal et al. [2] improved the MRL sketch [32] by introducing internal randomness, making the sketch unbiased and the rank error sub-Gaussian. Karnin et al. [26] presented the KLL sketch by improving the previous work with the idea in GK sketch and proved that the KLL sketch is asymptotically optimal. Our algorithm is implemented with the KLL sketch and keeps its mergeability, by recording the number of randomized compactions. Most recently, the ReqSketch [5] achieved the best-known asymptotic behavior in providing relative error approximation, i.e., more accurate estimates near extreme quantiles.

Recall that we mention the rank error of ReqSketch [5] and KLL [26] in Section 3.1, but not probabilistic filters of them. Indeed, probabilistic filters are first studied in this paper for determining exact quantiles. The difference is that [5] and [26] provide only approximate ranks without mentioning probabilistic filters (ranges) of quantiles.

The filter size analysis and pass estimation in our work are independent of quantiles. However, the compaction in [5] removes only some largest items in the compactor and makes the analyses for [5] depend on the quantile. It is thus not directly applicable to our proposal. Moreover, we use the randomized sketches of [26] but not the probabilistic filters of [26] which are proposed in Sec 3.1 of our study. The purpose of [26] is to obtain an approximate rank, whereas we aim to determine the ranges of exact quantiles by the proposed probabilistic filters.

8 CONCLUSIONS

In this paper, we study the determination of exact quantile with limited memory space by multiple passes of data. It gradually shrinks the range of the queried quantile, i.e., filter, till the remaining data are small enough to rank in memory. Rather than the conservative deterministic filter, we use randomized summaries to shrink more aggressively the ranges of the result. The probabilistic filter obtained from the randomized sketch has a failure probability of the quantile not being in the range, each of which incurs an extra pass for determining the exact quantile. Our solution is to estimate the number of passes required for the given data size and memory limit, and choose a failure probability with the minimum estimated passes. The method can terminate in P passes with $1-\delta$ confidence, storing $O(N^{1/P}\log^{\frac{P-1}{2P}}(\frac{1}{\delta}))$ items (Proposition 4.5), close to the lower bound $\Omega(N^{1/P})$ [34] for a fixed δ. The space bound is asymptotically better than that with GK sketch [16], the optimal deterministic comparison-based sketch [7], known for $P = 2$ passes. The proposed method has been deployed as the quantile function in an open-source time-series database Apache IoTDB. The implementation supports query processing of multiple quantiles at a time, to share the passes of data, and pre-computation of the randomized summaries for immutable SSTables in the LSM-tree based storage. Experiments on real and synthetic datasets demonstrate the superiority of our proposal.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (92267203, 62021002, 62072265, 62232005), the National Key Research and Development Plan (2021YFB3300500), and Beijing Key Laboratory of Industrial Big Data System and Application. Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

REFERENCES

[1] Ildar Absalyamov, Michael J. Carey, and Vassilis J. Tsotras. 2018. Lightweight Cardinality Estimation in LSM-based Systems. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 841–855. https://doi.org/10.1145/3183713.3183761

[2] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable summaries. *ACM Trans. Database Syst.* 38, 4 (2013), 26. https://doi.org/10.1145/2500128

[3] Zhiwei Chen, Shaoxu Song, Ziheng Wei, Jingyun Fang, and Jiang Long. 2021. Approximating Median Absolute Deviation with Bounded Error. *Proc. VLDB Endow.* 14, 11 (2021), 2114–2126. https://doi.org/10.14778/3476249.3476266

[4] Experiment Code and Data. 2023. https://github.com/thssdb/exact-quantile.

[5] Graham Cormode, Zohar S. Karnin, Edo Liberty, Justin Thaler, and Pavel Veselý. 2021. Relative Error Streaming Quantiles. In *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo (Eds.). ACM, 96–108. https://doi.org/10.1145/3452021.3458323

[6] Graham Cormode, Abhinav Mishra, Joseph Ross, and Pavel Veselý. 2021. Theory meets Practice at the Median: A Worst Case Comparison of Relative Error Quantile Algorithms. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 2722–2731. https://doi.org/10.1145/3447548.3467152

[7] Graham Cormode and Pavel Veselý. 2020. A Tight Lower Bound for Comparison-Based Quantile Summaries. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2020, Portland, OR, USA, June 14-19, 2020*, Dan Suciu, Yufei Tao, and Zhewei Wei (Eds.). ACM, 81–93. https://doi.org/10.1145/3375395.3387650

[8] Bitcoin Dataset. 2018. https://www.kaggle.com/shiheyingzhe/bitcoin-transaction-data-from-2009-to-2018.

[9] Binance Dataset. 2022. https://www.kaggle.com/datasets/jorijnsmit/binance-full-history.

[10] Electric Dataset. 2022. https://www.kaggle.com/datasets/jeanmidev/smart-meters-in-london.

[11] IBM Dataset. 2023. https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml.

[12] Thruster Dataset. 2022. https://www.kaggle.com/datasets/patrickfleith/spacecraft-thruster-firing-tests-dataset.

[13] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (2008), 107–113. https://doi.org/10.1145/1327452.1327492

[14] Ted Dunning. 2021. The t-digest: Efficient estimates of distributions. *Softw. Impacts* 7 (2021), 100049. https://doi.org/10.1016/j.simpa.2020.100049

[15] Ted Dunning and Otmar Ertl. 2019. Computing Extremely Accurate Quantiles Using t-Digests. *CoRR* abs/1902.04023 (2019). arXiv:1902.04023 http://arxiv.org/abs/1902.04023

[16] Michael Greenwald and Sanjeev Khanna. 2001. Space-Efficient Online Computation of Quantile Summaries. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, Sharad Mehrotra and Timos K. Sellis (Eds.). ACM, 58–66. https://doi.org/10.1145/375663.375670

[17] Joseph M. Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar. 2012. The MADlib Analytics Library or MAD Skills, the SQL. *Proc. VLDB Endow.* 5, 12 (2012), 1700–1711. https://doi.org/10.14778/2367502.2367510

[18] C. A. R. Hoare. 1962. Quicksort. *Comput. J.* 5, 1 (1962), 10–15. https://doi.org/10.1093/comjnl/5.1.10

[19] Configuration in Apache IoTDB. 2023. https://iotdb.apache.org/UserGuide/Master/Reference/Common-Config-Manual.html.

[20] Document in Apache IoTDB. 2023. https://iotdb.apache.org/UserGuide/Master/Operators-Functions/Data-Profiling.html#quantile.

[21] Implementation in Apache IoTDB. 2023. https://github.com/apache/iotdb/tree/research/exact-quantile.

[22] Yannis E. Ioannidis. 2003. The History of Histograms (abridged). In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer (Eds.). Morgan Kaufmann, 19–30. https://doi.org/10.1016/B978-012722442-8/50011-2

[23] Apache IoTDB. 2023. https://iotdb.apache.org/.

[24] Nikita Ivkin, Edo Liberty, Kevin J. Lang, Zohar S. Karnin, and Vladimir Braverman. 2019. Streaming Quantiles Algorithms with Small Space and Update Time. *CoRR* abs/1907.00236 (2019). arXiv:1907.00236 http://arxiv.org/abs/1907.00236

[25] Yuyuan Kang, Xiangdong Huang, Shaoxu Song, Lingzhe Zhang, Jialin Qiao, Chen Wang, Jianmin Wang, and Julian Feinauer. 2022. Separation or Not: On Handing Out-of-Order Time-Series Data in Leveled LSM-Tree. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 3340–3352. https://doi.org/10.1109/ICDE53745.2022.00315

[26] Zohar S. Karnin, Kevin J. Lang, and Edo Liberty. 2016. Optimal Quantile Approximation in Streams. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, Irit Dinur (Ed.). IEEE Computer Society, 71–78. https://doi.org/10.1109/FOCS.2016.17
[27] J. Kiefer. 1953. Sequential minimax search for a maximum. *Proc. Amer. Math. Soc.* 4 (01 1953), 502–505.
[28] YongChul Kwon, Magdalena Balazinska, Bill Howe, and Jerome A. Rolia. 2011. A Study of Skew in MapReduce Applications. https://api.semanticscholar.org/CorpusID:134490
[29] Yuanzhen Li, Shengjie Zheng, Zi-Xin Tan, Tuo Cao, Fei Luo, and Chunxia Xiao. 2023. Self-Supervised Monocular Depth Estimation by Digging into Uncertainty Quantification. *J. Comput. Sci. Technol.* 38, 3 (2023), 510–525. https://doi.org/10.1007/S11390-023-3088-Y
[30] Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. 2020. MCUNet: Tiny Deep Learning on IoT Devices. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html
[31] Chen Luo and Michael J. Carey. 2020. LSM-based storage techniques: a survey. *VLDB J.* 29, 1 (2020), 393–418. https://doi.org/10.1007/s00778-019-00555-y
[32] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. 1999. Random Sampling Techniques for Space Efficient Online Computation of Order Statistics of Large Datasets. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh (Eds.). ACM Press, 251–262. https://doi.org/10.1145/304182.304204
[33] Charles Masson, Jee E. Rim, and Homin K. Lee. 2019. DDSketch: A Fast and Fully-Mergeable Quantile Sketch with Relative-Error Guarantees. *Proc. VLDB Endow.* 12, 12 (2019), 2195–2205. https://doi.org/10.14778/3352063.3352135
[34] J. Ian Munro and Mike Paterson. 1980. Selection and Sorting with Limited Storage. *Theor. Comput. Sci.* 12 (1980), 315–323. https://doi.org/10.1016/0304-3975(80)90061-4
[35] Karl-Dietrich Neubert. 1998. The Flashsort1 Algorithm. *Dr. Dobb's Journal* (02 1998), 123–125.
[36] Patrick E. O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth J. O'Neil. 1996. The Log-Structured Merge-Tree (LSM-Tree). *Acta Informatica* 33, 4 (1996), 351–385. https://doi.org/10.1007/s002360050048
[37] Helmut Prodinger. 1995. Multiple Quickselect - Hoare's Find Algorithm for Several Elements. *Inf. Process. Lett.* 56, 3 (1995), 123–129. https://doi.org/10.1016/0020-0190(95)00150-B
[38] InfluxDB Quantile. 2023. https://github.com/influxdata/flux/blob/8ad36639ede4826242455389fff4810adfc4e884/stdlib/universe/quantile.go#L404.
[39] PostgreSQL Quantile. 2023. https://www.postgresql.org/docs/current/functions-aggregate.html.
[40] Supplementary. 2023. https://github.com/thssdb/exact-quantile/blob/main/Appendix.pdf.
[41] Chen Wang, Jialin Qiao, Xiangdong Huang, Shaoxu Song, Haonan Hou, Tian Jiang, Lei Rui, Jianmin Wang, and Jiaguang Sun. 2023. Apache IoTDB: A Time Series Database for IoT Applications. *Proc. ACM Manag. Data* 1, 2 (2023), 195:1–195:27. https://doi.org/10.1145/3589775
[42] Wolfgang Weiss, Víctor Juan Expósito Jiménez, and Herwig Zeiner. 2020. Dynamic Buffer Sizing for Out-of-order Event Compensation for Time-sensitive Applications. *ACM Trans. Sens. Networks* 17, 1 (2020), 1:1–1:23. https://doi.org/10.1145/3410403
[43] Haitao Yuan and Guoliang Li. 2021. A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. *Data Sci. Eng.* 6, 1 (2021), 63–85. https://doi.org/10.1007/s41019-020-00151-z
[44] Kangfei Zhao, Zongyan He, Jeffrey Xu Yu, and Yu Rong. 2023. Learning with Small Data: Subgraph Counting Queries. *Data Sci. Eng.* 8, 3 (2023), 292–305. https://doi.org/10.1007/S41019-023-00223-W

Received July 2023; revised October 2023; accepted November 2023