Moment-Based Quantile Sketches for Efficient High Cardinality Aggregation Queries (Extended)

Edward Gan, Jialin Ding, Kai Sheng Tai, Vatsal Sharan, Peter Bailis
Stanford InfoLab

ABSTRACT

Interactive analytics increasingly involves querying for quantiles over specific sub-populations of high cardinality datasets. Data processing engines such as Druid and Spark use mergeable summaries to estimate quantiles on these large datasets, but summary merge times are a bottleneck during high-cardinality aggregation. We show how a compact and efficiently mergeable quantile sketch can support aggregation workloads. This data structure, which we refer to as the moments sketch, operates with a small memory footprint (200 bytes) and computationally efficient (50ns) merges by tracking only a set of summary statistics, notably the sample moments. We demonstrate how we can efficiently estimate quantiles using the method of moments and the maximum entropy principle, and show how the use of a cascade further improves query time for threshold predicates. Empirical evaluation shows that the moments sketch can achieve less than 1 percent quantile error with 15× less overhead than comparable summaries, improving end query time in the MacroBase engine by up to 7× and the Druid engine by up to 60×.

PVLDB Reference Format:
Edward Gan, Jialin Ding, Kai Sheng Tai, Vatsal Sharan, and Peter Bailis. Moment-Based Quantile Sketches for Efficient High Cardinality Aggregation Queries. *PVLDB*, 11 (5): xxxx-yyyy, 2018.
DOI: https://doi.org/TBD

1. INTRODUCTION

Performing interactive multi-dimensional analytics over data from sensors, devices, and servers increasingly requires computing aggregate statistics for specific subpopulations and time windows [4, 29, 65]. In applications such as A/B testing [38, 42], exploratory data analysis [8, 74], and operations monitoring [2, 14], analysts frequently perform aggregation queries to better understand how particular user cohorts, device types, and feature flags are behaving. In particular, computing quantiles over these subpopulations is an essential part of debugging and real-time monitoring workflows [26].

Proceedings of the VLDB Endowment, Vol. 11, No. 5

DOI: https://doi.org/TBD

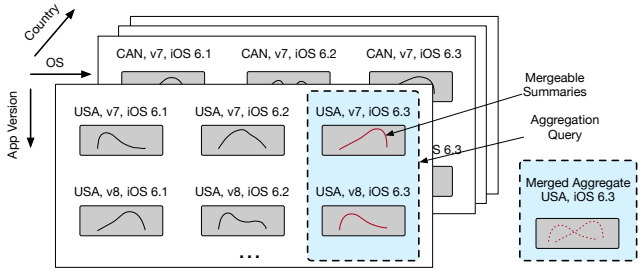

Figure 1: Given a data cube with pre-aggregated summaries, we can compute roll-ups along specific dimensions by merging the relevant summaries. Efficiently mergeable summaries enable scalable aggregations.

As an example of this quantile-driven analysis, our collaborators on a Microsoft application monitoring team collect billions of telemetry events daily from millions of heterogeneous mobile devices. Each device tracks multiple metrics including request latency and memory usage, and is associated with dimensional metadata such as application version and hardware model. Engineers issue quantile queries on a Druid-like [82] in-memory data store, aggregating across different dimensions to monitor their application (e.g., examine memory trends across device types) and debug regressions (e.g., examine tail latencies across versions). Querying for a single percentile in this deployment can require aggregating hundreds of thousands of dimension value combinations.

When users wish to examine the quantiles of specific slices of a dataset, OLAP engines such as Druid and Spark support computing approximate quantiles using compressed representations (*summaries*) of the data values [39, 67, 82, 83]. By pre-aggregating a summary for each combination of dimension values, Druid and similar engines can reduce query times and memory usage by operating over the relevant summaries directly, effectively constructing a data cube [33, 65].

Figure 1 illustrates how roll-up queries can be processed by combining pre-aggregated *mergeable* [3] summaries without scanning over the raw data. Given a time interval and a metric with d associated dimensions, Druid maintains one summary for each d-tuple of dimension values. These summaries are kept in RAM across a number of nodes, with each node scanning and merging relevant summaries to process subsets of the data specified by a user query.

More concretely, a Druid-like data cube in our Microsoft

deployment with 6 dimension columns, each with 10 distinct values, is stored as a set of up to 10^6 summaries per time interval. On this cube, computing the 99-th percentile latency for a specific app version can require 100,000 merges, or even more for aggregation across complex time ranges. When there are a limited number of dimensions but enormous data volumes, it is cheaper to maintain these summaries than scan over billions of raw datapoints.

Many quantile summaries support the merge operation [3, 28,34], but their runtime overheads can lead to severe performance penalties on high-cardinality datasets. Based on our experiments (Section 6.2.1), one million 1KB GK-sketches [34] require more than 3 seconds to merge sequentially, limiting the types of queries users can ask interactively. The merging can be parallelized, but additional worker threads still incur additional coordination and resource usage overheads. Materialized views [37, 44, 51, 61], sliding window sketches [25], and dyadic intervals can also reduce this overhead. However, dyadic intervals only apply to ordered dimensions, and the real-time ingest requirements at Microsoft make maintaining multiple materialized views prohibitively expensive, so merge time remains a relevant bottleneck.

In this paper, we enable interactive quantile queries over high-cardinality aggregates by introducing a compact and efficiently mergeable quantile sketch and associated quantile estimation routines. We draw a connection between the classic *method of moments* for parameter estimation in statistics [79] and the need for efficient summary data structures. We show that storing the sample moments $\mu_i = \frac{1}{n}\sum x^i$ and log-moments $\nu_i = \frac{1}{n}\sum \log^i(x)$ can enable accurate quantile estimation over a range of real-world datasets while utilizing fewer than 200 bytes of memory and incurring merge times of less than 50 nanoseconds. In the context of quantile estimation, we refer to our proposed summary data structure as the *moments sketch*.

While constructing the moments sketch is straightforward, the inverse problem of estimating quantiles from the summary is more complex. The statistics in a moments sketch provide only loose constraints on the distribution of values in the original dataset: many distributions might match the moments of a moments sketch but fail to capture the dataset structure. Therefore, we make use of the *principle of maximum entropy* [41] to compute a "least-biased" quantile estimate for a moments sketch. On continuous real-valued datasets, we find that this approach yields more accurate estimates than alternative methods, achieving $\epsilon \leq 1\%$ error with 200 bytes of memory. To achieve this, we also describe a series of practical optimizations to standard entropy maximization that allow us to compute quantile estimates in under 1 millisecond on a range of real-world datasets.

These query times make the moments sketch a suitable summary when many merges (hundreds of thousands) are required, memory per-summary may be limited to less than 1 kilobyte, and $\epsilon = .01$ error is acceptable. The moments sketch and our maximum entropy estimate is most useful in datasets without strong discretization and when very small $< 10^{-4}$ error is not required. The maximum entropy principle is less accurate when there are clusters of discrete values in a dataset (See Appendix A.4 in [30]), and numerical stability (See Section 4.3) limits the minimum achievable error using this approach.

Moving beyond simple quantile queries, many complex queries depend on the quantile estimates of multiple subpopulations. For example, data exploration systems such as MacroBase [8] are interested in finding all subpopulations that match a given threshold condition (e.g., subpopulations where the 95th percentile latency is greater than the global 99th percentile latency). Given a large number of subpopulations, the cost of millisecond-level quantile estimates on thousands of subgroups will accumulate. Therefore, to support threshold queries over multiple populations, we extend our quantile estimator with a *cascade* [75], or sequence of increasingly precise and increasingly expensive estimates based on bounds such as the Markov inequalities. For queries with threshold conditions, the cascades dramatically reduce the overhead of quantile estimation in a moments sketch, by up to 25×.

We implement the moments sketch both as a reusable library and as part of the Druid and MacroBase analytics engines. We empirically compare its accuracy and efficiency with alternative mergeable quantile summaries on a variety of real-world datasets. We find that the moments sketch offers 16 to 50× faster merge times than alternative summaries with comparable accuracy. This enables 15 to 50× faster query times on real datasets. Moreover, the moments sketch enables up to 7× faster analytics queries when integrated with MacroBase and 60× faster end-to-end queries when integrated with Druid.

In summary, we make the following contributions:

- We illustrate how statistical moments are useful as efficient mergeable quantile sketches in aggregation and threshold-based queries over high-cardinality data.
- We demonstrate how statistical and numerical techniques allow us to solve for accurate quantile estimates in less than 1 ms, and show how the use of a cascade further improves estimation time on threshold queries by up to 25×.
- We evaluate the use of moments as quantile summaries on a variety of real-world datasets and show that the moments sketch enables 15 to 50× faster query times in isolation, up to 7× faster queries when integrated with MacroBase and up to 60× faster queries when integrated with Druid over comparably-accurate quantile summaries.

The remainder of this paper proceeds as follows. In Section 2, we discuss related work. In Section 3, we review relevant background material. In Section 4, we describe the proposed moments sketch. In Section 5, we describe a cascade-based approach for efficiently answering threshold-based queries. In Section 6, we evaluate the moments sketch in a series of microbenchmarks. In Section 7, we evaluate the moments sketch as part of the Druid and MacroBase systems, and measure its performance in sliding window and low-precision environments. We conclude in Section 8. Supplemental details in Appendix A are available in an extended technical report [30].

2. RELATED WORK

High-performance aggregation. The aggregation scenarios in Section 1 are found in many existing streaming data systems [8, 16, 24, 65, 82], as well as data cube [33, 69],

data exploration [2], and visualization [17] systems. In particular, these systems are can perform interactive aggregations over time windows and along many cube dimensions, motivating the design of our sketch. Many of these systems use approximate query processing [4,35,59] and use sampling and summaries to improve query performance, but do not address data structures specific to quantiles. We believe the moments sketch serves as a useful primitive in these engines.

Sensor networking is a rich source of algorithms for heavily resource-constrained settings. Sensor network aggregation systems [53] support large scale roll-ups, but seminal work in this area is largely focused on the complementary problem of communication plans over a network [21,45,54]. Mean, min, max, and standard deviation in particular are used in [53] as functions amenable to computation-constrained environments, but the authors do not consider higher moments or their application to quantile estimation.

Several database systems make use of summary statistics in general-purpose analytics. Muthukrishan et al [60] observe that the moments are a convenient statistic in streaming settings and use it to fill in missing integers. Data Canopy [78] uses first and second moments as an efficiently mergeable statistic for computing standard deviations and linear regressions. Similarly, systems on probabilistic data cubes such as [81] use the first and second moments to prune queries over cube cells that store distributions of values. In addition, many methods use compressed data representations to perform statistical analyses such as linear regression, logistic regression, and PCA [19,63,70,80]. We are not aware of prior work utilizing higher moments to efficiently estimate quantiles for high-dimensional aggregation queries.

Quantile summaries. There are a variety of summary data structures for the ϵ-approximate quantile estimation problem [18, 23, 34, 71]. Some of these summaries assume values from a fixed universe [23,71], while others operate using only comparisons [3, 34]. Our proposed moments sketch and others [12, 28] operate on real values. Agarwal et al. [3] provide the initial motivation for mergeable summaries, as well as a proposed mergeable quantile sketch. The authors in [52, 77] benchmark a variety of quantile summaries but do not directly evaluate merge time. Zhuang [84] evaluates merge performance of a variety of quantile summaries in a distributed setting, finding the `Random` summary to be the fastest. To our knowledge we are the first to introduce and evaluate the moments sketch for fast merge times and low space overhead.

Method of moments. The method of moments is a well-established statistical technique for estimating the parameters of probability distributions [79]. The main idea behind this approach is that the parameters of a distribution of interest P can be related to the expectations of functions of the random variable $X \sim P$. As a general method for consistent statistical parameter estimation, the method of moments is used across a wide range of fields, including econometrics [36], physics [32,57], and machine learning [7,11,43]. In this work, we demonstrate how the method of moments, applied in conjunction with practical performance optimizations, can scale to support real-world latency-sensitive query processing workloads.

Maximum entropy principle. The maximum entropy principle prescribes that one should select the least informative distribution that is consistent with the observed data. In the database community, this principle has been applied to estimating cardinality [73] and predicate selectivity [55]. Mead and Papanicolaou [57] apply the maximum entropy principle to the problem of estimating distributions subject to moment constraints; follow-up work proposes the use of Chebyshev polynomials for stability [10, 72] and faster approximation algorithms [9], though we have not seen any practical implementations suitable for use in a database. The maximum entropy principle is also used in machine learning, notably in the context of *maximum entropy models* [13]. For example, in natural language processing, maximum entropy models are a popular choice for tasks such as text classification [62] and sequence tagging [48].

3. BACKGROUND

In this section, we review the approximate quantile estimation problem, mergeable quantile summaries, and our target query cost model.

3.1 Quantile Queries

Given a dataset D with n elements, for any $\phi \in (0, 1)$, the *ϕ-quantile* of D is the item $x \in D$ with rank $r(x) = \lfloor \phi n \rfloor$, where the rank of an element x is the number of elements in D smaller than x.

An ϵ-approximate ϕ-quantile is an element with rank between $(\phi - \epsilon)n$ and $(\phi + \epsilon)n$ [3]. Given an estimated ϕ-quantile q_ϕ, we can also define its *quantile error* ε [52] as the following:

$$\varepsilon = \frac{1}{n} \left| \text{rank}\left(q_\phi\right) - \lfloor \phi n \rfloor \right|, \tag{1}$$

such that an ϵ-approximate quantile has error at most ε. For example, given a dataset $D = \{1, \ldots, 1000\}$, an estimate $q_{0.5} = 504$ for the $\phi = 0.5$ quantile would have error $\varepsilon = 0.003$. In this paper, we consider datasets D represented by collections of real numbers $D \subset \mathbb{R}$.

Quantile summaries are data structures that provide approximate quantile estimates for a dataset given space sublinear in n. These summaries usually have a parameter k_ϵ that trades off between the size of the summary and the accuracy of its estimates. An ϵ-approximate quantile summary provides ϵ approximate ϕ-quantiles, where ϵ can be a function of space usage and the dataset [18, 23, 34, 71].

3.2 Mergeable Summaries

Agarwal et al. [3] introduce the concept of *mergeability* to accurately combine summaries in distributed settings. Formally, for a summary with parameter k_ϵ, we use $S(D, k_\epsilon)$ to denote a valid summary for a dataset D. For any pair of datasets D_1 and D_2, the summarization routine S is *mergeable* if there exists an algorithm (i.e., the "merge" procedure) that produces a combined summary

$$S(D_1 \uplus D_2, k_\epsilon) = \text{merge}(S(D_1, k_\epsilon), S(D_2, k_\epsilon))$$

from any two input summaries, where $\uplus$ denotes multiset addition.

Intuitively, a summary is mergeable if there is no accuracy cost to combining pre-computed summaries compared with constructing a summary on the raw data. Thus, mergeable summaries are *algebraic* aggregate functions in the data cube literature [33]. As an example, an equi-depth histogram [22] on its own is not mergeable because there is no way

to accurately combine two overlapping histogram buckets without access to additional data.

Mergeable summaries can be incorporated naturally into a variety of distributed systems. In the MapReduce paradigm, a "map" function can construct summaries over shards while a "reduce" function merges them to summarize a complete dataset [3]. In the GLADE system [68], mergeable summaries are an example of a *Generalized Linear Aggregate* (GLA), a user-defined computation that can be incrementally aggregated across nodes.

3.3 Query Model

As described in Section 1, we focus on improving the performance of quantile queries over aggregations on high cardinality datasets. Given a dataset with d categorical dimensions, we consider data cubes that maintain summaries for every d-way dimension value tuple as one natural setting for high performance aggregations, and many other settings are also applicable [78]. In these settings, query time is heavily dependent on the number of merges and the time per merge.

We consider two broad classes of queries in this paper. First, *single quantile* queries ask for quantile estimates for a single specified population. For example, we can query the 99th percentile of latency over the last two weeks for a given version of the application:

```
SELECT percentile(latency, 99) FROM requests
WHERE time > date_sub(curdate(), 2 WEEK)
AND app_version = "v8.2"
```

To process this query in time t_{query}, we would need to merge n_{merge} summaries, each with runtime overhead t_{merge}, and then estimate the quantile from the merged summary with runtime cost t_{est}. This results in total query time:

$$t_{\text{query}} = t_{\text{merge}} \cdot n_{\text{merge}} + t_{\text{est}}. \quad (2)$$

We evaluate the different regimes where queries are bottlenecked on merges and estimation in Figure 6 in Section 6.2.1: merge time begins to dominate at around $n_{\text{merge}} \geq 10^4$.

We also consider *threshold* queries which are conditioned on sub-groups or windows with percentiles above a specified threshold. For example, we may be interested in combinations of application version and hardware platform for which the 99th percentile latency exceeds 100ms:

```
SELECT app_version, hw_model,
    PERCENTILE(latency, 99) as p99
FROM requests
GROUP BY app_version, hw_model
HAVING p99 > 100
```

Such queries are very useful for debugging and data exploration [8], but have additional runtime cost that depends on the number of groups n_{groups} since t_{est} can be significant when one is searching for high quantiles over thousands of sub-groups. This results in total query time:

$$t_{\text{query}} = t_{\text{merge}} \cdot n_{\text{merge}} + t_{\text{est}} \cdot n_{\text{groups}}. \quad (3)$$

4. THE MOMENTS SKETCH

In this section, we describe how we perform quantile estimation using the moments sketch. First, we review the summary statistics stored in the moments sketch and describe how they comprise an efficiently mergeable sketch. Second, we describe how we can use the method of moments and the maximum entropy principle to estimate quantiles from the moments sketch, with details on how we resolve practical difficulties with numerical stability and estimation time. We conclude with a discussion of theoretical guarantees on the approximation error of quantiles estimated from the sketch.

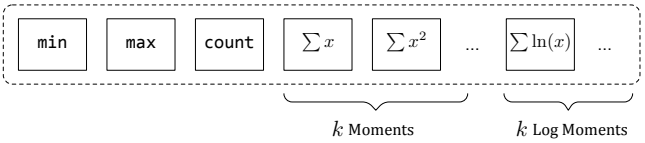

Figure 2: The moments sketch is an array of floating point values.

Algorithm 1: Moments sketch operations

input: number of moments k
function `Init`(x)
  $x_{\min}, x_{\max} \leftarrow \infty, -\infty$
  $\vec{\mu}, \vec{\nu} \leftarrow \vec{0}, \vec{0}$
  $n \leftarrow 0$
function `Accumulate`(x)
  $x_{\min}, x_{\max} \leftarrow \min\{x, x_{\min}\}, \max\{x, x_{\max}\}$
  $n \leftarrow n + 1$
  for $i \in \{1, \ldots, k\}$ **do**
    $\mu_i \leftarrow \frac{n-1}{n}\mu_i + \frac{1}{n}x^i$ ▷ Standard moments
    if $x > 0$ **then**
      $\nu_i \leftarrow \frac{n-1}{n}\nu_i + \frac{1}{n}\log^i(x)$ ▷ Log-moments
function `Merge`(o) ▷ o another sketch
  $x_{\min} \leftarrow \min\{o.x_{\min}, x_{\min}\}$
  $x_{\max} \leftarrow \max\{o.x_{\max}, x_{\max}\}$
  $\vec{\mu}, \vec{\nu} \leftarrow \vec{\mu} + o.\vec{\mu}, \vec{\nu} + o.\vec{\nu}$

4.1 Moments Sketch Statistics

The moments sketch of a dataset D is a set of floating point values: the minimum value $x_{\min}$, the maximum value $x_{\max}$, the count n, the sample moments $\mu_i = \frac{1}{n}\sum_{x \in D} x^i$ and the sample logarithmic moments $\nu_i = \frac{1}{n}\sum_{x \in D} \log^i(x)$ for $i \in \{1, \ldots, k\}$ (Figure 2). The moments sketch has an integer parameter k, which describes the highest power used in the moments. We refer to k as the *order* of a moments sketch. Each sample moment provides additional information about the distribution, so higher-order moments sketches are more precise but have higher space and computation time overheads.

The moments sketch supports a number of basic operations: `init` creates an empty sketch, `accumulate` updates the sketch via point-wise additions, and `merge` updates the sketch by merging it with another moments sketch. One can construct a moments sketch over a dataset using either `accumulate` or `merge`. When accumulating elements pointwise, we update the minimum and maximum, then add to the counts and moments. As an implementation detail, we can accumulate the unscaled sums $\sum x^i$ and $\sum \log^i(x)$ instead of the μ_i, ν_i. We merge two moments sketches by combining the minimum, maximum, count, and the moments via comparison and potentially vectorized addition. This merge operation preserves the property that an moments sketch constructed using only `accumulate` is identical (up to floating point precision) to an moments sketch constructed from merging existing sketches of partitions of the data, so there is no accuracy loss in pre-aggregating the data. Pseudocode for these operations is provided in Algorithm 1. In order

to provide final answers to queries moments sketch additionally supports quantile estimation routines described in Section 4.2. The moments sketch thus supports all of the basic user-defined aggregate operations [20, 68] and can be incorporated into data systems using this API.

Log-moments. The moments sketch records logarithmic moments (log-moments) in order to recover better quantile estimates for long-tailed datasets. In particular, taking the logarithm of data points is useful when values in the dataset can vary over several orders of magnitude. In general, when updating a moments sketch in a streaming manner or when maintaining multiple moments sketches in a distributed setting, we cannot know *a priori* whether standard moments or log-moments are more appropriate for the given dataset. Therefore, our default approach is to store both sets of moments up to the same order k. Given additional prior knowledge of the data distribution, we may choose to maintain a moments sketch with only a single set of moments.

Data points with negative values pose a potential problem for the log-moments since the logarithm is undefined for these points. There are several strategies for addressing this, including storing separate sums for positive and negative values and shifting all values so that they are positive. In this paper, we adopt the simple approach of ignoring the log sums when there are any negative values, and computing estimates using the remaining statistics.

Remark on pathological distributions. The moments of certain "pathological" distributions may be undefined; for example, the Cauchy distribution $f(x) = \pi^{-1}\left(1+x^2\right)^{-1}$ does not have finite moments of any order. However, the moments sketch tracks the moments of an empirical dataset, which are always well-defined. This suits our goal of estimating quantiles of the dataset, rather than an underlying distribution.

4.2 Estimating Quantiles

Method of moments. To estimate quantiles from a moments sketch, we apply the *method of moments* [7, 11, 43, 79] to construct a distribution $f(x)$ whose moments match those recorded in the sketch. Specifically, given a moments sketch with minimum $x_{\min}$ and maximum $x_{\max}$, we solve for a pdf $f(x)$ supported on $[x_{\min}, x_{\max}]$ such that $\int_{x_{\min}}^{x_{\max}} x^i f(x)\,dx = \mu_i$ and $\int_{x_{\min}}^{x_{\max}} \log^i(x) f(x)\,dx = \nu_i$. We can then report the quantiles of $f(x)$ as estimates for the quantiles of the dataset.

In general, a finite set of moments does not uniquely determine a distribution [5]. That is, there are often many possible distributions with varying quantiles that each match a given set of sample moments. Therefore, we must disambiguate between them.

Maximum entropy. In this work, we use the *principle of maximum entropy* [41] to select a unique distribution that satisfies the given moment constraints. Intuitively, the differential Shannon entropy H of a distribution with pdf $f(x)$, defined as $H[f] = -\int_{\mathcal{X}} f(x)\log f(x)\,dx$, is a measure of the degree of *uninformativeness* of the distribution. For example, a uniform distribution has higher entropy than a point mass distribution. Thus, the maximum entropy distribution can be seen as the distribution that encodes the *least* additional information about the data beyond that captured by the given moment constraints.

Applying the maximum entropy principle to the moments sketch, we estimate quantiles by solving for the pdf f that maximizes the entropy while matching the moments in the sketch. Following, we estimate quantiles using numeric integration and the Brent's method for root finding [64] .

In practice, we find that the use of maximum entropy distributions yields quantile estimates with comparable accuracy to alternative methods on a range of real-world datasets, unless the datasets are more discrete than continuous. We discuss our empirical results further in Section 6.2.3.

Optimization. We now describe how to solve for the maximum entropy distribution f. We trade off between accuracy and estimation time by solving for f subject to the first k_1 standard moments and k_2 log-moments stored in the sketch; incorporating more moments leads to more precise estimates but more computationally expensive estimation. As previously noted, for datasets with non-positive values (i.e., $x_{\min} \leq 0$), we set $k_2 = 0$. Therefore, our goal is to find the solution f to the following optimization problem:

$$\begin{aligned} \underset{f \in \mathcal{F}[x_{\min}, x_{\max}]}{\text{maximize}} \; & H[f] \qquad (4) \\ \text{subject to} \; & \int_{x_{\min}}^{x_{\max}} x^i f(x)\,dx = \mu_i, \qquad i \in \{1, \ldots, k_1\} \\ & \int_{x_{\min}}^{x_{\max}} \log^i(x) f(x)\,dx = \nu_i, \quad i \in \{1, \ldots, k_2\} \end{aligned}$$

where $\mathcal{F}[x_{\min}, x_{\max}]$ denotes the set of distributions supported on $[x_{\min}, x_{\max}]$.

It is well known that the solution to Problem (4) is a member of the class of exponential family distributions [41]:

$$f(x;\theta) = \exp\left(\theta_0 + \sum_{i=1}^{k_1} \theta_i x^i + \sum_{i=1}^{k_2} \theta_{k_1+i} \log^i(x)\right),$$

where θ_0 is a normalization constant such that $f(x;\theta)$ integrates to 1 over the domain $[x_{\min}, x_{\max}]$. The maximum entropy distribution is determined by the parameter θ such that $f(x;\theta)$ satisfies the moment constraints in Problem (4).

In order to solve for θ, we define the potential function $L(\theta)$ from [57]:

$$\begin{aligned} L(\theta) = \int_{x_{\min}}^{x_{\min}} \exp\left(\sum_{i=0}^{k_1} \theta_i x^i + \sum_{i=1}^{k_2} \theta_{k_1+i} \log^i x\right) dx \qquad (5) \\ - \sum_{i=0}^{k_1} \theta_i \mu_i - \sum_{i=1}^{k_2} \theta_{k_1+i} \nu_i \end{aligned}$$

$L(\theta)$ is a convex function over θ and is constructed so that the minimizing solution $\theta_{\text{opt}} = \arg\min_{\theta \in \mathbb{R}^{k_1+k_2-1}} L(\theta)$ is exactly the set of coefficients which satisfy the constraints in Problem (4). Equation (5) thus transforms the constrained optimization problem in (4) into an unconstrained convex optimization problem which we solve using Newton's method [15]. We show the explicit formulas for the gradient and Hessian that of Equation (5) in Appendix A.1 in [30]. First-order optimization routines such as SGD and BFGS [50] are also viable: they do not use the Hessian but require more steps to achieve convergence. As we will describe in Section 4.3, each additional entry in our Hessian can be computed efficiently using Chebyshev approximations, making second order methods more efficient overall. We provide a lesion study comparison in Section 6.3.

4.3 Practical Implementation

In this section, we outline implementation details that are important for solving the constrained entropy maximization problem in practice. Due to space constraints, some equations are omitted here and provided in Appendix A.1 in [30]. The Hessian of L takes the form:

$$\nabla^2 L(\theta)_{ij} = \int_{x_{\min}}^{x_{\max}} m_i(x) m_j(x) f(x;\theta)\, dx, \tag{6}$$

where the functions $m_i(x)$ range over the set of functions

$$\{x^i : i \in \{1, \ldots, k_1\}\} \cup \{\log^i(x) : i \in \{1, \ldots, k_2\}\}.$$

There are two main challenges in computations involving the Hessian in each Newton step. First, $\nabla^2 L$ can be nearly singular and cause numerical instabilities in Newton's method that prevent or slow down convergence. Second, since the integral in Eq. (6) has no closed form, the cost of performing $O(k^2)$ numerical integrations to compute $\nabla^2 L$ in each iteration can be expensive. We describe our solutions to each of these issues in turn, and then discuss how they influence our choice of k and k_1, k_2 during optimization.

Numerical Stability. To quantify the degree of numerical instability, we use the *condition number* of the Hessian $\nabla^2 L$. The condition number $\kappa(A)$ of a matrix A describes how close a matrix is to being singular: matrices with high condition number are close to being singular, and $\log_{10} \kappa$ provides an estimate of how many digits of precision are lost when inverting A. In particular, the use of the powers $m_i(x) \in \{x^i : i \in \{1, \ldots, k_1\}\}$ can result in ill-conditioned Hessians [31]. For example, when solving for a maximum entropy distribution with $k_1 = 8, k_2 = 0, x_{\min} = 20$, and $x_{\max} = 100$, we encountered $\kappa(\nabla^2 L) \approx 3 \cdot 10^{31}$ at $\theta = 0$, making even the very first Newton step infeasible using double precision.

We mitigate this issue by using a change of basis from the functions $m_i(x) = x^j$ and $m_i(x) = \log^j(x)$ to the basis of Chebyshev polynomials $T_i(x)$. Chebyshev polynomials are bounded polynomials supported on $[-1, 1]$ and are often used for polynomial approximation [10, 64]. Using them we define the new basis $\tilde{m}_i$ as follows:

$$\tilde{m}_i(x) = \begin{cases} T_i(s_1(x)), & i \in \{1, \ldots, k_1\} \\ T_{i-k_1}(s_2(\log(x))), & i \in \{k_1 + 1, \ldots, k_1 + k_2\} \end{cases}$$

where s_1, s_2 are linear scaling functions that map to $[-1, 1]$. The new basis functions $\tilde{m}_i(x)$ can be expressed in terms of x^j and $\log^j(x)$ using standard formulae for Chebyshev polynomials and the binomial expansion [56]. Using this new basis for m_i Equation (6), we found that the condition number for the above example is reduced to $\kappa \approx 11.3$, making precision loss during each Newton step less of a concern.

Efficient Integration. Naïvely computing the Hessian in Equation (6) requires evaluating $O(k^2)$ numerical integrals per iteration, which can lead to prohibitively slow estimation time. We reduce this computational overhead by using polynomial approximations of the functions appearing in the integrands. If the integrands $\tilde{m}_i(x)\tilde{m}_j(x)f(x;\theta)$ were expressible as polynomials in x, then each integral can be evaluated in closed form. The factors in the integrand that do not appear as polynomials in x are $\tilde{m}_i(x), i \in \{k_1 + 1, \ldots, k_1 + k_2\}$, which are polynomials in $\log(x)$, and the pdf $f(x;\theta)$. Therefore, we compute Chebyshev polynomial approximations of these factors and replace each instance in the integrands with its corresponding approximation.[1]

Approximating each of the factors with a degree n_c polynomial takes $O(n_c \cdot \log n_c)$ using a fast cosine transform [64], so computing the Hessian can be done in $O(k_2 n_c \log n_c + n_c k_1 k_2)$. This is not an asymptotic improvement over naive numeric integration, but the number of complex function evaluations (i.e. $\cos(x), e^x$) is reduced substantially. As we show in our empirical evaluation (Section 6.3), polynomial approximations reduce solve times $20\times$ compared to numerically integrating each entry of the Hessian independently. We find in our experiments that the major bottleneck during maximum entropy optimization is the cosine transform used to construct the polynomial approximations.

4.3.1 Choosing k

The two optimizations we have described improve the stability and runtime of our maximum entropy solver. However, for large k these can still be an issue. In this section we discuss how k affects the operation of our estimator implementation.

Given a moments sketch with k moments and log moments, users can configure how many of each k_1, k_2 they would like to make use of when estimating quantiles. In our implementation we use a set of default heuristics to select k_1, k_2 with condition number below a threshold $\kappa_{\max}$. Our heuristics greedily increment k_1 and k_2, favoring moments where we do not detect precision loss and which are closer to the moments expected from a uniform distribution.

Numeric floating point stability limits the range of useful k. Both our estimation routine and error bounds (Section 4.4) use moments corresponding to data shifted onto the range $[-1, 1]$. On scaled data with range $[c - 1, c + 1]$, this leads to numeric error ϵ_k in the k-th shifted moment, bounded by $\epsilon_k \leq 2^k (|c| + 1)^k \epsilon_s$ where ϵ_s is the relative error in the raw moments sketch power sums. This shift is the primary source of precision loss. We relate the loss to the error bounds in Section 4.4 to show that when using double precision floating point moments up to around $k \leq \frac{13.06}{0.78 + \log_{10}(|c|+1)}$ provide numerically useful values. Data centered at 0 ($c = 0$) have more stable higher moments, and in practice we encounter issues when $k \geq 15$. We provide derivations and evaluations of this formula in Appendix A.2 and A.3 in [30]

4.4 Quantile Error Bounds

Recall that we estimate quantiles using the maximum entropy distribution subject to the moment constraints recorded in the moments sketch. Since the true empirical distribution is in general not equal to the estimated maximum entropy distribution, to what extent can the quantiles estimated from the sketch deviate from the true quantiles? In this section, we discuss worst-case bounds on the discrepancy between *any* two distributions which share the same moments, and relate these to bounds on the quantile estimate errors. In practice, error on non-adversarial datasets is lower than these bounds suggest.

We consider distributions supported on $[-1, 1]$: we can scale and shift any distribution with bounded support to match. By Proposition 1 in Kong and Valiant [47], any two distributions supported on $[-1, 1]$ with densities f and

[1] Compare with Clenshaw-Curtis integration [64].

g and standard moments μ_f, μ_g, the Wasserstein distance (or Earth Mover's distance) $W_1(f,g)$ between f and g is bounded by:

$$W_1(f,g) \leq O\left(\frac{1}{k} + 3^k \|\mu_f - \mu_g\|_2\right).$$

For univariate distributions f and g, the Wasserstein distance between the distributions is equal to the L1 distance between their respective cumulative distribution functions F and G (see Theorem 6.0.2 in [6]). Thus:

$$W_1(f,g) = \int_{-1}^{+1} |F(x) - G(x)|\, dx.$$

If f is the true dataset distribution, we estimate q_ϕ by calculating the ϕ-quantile of the maximum entropy distribution $\hat{f}$. The quantile error $\varepsilon(q_\phi)$ is then equal to the gap between the CDFs: $\varepsilon(q_\phi) = |F(q_\phi) - \hat{F}(q_\phi)|$. Therefore, the average quantile error over the support $[-1,1]$ is bounded as follows:

$$\int_{-1}^{+1} \varepsilon(x)\, dx \leq O\left(\frac{1}{k} + 3^k \|\mu_f - \mu_{\hat{f}}\|_2\right). \tag{7}$$

Since we can run Newton's method until the moments μ_f and $\mu_{\hat{f}}$ match to any desired precision, the $3^k\|\mu_f - \mu_{\hat{f}}\|_2$ term is negligible.

Equation (7) does not directly apply to the ϵ_{avg} used in Section 6, which is averaged over ϕ for uniformly spaced ϕ-quantiles rather than over the support of the distribution. Since $q_\phi = \hat{F}^{-1}(\phi)$ and $\phi = \hat{F}(x)$, we can relate ϵ_{avg} to Eq. (7) using our maximum entropy distribution $\hat{f}$:

$$\int_0^1 \varepsilon(q_\phi)\, d\phi = \int_{-1}^{+1} \varepsilon(x)\hat{f}(x)\, dx \leq O\left(\frac{\hat{f}_{\max}}{k}\right)$$

where $\hat{f}_{\max}$ is the maximum density of our estimate. Thus, we expect the average quantile error ϵ_{avg} to have a decreasing upper bound as k increases, with higher potential error when $\hat{f}$ has regions of high density relative to its support. Though these bounds are too conservative to be useful in practice, they provide useful intuition on how worst case error can vary with k and $\hat{f}$ (c.f. Figure 20).

5. THRESHOLD QUERIES

We described in Section 3.3 two types of queries: single quantile queries and threshold queries over multiple groups. The optimizations in Section 4.3 can bring quantile estimation overhead down to $\leq$ 1ms, which is sufficient for interactive latencies on single quantile queries. In this section we show how we can further reduce quantile estimation overheads on threshold queries. Instead of computing the quantile on each sub-group directly, we compute a sequence of progressively more precise bounds in a *cascade* [75], and only use more expensive estimators when necessary. We first describe a series of bounds relevant to the moments sketch in Section 5.1 and then show how they can be used in end-to-end queries in Section 5.2.

5.1 Moment-based inequalities

Given the statistics in a moments sketch, we apply a variety of classical inequalities to derive bounds on the quantiles. These provide some worst-case error guarantees on quantile estimates which we can report to users along with the quantile estimate, and are also essential in allowing fast query processing on threshold queries over multiple groups.

One simple inequality we make use of is Markov's inequality. Given a non-negative dataset D with moments μ_i Markov's inequality tells us that for any value x, $\text{rank}(x) \geq n\left(1 - \frac{\mu_k}{t^k}\right)$ where the rank is the number of elements in D less than x. We apply this to arbitrary datasets using the transformation $D' = \{x - x_{\min} : x \in D\}$ since one can compute the moments μ_i' of D' using the binomial formula. Similarly we can use the transformations $D^- = \{x_{\max} - x : x \in D\}$ and $D^l = \{\log(x) : x \in D\}$ to obtain lower bounds and analogous bounds on the log moments.

We apply Markov's inequality to a quantile estimate $x = q_\phi$ on each of the transformed datasets and for each moment μ_k above to obtain upper or lower bounds on the rank $r(t)$. These upper and lower bounds on the rank then provide an upper bound on the quantile error ϵ for a ϕ-quantile estimate. We refer to this procedure as the MarkovBound procedure.

The authors in [66] provide a procedure (Section 3 Figure 1 in [66]) for computing tighter but more expensive bounds on the CDF $F(t)$ of a distribution given its moments. We refer to this procedure as the RTTBound procedure, and as with the Markov bounds, use it to bound the rank of a computed quantile estimate q_ϕ. The RTTBound procedure does not make use of the standard moments and log moments simultaneously. When applying RTTBound to the moments sketch, we run the RTTBound procedure once on the standard moments and once on log moments and take the tighter of the bounds.

Algorithm 2: Threshold Query Cascade

```
macro CheckBound(r_lower, r_upper, r_t)
    if r_lower > r_t then
        return true
    else if r_upper < r_t then
        return false
function Threshold(threshold t, quantile φ)
    if t > x_max then
        return false
    if t < x_min then
        return true
    r_lower, r_upper ← MarkovBound(t)   ▷ Markov Bound
    CheckBound(r_lower, r_upper, nφ)
    r_lower, r_upper ← RTTBound(t)      ▷ RTT Bound
    CheckBound(r_lower, r_upper, nφ)
    q_φ ← MaxEntQuantile(φ)             ▷ Maximum Entropy
    return q_φ > t
```

5.2 Cascades for Threshold queries

Given a moments sketch, Algorithm 2 shows how we calculate `Threshold`(t, ϕ): whether the dataset has quantile estimate q_ϕ above a fixed cutoff t. We use this routine whenever we answer queries on groups with a predicate $q_\phi > t$, allowing us to check whether a subgroup should be included in the results without computing the quantile estimate directly. The threshold check routine first performs a simple filter on whether the threshold t falls in the range $[x_{\min}, x_{\max}]$. Then, we can use the Markov inequalities `MarkovBound` to calculate lower and upper bounds on the rank of the threshold

rank(t) in the subpopulation. Similarly the `RTTBound` routine uses more sophisticated inequalities in [66] to bound the rank. These bounds are used to determine if we can resolve the threshold predicate immediately. If not, we solve for the maximum entropy distribution as described in Section 4.2 (`MaxEntQuantile`) to estimate q_ϕ.

The Markov and RTTBound bounds are cheaper to compute than our maximum entropy estimate, making threshold predicates cheaper to evaluate than explicit quantile estimates. The bounds apply to any distribution or dataset that matches the moments in a moments sketch, so this routine has no false negatives and is consistent with calculating the maximum entropy quantile estimate up front.

6. EVALUATION

In this section we evaluate the efficiency and accuracy of the moments sketch in a series of microbenchmarks, and then show how the moments sketch provides end-to-end performance improvements in the Druid and Macrobase data analytics engines [8, 82].

This evaluation demonstrates that:

1. The moments sketch supports 15 to 50× faster query times than comparably accurate summaries on quantile aggregations.
2. The moments sketch provides $\epsilon_{\text{avg}} \leq 0.01$ estimates across a range of real-world datasets using less than 200 bytes of storage.
3. Maximum entropy estimation is more accurate than alternative moment-based quantile estimates, and our solver improves estimation time by 200× over naive solutions.
4. Integrating the moments sketch as a user-defined sketch provides 7× faster quantile queries than the default quantile summary in Druid workloads.
5. Cascades can provide 25× higher query throughput compared to direct moments sketch usage in Macrobase threshold queries.

Throughout the evaluations, the moments sketch is able to accelerate a variety of workloads when the number of summary aggregations exceeds the number of queries, when space is at a premium, and when interactive sub-100ms query response times are essential.

6.1 Experimental Setup

We implement the moments sketch and its quantile estimation routines in Java[2]. This allows for direct comparisons with the open source quantile summaries [1,67] and integration with the Java-based Druid [82] and MacroBase [8] systems. In our experimental results, we use the abbreviation M-Sketch to refer to the moments sketch.

We compare against a number of alternative quantile summaries: a mergeable equi-width histogram (`EW-Hist`) using power-of-two ranges [65], the 'GKArray' (`GK`) variant of the Greenwald Khanna [34, 52] sketch, the AVL-tree T-Digest (`T-Digest`) [28] sketch, the streaming histogram (`S-Hist`) in [12] as implemented in Druid, the 'Random' (`RandomW`) sketch from [52,77], reservoir sampling (`Sampling`) [76], and the low-discrepancy mergeable sketch (`Merge12`) from [3], both implemented in the Yahoo! datasketches library [1].

[2] `https://github.com/stanford-futuredata/msketch`

	milan	hepmass	occupancy	retail	power	expon
size	81M	10.5M	20k	530k	2M	100M
min	$2.3e{-6}$	-1.961	412.8	1	0.076	$1.2e{-7}$
max	7936	4.378	2077	80995	11.12	16.30
mean	36.77	0.0163	690.6	10.66	1.092	1.000
stddev	103.5	1.004	311.2	156.8	1.057	0.999
skew	8.585	0.2946	1.654	460.1	1.786	1.994

Table 1: Dataset Characteristics. The evaluation datasets cover a range of distribution types.

The `GK` sketch is not usually considered mergeable since its size can grow upon merging [3], this is especially dramatic in the production benchmarks in Appendix A.8 in [30]. We do not compare against fixed-universe quantile summaries such as the Q-Digest [71] or Count-Min sketch [23] since they would discretize continuous values.

Each quantile summary has a *size parameter* controlling its memory usage, which we will vary in our experiments. Our implementations and benchmarks use double precision floating point values. During moments sketch quantile estimation we run Newton's method until the moments match to within $\delta = 10^{-9}$, and select k_1, k_2 using a maximum condition number $\kappa_{\max} = 10^4$. We construct the moments sketch to store both standard and log moments up to order k, but choose at query time which moments to make use of as described in Section 4.3.

We quantify the accuracy of a quantile estimate using the quantile error ε as defined in Section 3.1. Then, as in [52, 77] we can compare the accuracies of summaries on a given dataset by computing their *average error* ϵ_{avg} over a set of uniformly spaced ϕ-quantiles. In the evaluation that follows, we test on 21 equally spaced ϕ between 0.01 and 0.99.

We evaluate each summary via single-threaded experiments on a machine with an Intel Xeon E5-4657L 2.40GHz processor and 1TB of RAM, omitting the time to load data from disk.

6.1.1 Datasets

We make use of six real-valued datasets in our experiments, whose characteristics are summarized in Table 1. The *milan* dataset consists of Internet usage measurements from Nov. 2013 in the Telecom Italia Call Data Records [40]. The *hepmass* dataset consists of the first feature in the UCI [49] HEPMASS dataset. The *occupancy* dataset consists of CO2 measurements from the UCI Occupancy Detection dataset. The *retail* dataset consists of integer purchase quantities from the UCI Online Retail dataset. The *power* dataset consists of Global Active Power measurements from the UCI Individual Household Electric Power Consumption dataset. The *exponential* dataset consists of synthetic values from an exponential distribution with $\lambda = 1$.

6.2 Performance Benchmarks

We begin with a series of microbenchmarks evaluating the moments sketch query times and accuracy.

6.2.1 Query Time

Our primary metric for evaluating the moments sketch is total query time. We evaluate quantile aggregation query times by pre-aggregating our datasets into cells of 200 values and maintaining quantile summaries for each cell. Then we measure the time it takes to performing a sequence of merges

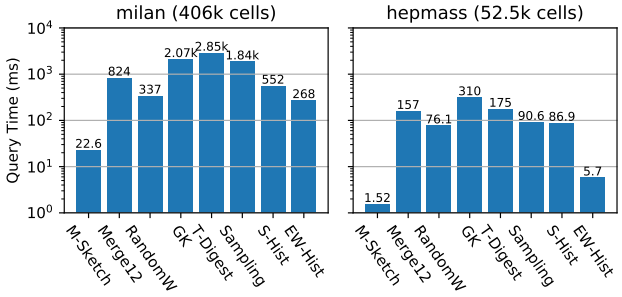

Figure 3: Total query time using different summaries to estimate quantiles with $\epsilon_{\text{avg}} \leq .01$. The moments sketch enables significantly faster queries at this accuracy.

dataset	milan		hepmass	
sketch	param	size (b)	param	size (b)
`M-Sketch`	$k = 10$	200	$k = 3$	72
`Merge12`	$k = 32$	5920	$k = 32$	5150
`RandomW`	$\epsilon = \frac{1}{40}$	3200	$\epsilon = \frac{1}{40}$	3375
`GK`	$\epsilon = \frac{1}{60}$	720	$\epsilon = \frac{1}{40}$	496
`T-Digest`	$\delta = 5.0$	769	$\delta = 1.5$	93
`Sampling`	1000 samples	8010	1000	8010
`S-Hist`	100 bins	1220	100	1220
`EW-Hist`	100 bins	812	15	132

Table 2: Summary size parameters used in Figure 3. We use these parameters to compare the query times at $\epsilon_{\text{avg}} \leq .01$ accuracy.

and estimate a quantile. In this performance microbenchmark, the cells are grouped based on their sequence in the dataset, while the evaluations in Section 7 group based on column values. We divide the datasets into a large number of cells to simulate production data cubes, while in Appendix A.7 and A.8 in [30] we vary the cell sizes. Since the moments sketch size and merge time are data-independent, the results generalize as we vary cell size.

Figure 3 shows the total query time to merge the summaries and then compute a quantile estimate when each summary is instantiated at the smallest size sufficient to achieve $\epsilon_{\text{avg}} \leq .01$ accuracy. We provide the parameters we used and average observed space usage in Table 2. On the long-tailed milan dataset, the `S-Hist` and `EW-Hist` summaries are unable to achieve $\epsilon_{\text{avg}} \leq .01$ accuracy with less than 100 thousand buckets, so we provide timings at 100 buckets for comparison. The moments sketch provides 15 to 50× faster query times than `RandomW`, the next fastest accurate summary. As a baseline, sorting the milan dataset takes 7.0 seconds, selecting an exact quantile online takes 880ms, and streaming the data pointwise into a `RandomW` sketch with $\epsilon = 1/40$ takes 220ms. These methods do not scale as well as using pre-aggregated moments sketches as dataset density grows but the number of cells remains fixed.

6.2.2 *Merge and Estimation Time*

Recall that for a basic aggregate quantile query $t_{\text{query}} = t_{\text{merge}} \cdot n_{\text{merge}} + t_{\text{est}}$. Thus we also measure t_{merge} and t_{est} to quantify the regimes where the moments sketch performs well. In these experiments, we vary the summary size parameters, though many summaries have a minimum size, and the moments sketch runs into numeric stability issues

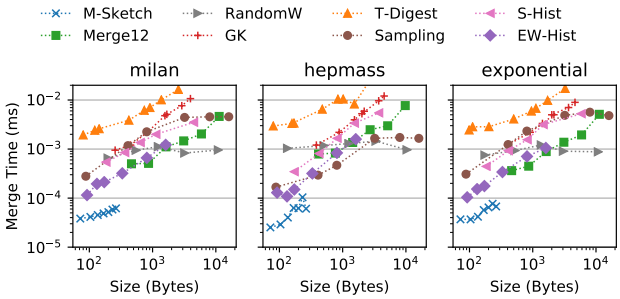

Figure 4: Per-merge latencies. The moments sketch provides faster merge times than alternative summaries at the same size.

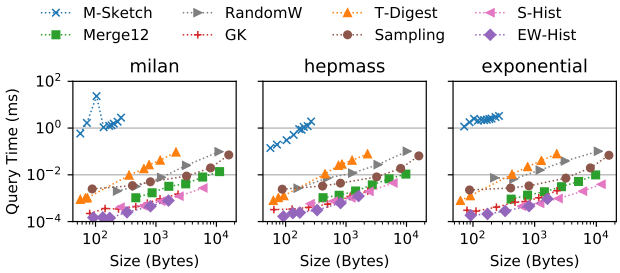

Figure 5: Quantile Estimation time. Estimation time on the moments sketch is slower than other sketches but under 3ms for $k = 10$.

past $k \geq 15$ on some datasets (see Section 4.3.1).

In Figure 4 we evaluate the average time required to merge one of the cell summaries. Larger summaries are more expensive to merge, and the moments sketch has faster (< 50ns) merge times throughout its size range. When comparing summaries using the parameters in Table 2, the moments sketch has up to 50× faster merge times than other summaries with the same accuracy.

One can also parallelize the merges by sharding the data and having separate nodes operate over each partition, generating partial summaries to be aggregated into a final result. Since each parallel worker can operate independently, in these settings the moments sketch maintains the same relative performance improvements over alternative summaries when we can amortize fixed overheads, and we include supplemental parallel experiments in Appendix A.6 in [30].

The other major contributor to query time is estimation time. In Figure 5 we measure the time to estimate quantiles given an existing summary. The moments sketch provides on average 2 ms estimation times, though estimation time can be higher for small k when our estimator chooses higher k_1, k_2 to achieve better accuracy. This is the cause for the spike at $k = 4$ in the milan dataset and users can can mitigate this by lowering the condition number threshold $\kappa_{\max}$. Other summaries support microsecond estimation times. The moments sketch thus offers a tradeoff of better merge time for worse estimation time. If users require faster estimation times, the cascades in Section 5.2 and the alternative estimators in Section 6.3 can assist.

We show how the merge time and estimation time tradeoff define regimes where each component dominates depending on the number of merges. In Figure 6 we measure how the query time changes for queries requiring different numbers of merges over pre-aggregated summaries on cells of 200 rows. We use the moments sketch with $k = 10$ and compare

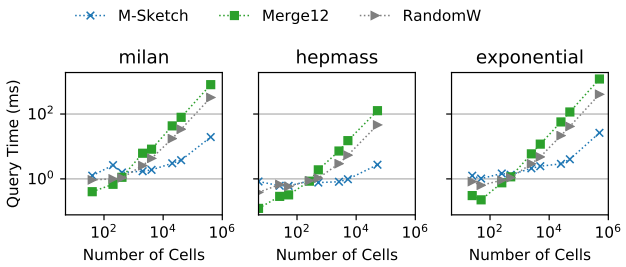

Figure 6: Comparing total query time using different mergeable summaries as we vary the number of merges. The moments sketch provides performance improvements when $n_{\text{merge}} \geq 10^4$.

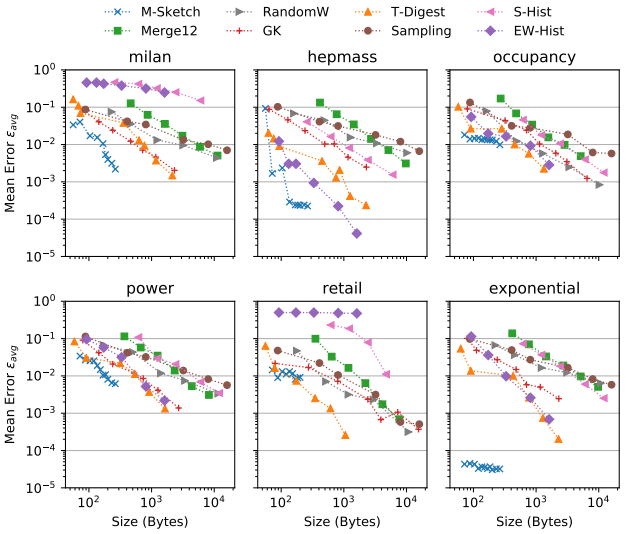

Figure 7: Average error for summaries of different sizes. The moments sketch delivers consistent $\epsilon_{\text{avg}} \leq 0.015$ with fewer than 200 bytes.

against the mergeable `Merge12` and `RandomW` summaries with parameters from Table 2. Merge time begins to dominate for $n_{\text{merge}} \geq 10^4$ and beyond that the moments sketch provides better performance than alternative summaries. However, the moments sketch estimation times dominate when $n_{\text{merge}} \leq 100$.

6.2.3 Accuracy

The moments sketch accuracy is dataset dependent, so in this section we compare the average quantile error on our evaluation datasets.

Figure 7 illustrates the average quantile error ϵ_{avg} for summaries of different sizes constructed using pointwise accumulation on the complete dataset. The moments sketch achieves $\epsilon \leq 10^{-4}$ accuracy on the synthetic exponential dataset, and $\epsilon \leq 10^{-3}$ accuracy on the high entropy hepmass dataset. On other datasets it is able to achieve $\epsilon_{\text{avg}} \leq 0.01$ with fewer than 200 bytes of space. On the integer retail dataset we round estimates to the nearest integer. The `EW-Hist` summary, while efficient to merge, provides less accurate estimates than the moments sketch, especially in the long-tailed milan and retail datasets.

In our supplemental evaluations in we provide further experiments showing how the moments sketch worst-case error

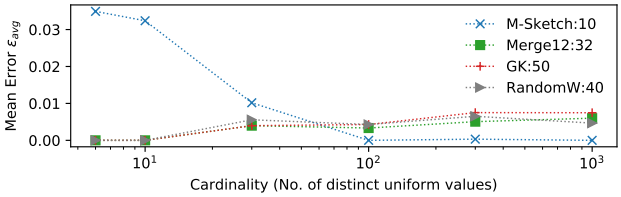

Figure 8: Accuracy of maximum entropy estimates on distributions with varying cardinality. The moments sketch is less accurate on discretized datasets, and fails to converge for cardinalities $n < 5$.

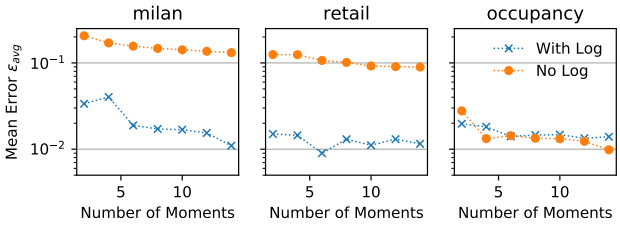

Figure 9: Accuracy with and without log moments. Given the same total space budget, log moments improve accuracy on the long-tailed milan and retail datasets, and do not affect accuracy significantly on other datasets such as occupancy

bounds are comparable to other summaries (Appendix A.5), that the moments sketch is robust to changes in skew and the presence of outliers (Appendix A.4), and that the moments sketch generalizes to a production workload (Appendix A.8). However, on datasets with low-entropy, in particular datasets consisting of a small number of discrete point masses, the maximum entropy principle provides poor accuracy. In the worst case, the maximum entropy solver can fail to converge on datasets with too few distinct values. Figure 8 illustrates how the error of the maximum entropy estimate increases as we lower the cardinality of a dataset consisting of uniformly spaced points in the range $[-1, 1]$, eventually failing to converge on datasets with fewer than five distinct values. If users are expecting to run queries on primarily low-cardinality datasets, fixed-universe sketches or heavy-hitters sketches may be more appropriate.

6.3 Quantile Estimation Lesion Study

To evaluate each component of our quantile estimator design, we compare the accuracy and estimation time of a variety of alternative techniques on the milan and hepmass datasets. We evaluate the impact of using log moments, the maximum entropy distribution, and our optimizations to estimation.

To examine effectiveness of log moments, we compare our maximum entropy quantile estimator accuracy with and without log moments. For a fair comparison, we compare the estimates produced from k standard moments and no log moments with those produced from up to $\frac{k}{2}$ of each. Figure 9 illustrates how on some long-tailed datasets, notably milan and retail, log moments reduce the error from $\epsilon > .15$ to $\epsilon < .015$. On other datasets, log moments do not have a significant impact.

We compare our estimator (`opt`) with a number of other estimators that make use of the same moments. The gaussian estimator fits a Gaussian distribution to the mean and

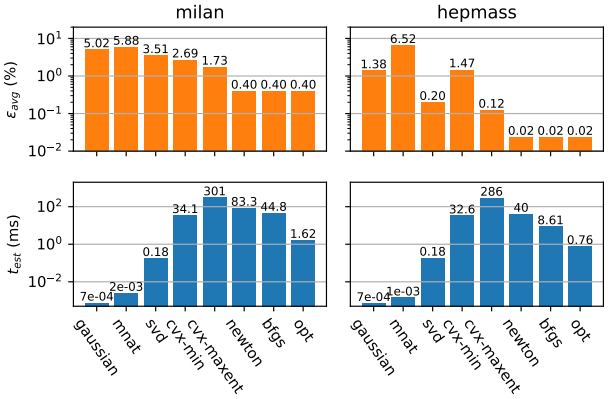

Figure 10: Lesion study comparing our optimized maximum entropy solver to other estimators. Our `opt` estimator provides at least 5× less error than estimators that do not use maximum entropy, and up to 200× faster estimation times than naive maximum entropy solvers.

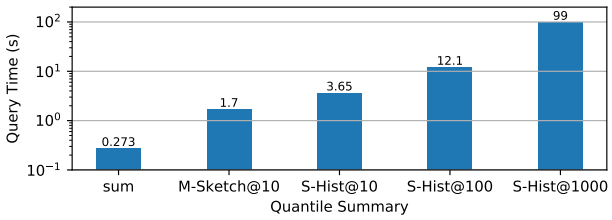

Figure 11: Druid end-to-end query benchmark. The moments sketch allows for faster query times than the comparable `S-Hist` summary with 100 bins. Runtime for a native sum operation is a lower bound on query time.

standard deviation. The mnat estimator uses the closed form discrete CDF estimator in [58]. The svd estimator discretizes the domain and uses singular value decomposition to solve for a distribution with matching moments. The cvx-min estimator also discretizes the domain and uses a convex solver to construct a distribution with minimal maximum density and matching moments. The cvx-maxent estimator discretizes the domain and uses a convex solver to maximize the entropy, as described in Chapter 7 in [15]. The newton estimator implements our estimator without the integration techniques in Sec. 4.3, and uses adaptive Romberg integration instead [64]. The bfgs estimator implements maximum entropy optimization using the first-order L-BFGS [50] method as implemented in a Java port of liblbfgs [46].

Figure 10 illustrates the average quantile error and estimation time for these estimators. We run these experiments with $k = 10$ moments. For uniform comparisons with other estimators, on the milan dataset we only use the log moments, and on the hepmass dataset we only use the standard moments. We perform discretizations using 1000 uniformly spaced points and make use of the ECOS convex solver [27]. Solvers that use the maximum entropy principle provides at least 5× less error than estimators that do not. Furthermore, our optimizations are able to improve the estimation time by a factor of up to 200× over an implementation using generic solvers, and provide faster solve times than naive Newton's method or BFGS optimizers. As described in Section 4.3, given the computations needed to calculate the gradient, one can compute the Hessian relatively cheaply, so our optimized Newton's method is faster than BFGS.

7. APPLYING THE MOMENTS SKETCH

In this section, we evaluate how the moments sketch affects performance when integrated with other data systems. We examine how the moments sketch improves query performance in the Druid analytics engine, as part of a cascade in the Macrobase feature selection engine [8], and as part of exploratory sliding window queries.

7.1 Druid Integration

To illustrate the utility of the moments sketch in a modern analytics engine, we integrate the moments sketch with Druid [82]. We do this by implementing moments sketch as an user-defined aggregation extension, and compare the total query time on quantile queries of the moments sketch with the default `S-Hist` summary used in Druid and introduced in [12]. The authors in [12] observe on average 5% error for an `S-Hist` with 100 centroids, so we benchmark a moments sketch with $k = 10$ against `S-Hists` with 10, 100, and 1000 centroids.

In our experiments, we deploy Druid on a single node – the same machine described in section 6.1 – with the same base configuration used in the default Druid quickstart. In particular, this configuration dedicates 2 threads to process aggregations. Then, we ingest 26 million entries from the milan dataset at a one hour granularity and construct a cube over the grid ID and country dimensions, resulting in 10 million cells.

Figure 11 compares the total time to query for a quantile on the complete dataset using the different summaries. The moments sketch provides 7× lower query times than a `S-Hist` with 100 bins. Furthermore, as discussed in Section 6.2.1, any `S-Hist` with fewer than 10 thousand buckets provides worse accuracy on milan data than the moments sketch. As a best-case baseline, we also show the time taken to compute a native sum query on the same data. The 1 ms cost of solving for quantile estimates from the moments sketch on this dataset is negligible here.

7.2 Threshold queries

In this section we evaluate how the cascades described in Section 5.2 improve performance when processing queries with threshold predicates. First we show in Section 7.2.1 how data exploration engines such as MacroBase benefit from using the moments sketch to search for anomalous dimension values. Then, we show in Section 7.2.2 how historical analytics queries can use the moments sketch to search and alert on sliding windows.

7.2.1 *MacroBase Integration*

The MacroBase engine searches for dimension values with unusually high outlier rates in a dataset [8]. For example, given an overall 2% outlier rate, MacroBase may report when a specific app version has an outlier rate of 20%. We integrate the moments sketch with a simplified deployment of MacroBase where all values greater than the global 99th percentile t_{99} are considered outliers. We then query Mac-

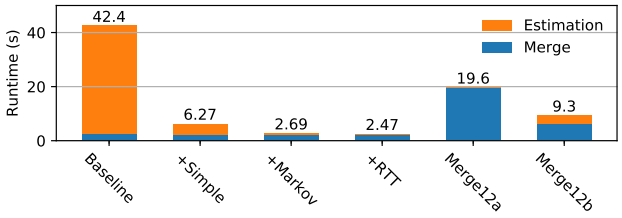

Figure 12: Runtime of MacroBase queries: the final moments sketch cascade outperforms queries using alternate sketches.

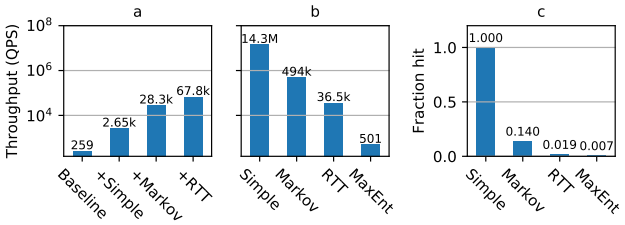

Figure 13: Cascades in MacroBase: (a) as we incrementally add cascade stages, threshold query throughput increases. (b) The cascade proceeds from faster to slower estimates. (c) Each stage of the cascade processes a smaller fraction of queries.

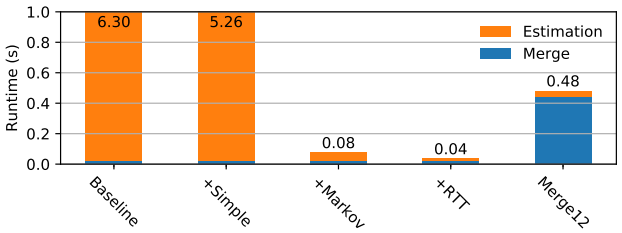

Figure 14: Sliding window query: moments sketch with cascades runs 13× faster than `Merge12`.

roBase for all dimension values with outlier rate at least $r = 30\times$ greater than the overall outlier rate. This is equivalent to finding subpopulations whose 70th percentile is greater than t_{99}.

Given a cube with pre-aggregated moments sketches for each dimension value combination and no materialized rollups, MacroBase merges the moments sketches to calculate the global t_{99}, and then runs Algorithm 2 on every dimension-value subpopulation, searching for subgroups with $q_{.7} > t_{99}$. We evaluate the performance of this query on 80 million rows of the milan internet usage data from November 2013, pre-aggregated by grid ID, country, and at a four hour granularity. This resulted in 13 million cube cells, each with its own moments sketch.

Running the MacroBase query produces 19 candidate dimension values. We compare the total time to process this query using direct quantile estimates, our cascades, and the alternative `Merge12` quantile sketch. In the first approach (`Merge12a`), we merge summaries during MacroBase execution as we do with a moments sketch. In the second approach (`Merge12b`), we calculate the number of values greater than the t_{99} for each dimension value combination and accumulate these counts directly, instead of the sketches. This approach incurs a higher query time but avoids much of the cost of merging. We present this as an optimistic baseline, and is not always a feasible substitute for merging summaries.

Figure 12 shows the query times for these different methods: the baseline method calculates quantile estimates directly for every threshold, we show the effect of incrementally adding each stage of our cascade ending with +RTTBound, and we also compare against the two usages of the `Merge12` sketch. Each successive stage of the cascade improves query time substantially. With the complete cascade, estimation time is negligible compared to merge time. Furthermore, the moments sketch with cascades has 7.9× lower query times than using the `Merge12` sketch, and even 3.7× lower query times than the `Merge12b` baseline.

In Figure 13 we examine the impact the cascade has on estimation time directly. Each additional cascade stage improves threshold query throughput and is more expensive than the last. The complete cascade is over 250× faster than this baseline, and 25× faster than just using a simple range check.

7.2.2 *Sliding Window Queries*

Threshold predicates are broadly applicable in data exploration queries. In this section, we evaluate how the moments sketch and our cascade improves performance on queries for sliding windows that exceed an alert threshold. This is useful when, for instance, users are searching for time windows of unusually high CPU usage spikes.

For this benchmark, we aggregated the 80 million rows of the milan dataset at a 10-minute granularity, which produced 4320 panes that spanned the month of November. We augmented the milan data with two spikes corresponding to hypothetical anomalies. Each spike spanned a two-hour time frame and contributed 10% more data to those time frames. Given a global 99th percentile of around 500 and a maximum value of 8000, we added spikes with values $x = 2000$ and $x = 1000$

We then queried for the 4-hour time windows whose 99th percentile was above a threshold $t = 1500$. When processing this query using a moments sketch, we can update sliding windows using turnstile semantics, subtracting the values from the oldest pane and merging in the new one.

Figure 14 shows the runtime of the sliding window query using both the moments sketch and `Merge12`. The cascade improves the estimation time of the moments sketch by filtering out windows that cannot pass the threshold. Faster moments sketch merge times and the use of turnstile semantics then allow for 13× faster queries than `Merge12`.

8. CONCLUSION

In this paper, we show how to improve the performance of quantile aggregation queries using statistical moments. Low merge overhead allows the moments sketch to outperform comparably accurate existing summaries when queries aggregate more than 10 thousand summaries. By making use of the method of moments and the maximum entropy principle, the moments sketch provides $\epsilon_{\text{avg}} \leq 0.01$ accuracy on real-world datasets, while the use of numeric optimizations and cascades keep query times at interactive latencies.

9. REFERENCES

[1] Yahoo! data sketches library, 2017. `https://datasketches.github.io/`.

[2] L. Abraham, J. Allen, O. Barykin, V. Borkar, B. Chopra, C. Gerea, D. Merl, J. Metzler, D. Reiss, S. Subramanian, J. L. Wiener, and O. Zed. Scuba: Diving into data at facebook. *VLDB*, 6(11):1057–1067, 2013.

[3] P. K. Agarwal, G. Cormode, Z. Huang, J. Phillips, Z. Wei, and K. Yi. Mergeable summaries. In *PODS*, 2012.

[4] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In *EuroSys*, pages 29–42, 2013.

[5] N. Akhiezer. *The Classical Moment Problem and Some Related Questions in Analysis*. Oliver & Boyd, 1965.

[6] L. Ambrosio, N. Gigli, and G. Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

[7] A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. In *COLT*, pages 33–1, 2012.

[8] P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri. MacroBase: Prioritizing attention in fast data. In *SIGMOD*, pages 541–556, 2017.

[9] A. Balestrino, A. Caiti, A. Noe', and F. Parenti. Maximum entropy based numerical algorithms for approximation of probability density functions. In *2003 European Control Conference (ECC)*, pages 796–801, 2003.

[10] K. Bandyopadhyay, A. K. Bhattacharya, P. Biswas, and D. A. Drabold. Maximum entropy and the problem of moments: A stable algorithm. *Phys. Rev. E*, 71:057701, May 2005.

[11] M. Belkin and K. Sinha. Polynomial learning of distribution families. In *FOCS*, 2010.

[12] Y. Ben-Haim and E. Tom-Tov. A streaming parallel decision tree algorithm. *Journal of Machine Learning Research*, 11(Feb):849–872, 2010.

[13] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[14] B. Beyer, C. Jones, J. Petoff, and N. Murphy. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Incorporated, 2016.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[16] L. Braun, T. Etter, G. Gasparis, M. Kaufmann, D. Kossmann, D. Widmer, A. Avitzur, A. Iliopoulos, E. Levy, and N. Liang. Analytics in motion: High performance event-processing and real-time analytics in the same database. In *SIGMOD*, pages 251–264, 2015.

[17] M. Budiu, R. Isaacs, D. Murray, G. Plotkin, P. Barham, S. Al-Kiswany, Y. Boshmaf, Q. Luo, and A. Andoni. Interacting with Large Distributed Datasets Using Sketch. In *Eurographics Symposium on Parallel Graphics and Visualization*, 2016.

[18] C. Buragohain and S. Suri. Quantiles on streams. In *Encyclopedia of Database Systems*, pages 2235–2240. Springer, 2009.

[19] Y. Chen, G. Dong, J. Han, J. Pei, B. W. Wah, and J. Wang. Regression cubes with lossless compression and aggregation. *TKDE*, 18(12):1585–1599, 2006.

[20] S. Cohen. User-defined aggregate functions: bridging theory and practice. In *SIGMOD*, pages 49–60, 2006.

[21] G. Cormode and M. Garofalakis. Streaming in a connected world: querying and tracking distributed data streams. In *SIGMOD*, pages 1178–1181, 2007.

[22] G. Cormode, M. Garofalakis, P. J. Haas, and C. Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[23] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, Apr. 2005.

[24] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A stream database for network applications. In *SIGMOD*, pages 647–651, 2003.

[25] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6):1794–1813, 2002.

[26] J. Dean and L. A. Barroso. The tail at scale. *Commun. ACM*, 56(2):74–80, 2013.

[27] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[28] T. Dunning and O. Ertl. Computing extremeley accurate quantiles using t-digests. `https://github.com/tdunning/t-digest`, 2017.

[29] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. Streamcube: Hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream. In *ICDE*, pages 1561–1572, 2015.

[30] E. Gan, J. Ding, K. S. Tai, V. Sharan, and P. Bailis. Moment-based quantile sketches for efficient high cardinality aggregation queries (extended). Technical report, Stanford University, 2018. `edgan8.github.io/assets/papers/moments-ext.pdf`.

[31] W. Gautschi. Questions of numerical condition related to polynomials. In C. D. Boor and G. H. Golub, editors, *Recent Advances in Numerical Analysis*, pages 45 – 72. Academic Press, 1978.

[32] W. C. Gibson. *The method of moments in electromagnetics*. CRC press, 2014.

[33] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53, 1997.

[34] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *SIGMOD*, volume 30, pages 58–66, 2001.

[35] A. Hall, A. Tudorica, F. Buruiana, R. Hofmann, S.-I. Ganceanu, and T. Hofmann. Trading off accuracy for speed in powerdrill. In *ICDE*, pages 2121–2132, 2016.

[36] L. P. Hansen. Large sample properties of generalized method of moments estimators. *Econometrica*, pages 1029–1054, 1982.

[37] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *SIGMOD*, SIGMOD '96, pages 205–216, 1996.

[38] D. N. Hill, H. Nassif, Y. Liu, A. Iyer, and S. Vishwanathan. An efficient bandit algorithm for realtime multivariate optimization. In *KDD*, pages 1813–1821, 2017.

[39] T. Hunter, H. Falaki, and J. Bradley. Approximate algorithms in apache spark: Hyperloglog and quantiles. `https://databricks.com/blog/2016/05/19/approximate-algorithms-in-apache-spark-hyperloglog-and-quantiles.html`, 2016.

[40] T. Italia. Telecommunications - sms, call, internet - mi, 2015. `http://dx.doi.org/10.7910/DVN/EGZHFV`.

[41] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

[42] R. Johari, P. Koomen, L. Pekelis, and D. Walsh. Peeking at a/b tests: Why it matters, and what to do about it. In *KDD*, pages 1517–1525, 2017.

[43] A. T. Kalai, A. Moitra, and G. Valiant. Efficiently learning mixtures of two gaussians. In *STOC*, pages 553–562, 2010.

[44] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *IDCE*, pages 472–483, 2014.

[45] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491, 2003.

[46] V. Khuc. lbfgs4j, 2017. `https://github.com/vinhkhuc/lbfgs4j`.

[47] W. Kong and G. Valiant. Spectrum estimation from samples. *Ann. Statist.*, 45(5):2218–2247, 10 2017.

[48] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[49] M. Lichman. UCI machine learning repository, 2013.

[50] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989.

[51] S. Liu, B. Song, S. Gangam, L. Lo, and K. Elmeleegy. Kodiak: Leveraging materialized views for very low-latency analytics over high-dimensional web-scale data. *VLDB*, 9(13):1269–1280, 2016.

[52] G. Luo, L. Wang, K. Yi, and G. Cormode. Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *VLDB*, 25(4):449–472, 2016.

[53] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. 2002.

[54] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD*, pages 287–298, 2005.

[55] V. Markl, P. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tran. Consistent selectivity estimation via maximum entropy. *VLDB*, 16(1):55–76, 2007.

[56] J. C. Mason and D. C. Handscomb. *Chebyshev polynomials*. CRC Press, 2002.

[57] L. R. Mead and N. Papanicolaou. Maximum entropy in the problem of moments. *Journal of Mathematical Physics*, 25(8):2404–2417, 1984.

[58] R. M. Mnatsakanov. Hausdorff moment problem: Reconstruction of distributions. *Statistics & Probability Letters*, 78(12):1612 – 1618, 2008.

[59] D. Moritz, D. Fisher, B. Ding, and C. Wang. Trust, but verify: Optimistic visualizations of approximate queries for exploring big data. In *CHI*, pages 2904–2915, 2017.

[60] S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, Aug. 2005.

[61] A. Nandi, C. Yu, P. Bohannon, and R. Ramakrishnan. Distributed cube materialization on holistic measures. In *ICDE*, pages 183–194. IEEE, 2011.

[62] K. Nigam. Using maximum entropy for text classification. In *IJCAI Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[63] C. Ordonez, Y. Zhang, and W. Cabrera. The gamma matrix to summarize dense and sparse data sets for big data analytics. *TKDE*, 28(7):1905–1918, 2016.

[64] W. H. Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[65] A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman. Aggregation and degradation in jetstream: Streaming analytics in the wide area. In *NSDI*, pages 275–288, 2014.

[66] S. Racz, A. Tari, and M. Telek. A moments based distribution bounding method. *Mathematical and Computer Modelling*, 43(11):1367 – 1382, 2006.

[67] N. Ray. The art of approximating distributions: Histograms and quantiles at scale. `http://druid.io/blog/2013/09/12/the-art-of-approximating-distributions.html`, 2013.

[68] F. Rusu and A. Dobra. Glade: A scalable framework for efficient analytics. *SIGOPS Oper. Syst. Rev.*, 46(1):12–18, Feb. 2012.

[69] S. Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, volume 2000, pages 307–316, 2000.

[70] J. Shanmugasundaram, U. Fayyad, and P. S. Bradley. Compressed data cubes for olap aggregate query approximation on continuous dimensions. In *KDD*, pages 223–232, 1999.

[71] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *International conference on Embedded networked sensor systems*, pages 239–249, 2004.

[72] R. Silver and H. Röder. Calculation of densities of states and spectral functions by chebyshev recursion and maximum entropy. *Physical Review E*, 56(4):4822, 1997.

[73] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran. Isomer: Consistent histogram construction using query feedback. In *ICDE*, pages 39–39, 2006.

[74] M. Vartak, S. Rahman, S. Madden, A. Parameswaran, and N. Polyzotis. Seedb: Efficient data-driven visualization recommendations to support visual analytics. *VLDB*, 8(13):2182–2193, 2015.

[75] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages I–511–I–518. IEEE, 2001.

[76] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

[77] L. Wang, G. Luo, K. Yi, and G. Cormode. Quantiles over data streams: An experimental study. In *SIGMOD*, SIGMOD '13, pages 737–748, 2013.

[78] A. Wasay, X. Wei, N. Dayan, and S. Idreos. Data canopy: Accelerating exploratory statistical analysis. In *SIGMOD*, pages 557–572, 2017.

[79] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010.

[80] R. Xi, N. Lin, and Y. Chen. Compression and aggregation for logistic regression analysis in data cubes. *TKDE*, 21(4):479–492, 2009.

[81] X. Xie, X. Hao, T. B. Pedersen, P. Jin, and J. Chen. Olap over probabilistic data cubes i: Aggregating, materializing, and querying. In *ICDE*, pages 799–810, May 2016.

[82] F. Yang, E. Tschetter, X. Léauté, N. Ray, G. Merlino, and D. Ganguli. Druid: A real-time analytical data store. In *SIGMOD*, pages 157–168, 2014.

[83] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 2–2, 2012.

[84] Z. Zhuang. An experimental study of distributed quantile estimation. *ArXiv*, abs/1508.05710, 2015.

APPENDIX

A. SUPPLEMENTAL DETAILS

A.1 Maximum Entropy Estimation

A.1.1 Newton's Method Details

Recall that we wish to solve the following optimization problem:

$$\underset{f \in \mathcal{F}[x_{\min}, x_{\max}]}{\text{maximize}} \; H[f]$$
$$\text{subject to} \int_{x_{\min}}^{x_{\max}} x^i f(x)\,dx = \mu_i, \qquad i \in \{1, \ldots, k_1\}$$
$$\int_{x_{\min}}^{x_{\max}} \log^i(x) f(x)\,dx = \nu_i, \quad i \in \{1, \ldots, k_2\}$$

.

Throughout this section it is easier to reformulate this problem in more general terms, using the functions

$$m_i(x) = \begin{cases} x^i & 0 \le i \le k_1 \\ h(x)^{i-k_1} & k_1 + 1 \le i \le k_1 + k_2. \end{cases} \tag{8}$$

where $h(x) = \log(x)$ or $h(x) = e^x$ depending on whether we work using the x or log-transformed $x' = \log(x)$ as our primary metric. Letting $k_t = k_1 + k_2$, and folding the ν_i into a larger $\vec{\mu}$ vector, our optimization problem is then:

$$\underset{f \in \mathcal{F}[x_{\min}, x_{\max}]}{\text{maximize}} \; H[f] \tag{9}$$
$$\text{subject to} \int_{x_{\min}}^{x_{\max}} m_i(x) f(x)\,dx = \mu_i, \quad i \in \{0, \ldots, k_t\}$$

.

Functional analysis [41] tells us that a maximal entropy solution to Eq. (9) has the form:

$$f(x; \theta) = \exp\left(\sum_{i=0}^{k_t} \theta_i m_i(x)\right),$$

Then if we define the potential function $L(\theta)$ from [57]:

$$L(\theta) = \int_{x_{\min}}^{x_{\min}} \exp\left(\sum_{i=0}^{k_t} \theta_i m_i(x)\right) - \sum_{i=0}^{k_t} \theta_i \mu_i \tag{10}$$

We can calculate the gradient and Hessian of $L(\theta)$ as follows:

$$\frac{\partial L}{\partial \theta_i} = \int_{x_{\min}}^{x_{\min}} m_i(x) \exp\left(\sum_{i=0}^{k_t} \theta_i m_i(x)\right) - \mu_i \tag{11}$$

$$\frac{\partial^2 \Gamma}{\partial \theta_i \partial \theta_j} = \int_{x_{\min}}^{x_{\min}} m_i(x) m_j(x) \exp\left(\sum_{i=0}^{k_t} \theta_i m_i(x)\right) \tag{12}$$

Note that when the gradient given in Eq. (11) is zero then the constraints in Eq. (9) are satisfied. Since $L(\theta)$ is convex and has domain $\mathbb{R}^{k_d}$, this means that by solving the unconstrained minimization problem over $L(\theta)$ we can find a solution θ we can find a solution to the constrained maximum entropy problem. Since Newton's method is a second order method, we can use Equations (10), (11), (12) are to execute Newton's method with backtracking line search [15].

A.1.2 Practical Implementation Details

Chebyshev Polynomial Basis Functions.

As described in Section 4.3, we can improve the stability of our optimization problem by using Chebyshev polynomials. To do so, we must redefine our $m_i(x)$

$$m_i(x) = \begin{cases} T_i(s_1(x)), & i \in \{1, \ldots, k_1\} \\ T_{i-k_1}(s_2(h(x))), & i \in \{k_1 + 1, \ldots, k_1 + k_2\} \end{cases}$$

where $T_i(x)$ are Chebyshev polynomials of the first kind [64] and the s_1, s_2 are linear scaling functions to map onto $[-1, 1]$ defined as:

$$s_1(x) = \left(x - \frac{x_{\max} + x_{\min}}{2}\right) / \left(\frac{x_{\max} - x_{\min}}{2}\right)$$
$$s_2(x) = \left(x - \frac{h(x_{\max}) + h(x_{\min})}{2}\right) / \left(\frac{h(x_{\max}) - h(x_{\min})}{2}\right).$$

The formulae for $\mu_i, \Gamma, \nabla\Gamma, \nabla^2\Gamma$ still hold, but now the μ_i are

$$\mu_i = \frac{1}{n} \begin{cases} \sum_x T_i(s_1(x)) & 0 \le i \le k_1 \\ \sum_x T_{i-k_a}(s_2(h(x))) & k_1 + 1 \le i \le k_1 + k_2. \end{cases} . \tag{13}$$

These can be computed from the quantities $\mu_i = \sum_x x^i$, $\nu_i = \sum_x h(x)^i$ originally stored in the moments sketch by using the binomial expansion and standard formulae for expressing Chebyshev polynomials in terms of standard monomials [56].

Chebyshev Polynomial Integration.

In this section we will show how Chebyshev approximation provides for efficient ways to compute the gradient and Hessian. Here it is easier to work with the change of variables $u = s_1(x)$ so that $f^u(u) = f(s_1^{-1}(u))$ has domain $[-1, 1]$. First, we will examine the case when $k_2 = 0$. If we can approximate $f(u; \theta)$ as a linear combination of chebyshev polynomials:

$$f^u(u; \theta) = \exp\left(\sum_{i=0}^{k_t} \theta_i m_i(u)\right) \tag{14}$$
$$\approx \sum_{j=0}^{n_c} c_j T_j(u) \tag{15}$$

Then using polynomial identities such as $T_i(x)T_j(x) = \frac{1}{2}(T_{i+j}(x) + T_{|i-j|}(x))$ we can evaluate the Gradient and Hessian in Eqs. (11), (12) using $O(k_1 n_c)$ algebraic operations.

We can approximate $f^u(u; \theta)$ using the Clenshaw Curtis quadrature formulas to approximate a function f supported on $[-1, 1]$ (Eq. 5.9.4 in [64]):

$$a_j = \frac{2}{n_c}\left(\frac{f(1)}{2} - \frac{f(-1)}{2} + \sum_{i=1}^{n_c - 1} f\left[\cos\left(\frac{\pi i}{n_c}\right)\right] \cos\left(\frac{\pi i}{n_c}\right)\right) \tag{16}$$

Then

$$f(x) \approx \frac{1}{2} T_0(x) + \sum_{i=1}^{n_c} a_i T_i(x) \tag{17}$$

where Eq. (16) can be evaluated in $n_c \log n_c$ time using the Fact Cosine Transform [64]. The case when $k_2 > 0$ is similar except we need to approximate not just $f^u(u; \theta)$ but also $T_i(s_2(h(s_1^{-1}(u)))) f^u(u; \theta)$ for $i \le k_2$.

A.2 Numeric Stability of Moments

As mentioned in Section 4.3.1, floating point arithmetic limits the usefulness of higher moments. This is because the raw moments $\frac{1}{n}\sum x^i$ are difficult to optimize over and analyze: both maximum entropy estimation (Section 4.3) and theoretical error bounds (Section 4.4) apply naturally to moments on data in the range $[-1, 1]$. In particular, shifting the data improves the conditoning of the optimization problem dramatically. However, when merging sketches from different datasets, users may not know the full range of the data ahead of time, so the power sums stored in a moments sketch correspond to data in an arbitrary range $[a, b]$. Thus, we will analyze how floating point arithmetic affects the process of scaling and shifting the data so that it is supported in the range $[-1, 1]$. This is similar to the process of calculating a variance by shifting the data so that it is centered at zero.

We can calculate the moments of scaling data $x_{scale} = k \cdot x$ with error only in the last digit, so we can assume we have data in the range $[c-1, c+1]$. Let μ_i be the moments of the x_{scale}, and let μ_i^s be the moments of the shifted data $x_{shift} = x_{scale} - c$. Using the binomial expansion

$$\mu_k^s = \frac{1}{n}\sum_{x \in x_{scale}} (x-c)^k \tag{18}$$

$$= \sum_{i=0}^{k} \binom{k}{i} \mu_i (-c)^{k-i} \tag{19}$$

Using numerically stable addition, we can calculate μ_k to a relative precision ϵ close to machine precision $\epsilon \approx 2^{-53}$. Then, the absolute error ϵ_k in estimating μ_k^s is bounded by:

$$\epsilon_k \leq \sum_{i=0}^{k} \binom{k}{i} |\mu_i| |c|^{k-i} \epsilon \tag{20}$$

$$\epsilon_k \leq \sum_{i=0}^{k} \binom{k}{i} (|c|+1)^k \epsilon \tag{21}$$

$$\leq 2^k (|c|+1)^k \epsilon \tag{22}$$

We know that the average quantile error is bounded by $O\left(\frac{1}{k} + 3^k \|\mu_f - \mu_{\hat{f}}\|_2\right)$ (Equation 7 in Section 4.4), so if we can calculate all of the μ_i^s to within precision $3^{-k}\left(\frac{1}{k-1} - \frac{1}{k}\right)$ then we have enough precision to bound the quantile estimate by $O\left(\frac{1}{k-1}\right)$. This way, we can show that the error bound from using the first k moments will be at least as tight as the bound from using the first $k-1$ moments. As k and $|c|$ grow, achieving this precision becomes more and more difficult, and we can solve for the cutoff point using base-10 log.

$$2^k (|c|+1)^k \epsilon \leq 3^{-k}\left(\frac{1}{k-1} - \frac{1}{k}\right) \tag{23}$$

$$k\left(\log 6 + \log\left(|c|+1\right)\right) \leq \log\frac{1}{\epsilon} - \log\left(k^2 - k\right)) \tag{24}$$

Plugging in double precision for ϵ into Eq. (24), we know that $k \leq \frac{53 \log 2}{\log 6} \leq 20$, so $\log\left(k^2 - k\right) \leq 2.58$

$$k \leq \frac{53 \log 2 - 2.58}{\log 6 + \log\left(|c|+1\right)} \tag{25}$$

$$\leq \frac{13.35}{.78 + \log\left(|c|+1\right)} \tag{26}$$

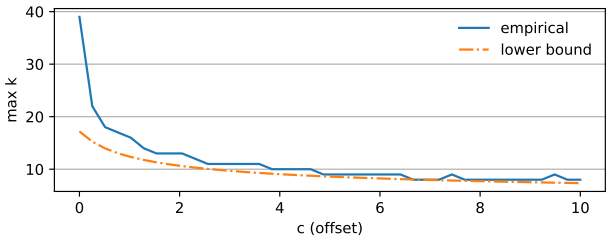

Figure 15: Highest order usable moments for data centered at different locations. Our data-independent bound is conservative compared to values on a uniform dataset.

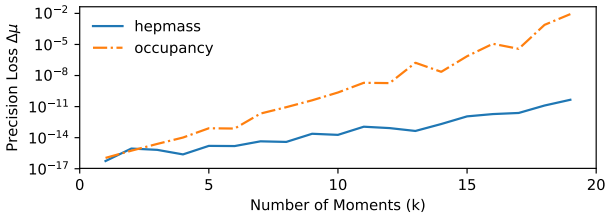

Figure 16: Precision loss from shifting and converting higher moments to chebyshev moments. The occupancy dataset exhibits more precision loss because it is centered further away from zero.

Equation 26 is a conservative bound on the number of numerically stable moments we can extract from an moments sketch, and suggests that when our data is centered at 0, we have at least 17 stable moments. When the raw data have range $[x_{\min}, 3x_{\min}]$, then $c = 2$, and we have at least 10 stable moments. In our evaluations, 10 stable moments are enough to achieve quantile error $\approx .01$. Figure 15 describes how the bound in Equation 26 varies with c, and compares it with the highest order stable moment of a uniform distribution supported on $[c-1, c+1]$. This confirms that our formula is a conservative bound on the true precision loss due to the shift. If the raw data are centered even further from 0, users can consider pre-shifting all of their data to make better use of numeric precision.

As a measure of the downstream impact of this effect on some of our evaluation datasets, Figure 16 shows the precision loss during Chebyshev polynomial calculation $\Delta\mu = |\mu_i - \hat{\mu}_i|$ where μ_i is the true Chebyshev moment and $\hat{\mu}_i$ is the value calculated from the moments sketch. Precision loss is more severe on the occupancy dataset which is centered away from zero ($c \approx 1.5$) compared with the hepmass dataset ($c \approx 0.4$). See Table 1.

A.3 Low-precision storage

In section A.2 we discussed how floating point precision limits the usability of higher moments. Conversely, in settings where space is heavily constrained, the data is well-centered, and we only need a limited number of moments, the moments sketch can be compressed by reducing the precision of the sketch contents using randomized rounding.

As a proof-of-concept of this approach, we created an encoder that compresses the double precision floating point values in a moments sketch using reduced floating point precision, quantizing the significand and removing unused bits in the exponent. This low-precision representation has a

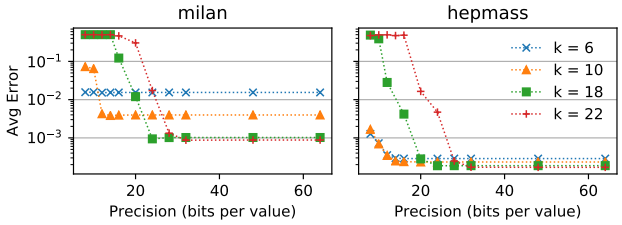

Figure 17: Average error for low-precision moments sketches after 100 thousand merges. Twenty bits of precision is sufficient to maintain accuracy for both datasets.

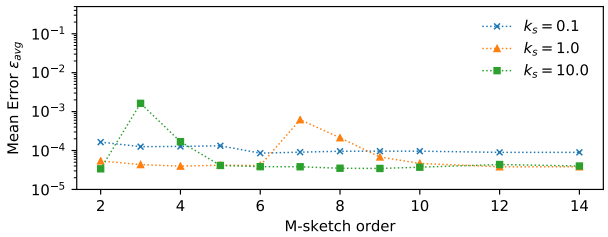

Figure 18: Accuracy of estimates on Gamma distributions with varying shape parameter k_s. The maximum entropy principle is able to construct an accurate estimate across a range of parameters.

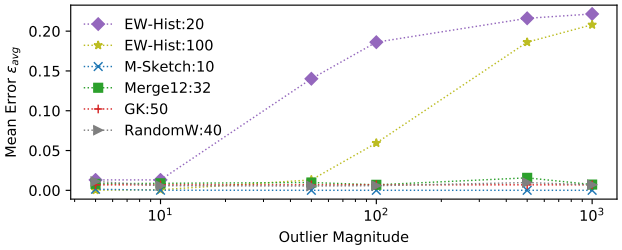

Figure 19: Mean error on a Gaussian dataset with outliers of different magnitudes added. The moments sketch remains accurate for large outliers, but the `EW-Hist` accuracy degrades.

negligible impact on merge times since we can convert them to and from native double precision using simple bit manipulation.

We evaluate the encoding by constructing 100 thousand pre-aggregated moments sketches, reducing their precision, and then merging them and querying for quantiles on the aggregation. Figure 17 illustrates how the quality of the final estimate remains stable as the precision is decreased until we reach a minimum threshold, after which accuracy degrades. On the milan dataset, a moments sketch with $k = 10$ can be stored with 20 bits per value without noticeably affecting our quantile estimates, representing a 3× space reduction compared to standard double precision floating point.

These results are consistent with the bounds in Section A.2 and show how higher moments require more bits of precision. However, the bounds are conservative since they only consider the midpoint c of a dataset and are otherwise both dataset-independent and agnostic to the maximum entropy principle.

A.4 Varying Distribution Parameters

Our usage of log-moments greatly reduces the impact of data skew on the accuracy of moments sketch quantile estimates. In Figure 18 we vary the shape parameter k_s of a Gamma distribution with scale factor $\theta = 1$. The skew of this distribution is $\frac{2}{\sqrt{k_s}}$ so $k_s = 0.1$ corresponds to very high skew. For $k_s = 0.1, 1.0, 10.0$, our estimator can achieve $\epsilon_{avg} \leq 10^{-3}$ error. The accuracy regressions on orders 3 and 7 occur when the solver stops making use of all available moments to reduce the condition number of the Hessian (Section 4.3). In this specific case, our solver uses a heuristic to decide that given a maximum condition number, optimizing using 3 log moments is more valuable than 2 log moments and 2 standard moments. This choice leads to worse accuracy on a Gamma distribution, but in general it is difficult to know which subset of moments will yield the most accurate estimate. More effective heuristics for choosing subsets of moments that do not exceed a condition number threshold is an open direction for future research.

The moments sketch, unlike histogram-based sketches, is also somewhat robust to the presence of large outlier values in a dataset. In Figure 19 we evaluate the effect of adding a fixed fraction $\delta = 0.01$ of outlier values from a Gaussian with mean μ_o and standard deviation $\sigma = 0.1$ to a dataset of 10 million standard Gaussian points. As we increase the magnitude μ_o of the outliers, the `EW-Hist` summaries with 20 and 100 bins lose accuracy though a moments sketch with $k = 10$ remains accurate. The `Merge12` sketch is agnostic to value magnitudes and is unaffected by the outliers. If extremely large outliers are expected, floating point precision suffers and the moments sketch can be used in conjunction with standard outlier removal techniques.

A.5 Error Upper Bounds

Thus far we have evaluated observed accuracy. For comparison, Figure 20 shows the average guaranteed upper bound error provided by different summaries constructed using pointwise accumulation on the datasets (no merging). These are in general higher than the observed errors. We use the RTTBound routine in Section 5.1 to bound the moments sketch error. We omit the `S-Hist` since it does not provide upper bounds. When merging is not a concern, the `GK` summary provides the best guaranteed error.

A.6 Parallel Merges

In Section 6.2.1, we evaluated merge time through single-threaded experiments. We evaluate how well throughput generalizes to parallel aggregation by sharding pre-computed

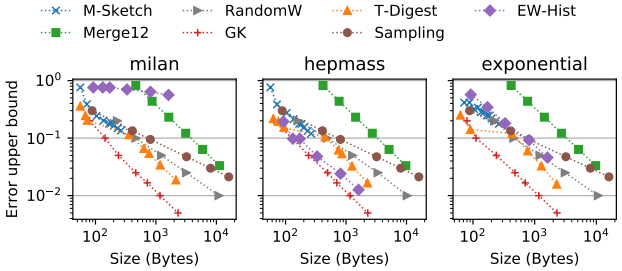

Figure 20: Average bound size for summaries of different sizes. No summary is able to provide $\epsilon_{bound} \leq .01$ guarantees with less than 1000 bytes.

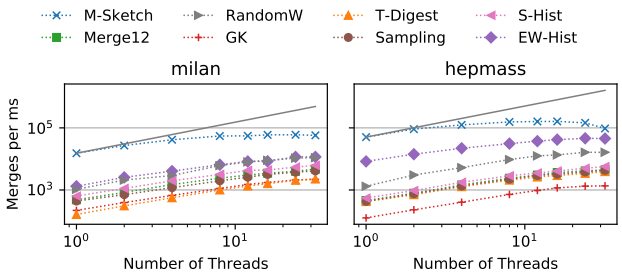

Figure 21: Strong scaling of parallel merging. For fixed number of merges, the throughput of the moments sketch scales with the number of threads available up to 8-way parallelism, and remains faster than alternatives. The solid line shows ideal moments sketch scaling.

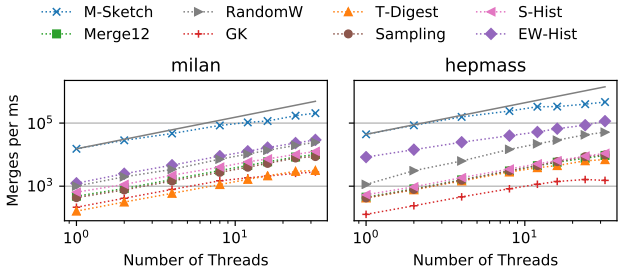

Figure 22: Weak scaling of parallel merging. For fixed number of merges per thread, the moments sketch and other summaries scale nearly linearly with parallelism.

summaries into equal sized batches, and merging the summaries in each batch on an independent worker thread. After all threads have completed, we combine the merged result from each batch using a sequential merge to obtain a final summary for the complete dataset.

In Figure 21 we evaluate strong scalability by measuring the total throughput in merging 400 thousand summaries (constructed from blocks of 200 elements) as we increase the number of threads. In our experiments we needed to duplicate the hepmass dataset to yield 400 thousand summaries, and initialized summaries using the parameters in Table 2. The moments sketch remains faster than alternate summaries as we increase the amount of parallelism, though thread overheads and variance in stragglers limits parallelism on these datasets past 8 threads when there is less work per thread. In Figure 22 we evaluate weak scalability by performing a similar experiment but increase the dataset size alongside thread count, keeping number of merges per thread constant. Under these conditions, the moments sketch and other summaries achieve even better scalability.

These experiments confirm our intuition that since merges can be performed independently, single-threaded performance is indicative of parallel performance, and the relative speedup provided by the moments sketch remains stable in parallel settings. The moments sketch and other summaries can be used in more sophisticated distributed aggregation plans as well, such as in [17, 68], though since the moments sketch is so compact and cheap to merge, multi-level hierarchical aggregation is only profitable when enormous numbers of cores are available.

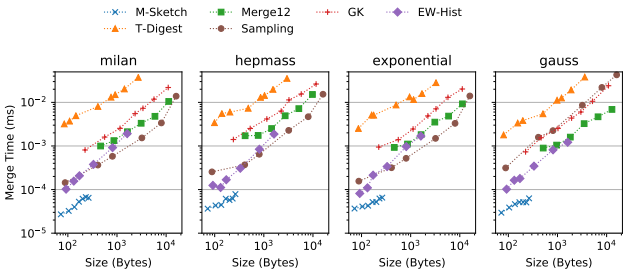

Figure 23: Merge times on with sketches on cells of 2000 elements, and on a Gaussian dataset with cells of 10000 elements. Since the moments sketch has a fixed size, its per-merge times remain faster than alternative sketches with comparable accuracy.

A.7 Varying Cell Size

In our main evaluations, we group our datasets into cells of 200 elements and construct sketches for each cell to maintain a pre-aggregated collection of data summaries. We do not target deployments where very few elements can be pre-aggregated per summary: in these cases merging moments sketches is relatively expensive. On the other hand production data systems can have much larger data volumes and opportunities to pre-aggregate more elements per cell. Since the moments sketch is fixed-size regardless of the data, increasing the number of elements per cell does not affect its merge time performance, while other sketches which which have not reached their maximum capacity will be correspondingly larger and slower to merge.

In Figure 23 we measure the time taken per merge for different summaries constructed on cells of 2000 elements for the milan, hepmass, and exponential dataset, and cells of 10000 elements on a synthetic Gaussian dataset with 1 billion points. The relative performance of different sketches matches closely with Figure 4, except that larger `Sampling` and `Merge12` summaries are now slower when constructed on more than 200 elements.

A.8 Experiments on a Production Dataset

In this section we evaluate merge time and accuracy on a production workload from Microsoft that contains 165 million rows of application telemetry data for an integer-valued performance metric. We group and pre-aggregate by four columns that encode information about application version, network type, location, and time, resulting in 400 thousand cells. These cells have a minimum size of 5 elements, a maximum size of 722044 elements, and an average size of 2380 elements. Figure 24 illustrates the distribution of integer data values and the cell sizes.

Then, we measure the performance and accuracy of merging the cells to perform a quantile aggregation query. Figure 25 illustrates that on this workload with variable sized cells, the moments sketch still provides faster merge times than comparably accurate summaries (c.f. Appendix A.7). The moments sketch achieves $\epsilon_{avg} < .01$ error when we round estimates to the nearest integer on this integral dataset. Since the `GK` sketch is not strictly mergeable [3], it grows considerably when merging the heterogenous summaries in this

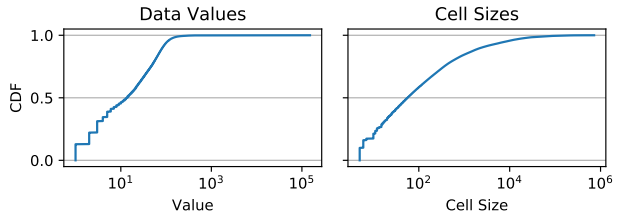

Figure 24: Microsoft data values and cell sizes.

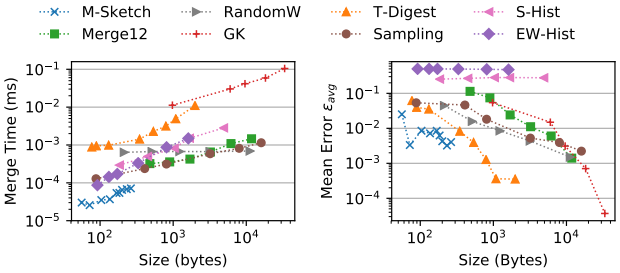

Figure 25: Merge times and accuracy on the Microsoft dataset. The merge performance of the moments sketch generalizes to workloads with variable sized cells, and exhibits an error rate of $\epsilon_{avg} < .01$.

workload to preserve its accuracy.