Evaluating Query Estimation Errors Using Bootstrap Sampling

by

Semih Cal

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

August, 2021

Evaluating Query Estimation Errors Using Bootstrap Sampling

Semih Cal

Signature:

Semih Cal, Student Date

Approvals:

Dr. Feng Yu, Thesis Advisor Date

Dr. John R Sullins, Committee Member Date

Dr. Yong Zhang, Committee Member Date

Dr. Salvatore A. Sanders, Dean of Graduate Studies Date

Abstract

Big data embodies massive amount of knowledge. Many businesses now rely on big data information mining to forecast the viability of future business operations. Information mining of big data can require a large investment of time. To reduce time requirements, sampling is one of the most preferred methods. Evaluation of quality (e.g. query prediction error) for the query estimates is crucial for meaningful results. The main method used in the past to solve this problem is based on bootstrap sampling. Existing work typically makes strong dataset assumptions that may not apply to real-world datasets. This research aims to evaluate query estimation errors using the bootstrap sampling method. There exist different kinds of bootstrap methods. In this work, we used the non-parametric bootstrap sampling to calculate the error distribution of the queries that we choose. Then we calculated the confidence intervals to find out the hit ratio. Even though the bootstrap sampling method is one of the main approaches for finding the error in statistic estimates, it is computationally expensive on large data. To solve this problem, we test both memory and disk as storage for optimizing bootstrap sampling. Furthermore, two different total number of bootstrap samples (B=2000, and B=200) have been tested to reduce bootstrap computation with reliable results for optimization purposes. In the experiment part, we use three different sizes of data (100MB, 1GB, and 10GB) as well as three different sampling ratios (0.1%, 0.5%, and 1%) to analyze the data that we generated on the TPC-H benchmark in terms of accuracy and performance.The results demonstrate that the hit ratios are very high even with a 0.1% sampling ratio. The optimization strategies that were used reduced the bootstrap sample computation time adequately.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Feng Yu. His insightful remarks encouraged me to strengthen my thinking and elevate my work. I appreciate the chance he gave me to work with him, as well as the aid, patience, and knowledge he provided to help me accomplish my research.

I would like to express my deepest appreciation to my committee members Dr. John Sullins, and Dr. Yong Zhang for taking time out of their schedule to share their insights and knowledge. I would also like to give special thanks to Dr. Alina Lazar and Ms. Connie Frisby, who both were very helpful during my graduate study.

I would also like to special thanks to my family and friends for their support during my studies and for providing me with a firm platform on which to thrive in my studies and as a person.

Contents

List of Figures

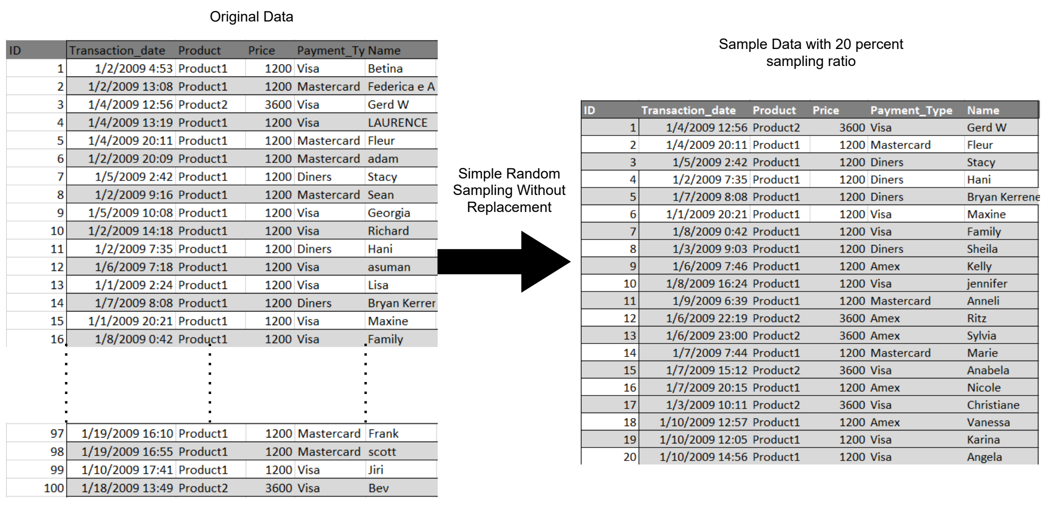
1 Introduction

The objective of query estimation is to provide approximate answers to database queries. Accurate estimates of the expenses of different plans are required for the best plan selection during query optimization in a database [10]. Selectivity, or the number of bundles that meet a specific predicate, is one of the most critical elements influencing plan cost. As a result, in most circumstances, the accuracy of the selectivity estimates has a direct influence on the selection of the optimal plan. In today's world, most companies are generating a mass amount of data, which is considered as big data [4] [20]. Since querying in big data is very costly in terms of time, query estimation methods gain great importance today. The estimation is calculated by using the samples from the population. For instance, let's say we are going to find out how many people are younger than 40 years old from the population of 60,000,000. We can take 1% of the population as a sample to estimate people who are younger than 40 years old. When we divide the result with the sampling ratio, in this case, it is 1%. we find the estimate according to the original size of the population. If we know the exact result of the problem, the difference between the original result and the sample result is the sampling error. We couldn't identify the exact amount of error associated with any given sample if we didn't know the result from the original data. However, we can estimate the degree of errors that may be expected with the given sample size. For instance, we could do this by obtaining a sample of 600,000 randomly selected people (which is 1% of all population) from the population and finding the results from each sample. Each result is an estimation of the problem given. If we do the same process 100 times or more, we will obtain different results. These different results will help us to create a graph and see the standard deviation of the population. The standard deviation of the population is called standard error at the same time. In this way, we can estimate the error. Error estimation is a very important topic in statistics, and data analytic to find out the accuracy of the estimate. It ensures reliable observations from a given population. Also, querying a large data set is expensive computation by using smaller samples from the original data reduces the time cost.

2 Background

2.1 Simple Random Sampling for Query Estimation

When analyzing the dataset, it is not required to analyze the complete dataset, and examining the complete dataset is deemed too costly in terms of response time or resource consumption. When simple random sampling is utilized, there will be savings. The savings from sampling can come from lower data retrieval costs or further post-processing of the sample. When working with vast administrative or scientific data sets, access costs can be significant. Post-processing of the sample may include costly statistical calculations or additional physical verification of the real-world entities identified by the sample. As an example, physical audits of annual salary records of employees or analyzing scientific records that have been collected. Random sampling has different kinds of techniques to establish sampling [15] such as a binomial random sample, a simple random sample without replacement (SRSWOR), a simple random sample with replacement (SRSWR), a stratified random


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: Simple Random Sampling 20%

sample, weighted random sample, and more. In this paper, a simple random sample without replacement was used. In SRSWOR's definition, a population in which every constituent of the population has an equal chance of being included in the example Duplication is not permitted. Thus, there is no duplication in the sample data set. SRSWOR is helpful for query estimations and it increases efficiency[16]. Another study using SRSWOR shows that it is useful to use SRSWOR for query size estimation [23]. SRSWOR to estimate query formula to predict query size (COUNT function) is $Y_s = \sum_{i=1}^{n} y_i$ where Y_s is the query result of the sample table with COUNT aggregation function, y_i is the query result of each sample tuple and n is the size of sample table.

In Figure 1, there is an example data set θ with 100 records in it. To create a sample data set using SRSWOR with a 20% sampling ratio, first getting two random numbers between one to twenty, and sort them, then selecting the records based on the unique id of the records using the twenty random numbers that are generated. As result, the sample data set $\hat{\theta}$ with twenty records has been obtained.

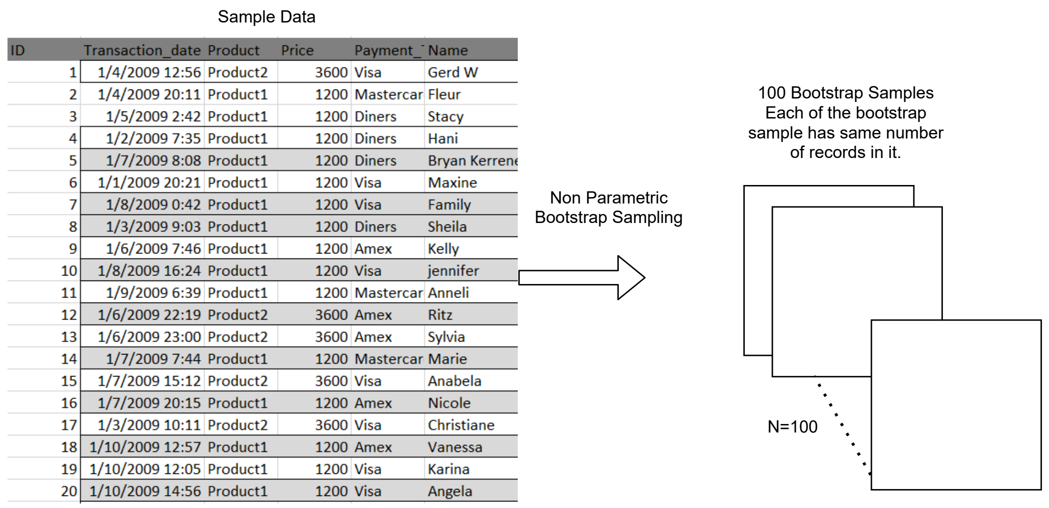
2.2 Bootstrap Sampling (BS)

The bootstrap method is one of the most widely used statistical approaches [8]. The primary goal of bootstrap methods or algorithms is to build big data sets and re-sample from them. The bootstrap approach generates an interpretation by redrawing some parameters based on statistical inferences. This technique is then done several times to increase its reliability. Variance estimations are generated effectively using the bootstrap approach, which is widely used for variance estimations. The bootstrap technique is a method that does not rely on complex mathematical formulas, has limited assumptions, and is simple to learn and apply. It produces consistent outcomes, even when the assumptions are inadequate. The bootstrap method is used in probability, confidence intervals, hypothesis testing, and regression analysis [21].

The bootstrap method is used to calculated standard errors. When other approaches are computationally difficult, unreliable, or unavailable, it can be useful for this purpose.[12] Let $\hat{\theta}$ is the estimate parameter, corresponded estimation of x th bootstrap sample is $\hat{\theta}^*_x$, and $\bar{\theta}^*$ is the mean of $\hat{\theta}^*_x$ Then the standard deviation of $\hat{\theta}^*_x$ is

$$s^*\left(\hat{\theta}\right) = \left(\frac{1}{N-1}\sum_{x=1}^{N}\left(\hat{\theta}^*_x - \bar{\theta}^*\right)^2\right)^{\frac{1}{2}}$$

To generate asymptotic confidence intervals or run asymptotic tests, we can use $s^*\left(\hat{\theta}\right)$, in the same manner, we would any other asymptotically valid standard error. For the bootstrap confidence interval, there are a lot of techniques. Davison and Hinkey explained


Sample Data

ID	Transaction_date	Product	Price	Payment_	Name
1	1/4/2009 12:56	Product2	3600	Visa	Gerd W
2	1/4/2009 20:11	Product1	1200	Mastercar	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	Mastercar	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	Mastercar	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 2: Bootstrap Sampling

confidence interval in [5].

For example, Let's use the data set θ, and $\hat{\theta}$ sample data set generated with SRSWOR. Sample data set $\hat{\theta}$ has 20 records in it, and we need a 100 bootstrap sample for the given $\hat{\theta}$. In this case, the number of the bootstrap sample (B) is 100. The bootstrap samples are constructed using nonparametric bootstrap[2].

There are different methods of bootstrapping to calculate the confidence interval.[18] To explain each of the methods, in increasing complexity, we describe the essential logic underlying them. We will assume that we want to estimate a population parameter in each sample θ, and the estimator of that parameter by the sample is $\hat{\theta}$. Standard bootstrapping generates B bootstrapping samples and computes an estimate of the population parameter for each. These estimates are $\hat{\theta}^*_1,...,\hat{\theta}^*_N$. .

For the standard bootstrapping or basic bootstrapping to find confidence interval, error of the estimate $\hat{\theta}^*_x$ is obtained with $\hat{\theta}^*_x - \hat{\theta}$. Then, lower e_l and upper e_u bounds encircle the center $100(1-\alpha)$ (if α represents the level of significance is 0.05 which means 95% CI) of these errors. After that, the confidence interval enclose the central of $100(1-\alpha)$ is obtained with $[\hat{\theta}-e_l, \hat{\theta}+e_u]$. The bootstrap distribution of errors is assumed to be a decent estimate for the true distribution of sampling errors in this method.

The percentile bootstrap has also used the found interval.[7] Let's assume that $\hat{\theta}_1^*$,...,$\hat{\theta}_N^*$ are the bootstrap samples, and they are ordered. After the estimate is calculated, CI can be calculated with $100(\frac{\alpha}{2})$th percentile of the distribution and $100(1-\frac{\alpha}{2})$th percentile distribution.

3 Bootstrap for Select Query Estimation

Query estimation has been a popular research area since the beginning of it [22]. Query estimation is not only becoming important in academia, but it also became very popular in the business world. Big data query systems started to support query estimations as well. In A Handbook for Building an Approximate Query Engine [14], the literature explains the details.

3.1 Methodology

There are numerous query selection methods that have been used [13]. In this experiment, a different approach was applied to increase the efficiency of the program. One of the aggregate functions, COUNT, has been used to get an estimation of the number of rows in the given condition (query size). Since bootstrapping was used on sample data set $\hat{\theta}$, the ground truth of the given query has been calculated prior to bootstrap calculation. Thus, The query result for each record has already been marked and saved in an array as ordered in-sample data set $\hat{\theta}$. Instead of using sample data to generate bootstrap samples, The resulting array is used which provides efficiency in terms of the query selection process. After the sample query result S_Q is calculated by executing query Q on sample table, Bootstrap samples can be generated. To calculate estimation of bootstrap replications the formula is

$$\hat{Y}_j = \frac{Y_j}{f} \tag{1}$$

where Y_j is the query result of the jth bootstrap sample and, f is the sampling ratio. For example if we want to calculate estimation of bootstrap replications with the COUNT function the formula is

$$\hat{Y}_j = \frac{1}{f}\sum_{i=1}^{n} y_{i,j} \tag{2}$$

where $y_{j,i}$ is the $i$the tuple result in the jth bootstrap sample.

3.2 Optimization of Bootstrap for Query Estimation

The bootstrap's broad applicability and automaticity have been demonstrated in both theory [1, 9] and practice [17]. Unfortunately, because hundreds or even thousands of bootstrap trials are often required to generate trustworthy estimates, bootstrap struggles from a large computational overhead [11]. First, the computation of bootstrap has been done by storing the sample data $\hat{\theta}$ in disk storage. However, it creates a bottleneck for the file access time on a disk stored as a CSV file especially when the original data set θ increases. There have been studies shows that reading data from memory for big data is more efficient than reading data from disk [19]. For this reason, while getting the sample data set $\hat{\theta}$ from the original data set θ, the sample data set store in the memory as an array with query results. To do that, first, the program finds the randomly selected record in the original data set θ, then evaluates the record with the given query. If the record satisfies the query, then the record is saved with one otherwise it is saved as zero. In this way, the bottleneck is avoided. Also, while doing the bootstrap sampling, the data have already been evaluated according to the given query. Hence, the speed performance increases significantly. Since the sample data set $\hat{\theta}$ is stored in the memory array, there is no need to use a sort algorithm to sort them because accessing array element is faster than accessing records stored in the disk as a CSV file. If the sample data is stored in a file, we need to loop through the find selected tuple. If the sample data stored in an array we can directly access to selected tuple with index number of the array. This also contributes to the efficiency of generating bootstrap samples. When the sample data set is stored in a CSV file, sorting is needed to avoid iteration to generate bootstrap samples. To do that, the quick sort was chosen because the average time complexity of it is $O(n \log n)$. Even though quick sort is efficient for sorting, still it does not provide expected efficiency for the big data set. Thus, using a memory array to store sample data $\hat{\theta}$ conduces generating bootstrap samples. Generating nonparametric bootstrap confidence intervals with different methods and their improvements have been discussed [6]. In this experiment, bootstrap sample size B has been tested with B = 2000 and B = 200, and the results were observed in terms of computation time and accuracy. When the results are analyzed in Figure 3 and Figure 9, The original data set size is 1GB and the sampling ratio is 0.1%. The graph results show that if the bootstrap sample size is lowered from 2000 to 200, the bootstrap computation time goes down from 12 seconds to 1.8 seconds which provides sufficient time optimization for bootstrap computation. When the sampling ratio increases from 0.1% to 0.5% in Figure 4 and Figure 10, the graph results show that the bootstrap computation time goes down from 62 seconds to 11.5 seconds. Lastly, When the sampling ratio increases from 0.5% to 1% in Figure 5 and Figure 11, the graph results show that the bootstrap computation time goes down from 1500 seconds to 190 seconds. Overall, the graphs illustrate the performance of the program increases, and when the sampling ratio increases, the performance of the program gets better with larger sample data set. In terms of accuracy, the figures in the hit ratio change part demonstrate the results. For example, if we analyze Figure 12 where B is 2000 as well as Figure 13 where B is 200, the results show that the hit ratio is almost the same with 100 MB data sets for both B is 2000 and B is 200. Another example, if we look at Figure 14 where B is 2000 as well as Figure 15 where B is 200, the figures show that the hit ratios affect from the bootstrap sampling size N. If we look at Figure 16 where B is 2000 as well as the Figure 17 where B is 200, the figures show that the hit ratios affect from the bootstrap sampling size B. As a result, lower B can be used to increase speed but accuracy will be the trade-off. If the data analysis requires sensitive calculation, B should be chosen 2000.

3.3 Implementation

This experiment consists of query processor, sampler bootstrap computing, and hit ratio computation. In this study, we take for the following query formulation:

SELECT Aggregation(attribute collection) FROM table name WHERE conditions

In query processor implementation, there are 10 queries (see the queries in Appendix A). The first 5 queries are large, and the other 5 of them are small queries. Each query runs 10 times. The queries are saved in a txt file. The queries were parsed from the txt file and saved into a dictionary. The query parser selects the associate attributes from the record and evaluates its records. In the program, instead of creating the sample data set with attributes of selected each record, the result of the query evaluation of each record is done by the query processor and saved into a memory array (See Figure 3).

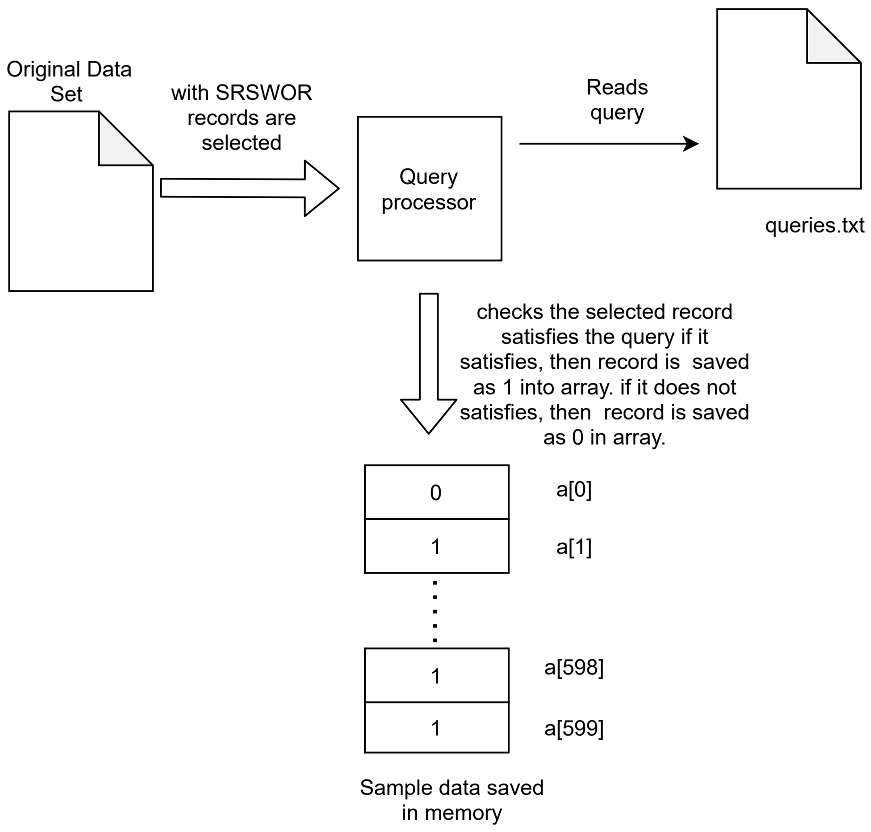

Original Data Set
with SRSWOR records are selected
Query processor
Reads query
queries.txt
checks the selected record satisfies the query if it satisfies, then record is saved as 1 into array. if it does not satisfies, then record is saved as 0 in array.
0
a[0]
1
a[1]
1
a[598]
1
a[599]
Sample data saved in memory

Figure 3: Query Processor

In the sampler part of the experiment. SRSWOR has been used. first, the program counts the number of records in the original data set *R* then it produces a random number according to the number of records in the original data set and saved into an array. Next, random numbers were sorted with the quick sort algorithm. The program compares each record number with each random number that has been saved into an array. If it matches then the query processor evaluates the query. After that, it saves each record into an array in the memory. As a result, the sample data S is created.

To create bootstrap samples $S^*_1,...,S^*_B$, The number of bootstrap samples, B, can be 2000 and 200 in the experiment. The sample data S was used to generate the nonparametric bootstrap sample. A sample of the same size as the data is taken from the data with replacement in the nonparametric bootstrap. The records randomly selected from sample data stored in array then, bootstrap samples are generated.

Hit ratio was calculated with using the bootstrap samples $S^*_1,...,S^*_B$. Calculation of the hit ratio consists of the following steps: calculating the ground truth Y_G of original data, calculating estimation of ground truth $\hat{Y}$ using the sample data, calculating the confidence interval (CI) with bootstrap samples, and check the ground truth within the confidence interval. This process was done 10 times for each query. The query result of the original data is the ground truth of original data. To calculate CI, first, finding the query estimation of each bootstrap sample which is $\hat{Y}_j$. Second, find the mean of bootstrap samples. The formula to find mean of bootstrap sample is

$$\mu = \frac{\hat{Y}_1 + \hat{Y}_2 + \cdots + \hat{Y}_B}{B} \tag{3}$$

The formula of standard deviation S_B of bootstrap samples is

$$S_B = \left[\frac{(\hat{Y}_1 - \mu)^2 + (\hat{Y}_2 - \mu)^2 + ... + (\hat{Y}_B - \mu)^2}{B - 1} \right]^{\frac{1}{2}} \tag{4}$$

After standard deviation is calculated, we use CI formula

$$CI = [\hat{Y} \pm S_B \cdot Z] \tag{5}$$

where $\hat{Y}$ is the query estimation, S_B is bootstrap standard error, and Z is the value from the standard normal distribution for the chosen confidence level. In this experiment, 95% confidence intervals are used with the Z value 1.96.

4 Experiment

4.1 Experiment Setup

For this experiment, one of the GPU servers in Datalab[1] at Youngstown State University had been used. The server has CentOS Linux release 7.9.2009 as an operating system. The hardware specifications are Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50GHz, 8GB of RAM, and GeForce GTX 1080 GPU.

Three data sets have been used for the experiment. The data sets are generated using TPC-H benchmark [3]. The sizes of the data sets are 100MB, 1GB, and 10GB

[1]datalab.ysu.edu

respectively. In this experiment, there is a total of 10 queries. The first 5 queries are large queries and the last 5 queries are small queries. Testing queries are located in Appendix A part of this paper. For the sample of the original data, three different sampling ratios are used and they are 0.1%, 0.5%, and 1%. The program checks if the ground truth of the original data set between the upper and lower confidence interval. If the original data between the confidence interval then it is a hit. Otherwise, it is a miss. These processes were applied for each query 10 times. After that, the hit ratio was calculated. You can access the source code on GitHub[2].

4.2 Speed Performance

The speed performance of the experiment was measured in terms of File access time, Simple random sampling time, and Bootstrapping time.

4.2.1 Simple Random Sampling Time

Sampling performance depends on the size of the data as well as the sampling ratio. If the sampling ratio is increased, the sampling process will increase as well. As we can see the figures below.

In Figure 4, it shows 1GB data with a 0.1% sampling ratio. The simple random sampling time takes approximately 5 to 6 seconds.

[2]https://github.com/YSU-Data-Lab/Semih_Cal_Thesis_Summer_2021

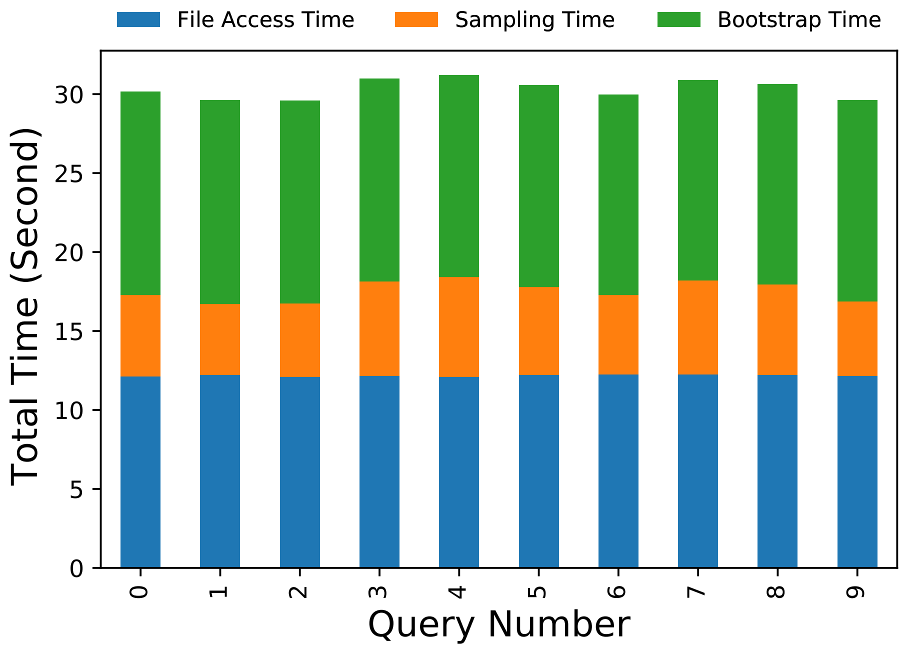


Figure 4: Time Graph 1GB with 0.1% B=2000

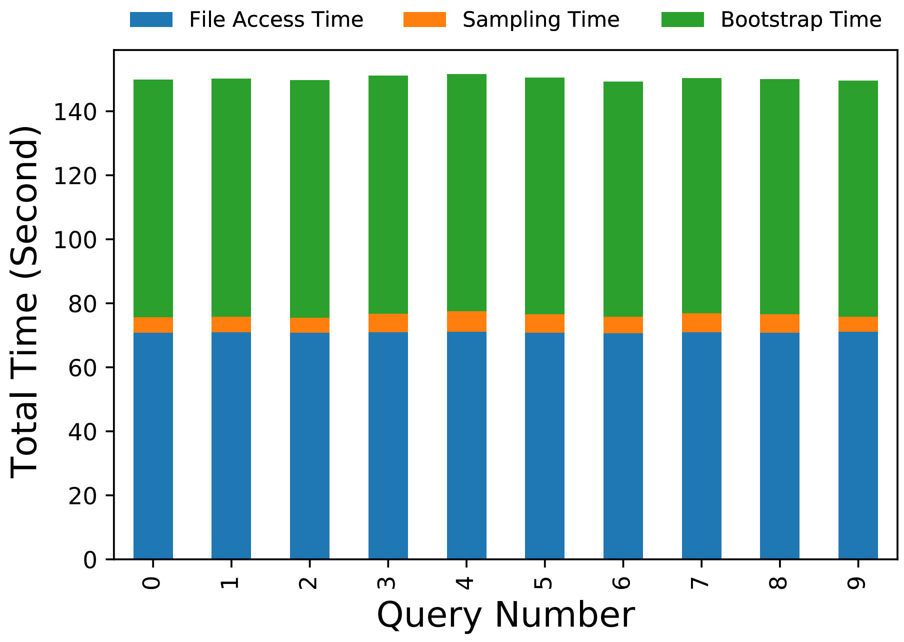


Figure 5: Time Graph 1GB with 0.5% B=2000

In Figure 5, the simple random sampling time increased a little bit higher than the previous figure. Because the sampling ratio is 5 times bigger than the previous one. This means that the number of records in the sample data is increased.

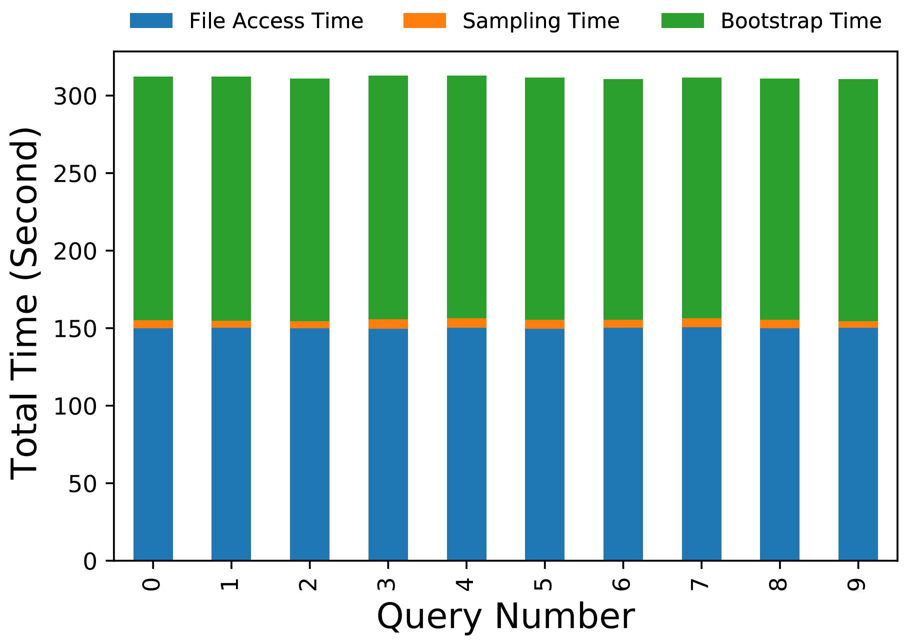


Figure 6: Time Graph 1GB with 1% B=2000

In Figure 6, the simple random sampling time takes the highest amount of time for the 1GB data. The sampling ratio is five times bigger than figure 4 and 10 times bigger than Figure 5. As a result, simple random sampling time takes more time when the sampling ratio increase.

4.2.2 Bootstrap Time

Bootstrap sampling time depends on sampling ratio and B is the number of bootstrap samples to produce. In the 3 figures below, bootstrap times with B is equal to 2000 illustrated. In the first figure, the sampling ratio is 0.1% bootstrap process faster than the other 2 graphs because the sample data that user has the smallest number of records and there are 2000 bootstrap samples created. In the second figure (Figure 8) the number of records increased in the sample data. If the number of records increases, the bootstrap sampling process increases as well. In the last figure(Figure 9), the sampling ratio is 1% and bootstrap sampling takes more time.

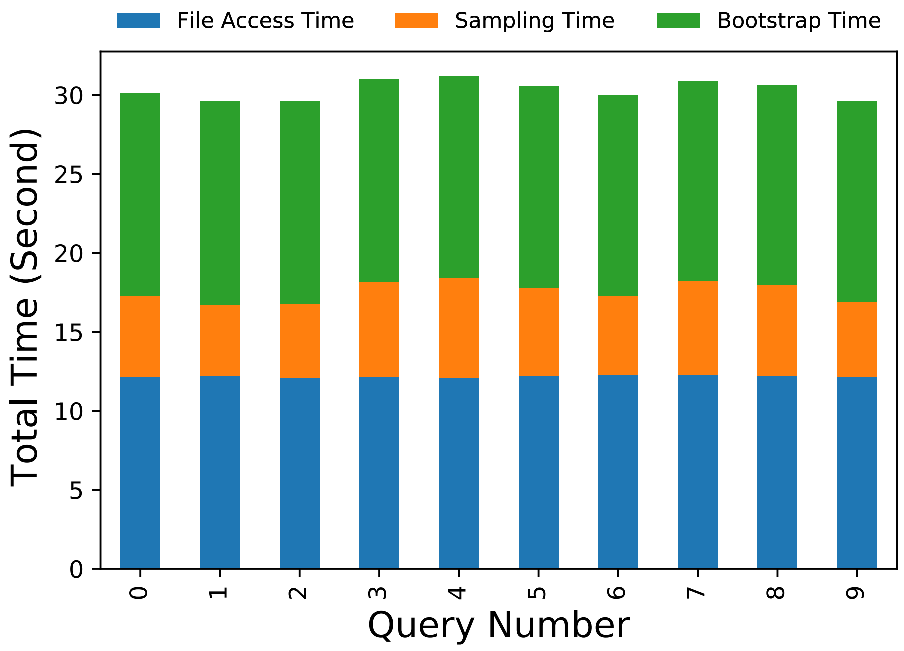


Figure 7: Time Graph 1GB with 0.1% B=2000

The figures (Figure 10,11,12) demonstrate the time graph with B is 200. Since the number of bootstrap samples decreased from 2000 to 200, bootstrap time decreases as well. However, when the sampling ratio increases in the figures, the bootstrap time increases too. Lowering the B provides faster calculation and the accuracy remains reliable as you can see in the accuracy performance section 4.3.

Consequently, bootstrap time depends on the sampling ratio and the number of bootstrap samples N. Both of them are direct proportional to bootstrap time.

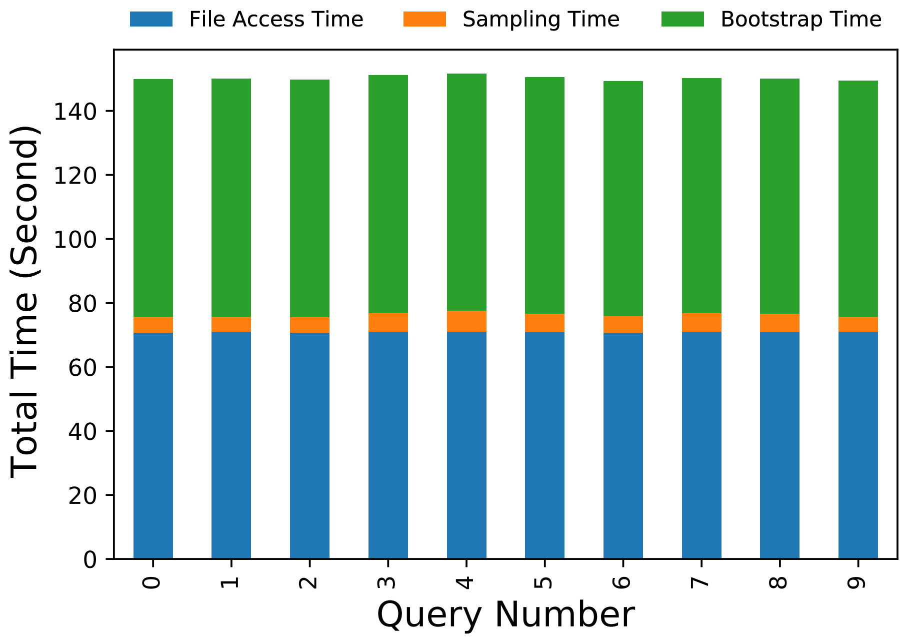


Figure 8: Time Graph 1GB with 0.5% B=2000

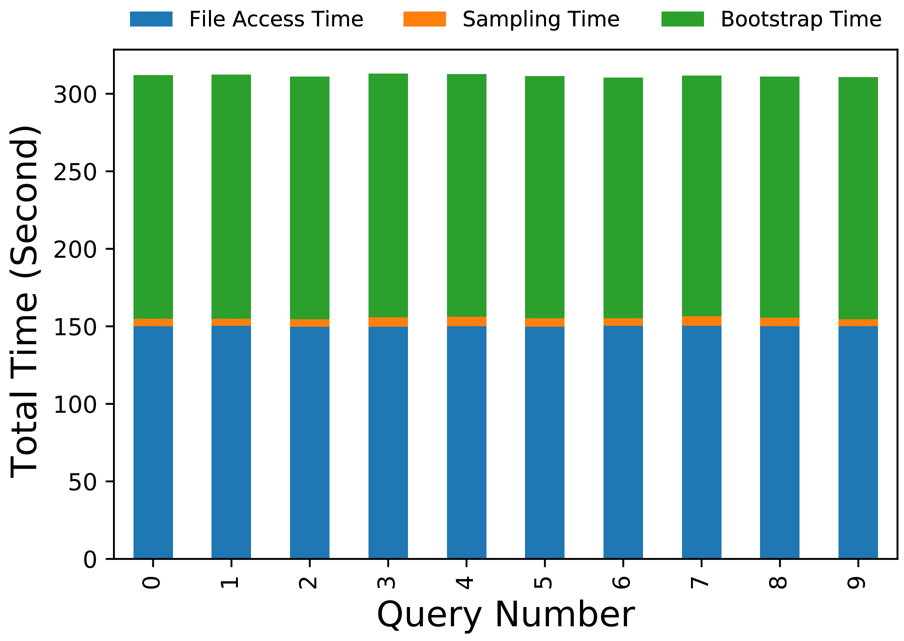


Figure 9: Time Graph 1GB with 1% B=2000

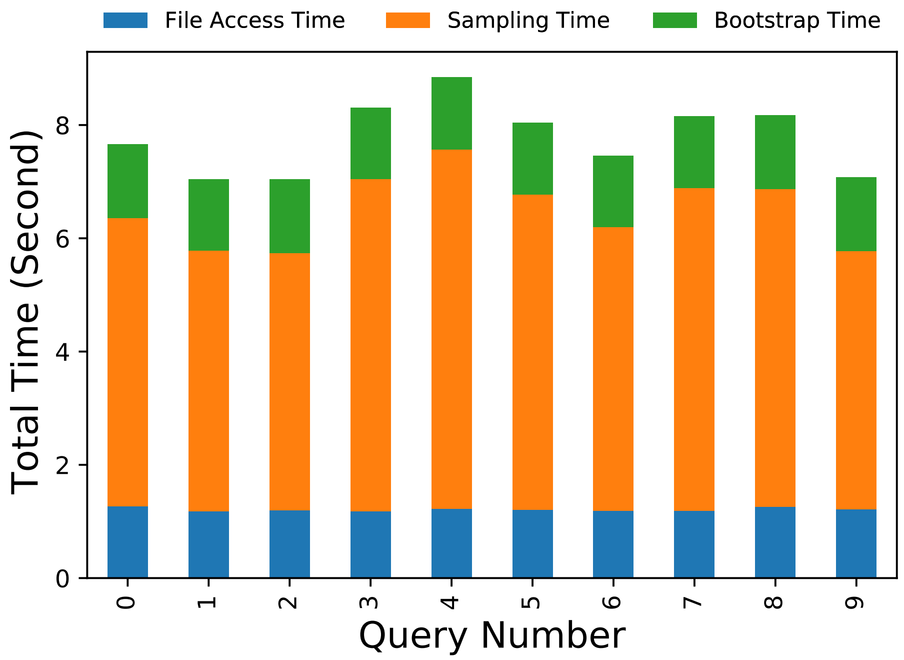


Figure 10: Time Graph 1GB with 0.1% B=200

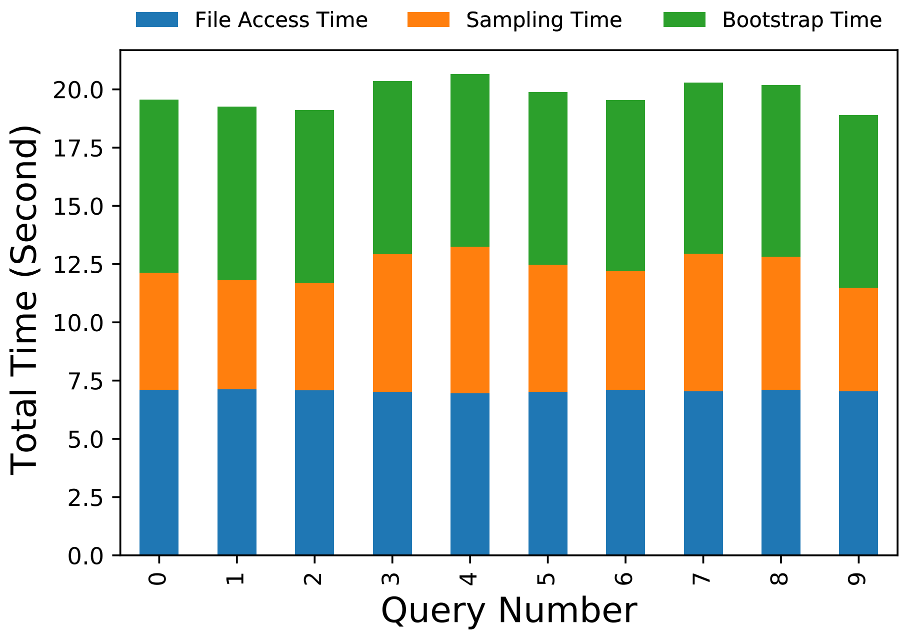


Figure 11: Time Graph 1GB with 0.5% B=200

4.3 Accuracy Performance

In this experiment, the number of hits are calculated with ground truth of original data and ground truth of sample data with different sampling ratio. the number of the hit for the

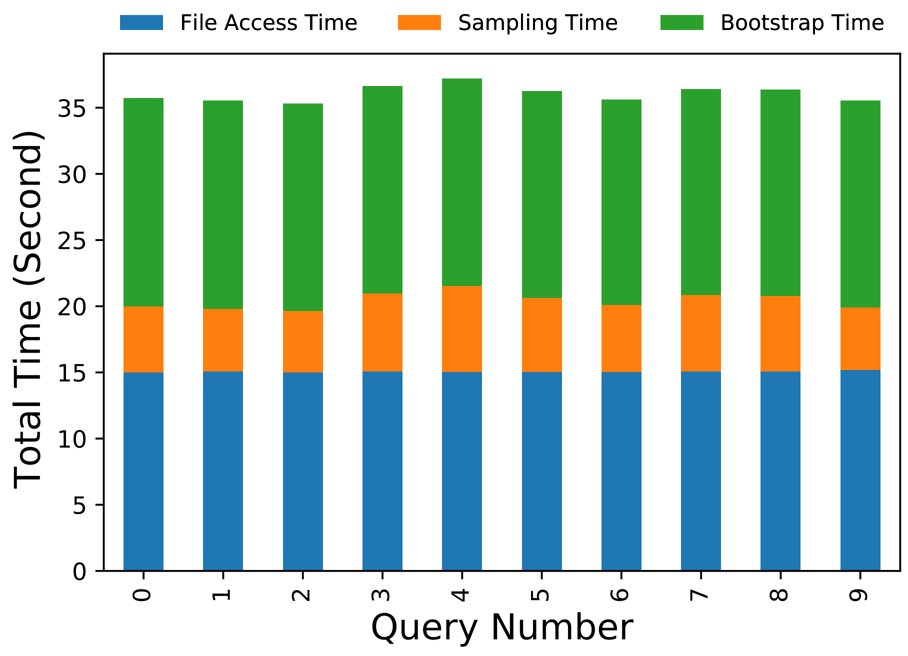


Figure 12: Time Graph 1GB with 1% B=200

sample data found by dividing the sample ratio by the number of the hit for the sample data to find the approximate number of hits. Then, bootstrap sampling is used to calculate CI (confidential interval). If the ground truth of the original data between the CI, that means it is a hit. Otherwise, it is not a hit. Based on this logic, the accuracy was calculated using 10 different queries, and each query was run 10 times then, the percentage of hit ratio was calculated for each query with a different sampling ratio. The calculation of hit ratio is

$$\text{hit ratio} = \frac{\text{times (CI hits)}}{\text{times(total experiment)}} \times 100\% \tag{6}$$

4.3.1 Hit Ratio Change

The hit percentage was calculated for 100MB, 1GB, and 10GB data sets with 0.1%, 0.5%, and 1% respectively. Beside of the size of the data sets and sampling ratio, two different number of a bootstrap sample (B=2000 and B=200) was used for observing if the hit ratio affected by N. In Figure 13, hit percentage of 100 MB data sets was used. The number of bootstrap samples is 2000. Each query run 10 times with three different sampling ratio. the blue bar is 0.1%, the orange bar is 0.5%, and the green bar is 0.1%. The graph shows that the hit ratio for the 100 MB data set is 100% for most of the queries and miss for each query no more than 20%. when the sampling ratio increases, the hit ratio also increases in most of the queries. In Figure 14, a hitting percentage of 100 MB data sets was used. The number

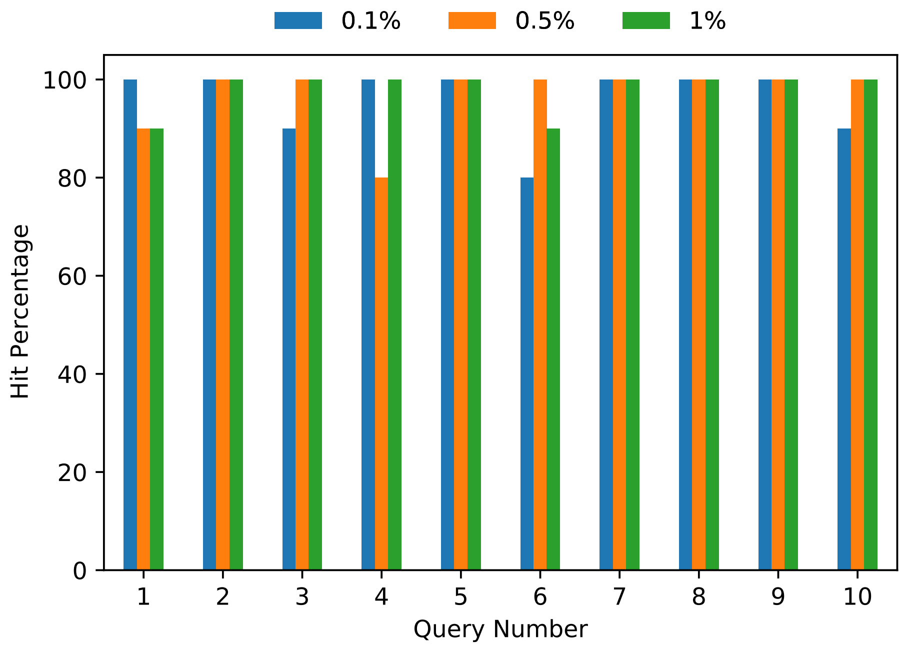


Figure 13: Hit Percentage Graph 100MB data set B=2000

of the bootstrap sample (B) is 200. The graph shows that the hit ratio for the 100 MB data set is also 100% for most of the queries and miss for each query no more than 20%. when the sampling ratio increases, the hit ratio also increases in most of the queries. When the B decreases from 2000 to 200, the hit percentage goes down a little bit in general. Because CI is calculated with Bootstrap samples. When the number of bootstrap decreases, The CI narrows down which affects the hit percentage.

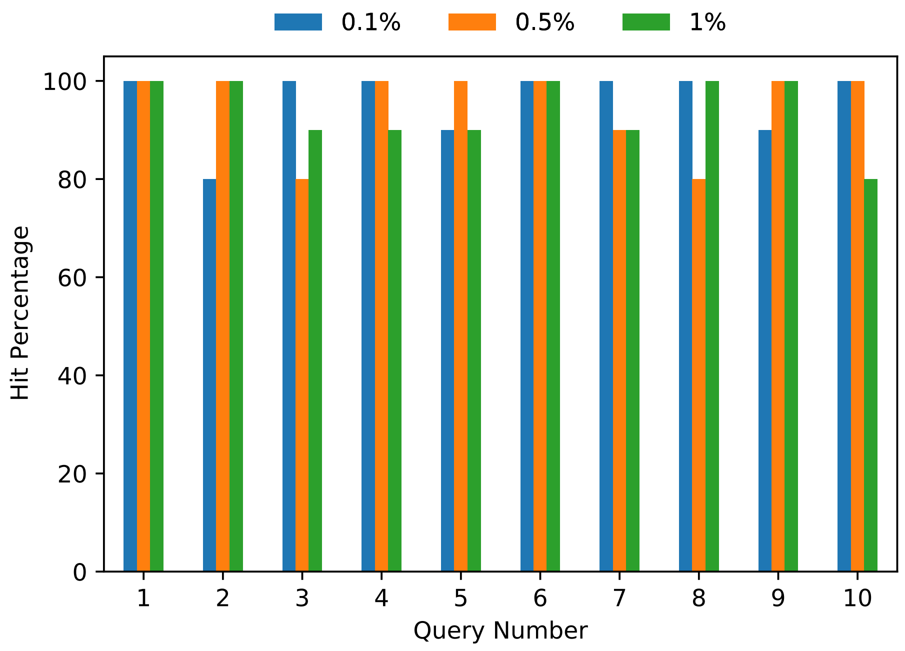


Figure 14: Hit Percentage Graph 100MB data set B=200

In the Figure 15, a hitting percentage of 1 GB data set was used. The number of bootstrap samples is 2000. Since the size of the data set increased, the results in the figure more accurate. The graph shows that the hit ratio for the 1 GB data set is 100% for most of the queries and miss for each query no more than 20%. When the sampling ratio increases, the hit ratio also increases in most of the queries.

In the Figure 16, a hit percentage of 1 GB data set was used. The number of bootstrap samples is 200. The graph shows that the hit ratio for the 1 GB data set is 100% for most of the queries and miss for each query no more than 30%. when the sampling ratio

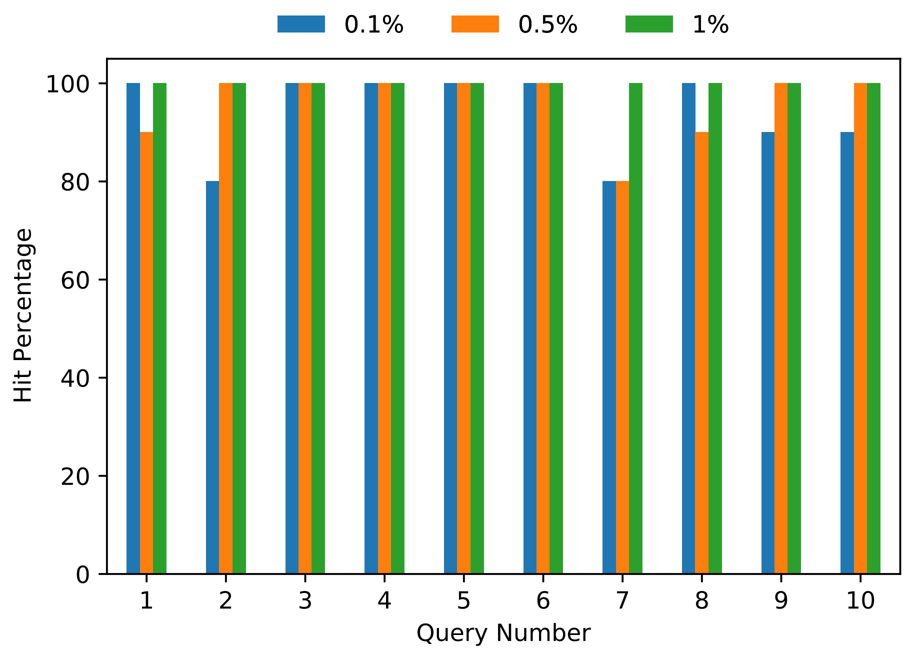


Figure 15: Hit Percentage Graph 1GB data set B=2000

increases, the hit ratio also increases in most of the queries. Since the B decreases from 2000 to 200, the hit ratio goes down on this figure because of the CI narrow down. Even though the data set increased, CI still affects by the number of bootstrap samples B. In query 7, the hit ratio goes down even though the sampling ratio increases. It is an exception that the simple random sampling selected the records that did not meet the query condition. It is a possibility that occurred on this figure.

In the Figure 17, a hit percentage of 10 GB data sets was used which is the largest data set that was used on this experiment. The number of bootstrap samples is 2000. The graph shows that the hit ratio for the 10GB MB data set is 100% for most of the queries and miss for each query no more than 20%. The hit ratio did not change for each query so, larger data sets can be used for query estimations. When the sampling ratio increases, the hit ratio also increases in most of the queries.

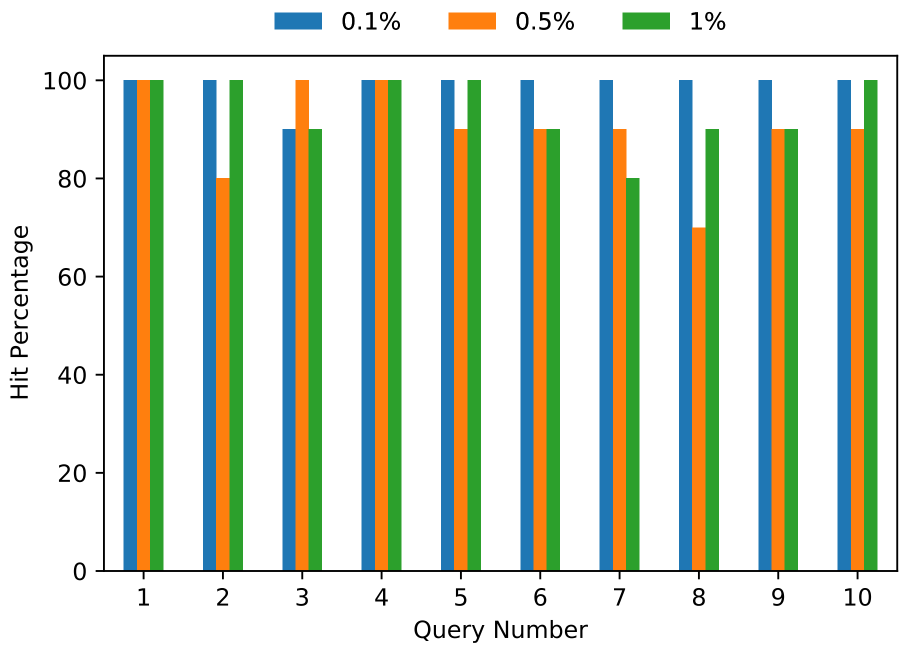


Figure 16: Hit Percentage Graph 1GB data set B=200

In the Figure 18 , a hit percentage of 10 GB data sets was used. The number of bootstrap samples is 200. The graph shows that hit ratio for the 10GB MB data set is 100% for most of the queries and the miss for each query no more than 20%. when the sampling ratio increases, the hit ratio also increases in most of the queries. Again, the figure shows similarities with 100 MB and 1GB data sets with B=200. When the B decreases from 2000 to 200, the hit ratio goes down because the CI narrows down. Even though the data set increased, CI still affects by the number of bootstrap samples N.

The hit ratio has been affected by the sampling ratio and Number of bootstrap samples B. In this experiment there are three different sampling ratios and they are 0.5%, 0.1%, and 1%. When the sampling ratio increases, the hit ratio also increases on most of the queries. So, increasing the sampling ratio gives better performance. However, it has a drawback in terms of speed. Because there will be more records to go through to create

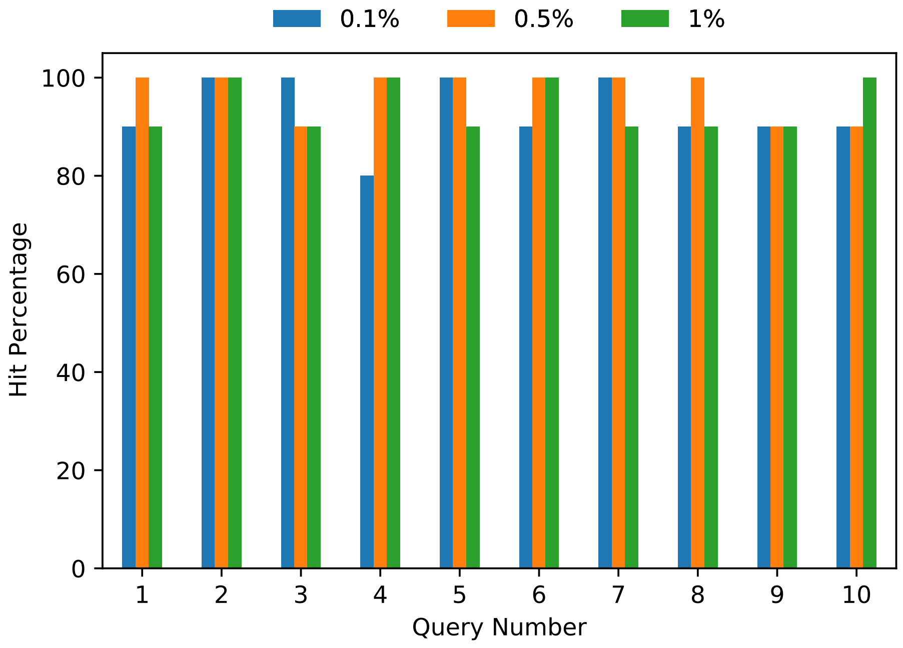


Figure 17: Hit Percentage Graph 10GB data set B=2000

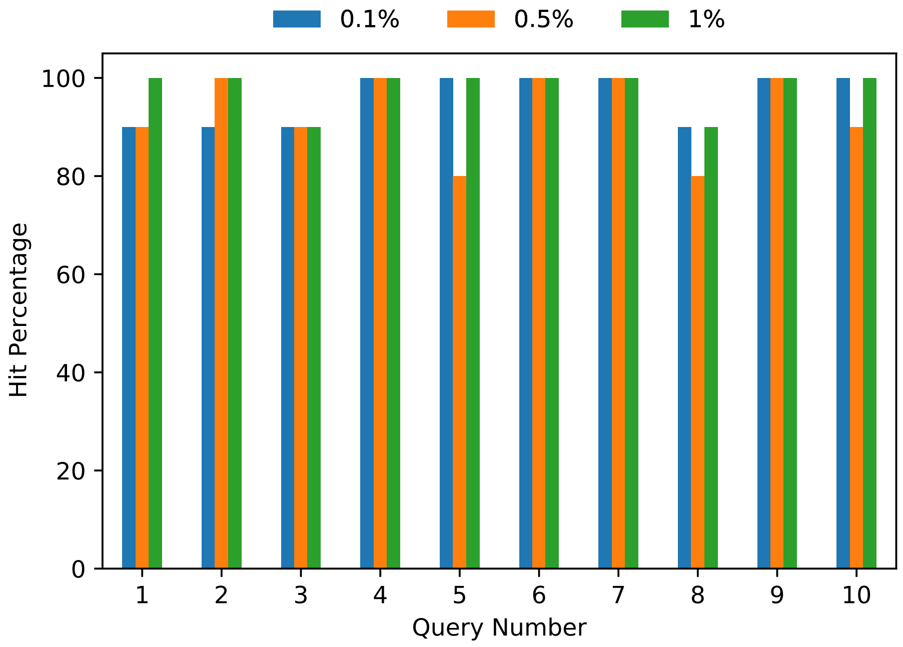


Figure 18: Hit Percentage Graph 10GB data set B=200

sample data set. Another factor that affects the hit ratio is B. When B decreases, the hit ratio goes down too in all figures. Because to get a hit the ground truth of the original table has to be between the CI which is calculated by bootstrap samples. Therefore, increasing the B gives better results.

4.4 Optimization Results

Implementation of a basic optimization method increased the performance of bootstrap sampling significantly. As you can see in the figures below (Figures 19,20,21), storing the sample data in an array and lowering the total number of bootstrap sample B, provides a more efficient bootstrap sampling calculation. The graphs show that the speed ratio is up to 6.47 times better than the previous implementation.

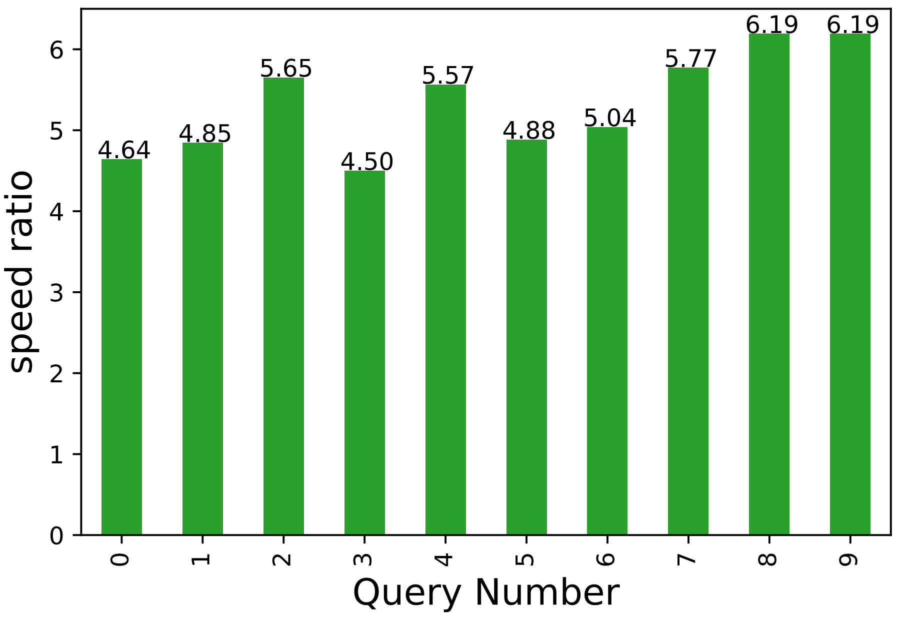


Figure 19: Sampling time of 1GB data with 0.1% sampling ratio

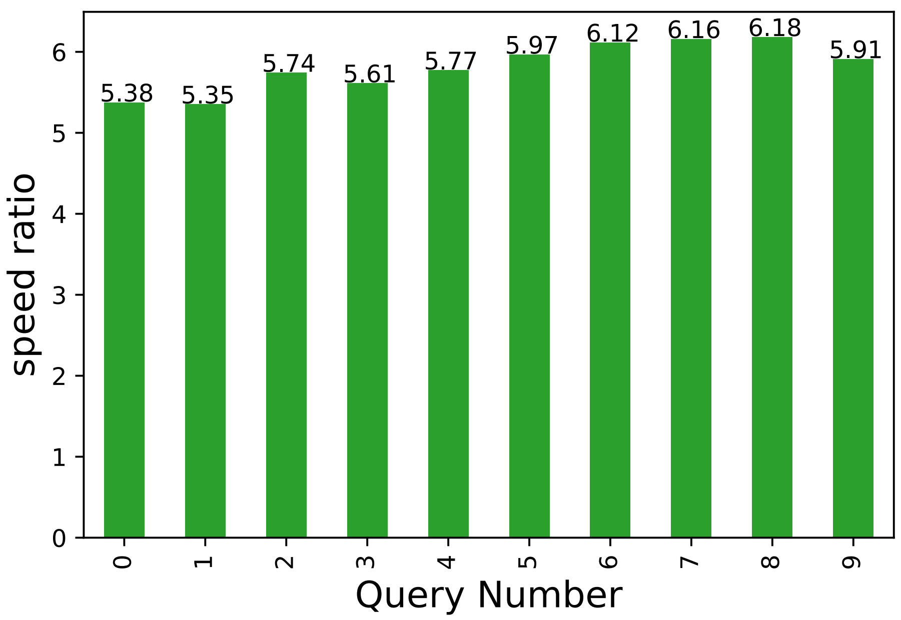


Figure 20: Sampling time of 1GB data with 0.5% sampling ratio

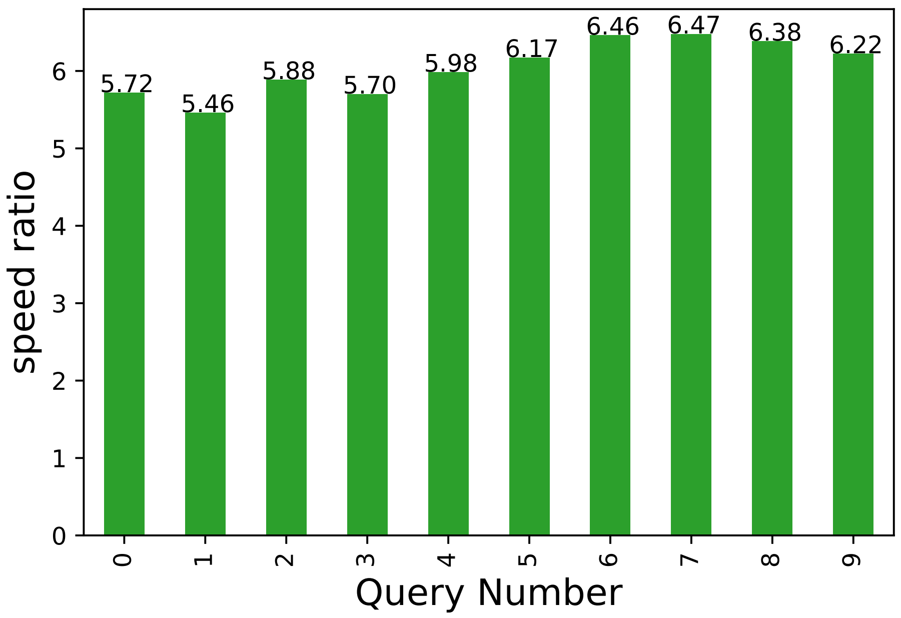


Figure 21: Sampling time of 1GB data with 1% sampling ratio

5 Conclusion

This research evaluates query estimation errors using bootstrap sampling. Querying the large size of data is timely expensive. Query estimation from the small sample data reduces the time cost. We designed a query processor mechanism and bootstrap sampling mechanism which can calculate the CI for query estimates. Three different sampling ratio (0.1%, 0.5%, 1%) were used in this work to determine hit ratio affects on the sample size. the non-parametric bootstrap sampling method was used to determine confidence intervals. It tested with 2000 and 200 Iterations. For the optimization on the bootstrap estimation is the number of samples lowered and the results show if we use 200 samples still the hit ratio remains reliable. Also, the performance increases since fewer samples were used. Another optimization approach that was tested on this experiment is saving the samples into memory instead of the disk as storage. In the hit ratio part of the experiment, different sampling ratio and different B values were used. When the sampling ratio increases, the hit ratio also increases but even with the small sampling ratio the results are still reliable. Even though the program has optimizations to lower computation time, using GPU for the bootstrap sampling will increase the time efficiency. Future research will be based on this idea to boost up program. GPU can compute bootstrap sampling faster than CPU since it has more threads than CPU and all of them are working in parallel. Also, smaller sampling ratios will be tested to determine what can be the smallest ratio to obtain reliable results.

References

[1] Bickel, P. J., and Freedman, D. A. Some asymptotic theory for the bootstrap. *The annals of statistics* (1981), 1196–1217.

[2] Carpenter, J., and Bithell, J. Bootstrap confidence intervals: when, which, what? a practical guide for medical statisticians. *Statistics in medicine 19*, 9 (2000), 1141–1164.

[3] Council, T. P. P. Tpc-h benchmark specification. *Published at http://www. tcp. org/hspec. html 21* (2008), 592–603.

[4] Davenport, T. H., and Dyché, J. Big data in big companies. *International Institute for Analytics 3* (2013), 1–31.

[5] Davison, A. C., and Hinkley, D. V. *Bootstrap methods and their application*. No. 1. Cambridge university press, 1997.

[6] Davison, A. C., Hinkley, D. V., and Young, G. A. Recent developments in bootstrap methodology. *Statistical Science* (2003), 141–157.

[7] Efron, B. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

[8] Efron, B., and LePage, R. *Introduction to bootstrap*. Wiley & Sons, New York, 1992.

[9] Ghosal, S., Sen, A., and Van Der Vaart, A. W. Testing monotonicity of regression. *Annals of statistics* (2000), 1054–1082.

[10] Kapoor, R., and Virk, R. Selectivity & cost estimates in query optimization in distributed databases. *International journal of enhanced research in management and computer applications* (2013).

[11] Kleiner, A., Talwalkar, A., Agarwal, S., Stoica, I., and Jordan, M. I. A general bootstrap performance diagnostic. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), pp. 419–427.

[12] MacKinnon, J. G. Bootstrap methods in econometrics. *Economic Record 82* (2006), S2–S18.

[13] Markovic, J., and Taylor, J. Bootstrap inference after using multiple queries for model selection. *arXiv preprint arXiv:1612.07811* (2016).

[14] Mozafari, B., and Niu, N. A handbook for building an approximate query engine. *IEEE Data Eng. Bull. 38*, 3 (2015), 3–29.

[15] Olken, F. *Random sampling from databases*. PhD thesis, University of California, Berkeley, 1993.

[16] Olken, F., and Rotem, D. Simple random sampling from relational databases.

[17] Pol, A., and Jermaine, C. Relational confidence bounds are easy with the bootstrap. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (2005), pp. 587–598.

[18] Puth, M.-T., Neuhäuser, M., and Ruxton, G. D. On the variety of methods for calculating confidence intervals by bootstrapping. *Journal of Animal Ecology 84*, 4 (2015), 892–897.

[19] Rasmussen, A., Lam, V. T., Conley, M., Porter, G., Kapoor, R., and Vahdat, A. Themis: an i/o-efficient mapreduce. In *Proceedings of the Third ACM Symposium on Cloud Computing* (2012), pp. 1–14.

[20] Schermann, M., Hemsen, H., Buchmüller, C., Bitter, T., Krcmar, H., Markl, V., and Hoeren, T. Big data. *Business & Information Systems Engineering 6*, 5 (2014), 261–266.

[21] Tibshirani, R. J., and Efron, B. An introduction to the bootstrap. *Monographs on statistics and applied probability 57* (1993), 1–436.

[22] Vrbsky, S., Smith, K., and Liu, J. An object-oriented semantic data model to support approximate query processing. In *Proceedings of IFIP TC2 Working Conference on Object-Oriented Database Semantics* (1990).

[23] Wilson, D. Correlated sample synopsis on big data, 2018.

Appendix A   Testing Queries HQL Code

- select count(*) from lineitem where L_QUANTITY < 20 and L_QUANTITY > 0
- select count(*) from lineitem where L_LINENUMBER < 3 and L_LINENUMBER > 0
- select count(*) from lineitem where L_LINENUMBER < 5 and L_LINENUMBER > 2
- select count(*) from lineitem where L_DISCOUNT < .07 and L_DISCOUNT > .02
- select count(*) from lineitem where L_EXTENDEDPRICE < 100000.00 and L_-EXTENDEDPRICE > 20000.00
- select count(*) from lineitem where L_DISCOUNT < .04 and L_DISCOUNT > 0.0
- select count(*) from lineitem where L_QUANTITY < 20 and L_QUANTITY > 10
- select count(*) from lineitem where L_DISCOUNT < .05 and L_DISCOUNT > .02
- select count(*) from lineitem where L_EXTENDEDPRICE < 15000.00 and L_-EXTENDEDPRICE > 0.0
- select count(*) from lineitem where L_LINENUMBER < 2 and L_LINENUMBER > 0