EPiC Series in Computing

Volume 77, 2021, Pages 144–153

Proceedings of ISCA 30th International Conference on Software Engineering and Data Engineering

Optimized Bootstrap Sampling for σ-AQP Error Estimation: A Pilot Study

Semih Cal[1], En Cheng[2], and Feng Yu[1]*

[1] Youngstown State University, Youngstown, OH, USA
scal@student.ysu.edu,fyu@ysu.edu
[2] University of Akron, Akron, OH, USA
echeng@uakron.edu

Abstract

Approximate query processing (AQP) aims to provide an approximated answer close to the exact answer efficiently for a complex query on large datasets, especially big data. It brings enormous benefits into many data science fields when the efficiency of query execution weighs more than the accuracy. However, assessing the accuracy of an approximated answer from AQP deserves more study. Existing work usually relies on strong dataset assumptions which may not work for real-world datasets. In this work, we employ bootstrap sampling to assess the estimation errors of the AQP for selection queries (called σ-AQP). We implement a prototype system which can calculate confidence intervals for the estimated query results. Experiment results demonstrated that the confidence intervals generated by the prototype system can cover the ground truth of the query results with high accuracy and low computing cost. In addition, we implement optimization strategies for the bootstrap sampling which have significantly improved the overall computing efficiency.

1 Introduction

Big data produces demanding challenges for modern data management systems to efficiently process complex queries within a limited time. Much work has been developed towards promptly executing data queries in both hardware and software platforms [8, 9, 14]. However, calculating the exact query answer can be expensive and possibly unnecessary in every scenario. For example, at the initial state of an exploratory data analysis (or EDA) project, one may only need to know approximated answers of some testing queries in a relatively short time.

Approximate query processing (or AQP) is an alternative scheme to provide estimated query answers with a satisfying accuracy and within a short time [2, 12, 13]. AQP doesn't need to run the original query but try to collect statistics to generate query estimations. Based on the fashion of statistics collection, AQP can be categorized into two groups including the *online AQP* and the *offline AQP* [3]. The online AQP [4, 10, 11], by the name, starts collecting

*Corresponding Author

statistics only after the target query for approximation is given. The online AQP usually relies on auxiliary data structures, such as indices and hash tables, in order to collect statistics quickly. Another drawback of the online AQP is its collected statistics are not reusable and must be re-collected for each different query. However, given enough time, the online AQP is able to run multiple times to approach to the desired estimation accuracy.

The offline AQP [1, 16], on the other hand, collects statistics before a target query is submitted. It usually needs the knowledge of the whole database schema in order to create a holistic statistical synopsis. One advantage of the offline AQP is it doesn't require auxiliary data structures to collect statistics because the statistics collection happens before the query is given and doesn't affect the system performance. Another advantage is the collected statistics by the offline AQP are usually reusable for future queries. This alleviates the data accessing costs. However, one open question for the offline AQP research is how to efficiently assess the estimation errors for an offline AQP system.

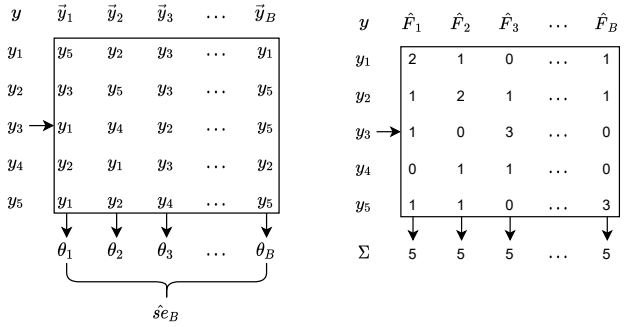
One of the most common AQP scheme is the σ-AQP which estimates selection queries. The challenge is that, for different conditions in a selection (σ) query, the underlying distributions of the result sets are different and difficult to predict. For the online AQP schemes, since the statistics are collected on-the-fly for each query, the distribution of the result sets is known. Therefore, an estimated variance of its statistic estimator can be derived. However, for offline AQP schemes, only the distribution of the query result sets from the original dataset is unknown. This creates an obstacle to the assess the estimation errors for offline AQP schemes.

Bootstrap sampling [15] is a special statistical method that can assess the errors of sample-based estimators. One advantage of bootstrap sampling is that it doesn't require the pre-knowledge of the data distribution, but can "pull itself up by its bootstrap". The bootstrap sampling performs a special sampling method, called re-sampling, which can generate a large number of replicated random samples (without replacement), called bootstrap replications, based on the original samples. By applying a statistical estimator on bootstrap samples, a set of estimations, called bootstrap replications, is obtained. Finally, the standard error of the bootstrap replications is obtained to assess the query estimation error.

In this work, we will focus on using bootstrap sampling to assess estimation errors for offline σ-AQP schemes. We propose a framework to implement bootstrap sampling for selection queries and perform estimation error assessments. A prototype system is implemented to simulate a real-world database system. This system is integrated with a bootstrap engine capable of generating confidence intervals for each estimated query answer. We test the performance of the prototype system on multiple datasets with various combinations of hyper-parameters to simulate real-world AQP scenarios. The experimental results show satisfying accuracy of the confidence intervals for testing queries. With the findings of the computing bottlenecks of the system, we propose optimization mechanisms for the bootstrap sampling pipeline in order to increase the overall system performance.

The rest of this work is organized as follows. Section 2 introduces the background knowledge of σ-AQP and bootstrap sampling schemes. 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 the proposed bootstrap sampling functionality is described in Section 4. Experimental results are presented in Section 5. The conclusion and future work are included in Section 6.

2 Background



(a) Bootstrap Samples (b) Bootstrap Distributions

Figure 1: Example: Bootstrap Sampling

2.1 σ-AQP

One of the common applications of AQP is to estimate selection (or σ) queries, named σ-AQP. Given a selection query Q on a table R, to get the ground truth query answer Y_{gt}, the traditional manner of query answering is to execute query Q on R which may take a long waiting time when R has a large volume. Instead of running query Q directly on R, σ-AQP takes a simple random sample without replacement (SRSOR) from table R, denoted by S, and runs the query Q on S to get a sample result Y_s. The ground truth, Y_{gt} can then be estimated by $Y_{gt} = \frac{Y_s}{f}$ where $f = \frac{|S|}{|R|}$ is the sampling ratio.

2.2 Bootstrap Sampling

The bootstrap sampling method was originally introduced by Bradley Efron in 1979 [7]. 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 assumptions on the data.

The unique statistical procedure in bootstrap sampling is resampling, which generates new distributions, called *bootstrap samples*, from a given data 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 statistic features of the original given data, even when the original distribution is unknown.

Figure 1 depicts a simple example of how bootstrap sampling is performed. When given a sample data $\vec{y} = (y_1, y_2, ..., y_n)$ from an unknown distribution F, a *bootstrap sample* $\vec{y}^* = (y_1^*, y_2^*, ..., y_n^*)$ 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 $\vec{y}_1 = (y_5, y_3, y_1, y_2, y_1)$, $\vec{y}_2 = (y_2, y_5, y_4, y_1, y_2)$, $\vec{y}_3 = (y_3, y_3, y_2, y_3, y_4)$, etc. These resamples are shown in Figure 1a. After summarizing the frequency of each sampled element, we obtain the distribution of a bootstrap sample, $\hat{F} = (\hat{f}_1, \hat{f}_2, \dots)$, where $\hat{f}_k = \#\{y_i^* = y_k\}/n$. Figure 1b depicts a bootstrap sample example.

It can be proved that $\hat{F}$ is a sufficient statistic to estimate the true distribution F since all the information about F contained in $\vec{y}$ is also contained in $\hat{F}$.

A good application of bootstrap sampling is to estimate the standard error of a sample estimator from an unknown distribution. Suppose we are interested in a parameter $\theta = t(F)$ calculated based on $\vec{y}$. Since usually we don't have all the information of F, but can only calculate an estimation of θ from the given sample $\vec{y}$ denoted by $\hat{\theta} = s(\vec{y})$. For each bootstrap sample $\vec{y}^*$ we can generate a *bootstrap replication* of $\hat{\theta}$, as $\hat{\theta}^* = s(\vec{y}^*)$. For example, when $\hat{\theta}$ is the sample mean $\overline{\vec{y}}$, a bootstrap replication $\hat{\theta}^*$ is be the sample mean on a bootstrap sample $\overline{\vec{y}^*}$.

After generating a number of B independent bootstrap samples, we can obtain the standard error of all $\hat{\theta}$, i.e. $\hat{se}_B(\hat{\theta})$, called the *bootstrap estimation of standard error*. When $B \to \infty$, $\hat{se}_B(\hat{\theta}) \to se_{\hat{F}}(\hat{\theta})$. $se_{\hat{F}}(\hat{\theta})$ is called the *ideal bootstrap estimation* of the ground truth standard error $se_{\hat{F}}(\hat{\theta})$. We say that $se_{\hat{F}}(\hat{\theta})$ is a *plug-in* estimate of $se_F(\hat{\theta})$ that uses the empirical distribution $\hat{F}$ in replacement of the unknown distribution F. $\hat{se}_B(\hat{\theta})$ can be calculated as

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

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

Both $\hat{se}_B(\hat{\theta})$ and $se_{\hat{F}}(\hat{\theta})$ 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:

```
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 query result y_i based on the aggregation function. For example, if the aggregation function is COUNT and the primary key is included in the attribute collection, then y_i will be 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 Y_{GT} is

$$\widehat{Y} = \frac{Y_s}{f} \tag{2}$$

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 sample relation S, then bootstrap samples $\{\vec{y}_j\}_{j=1}^{B}$ can be generated for error estimation, where B is the total

times of bootstrap sampling to perform. Each $\vec{y}_j = \{y_{j,i}\}_{i=1}^{n}$ a bootstrap sampling of S_Q, where each query result $y_{j,i}$, $1 \leq i \leq n$, is randomly sampled with replacement from S_Q.

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

$$\widehat{Y}_j = \frac{Y_{\vec{y}_j}}{f} \tag{3}$$

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

$$\widehat{Y}_j = \frac{1}{f} \sum_{i=1}^{n} y_{j,i} \tag{4}$$

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

$$\hat{se}_B = \left[\frac{1}{B-1} \sum_{j=1}^{B} (\widehat{Y}_j - \overline{\widehat{Y}}_B)^2 \right]^{\frac{1}{2}} \tag{5}$$

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 (5) is the *bootstrap estimation of the standard error* of $\widehat{Y}$ which estimates the query result Y_{GT} of query Q.

3.3 Confidence Interval for σ-AQP

There are different methods using bootstrapping to generate the confidence interval (or CI), such as the standard method and percentile method. There are also improved methods to increase the accuracy such as BC_a and ABC. In this work, we adopt the standard method which is a commonly adopted method, and because we aim to investigate the overall system performance on a large dataset.

Suppose the significance level is denoted as α, where α is a probabilistic value. Common choices of α include 5% for 90% level of significance and 2.5% for 95% level of significance, respectively. The standard method for bootstrap CI is calculated as

$$\left(\widehat{Y} - z^{(1-\alpha)} \cdot \hat{se}_B, \widehat{Y} + z^{(1-\alpha)} \cdot \hat{se}_B \right) \tag{6}$$

where $\widehat{Y}$ is the query estimation and $z^{(1-\alpha)}$ is the $100(1-\alpha)$th percentile of a standard normal distribution. For example, for 90% level of significance, $z^{(.95)}$=1.645, and for 95% level of significance, $z^{(.975)}$=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 2.

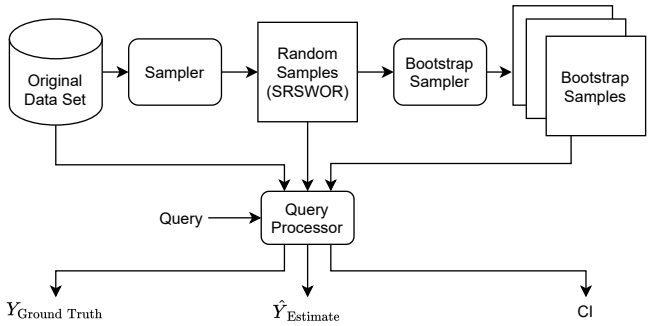


Figure 2: Prototype σ-AQP Framework with a Bootstrap Engine

The simple query processor will read the query structure from a plain text file and execute the query accordingly. The query execution engine will read the tuples from the base table data file and check if each tuple satisfies the selection condition.

The σ-AQP engine has two functions: generate a sample table using simple random sampling (simple random sampler) and provide query estimations using the sample table (sample estimator). The sample fraction f is pre-determined before the sampling is performed. 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 are too big to be fitted 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 later on.

After the sample tuples are drawn, the sample estimator of the σ-AQP engine can produce a query estimation by first execute the original query Q on the sample table S and get 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 ground truth, Y_{est}, will be calculated using Eq (2).

The bootstrap engine will perform bootstrap sampling on the sample query results S_Q, calculate the bootstrap standard error s_B, and produce the bootstrap confidence interval (CI).

5 Experiment

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 the C and Python language [1]. There are three data sets that have been used for the experiment. The datasets are generated using the TPC-H benchmark [5] which is widely used for query processing experiments[2]. To simulate the common data setting, we generated three TPC-H datasets with zero skewness and in volumes of 100MB, 1GB, and 10GB, respectively. We randomly generated 10 test queries with different selection ranges. The first 5 queries are large-range selection queries and last 5 queries are small-range selection queries.

[1] Experiment code available at https://github.com/YSU-Data-Lab/Semih_Cal_Thesis_Summer_2021
[2] We employed the TPC-H program available at https://github.com/YSU-Data-Lab/TPC-H-Skew

Table 1: Test Queries

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

5.2 Bootstrap Accuracy Tests

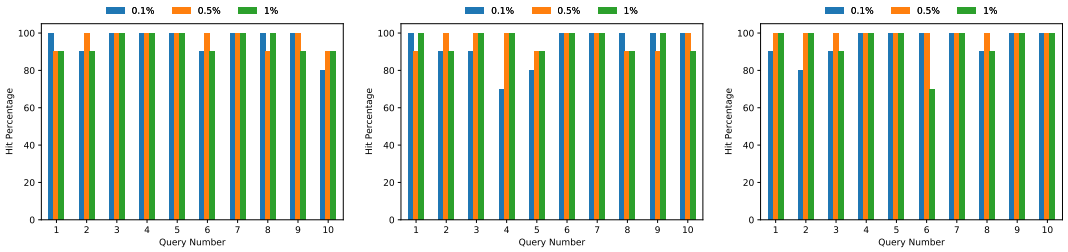
Given the sizes of the datasets, 100MB, 1GB, and 10GB, we use simple random sampling to estimate the query sizes and then use the proposed bootstrap method to estimate the estimation error. The estimation error is expressed in confidence intervals, i.e. CI. The ground truth of the query, Y_{GT} is also calculated 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 run the CI hit test 10 times and calculate the hit ratio of Bootstrap CIs as following.

$$\text{hit ratio} = \frac{\text{times(CI hits } Y_{GT})}{\text{times(total experiment)}} \times 100\% \tag{7}$$

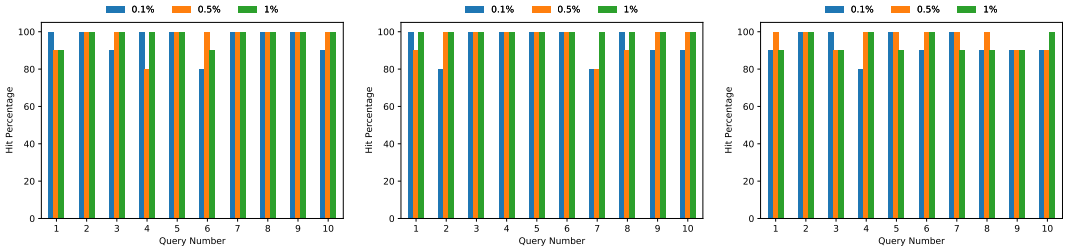
Figure 3 depicts the results of accuracy tests using Bootstrap sampling. To study how the bootstrap iterations (B) affects the hit ratios, we choose B=2000 which is recommended in [6] and B=200 for comparison. Comparing the first three figures with B=200 and the next three figures with B=2000 in Figure 3, no significant differences of hit ratios are observed. However, the variance of the group with B=200 is slightly larger than the group with B=2000. This means using a less number of bootstrap iterations can save computing costs and achieve the acceptable accuracy with a sacrifice of stability.

5.3 Speed Performance Tests

We present the speed performance results when estimating the test queries on the 1GB dataset in Figure 4. The running time to answer each test query is composed of three parts including file access time, simple random sampling time, and bootstrap sampling time. The file accessing, random sampling, and bootstrap sampling procedures did not use any memory buffer in order to simulate the worst performance scenario. The groups of the first three figures and last three figures show that, when B stays the same and f increases, the bootstrap sampling time increases and becomes a major bottleneck than the random sampling time. The same trend is observed when f stays the same and B increases, for example comparing Figure 4a and 4d. Therefore, the performance of bootstrap sampling is majorly affected by the sampling ratio (f) and the total bootstrap iterations (B), especially when data resides out-of-core.

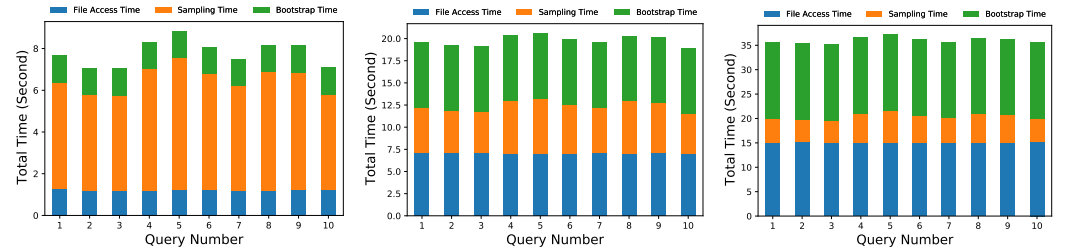


(a) B=200, 100MB (b) B=200, 1GB (c) B=200, 10GB

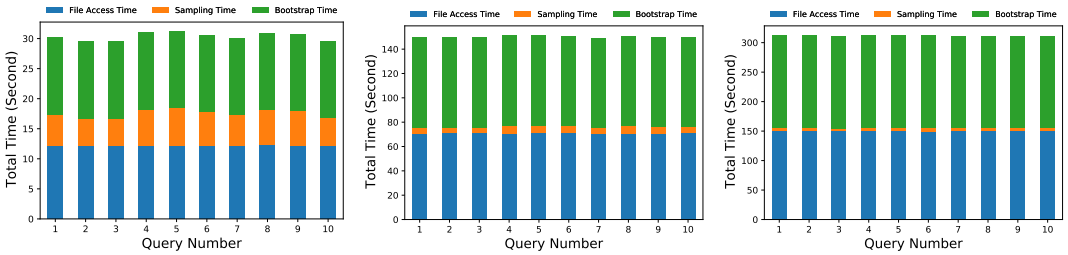


(d) B=2000, 100MB (e) B=2000, 1GB (f) B=2000, 10GB

Figure 3: Hit ratios of bootstrap confidence intervals (B: bootstrap iterations; sampling ratio: 0.1%, 0.5%, and 1%)



(a) B=200, f=0.1% (b) B=200, f=0.5% (c) B=200, f=1%

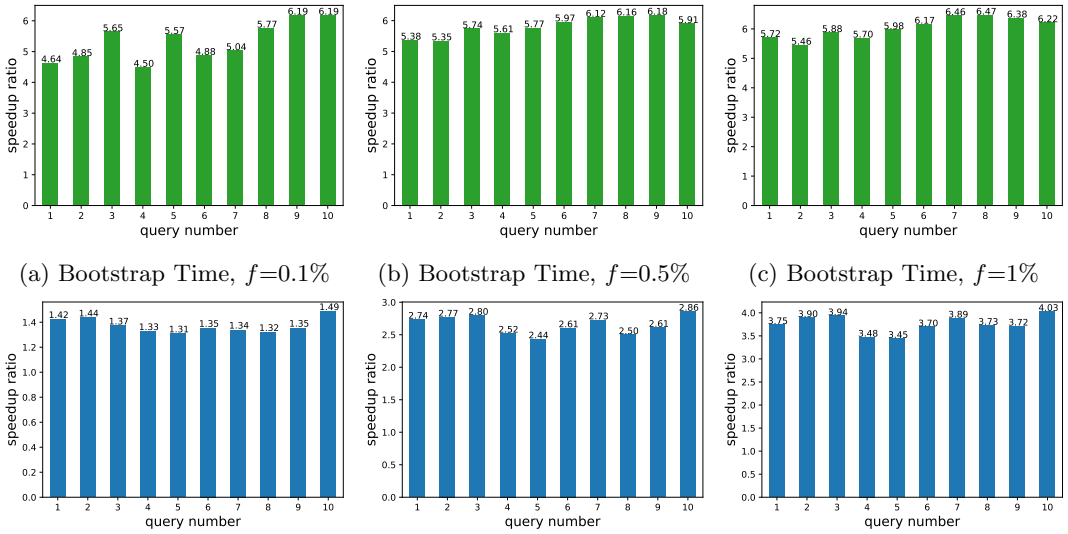


(d) B=2000, f=0.1% (e) B=2000, f=0.5% (f) B=2000, f=1%

Figure 4: Sampling Time of 1GB Data (B: bootstrap iterations, f: sampling ratio (%))

5.4 Tests of Optimized Bootstrap Sampling

We implement an optimized scheme of the bootstrap sampling in the following aspects.



(a) Bootstrap Time, f=0.1% (b) Bootstrap Time, f=0.5% (c) Bootstrap Time, f=1%

(d) Total Time, f=0.1% (e) Total Time, f=0.5% (f) Total Time, f=1%

Figure 5: Speedup Ratio using Optimized Bootstrap Sampling (f: sampling ratio (%))

1. 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 and passed on the bootstrap engine.
2. 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.

We perform the experiments on the 1GB test data and compare the execution speeds The results are shown in Figure 5. The speedup factor defined as time(original bootstrap sampling)/time(optimized bootstrap sampling). As observed from the experiment results, the optimized bootstrap method reached approximately a speed-up factor of 5. And the file access time reached approximately a speedup factor of 2. The speedup factor progressively increases with the sampling ratio.

6 Conclusion

In this work, we present a prototype system implementation for σ-AQP with the ability to assess the estimation errors using bootstrap sampling. The contributions are two folds. First, we implemented a prototype query processing framework along with a bootstrap sampling component that can calculate the confidence intervals for selection (or σ) query estimations. The experiment results show high accuracy of the confidence intervals even when the sampling ratios are small. The second is to study the bottlenecks of the prototype system on large datasets. We proposed multiple strategies in the experimental study to further optimize the bootstrap sampling speed. These strategies were shown effective in the additional experimental results. In the future, we will generalize the current framework to assess the errors of AQP for more complex queries, such as join and common aggregation queries, on large datasets.

7 Acknowledgement

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

References

[1] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join Synopses for Approximate Query Answering. In *Proc. SIGMOD'99*, pages 275–286, New York, NY, USA, 1999. ACM.

[2] Sameer Agarwal, Henry Milner, Ariel Kleiner, Ameet Talwalkar, Michael Jordan, Samuel Madden, Barzan Mozafari, and Ion Stoica. Knowing When You're Wrong: Building Fast and Reliable Approximate Query Processing Systems. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data - SIGMOD*, pages 481–492, 2014.

[3] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. Approximate query processing: No silver bullet. In *Proc. SIGMOD'17*, pages 511–519, 2017.

[4] Yu Chen and Ke Yi. Two-Level Sampling for Join Size Estimation. In *Proc. ICDE'17*, number 2 in SIGMOD '17, pages 759–774, New York, NY, USA, 2017. ACM.

[5] Transaction Processing Performance Council. Tpc-h benchmark. http://www.tpc.org/tpch/.

[6] B Efron. Bootstrap methods: another look at the jacknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[7] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[8] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, 35(2):137–144, 2015.

[9] D Le Quoc, I Ekin Akkus, Pramod Bhatotia, Spyros Blanas, Ruichuan Chen, Christof Fetzer, Thorsten Strufe, Do Le Quoc, Istemi Ekin Akkus, Pramod Bhatotia, Spyros Blanas, Ruichuan Chen, Christof Fetzer, and Thorsten Strufe. Approximate Distributed Joins in Apache Spark. *ArXiv e-prints*, abs/1805.0, may 2018.

[10] Viktor Leis, Bernhard Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. Cardinality Estimation Done Right : Index-Based Join Sampling. In *Proc. CIDR'17*, Chaminade, California, USA, 2017.

[11] Feifei Li, Bin Wu, Ke Yi, Zhuoyue Zhao, Lihong Li, Shawna Miles, Zephan Melville, Amalthiya Prasad, and Linda L Breeden. Wander Join: Online Aggregation via Random Walks. *Proc. SIGMOD'16*, pages 615–629, 2016.

[12] Kaiyu Li and Guoliang Li. Approximate query processing: what is new and where to go? *Data Science and Engineering*, 3(4):379–397, 2018.

[13] Qing Liu. Approximate Query Processing. In M TAMER LIU LING and ÖZSU, editor, *Encyclopedia of Database Systems*, pages 113–119. Springer US, Boston, MA, 2009.

[14] Marc Sch, Johannes Schildgen, and Stefan Deßloch. Sampling with Incremental MapReduce. In *Datenbanksysteme für Business, Technologie und Web (BTW)*, 2015.

[15] Robert J Tibshirani and Bradley Efron. An introduction to the bootstrap. *Monographs on statistics and applied probability*, 57:1–436, 1993.

[16] Feng Yu, Wen-Chi Hou, Cheng Luo, Dunren Che, and Mengxia Zhu. CS2: A New Database Synopsis for Query Estimation. In *Proc. SIGMOD'13*, pages 469–480, New York, NY, USA, 2013. ACM.