Non-Parametric Error Estimation for σ-AQP using Optimized Bootstrap Sampling

Feng Yu*†
Youngstown State University, Youngstown, OH 44555, USA
Semih Cal‡
Texas Tech University, Lubbock, TX 79409, USA
En Cheng§
University of Akron, Akron, OH 44325, USA
Lucy Kerns¶
Youngstown State University, Youngstown, OH 44555, USA
Weidong Xiong‖
Cleveland State University, Cleveland, OH 44115, USA

Abstract

Approximate query processing (or AQP) aims to quickly provide approximated answers for time-consuming search queries on large datasets. It brings enormous benefits in data science when the query execution efficiency weighs more than the accuracy. However, assessing the accuracy of an approximated answer from AQP still lacks study. Existing work usually relies on strict dataset assumptions that are often not satisfied in real-world datasets. In this work, we employ a non-parametric statistical method, called bootstrap sampling, to assess errors of an AQP system for selection queries (or σ-AQP). We implement a prototype AQP system integrated with a bootstrap sampling engine that can estimate the standard deviation and produce confidence intervals for selection query estimations. Extensive experiments operating the prototype system demonstrated that the confidence intervals generated can cover the ground truth query results with high accuracy and low computing costs. In addition, we introduce optimization strategies for bootstrap sampling which can improve the overall computing efficiency of the prototype AQP system.

Key Words: Approximate query processing, error estimation, non-parametric method, bootstrap sampling

1 Introduction

Efficient query processing of complex queries on big data posts a demanding challenge for modern data management systems. Much work has been developed towards promptly executing data queries on both hardware and software platforms [9, 21, 22]. However, calculating the exact answer for each data query is expensive and may not be necessary for all scenarios. For example, during the exploratory data analysis (or EDA), a user often only needs approximated answers for a collection of testing queries where the execution speed weighs more than the accuracy.

Approximate query processing (or AQP) is an alternative scheme to provide estimated query answers with satisfying accuracy and within a short time [2, 18, 19]. AQP doesn't need to run the query on the original dataset but can collect statistics to generate query estimations. A common application of AQP is to estimate selection (or σ) queries, called σ-AQP. For selection queries, simple random samples are usually employed for fast and accurate query estimations [1, 24].

An open question for the AQP research is how to efficiently assess the error of query estimation. The challenge is that, for different selection conditions, the underlying distributions of the result sets are different and difficult to predict. This creates an obstacle to efficiently assessing the estimation errors for AQP systems.

Bootstrap sampling [23] is a statistical technique that can assess the errors of sample-based estimators. One advantage of bootstrap sampling is that it doesn't rely on any knowledge of the data distribution to provide error estimation, but can "pull itself up by its bootstrap". It conducts a special sampling method, called resampling, which generates many replicated random samples with replacement, called bootstrap samples, from the original random samples used by AQP. Using the bootstrap samples, common error assessments, such as the standard deviation, of a query estimator can be calculated. An advantage of bootstrap sampling is it doesn't require restricted assumptions of data such as normal distribution used by large number theory. Statistical methods like these are commonly named as non-parametric methods [12].

In this work, we will focus on using bootstrap sampling to assess the estimation error of a σ-AQP system. The contributions of this work are as follows:

1. We propose a framework equipped with the bootstrap sampling method to assess the errors of an AQP system

*Email addresses are to be put first. fyu@ysu.edu, scal@ttu.edu, echeng@uakron.edu, xlu@ysu.edu, w.xiong15@csuohio.edu
†Department of Computer Science and Information Systems.
‡Department of Computer Science.
§Department of Computer Science.
¶Department of Mathematics and Statistics.
‖Department of Electrical Engineering and Computer Science.

for selection queries (or σ-AQP). A prototype system is implemented to simulate a real-world database system that can execute common selection queries. This system is integrated with a bootstrap sampling engine used for non-parametric error assessment.

2. We test the performance of the prototype system on multiple datasets with various combinations of hyper-parameters to simulate real-world scenarios. The experimental results show satisfying accuracy of error assessment.
3. With the findings of the computing bottlenecks of the bootstrap sampling procedure, we propose optimization schemes to improve the overall system performance.

Compared with the conference version work [4], additional contributions are made including:

1. We extended the sections of background and bootstrap sampling framework. We added the related work of AQP and bootstrap sampling.
2. We performed extended experiments of error assessment. Various datasets with skewness were employed in the accuracy tests of bootstrap confidence intervals.
3. Additional analysis of the error assessment experiments is included. The means and standard deviations of the confidence interval hit ratios and the bootstrap standard deviations are presented.

The rest of this work is organized as follows. Section 2 introduces the background of σ-AQP and bootstrap sampling. Section 3 describes how to use bootstrap sampling to assess estimation errors from a sample-based σ-AQP scheme. The implementation of the prototype system incorporating σ-AQP and a bootstrap sampling engine is described in Section 4. Experimental results are presented in Section 5. The related work is included in Section 6. The conclusion and future work are included in Section 7.

2 Background

2.1 σ-AQP

Approximate query processing (or AQP) is the technology to provide approximated answers to complex queries using statistical methods. It aims to provide accurate query estimations within a short time frame. σ-AQP is the AQP focusing on estimating selection (SELECT or σ) queries. Given a selection query Q on a table R, to get the ground truth query answer Y_{GT}, the traditional scheme of query answering is to execute query Q on R which may take a long time when R has a large volume and the query result size is big. Instead of running query Q directly on the original dataset R, σ-AQP takes a simple random sample without replacement (SRSOR) from R, denoted by S, and runs the query Q on S to get a sample result Y_s. In this case, the ground truth, Y_{GT} can then be estimated by

$$Y_{GT} = \frac{Y_s}{f} \tag{1}$$

where $f = |S|/|R|$ is the sampling ratio.

Figure 1 demonstrates an example of simple random sampling without replacement. If the original table includes 100 records, using a 20% sampling ratio, 20 records will be randomly selected without replacement and saved into a sample table. After that, σ-AQP will use the sample table to produce estimations for selection queries with a sampling ratio parameter set to 20%. In practice, the sampling ratio is usually tiny. For example, a sampling ratio of less than 1% is usually employed. For highly skewed data, a larger sampling ratio can be used to increase the accuracy of query estimation.

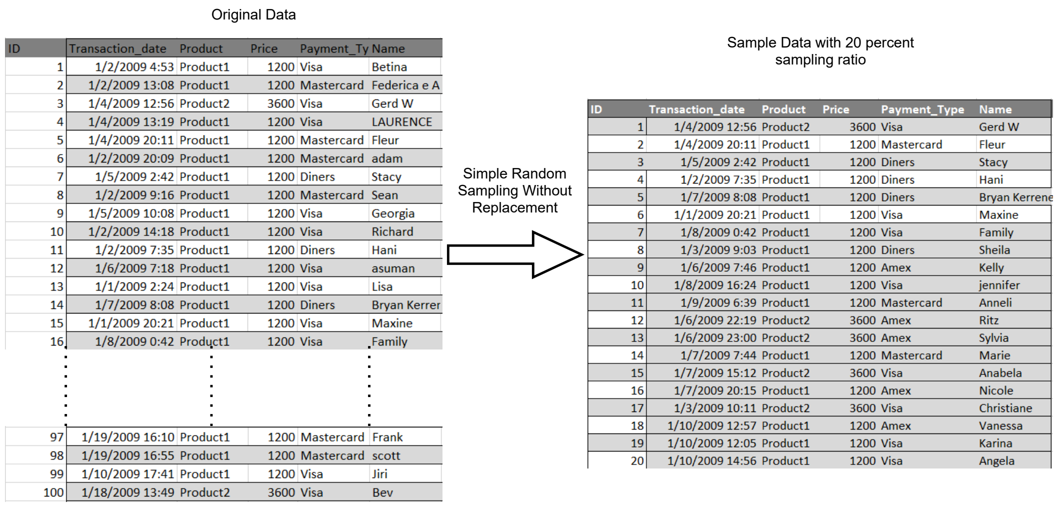
2.2 Bootstrap Sampling

Bootstrap sampling was originally introduced by Bradley Efron in 1979 [10]. It is a computer-assisted method designed to measure the quality of various statistical estimators. Bootstrap sampling generates a collection of new distributions from the original distribution and can derive their variance which can be used to quantify the accuracy of statistical estimators based on the observed data. It works well when the target data is drawn from unknown distributions, which is superior to deriving closed-form methods based on limited data assumptions.

A unique statistical feature in bootstrap sampling is *resampling*. This procedure generates new distributions, called *bootstrap samples*, from a given sampled dataset using simple random sampling with replacement (SRSWR). Each resampled new distribution can produce a scalar called a *bootstrap replication*. Bootstrap sampling generates a large number of bootstrap replications and can use them to estimate the statistical features, such as standard deviation, of the originally given dataset even when the original distribution is unknown.

Figure 2 depicts a simple example of how bootstrap sampling is performed. When given a sample data $\mathbf{y} = (y_1, y_2, ..., y_n)$ from an unknown distribution F, a *bootstrap sample* $\mathbf{y}^* = (y_1^*, y_2^*, ..., y_n^*)$ is a resampled collection obtained by randomly sample n times with replacement from the original sample y_1, y_2, ..., y_n. For instance, if $n = 5$, we might obtain different bootstrap samples, such as $\mathbf{y}_1^* = (y_5, y_3, y_1, y_2, y_1)$, $\mathbf{y}_2^* = (y_2, y_5, y_4, y_1, y_2)$, $\mathbf{y}_3^* = (y_3, y_3, y_2, y_3, y_4)$, etc. These resamples are shown in Figure 2a. Figure 2b depicts a bootstrap sample example.

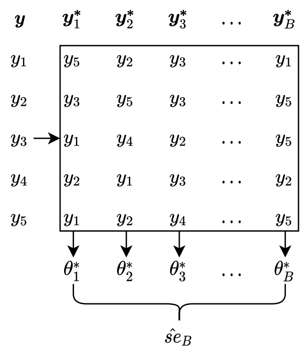
A useful application of bootstrap sampling is to estimate the standard deviation of a statistical estimator from an unknown distribution. Suppose we wish to estimate an unknown population parameter, $\theta = t(F)$, based on the sampled data $\mathbf{y}$, i.e., $\hat{\theta} = s(\mathbf{y})$. We first generate a number of B independent bootstrap samples. Given each bootstrap sample $\mathbf{y}^*$, a *bootstrap replication* of $\hat{\theta}$ is computed as $\hat{\theta}^* = s(\mathbf{y}^*)$. For example, if $\hat{\theta}$ is the sample mean $\bar{\mathbf{y}}$, a bootstrap replication $\hat{\theta}^*$ is the mean of a bootstrap sample $\overline{\mathbf{y}^*}$. The standard error of $\hat{\theta}^*$, i.e. $\widehat{se}_B(\hat{\theta}^*)$, called the *bootstrap estimation of standard error*, can be calculated from the B bootstrap replications as follows.


Original Data
Simple Random Sampling Without Replacement
Sample Data with 20 percent sampling ratio

ID	Transaction_date	Product	Price	Payment_Ty	Name
1	1/2/2009 4:53	Product1	1200	Visa	Betina
2	1/2/2009 13:08	Product1	1200	Mastercard	Federica e A
3	1/4/2009 12:56	Product2	3600	Visa	Gerd W
4	1/4/2009 13:19	Product1	1200	Visa	LAURENCE
5	1/4/2009 20:11	Product1	1200	Mastercard	Fleur
6	1/2/2009 20:09	Product1	1200	Mastercard	adam
7	1/5/2009 2:42	Product1	1200	Diners	Stacy
8	1/2/2009 9:16	Product1	1200	Mastercard	Sean
9	1/5/2009 10:08	Product1	1200	Visa	Georgia
10	1/2/2009 14:18	Product1	1200	Visa	Richard
11	1/2/2009 7:35	Product1	1200	Diners	Hani
12	1/6/2009 7:18	Product1	1200	Visa	asuman
13	1/1/2009 2:24	Product1	1200	Visa	Lisa
14	1/7/2009 8:08	Product1	1200	Diners	Bryan Kerrer
15	1/1/2009 20:21	Product1	1200	Visa	Maxine
16	1/8/2009 0:42	Product1	1200	Visa	Family
	⋮	⋮			⋮
97	1/19/2009 16:10	Product1	1200	Mastercard	Frank
98	1/19/2009 16:55	Product1	1200	Mastercard	scott
99	1/10/2009 17:41	Product1	1200	Visa	Jiri
100	1/18/2009 13:49	Product2	3600	Visa	Bev

ID	Transaction_date	Product	Price	Payment_Type	Name
1	1/4/2009 12:56	Product2	3600	Visa	Gerd W
2	1/4/2009 20:11	Product1	1200	Mastercard	Fleur
3	1/5/2009 2:42	Product1	1200	Diners	Stacy
4	1/2/2009 7:35	Product1	1200	Diners	Hani
5	1/7/2009 8:08	Product1	1200	Diners	Bryan Kerrene
6	1/1/2009 20:21	Product1	1200	Visa	Maxine
7	1/8/2009 0:42	Product1	1200	Visa	Family
8	1/3/2009 9:03	Product1	1200	Diners	Sheila
9	1/6/2009 7:46	Product1	1200	Amex	Kelly
10	1/8/2009 16:24	Product1	1200	Visa	jennifer
11	1/9/2009 6:39	Product1	1200	Mastercard	Anneli
12	1/6/2009 22:19	Product2	3600	Amex	Ritz
13	1/6/2009 23:00	Product2	3600	Amex	Sylvia
14	1/7/2009 7:44	Product1	1200	Mastercard	Marie
15	1/7/2009 15:12	Product2	3600	Visa	Anabela
16	1/7/2009 20:15	Product1	1200	Amex	Nicole
17	1/3/2009 10:11	Product2	3600	Visa	Christiane
18	1/10/2009 12:57	Product1	1200	Amex	Vanessa
19	1/10/2009 12:05	Product1	1200	Visa	Karina
20	1/10/2009 14:56	Product1	1200	Visa	Angela

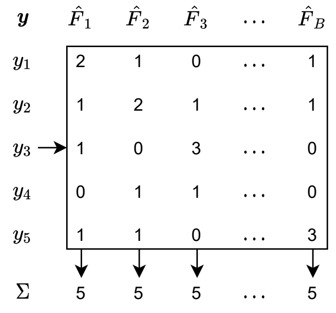
Figure 1: Example: simple random sampling without replacement (SRSWOR) using 20% sampling ratio



$$\begin{array}{c|ccccc} \boldsymbol{y} & \boldsymbol{y}_1^* & \boldsymbol{y}_2^* & \boldsymbol{y}_3^* & \cdots & \boldsymbol{y}_B^* \\ \hline y_1 & y_5 & y_2 & y_3 & \cdots & y_1 \\ y_2 & y_3 & y_5 & y_3 & \cdots & y_5 \\ y_3 & y_1 & y_4 & y_2 & \cdots & y_5 \\ y_4 & y_2 & y_1 & y_3 & \cdots & y_2 \\ y_5 & y_1 & y_2 & y_4 & \cdots & y_5 \\ \hline & \theta_1^* & \theta_2^* & \theta_3^* & \cdots & \theta_B^* \end{array}$$

$$\underbrace{\theta_1^* \quad \theta_2^* \quad \theta_3^* \quad \cdots \quad \theta_B^*}_{\hat{se}_B}$$

(a) Bootstrap samples



$$\begin{array}{c|ccccc} \boldsymbol{y} & \hat{F}_1 & \hat{F}_2 & \hat{F}_3 & \cdots & \hat{F}_B \\ \hline y_1 & 2 & 1 & 0 & \cdots & 1 \\ y_2 & 1 & 2 & 1 & \cdots & 1 \\ y_3 & 1 & 0 & 3 & \cdots & 0 \\ y_4 & 0 & 1 & 1 & \cdots & 0 \\ y_5 & 1 & 1 & 0 & \cdots & 3 \\ \hline \Sigma & 5 & 5 & 5 & \cdots & 5 \end{array}$$

(b) Bootstrap distributions

Figure 2: Example: bootstrap sampling

$$\widehat{se}_B(\hat{\theta}^*) = \left[\frac{1}{B-1} \sum_{i=1}^{B} \left(\hat{\theta}_i^* - \bar{\theta}^* \right)^2 \right]^{\frac{1}{2}} \tag{2}$$

where $\bar{\theta}^* = \sum_{i=1}^{B} \hat{\theta}_i^* / B$.

When $B \to \infty$, we have $\widehat{se}_B(\hat{\theta}^*) \to se_{\hat{F}}(\hat{\theta}^*)$, where $se_{\hat{F}}(\hat{\theta}^*)$ is called the *ideal bootstrap estimation* of the ground truth standard error of $\hat{\theta}$, i.e. $se_F(\hat{\theta})$. Both $se_{\hat{F}}(\hat{\theta}^*)$ and its approximation $\widehat{se}_B(\hat{\theta}^*)$ are called *non-parametric bootstrap* estimates since they are generated from the distributions, $\hat{F}$, which are non-parametric estimates of the ground truth population F.

3 Bootstrap for Selection Query Error Estimation

3.1 Selection Query Estimation

We consider the following query formulation in this research:

```
Q: SELECT Aggregation(attribute collection)
FROM table_name WHERE conditions
```

After a query Q is executed on the sample table S, each sample tuple $u_i \in S$ will produce a tuple query result y_i based on the aggregation function. For example, if the aggregation function is COUNT, then y_i is either 1 if u_i satisfies the selection condition or 0 otherwise. The query result Y_s on the sample table S is calculated as $Y_s = \sum_{i=1}^{n} y_i$, where $n = |S|$ is the sample size. Suppose the size of the original table R is N, and the sample fraction $f = \frac{n}{N}$, then the estimation of the query result ground truth is

$$\widehat{Y} = \frac{Y_s}{f} \tag{3}$$

This estimation works well if the original table R has low skewness and the sample S is uniformly collected from R. Otherwise, the accuracy may be low when data is highly skewed or the sample S is not uniformly distributed (or even includes correlation).

3.2 Bootstrap Sampling from Query Results

After the sample query results $S_Q = \{y_i\}_{i=1}^{n}$ are obtained by executing Q on the sample relation S, bootstrap samples $\{\mathbf{y}_j^*\}_{j=1}^{B}$ can be generated for error estimation, where B is the total times of bootstrap sampling. Each $\mathbf{y}_j^* = \{y_{j,i}^*\}_{i=1}^{n}$ a bootstrap sampling of S_Q, where each query result $y^*_{j,i}$, $1 \le i \le n$, is randomly sampled with replacement from S_Q.

To obtain the bootstrap replication, we use the same estimator in Eq (2) on each $\mathbf{y}^*_j$, $j = 1, ..., B$ as

$$\widehat{Y}^*_j = \frac{Y_{\mathbf{y}^*_j}}{f} \tag{4}$$

For example, if the aggregation is COUNT, then the estimator is

$$\widehat{Y}^*_j = \frac{1}{f}\sum_{i=1}^{n} y^*_{j,i} \tag{5}$$

After repeating the bootstrap sampling for B times, a collection of bootstrap replications is obtained, denoted by $\widehat{Y}^*_B = \{\widehat{Y}^*_j\}^B_{j=1}$. The bootstrap standard deviation is calculated as

$$\widehat{se}_B = \left[\frac{\sum_{j=1}^{B}(\widehat{Y}^*_j - \overline{\widehat{Y}^*_B})^2}{B-1}\right]^{\frac{1}{2}} \tag{6}$$

where $\overline{\widehat{Y}^*_B}$ is the sample mean of all bootstrap replications $\widehat{Y}^*_B$. By the theory of bootstrap sampling, we claim that Eq (6) is the *bootstrap estimation of the standard error* of $\widehat{Y}$ which estimates the query result Y_{GT} of query Q.

3.3 Computing the Confidence Interval

There are different methods in bootstrap sampling to generate a confidence interval (or CI), such as the normal-theory CI, bootstrap percentile CI, and basic bootstrap CI. There are also improved CI methods to increase the accuracy such as the Bias-Corrected and Accelerated interval (BC$_a$) and Approximate Bootstrap Confidence interval (ABC) [10]. In this work, we implemented the normal-theory CI method, which is calculated as

$$\left(\widehat{Y} - z_{\alpha/2}\cdot\widehat{se}_B, \widehat{Y} + z_{\alpha/2}\cdot\widehat{se}_B\right) \tag{7}$$

where $\widehat{Y}$ is the query estimation from AQP, $1-\alpha \in [0,1]$ is the confidence level, and $z_{\alpha/2}$ is the upper-$\alpha/2$ standard normal critical point. For example, for a 90% confidence level (i.e., $\alpha = .10$), $z_{.05} = 1.645$, and for a 95% confidence level, $z_{.025}$=1.960.

4 Implementation

We propose a prototype AQP system with the ability to generate error estimations using bootstrap sampling. The prototype system consists of the following parts: a simple query processor, a query execution engine for selection, a σ-AQP engine using simple random sampling, and a bootstrap engine for error estimation. The architecture of the query processor is depicted in Figure 3.

The implemented query processor reads queries from a plain text file and executes them accordingly. The query execution engine reads each tuple from the table data file and produces a tuple query result by checking whether it satisfies the selection

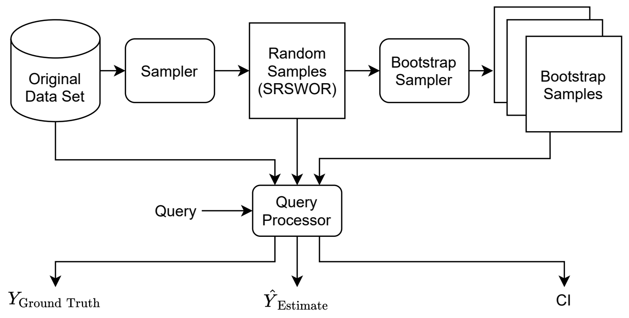


Figure 3: Prototype σ-AQP framework with a bootstrap sampling engine

condition. Summarizing all tuple query results will produce the final query result.

The σ-AQP engine of this system has two functions: generates a sample table S from the base table R using simple random sampling without replacement (simple random sampler) and provide query estimations using the sample table (sample estimator). When sampling starts, a series of random row numbers will be generated in an array and the sampler will access the base table file and retrieve only the tuples in the random number array. Depending on the volume of sample tuples, if the sample tuples cannot fit into the memory, the sampler will output the sampled tuples into the sample table file in batches. Otherwise, the sampled tuples will be read in one batch and saved into the sample table.

After the sample tuples are drawn, the sample estimator of the σ-AQP engine can produce a query estimation by first executing the original query Q on the sample table S and getting a sample result set Y_s. The query execution engine will be called to run the query on the sample table S and the sample query results S_Q will be generated. The estimation of the query result, $\widehat{Y}_{est}$, will be calculated using Eq (3). The bootstrap engine will perform bootstrap sampling on the sample query results S_Q, calculate the bootstrap standard deviation $\widehat{se}_B$, and produce the bootstrap confidence interval (CI) using Eq (7).

5 Experiment

We present the experimental results in this section. First, we test the error assessment accuracy of the implemented prototype AQP system. Second, we investigate the performance of the bootstrap sampling procedure during the accuracy tests. Finally, we present the performance results using optimized bootstrap sampling methods.

5.1 Experiment Setup

The experiment server is equipped with an Intel Xeon E5-1620 v4 CPU and 8GB of RAM and runs CentOS 7 Linux. The experiment code is written in C and Python languages. The major prototype components such as query parsing, query processing, AQP, and bootstrap sampling module

Table 1: Test queries for accuracy experiments

No.	Query
1	select count(*) from lineitem where L_QUANTITY <20 and L_QUANTITY >0
2	select count(*) from lineitem where L_LINENUMBER <3 and L_LINENUMBER >0
3	select count(*) from lineitem where L_LINENUMBER <5 and L_LINENUMBER >2
4	select count(*) from lineitem where L_DISCOUNT <.07 and L_DISCOUNT >.02
5	select count(*) from lineitem where L_EXTENDEDPRICE <100000.00 and L_EXTENDEDPRICE >20000.00
6	select count(*) from lineitem where L_DISCOUNT <.04 and L_DISCOUNT >0.0
7	select count(*) from lineitem where L_QUANTITY <20 and L_QUANTITY >10
8	select count(*) from lineitem where L_DISCOUNT <.05 and L_DISCOUNT >.02
9	select count(*) from lineitem where L_EXTENDEDPRICE <15000.00 and L_EXTENDEDPRICE >0.0
10	select count(*) from lineitem where L_LINENUMBER <2 and L_LINENUMBER >0

are implemented in C. The driver programs for experiments are written in Python. The source code of the prototype system and experiments are available on GitHub[1].

The tests datasets are generated using the TPC-H benchmark with skew[2] [8] which is widely used for data querying tests. We focused on testing SELECT (or σ) queries for σ-AQP error assessment and we chose the largest table, namely `LINEITEM`, in a TPC-H database. We generated multiple TPC-H datasets (only including the `LINEITEM` table) in volumes of 100MB, 1GB, and 10GB and with skewness of 0 (no skew) and 1 (highly skewed), respectively. We randomly generated 10 test queries with different selection ranges listed in Table 1. Among them, five queries are large-range selection queries and five queries are small-range selection queries.

5.2 Bootstrap Accuracy Tests

We estimate each test query using the implemented AQP system which produces a 95% level bootstrap CI as a range estimation. The implemented query processor computes the ground truth of the query, i.e. Y_{GT}, on the original dataset. If the Y_{GT} is contained in the CI, it's considered a "hit"; otherwise, it's a "miss". For each test query, we repeatedly generate the bootstrap CI for 10 times and calculate the averaged hit ratio as follows.

$$\text{hit ratio} = \frac{\text{count(CI includes } Y_{GT})}{\text{count(overall experiments)}} \times 100\% \qquad (8)$$

Figure 4 depicts the results of accuracy tests using bootstrap sampling. Each small figure depicts the result on one test dataset using different sampling ratios (f) including 0.1%, 0.5%, and 1%. To study how the bootstrap iterations (B) affect the hit ratios, we use compare tests with B=200 and B=2000 (recommended in [10]). In addition, to study how the data skew (z) affects the hit ratios, we compute hit ratios with different skewness values, z=0 (no skew) and z=1 (highly skewed).

[1]The experiment code is available at `https://github.com/YSU-Data-Lab/Semih_Cal_Thesis_Summer_2021`

[2]We employed the TPC-H toolkit available at `https://github.com/YSU-Data-Lab/TPC-H-Skew`

Table 2 includes the averaged hit ratios of all experiments. The overall averaged hit ratios range between 94.8% to 97.6%. As observed in the results, a higher sampling ratio usually generates higher hit ratios. However, comparing the hit ratio results with B=200 and those with B=2000, no significant differences in hit ratios were observed. For instance, the overall averaged hit ratios of z=1 when B=200 (97.9%) and B=2000 (97.6%) are both slightly higher than those with z=0 when B=200 (95.8%) and B=2000 (94.8%).

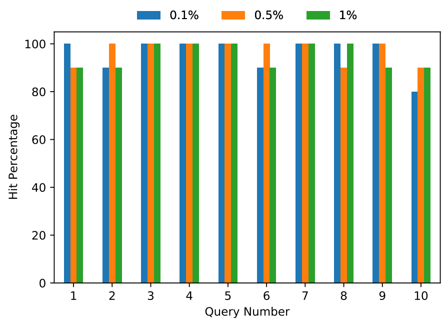
Table 3 includes the standard deviations of the hit ratios. The overall standard deviations range between 4.8 to 6.1. First, for each fixed value of B and z, the standard deviation of hit ratios generally decreases while the sampling ratio increases. For example, when B=2000 and z=1, the standard deviations for sampling ratios 0.1%, 0.5%, and 1% are 9.5, 6.7, 6.3, respectively. On the other hand, when B and z changed, no significant difference in the standard deviations were observed. The overall standard deviations of more skewed data when z=1 (STD=4.8 for B=200 and 5.6 for B=2000) are both slightly smaller than those when z=0 (STD=6.1 for B=200 and 6.4 for B=2000).

Table 4 includes the bootstrap standard deviations. A smaller bootstrap standard deviation means a smaller error estimation of the query estimation and a narrower bootstrap CI. As observed, the bootstrap standard deviations generally decrease with the sampling ratio increases. This demonstrates that query estimations are more consistent given higher sampling ratios. Changes of B and z do not significantly affect the bootstrap standard deviations.

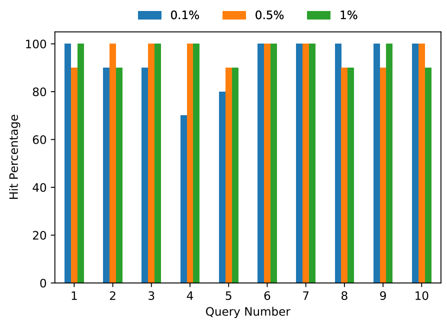
In general, the experiments show that higher sampling ratios help to improve bootstrap CI hit ratios. On the other hand, the changes of bootstrap iterations (B) or data skewness (z) do not significantly impact the error assessment accuracy.

5.3 Speed Performance Tests

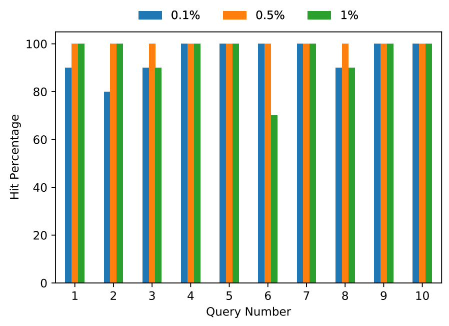
We present the speed performance results when estimating the test queries on the 1GB dataset in Figure 5. The running time to answer each test query is composed of three parts including the file access time, simple random sampling time, and bootstrap sampling time. The file accessing, random sampling, and



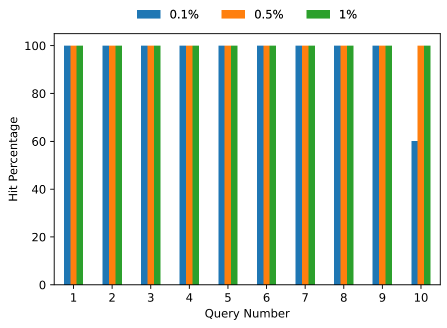
(a) B=200, z=0, 100MB



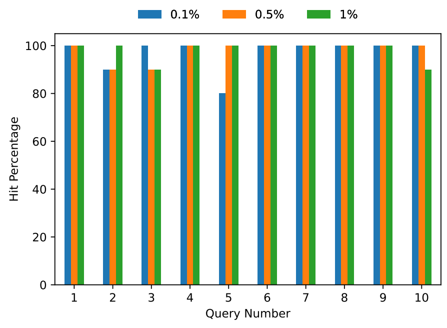
(b) B=200, z=0, 1GB



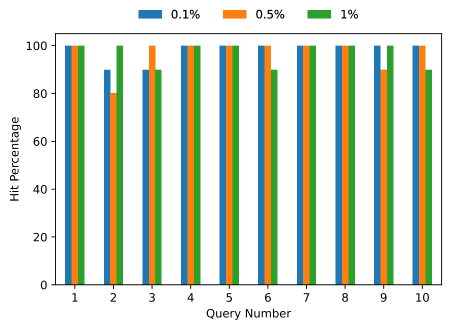
(c) B=200, z=0, 10GB



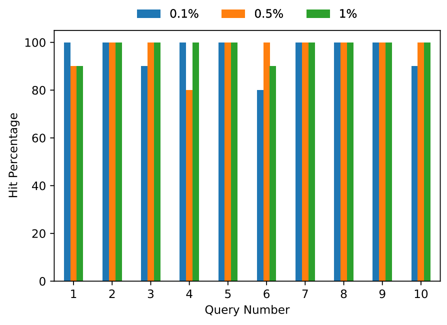
(d) B=200, z=1, 100MB



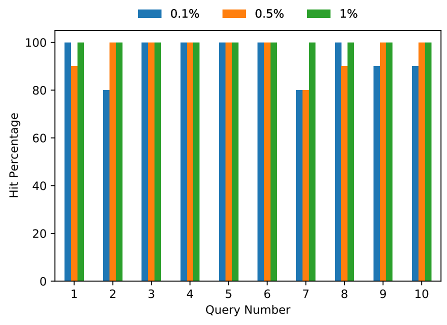
(e) B=200, z=1, 1GB



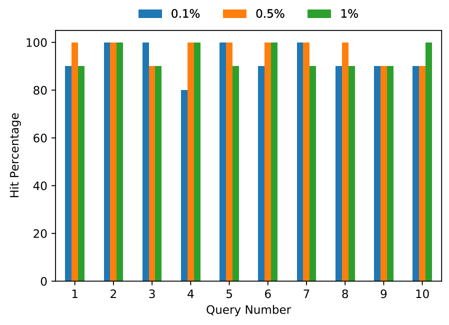
(f) B=200, z=1, 10GB



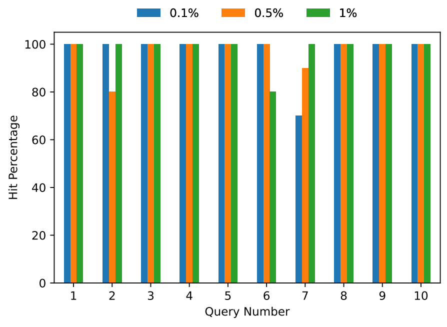
(g) B=2000, z=0, 100MB



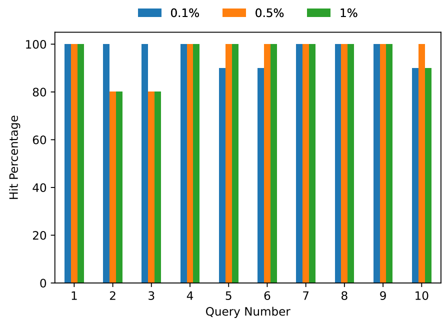
(h) B=2000, z=0, 1GB



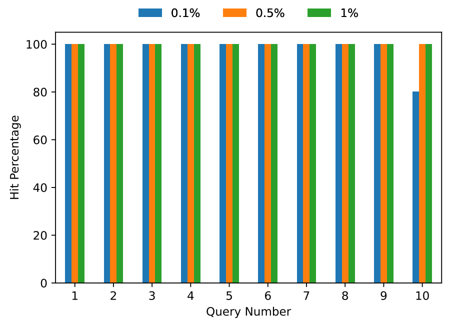
(i) B=2000, z=0, 10GB



(j) B=2000, z=1, 100MB



(k) B=2000, z=1, 1GB



(l) B=2000, z=1, 10GB

Figure 4: Hit ratios of 95% level bootstrap confidence intervals (B: bootstrap iterations; sampling ratio: 0.1%, 0.5%, and 1%; z - data skewness: 0 or 1, larger value is more skewed)

bootstrap sampling procedures did not use any memory buffer to simulate the worst performance scenario. The groups of the first three figures and the last three figures show that, when the total bootstrap iterations (B) stay the same and the sampling ratio (f) increases, the bootstrap sampling time increases and becomes a major bottleneck compared with the random sampling time. The same trend is also observed when f stays the same and B increases, for example, comparing Figure 5a and Figure 5d. Therefore, the performance of bootstrap sampling is mainly affected by values of B and f, especially when the data resides out-of-core.

5.4 Tests of Optimized Bootstrap Sampling

To increase the overall performance the prototype AQP system, we improve the bootstrap sampling procedure in the following aspects.

1. To lower the data access time, the tuples for bootstrap sampling are not directly accessed. It's only the query sample results that are calculated and stored in a memory array which are passed to the bootstrap engine. Since the sampling ratios for AQP are usually small (less than 1%), these arrays shall be small enough to fit into the main memory. If the sampled array is too large, other alternatives can be implemented such as using partitioned data arrays.
2. The sorting of the random numbers for resampling is omitted to save computation. During the resampling procedure, the bootstrap random numbers are kept unsorted. After that, a resample array of sample query results are extracted according to the bootstrap random number array by in-memory array mapping. For example, if the generated random number array for resampling is $\{5,3,1,2\}$, then the query results resampled shall be $\{y_5, y_3, y_1, y_2\}$.

We perform the same experiments on the 1GB test data using the prototype system with the optimized bootstrap sampling engine. Figure 6 depicts the execution time speedup factors comparing the optimized bootstrap sampling scheme with the original scheme. The speedup factor is defined as follows.

$$\text{speedup factor} = \frac{\text{time(original bootstrap sampling)}}{\text{time(optimized bootstrap sampling)}} \quad (9)$$

As observed from the experiment results, the optimized system reached an averaged speed-up factor of 5 comparing the bootstrap sampling execution times. It also reached an averaged speedup factor of 2 comparing the file access times. In addition, the speedup factor progressively increases with the sampling ratio.

6 Related Work

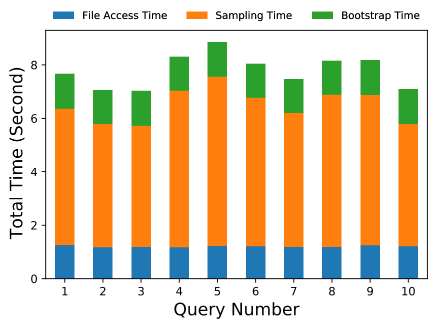
Based on the schemes of statistics collection, AQP can be categorized into two directions including the *online AQP* and the *offline AQP* [5]. The online AQP schemes [6, 16, 17], by the name, start collecting statistics only after the target query for approximation is given. Therefore, to reach a high statistics collecting speed, they usually rely on auxiliary data structures, such as indices and hash tables. Creating and maintaining these auxiliary data structures will generate heavy overheads especially for big data applications. Another drawback is that their collected statistics can only be used once for a given query, and must be re-collected for a different query which wastes computing resources. The offline AQP [1, 24], on the other hand, collects statistics before a query is submitted. It usually needs the knowledge of the whole database schema or the join graph, to create a holistic statistical synopsis. One advantage of the offline AQP is it doesn't rely on any auxiliary data structure or advanced hardware to collect statistics because the statistics collection happens before the target query is given and doesn't affect the run-time system performance. Another advantage is the statistics collected by offline AQP schemes are reusable for all future target queries given the database join graph is not changed.

Bootstrap sampling has a long history in statistics. [12] is a definitive book for its literature. However, this powerful method still waits to be fully utilized in modern database systems. Existing work [10, 11, 20, 7, 3, 15, 14, 13, 25, 26, 27] has made contribution to this direction. Among them, Pol and Jermaine in [20] focused on increasing the performance of bootstrap sampling by lowering the number of bootstrap iterations. To this end, a new data structure named resampling tree was introduced in their proposed ODM framework. Zeng et al. [25, 26] introduced an improved method called Analytical Bootstrap Method (ABM) that can avoid bootstrap iterations for limited types of database queries. Kleiner et al. [14] introduced a new bootstrap sampling method that can reduce bootstrap iterations on big datasets.

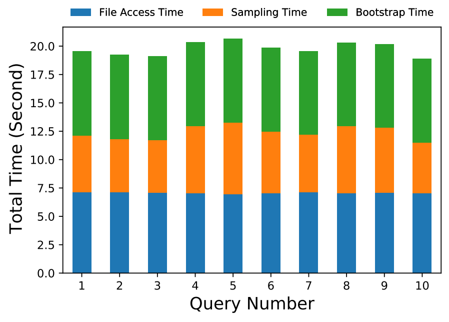
Our work concentrates on the empirical analysis of the bootstrap sampling for σ-AQP systems. We claim that the mentioned related work doesn't fully address the topics in this work. However, some methods [20, 25, 14] can help to improve the bootstrap sampling performance in this work.

7 Conclusion and Future Work

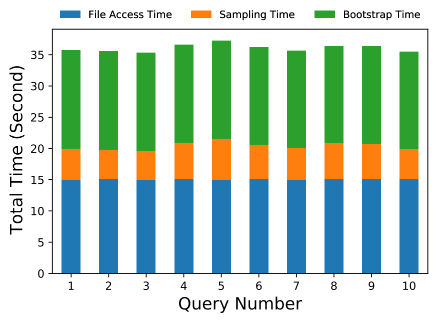
In this work, we employ a non-parametric statistical method, called bootstrap sampling, to assess the estimation errors of σ-AQP systems. The contributions are threefold. First, we developed a prototype σ-AQP system integrated with a bootstrap sampling engine that can produce confidence intervals for selection query estimations. Second, we performed extensive query estimation experiments using the implemented system. The results showed that the bootstrap confidence intervals produced are highly accurate even when small sampling ratios were used. Third, we studied the performance bottlenecks of the implemented system and proposed multiple strategies to optimize the bootstrap sampling procedure which were shown effective in experiments. In the future, we will



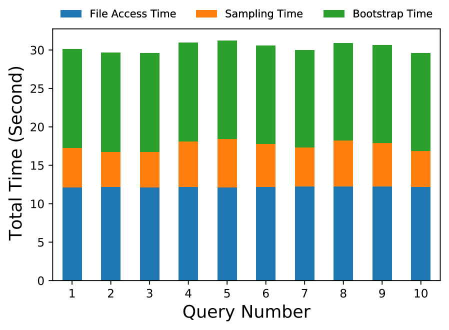
(a) B=200, *f*=0.1%



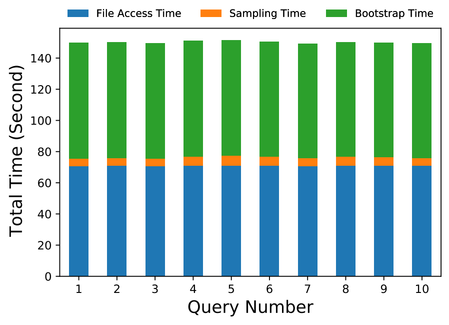
(b) B=200, *f*=0.5%



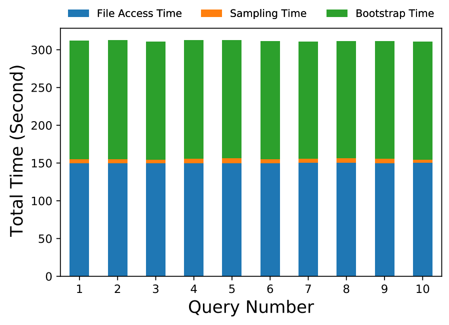
(c) B=200, *f*=1%



(d) B=2000, *f*=0.1%

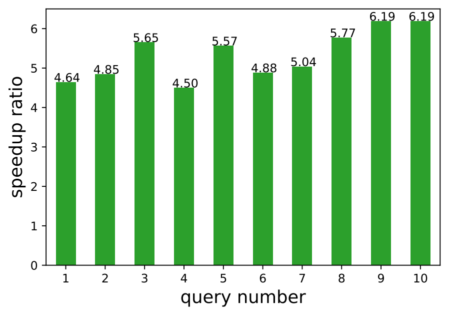


(e) B=2000, *f*=0.5%

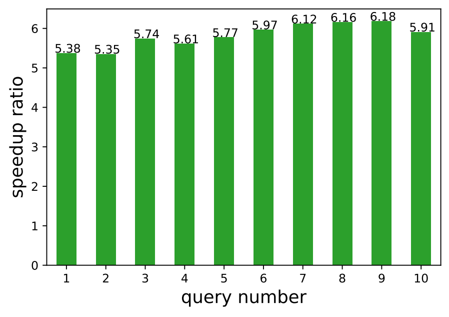


(f) B=2000, *f*=1%

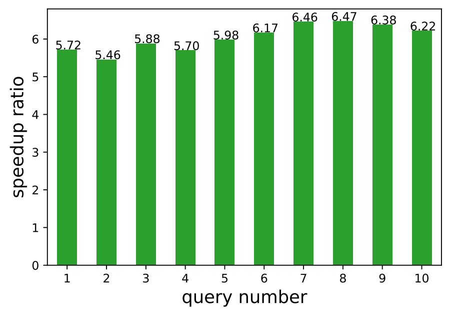
Figure 5: Time overheads in 1GB data tests (B: bootstrap iterations, *f*: sampling ratio (%))



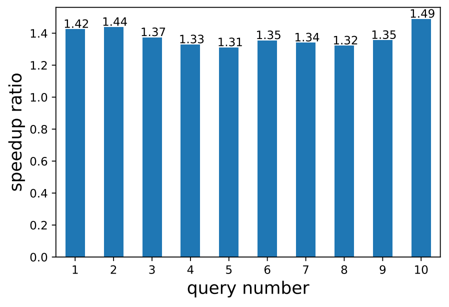
(a) Bootstrap sampling time, *f*=0.1%



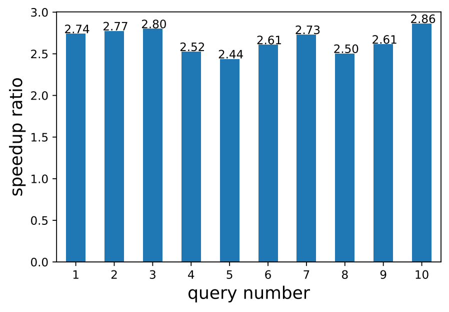
(b) Bootstrap sampling time, *f*=0.5%



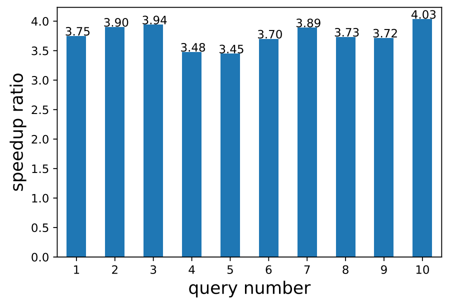
(c) Bootstrap sampling time, *f*=1%



(d) Total time, *f*=0.1%



(e) Total time, *f*=0.5%



(f) Total time, *f*=1%

Figure 6: Speedup ratios using optimized bootstrap sampling in 1GB data tests (*f*: sampling ratio (%))

Table 2: Hit ratio means (%, B: total bootstrap iterations, z: skewness value)

B	z	100 MB				1GB				10GB				Overall
		0.1%	0.5%	1%	avg	0.1%	0.5%	1%	avg	0.1%	0.5%	1%	avg	avg
200	0	96.0	97.0	95.0	96.0	93.0	96.0	96.0	95.0	95.0	99.0	95.0	96.3	95.8
200	1	96.0	100.0	100.0	98.7	97.0	98.0	98.0	97.7	98.0	97.0	97.0	97.3	97.9
2000	0	93.0	88.0	99.0	93.3	94.0	95.0	97.0	95.3	97.0	97.0	93.0	95.7	94.8
2000	1	97.0	97.0	98.0	97.3	97.0	96.0	95.0	96.0	98.0	100.0	100.0	99.3	97.6

Table 3: Hit ratio standard deviations (B: total bootstrap iterations, z: skewness value)

B	z	100 MB				1GB				10GB				Overall
		0.1%	0.5%	1%	avg	0.1%	0.5%	1%	avg	0.1%	0.5%	1%	avg	avg
200	0	7.0	4.8	5.3	5.7	10.6	5.2	5.2	7.0	7.1	0.0	9.7	5.6	6.1
200	1	12.6	0.0	0.0	4.2	6.7	4.2	4.2	5.0	4.2	6.7	4.8	5.2	4.8
2000	0	9.5	11.4	3.2	8.0	5.2	7.1	4.8	5.7	6.7	4.8	4.8	5.4	6.4
2000	1	9.5	6.7	6.3	7.5	4.8	8.4	8.5	7.2	6.3	0.0	0.0	2.1	5.6

Table 4: Bootstrap standard deviations (B: total bootstrap iterations, z: skewness value)

B	z	100 MB				1GB				10GB				Overall
		0.1%	0.5%	1%	avg	0.1%	0.5%	1%	avg	0.1%	0.5%	1%	avg	avg
200	0	10817.8	4844.3	3449.9	6370.7	34567.3	15368.4	10775.8	20237.2	108118.6	48151.8	34063.4	63444.6	30017.5
200	1	10821.6	4889.6	3477.0	6396.1	34044.4	15211.9	10851.6	20036.0	107887.7	47909.3	34450.6	63415.9	29949.3
2000	0	10894.7	4855.3	3427.2	6392.4	34198.9	15371.4	10823.5	20131.3	108599.9	48490.2	34193.6	63761.2	30095.0
2000	1	10864.4	4839.7	3427.8	6377.3	34237.1	15313.4	10811.0	20120.5	107740.9	48241.3	34143.8	63375.3	29957.7

generalize the current framework to assess the errors of AQP systems for more complex queries, such as join and common aggregation queries, on large datasets.

Acknowledgement

This research is partially supported by University Research Council Grant and Research Professorship Award at Youngstown State University.

References

[1] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. “Join Synopses for Approximate Query Answering”, In *Proc. SIGMOD ’99*, pp. 275–286, 1999.

[2] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari, and I. Stoica. “Knowing when You’re Wrong: Building Fast and Reliable Approximate Query Processing Systems”, In *Proc. SIGMOD 2014*, pp. 481–492, 2014.

[3] K. J. Archer and R. V. Kimes. “Empirical Characterization of Random Forest Variable Importance Measures”, *Computational Statistics and Data Analysis*, 52:2249–2260, 2008.

[4] S. Cal, E. Cheng, and F. Yu. “Optimized Bootstrap Sampling for -AQP Error Estimation: A Pilot Study”, In *Proc. of ISCA 30th International Conference on Software Engineering and Data Engineering*, 77:144–153, 2021.

[5] S. Chaudhuri, B. Ding, and S. Kandula. “Approximate Query Processing: No Silver Bullet”, In *Proc. SIGMOD’17*, pp. 511–519, 2017.

[6] Y. Chen and K. Yi. “Two-Level Sampling for Join Size Estimation”, In *Proc. SIGMOD 2017*, ACM, pp. 759–774, 2017.

[7] M. R. Chernick. *Bootstrap Methods: A Guide for Practitioners and Researchers*, John Wiley & Sons, 2008.

[8] T. P. P. Council. “TPC-H Benchmark”, `http://www.tpc.org/tpch/`.

[9] C. Doulkeridis and K. Nørvåg. “A Survey of Large-Scale Analytical Query Processing in MapReduce”, *The VLDB Journal*, 23:355–380, 6 2014.

[10] B. Efron. “Bootstrap Methods: Another Look at the Jackknife”, *The Annals of Statistics*, 7:1–26, 1979.

[11] B. Efron. “Second Thoughts on the Bootstrap”, *Statistical Science*, 18:135–140, 2003.

[12] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*, CRC Press, 1994.

[13] A. Kleiner, A. Talwalkar, S. Agarwal, I. Stoica, and M. I. Jordan. “A General Bootstrap Performance Diagnostic”, In *Proc. SIGKDD 2013*, pp. 419, 2013.

[14] A. Kleiner, A. Talwalkar, P. Sarkar, and M. Jordan. “The Big Data Bootstrap”, *arXiv preprint arXiv:1206.6415*, June 2012.

[15] N. Laptev, K. Zeng, and C. Zaniolo. “Early Accurate Results for Advanced Analytics on MapReduce”, *Proc. VLDB Endow.*, 5:1028–1039, June 2012.

[16] V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann. “Cardinality Estimation Done Right : Index-

based Join Sampling", In *Proc. CIDR'17*, 2017.

[17] F. Li, B. Wu, K. Yi, Z. Zhao, L. Li, S. Miles, Z. Melville, A. Prasad, and L. L. Breeden. "Wander Join: Online Aggregation via Random Walks", In *Proc. SIGMOD'16*, pp. 615–629, 2016.

[18] K. Li and G. Li. "Approximate Query Processing: What is New and Where to Go?", *Data Science and Engineering*, 3:379–397, 2018.

[19] Q. Liu. "Approximate Query Processing", In L. LIU and M. T. ÖZSU, Editors. *Encyclopedia of Database Systems*, Springer US, pp. 113–119, 2009.

[20] A. Pol and C. Jermaine. "Relational Confidence Bounds are Easy with the Bootstrap", In *Proc. SIGMOD'05*, pp. 587–598, 2005.

[21] D. L. Quoc, I. E. Akkus, P. Bhatotia, S. Blanas, R. Chen, C. Fetzer, and T. Strufe. "Approximate Distributed Joins in Apache Spark", *arXiv preprint arXiv:1805.05874* , May 2018.

[22] M. Sch, J. Schildgen, and S. Deßloch. "Sampling with Incremental MapReduce", *Datenbanksysteme für Business, Technologie und Web (BTW 2015)-Workshopband*, 2015.

[23] R. J. Tibshirani and B. Efron. *An Introduction to the Bootstrap*, Monographs on statistics and applied probability, 57:1–436, 1993.

[24] F. Yu, W.-C. Hou, C. Luo, D. Che, and M. Zhu. "CS2: A New Database Synopsis for Query Estimation", ACM, pp. 469–480, 2013.

[25] K. Zeng. "ABS: A System for Scalable Approximate Queries with Accuracy Guarantees", *Sigmod*, pp. 1067–1070, 2014.

[26] K. Zeng. "Approximation and Search Optimization on Massive Data Bases and Data Streams", PhD Thesis, University of California, Los Angeles, 2014.

[27] Z. Zhou, H. Zhang, S. Li, and X. Du. "Hermes: A Privacy-Preserving Approximate Search Framework for Big Data", *IEEE Access*, 6:20009–20020, 2018.

Feng Yu, Ph.D., is currently an associate professor of Computer Science and Information Systems at Youngstown State University, USA. He is a campus champion for NSF XSEDE. His current research interests include database systems, big data management, and cloud computing. He has served as a reviewer for many international conferences, such as DEXA, SSDBM, and IEEE Big Data, and scholarly journals, such as ACM TODS, Information Sciences, and DKE.

Semih Cal received the B.S. degree from the Department of Computer Science, Sam Houston State University, Texas, USA in 2017 and he received the M.S. degree from the Department of Computer Science, Youngstown State University, Ohio, USA in 2021, He is currently a Ph.D. student in the Department of Computer Science, Texas Tech University. His current research interests include IoT devices, routing protocols in FANET, and mobile data management.

En Cheng, Ph.D., is an associate professor in the Department of Computer Science at The University of Akron. Before joining The University of Akron, Dr. Cheng had the opportunity to experience internships in diverse research centers, including Microsoft Research Asia, IBM T.J. Watson Research Center, and Cleveland Clinic Foundation. Her current research interests include Data Integration, Big Data, Database Systems and Applications, Mobile Applications, Semantic Web, and Business Intelligence.

Lucy Kerns, Ph.D., is currently working as an associate professor in the Department of Mathematics and Statistics at Youngstown State University, where she also serves as the Statistics Coordinator and co-director of the Mathematical and Statistical Consulting Center. Her research interests have been focused mainly on simultaneous inferential techniques, environment risk assessment and environmental toxicology, statistical modeling, data analytics (machine learning and data mining), and data visualization.

Weidong Xiong is a visiting associate lecturer of the Department of Electrical Engineering and Computer Science at Cleveland State University. He received his Ph.D. in Computer Science from Southern Illinois University Carbondale, IL in 2018. His primary research interests include Concurrency Control Protocols in Database, 3D Programming, Volume Rendering, and Deep Learning. Prior to transitioning his career in academia, Xiong was a senior software engineer in IT industry with more than ten years of programming experience.