Experimental Evaluation of Selectivity Estimation on Big Spatial Data

Harry Chasparis
University of California, Riverside
900 University Ave
Riverside, California 92521
zchas001@ucr.edu

Ahmed Eldawy
University of California, Riverside
900 University Ave
Riverside, California 92521
eldawy@cs.ucr.edu

ABSTRACT

With the tremendous volume of spatial datasets, there is an increasing need to process and analyze spatial data. One of the fundamental spatial queries is the selectivity estimation problem where users want to quickly estimate the total number of records in a given query range. While there have been several approaches to solve this problem for big data, there is no systematic evaluation and comparison for these techniques.

In this work, we experimentally examine three of the most widely used techniques for selectivity estimation, namely, sampling, uniform binning, and non-uniform binning. This evaluation will be a basis for deciding when to use each of these techniques based on the application requirements. Furthermore, we study the trade-off between memory usage, preprocessing overhead, online query time and the accuracy of the results. With extensive experiments on large datasets, we provide an evaluation of these techniques and we reveal their benefits and their weaknesses.

KEYWORDS

Selectivity Estimation, Spark, Big Spatial Data

ACM Reference format:
Harry Chasparis and Ahmed Eldawy. 2017. Experimental Evaluation of Selectivity Estimation on Big Spatial Data. In *Proceedings of GeoRich'17 , Chicago, IL, USA, May 14, 2017,* 6 pages.
DOI: http://dx.doi.org/10.1145/3080546.3080553

1 INTRODUCTION

In the recent years, there has been a remarkable increase in the amount of big data in general, and spatial data in particular. According to recent studies, we generate 2.5 quintillion bytes of data everyday [14]. A recent McKinsey report [12] shows that 60% of this data is location-based with an end-user value of $700 billion in the current decade. Big spatial data is used in many applications including brain simulation [22, 29], climate studies [11], and geo-targeted advertising [28].

One of the fundamental spatial analysis techniques is *selectivity estimation* which tries to estimate the total number of objects in a given spatial range. Selectivity estimation has many applications in the area of spatial data management including index construction [19], query load balancing [5], and query optimization [9]. For example, ScalaGiST [19] uses those estimates to partition and index the data into equi-sized blocks. On the other hand, AQWA [5] uses selectivity estimation of both the data records and queries to better balance the load across machines. Another example is SpatialHadoop [9] which uses it for query optimization where it is used to direct highly-selective queries to use a single-machine algorithm while a low-selective query uses MapReduce. Other systems use it for load balancing in analytic jobs [20, 31]. Selectivity estimation can be also used to analyze Twitter trends where the number of tweets with a specific keyword is compared across different query regions and times in order to measure how the keyword is trending [16, 21].

Two of the most widely used classes of techniques that address the selectivity estimation problem are *sampling* and *binning*. In *sampling*-based techniques, a random sample is drawn from the input data and used to estimate the total number of records in any selected area. The performance and accuracy of sampling-based techniques rely on many factors including the type of the sample, i.e., biased or unbiased, and the size of the sample. In *binning*-based techniques, a two-dimensional histogram is constructed and used to estimate the number of records. The performance of these techniques depends on several factors including the number and shape of the bins. While they have been applied in big data systems [5, 9, 10, 19, 20, 31], there is a lack of a thorough comparison between these two classes of techniques that assists researchers and practitioners in choosing the most appropriate technique for a given application, system setting and dataset.

In this paper, we provide a comprehensive experimental evaluation of several selectivity estimation techniques based on sampling and binning. The goal of this study is to assist researchers and developers in choosing the most suitable technique for each application. The experiments are split into two phases, an *offline* phase that summarizes the data and an *online* phase that answers selectivity estimation queries. The offline phase runs only once in parallel in the cluster to summarize the data into a prespecified memory budget (B) which is then stored in the main memory of a single machine. The online phase runs on a single machine and uses the precomputed summary to produce answers to several selectivity estimation queries. We study the tradeoff between the running time of the offline phase, the running time of the online phase, and the quality of the results, while varying several parameters including, the memory budget allotted for the summary, the input data size, the number of machines, and the selectivity ratio. The results of this

GeoRich'17 , Chicago, IL, USA
© 2017 ACM. 978-1-4503-5047-1/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3080546.3080553

study will allows users to choose the most appropriate technique according to the system setup and application preferences.

In our experiments, we found that the sampling method works better in terms of quality and running time when the allotted memory is limited and the selectivity of the query range is small. As the query range gets bigger and the available memory becomes larger, the binning methods become faster and more accurate. This preliminary experimental evaluation opens the direction for more thorough experiments by the big data community.

The remainder of the paper is organized as follows. Section 2 summarizes the related work. Section 3 includes the experimental setup, giving details about the datasets, the machine setup and query workload. The results of the experimental evaluation is included in Section 4 and in Section 5 we summarize our findings. Finally, Section 6 concludes with future directions and extensions.

2 RELATED WORK

Selectivity estimation is a fundamental problem that has been studied for decades. In this section, we categorize the existing research efforts into three categories, namely, *selectivity estimation for non-spatial data*, *selectivity estimation for spatial data* and *experimental studies of selectivity estimation.*

Selectivity estimation for non-spatial data: There has been a lot of research in the area selectivity estimation for traditional non-spatial data. These techniques use either single-dimensional histograms [15, 26] or samples [18] to summarize the data and use the summarized data to answer the selectivity estimation problem. These techniques were designed for one-dimensional data and cannot directly apply to multidimensional data including spatial data.

Selectivity estimation for spatial data: The research work in this category tries to expand the single-dimensional techniques described above for multidimensional data. [24, 30] propose smart sampling techniques that improve over the regular uniform sampling techniques to provide better quality. Belussi and Faloutsos [6] are among the first ones who attempted to research the principles of spatial data and propose techniques which were addressed specifically for spatial data. [1, 17, 27] propose several histogram-based techniques which are applied to different types of spatial datasets. Furthermore, Ching-Tien Ho *et al* [13] improve histogram-based techniques where each bin represents the total number of records in a region of the input domain. Unlike other techniques which store the total number of records in the area covered by the corresponding cell, they store the total number of records that are to the top of left of that cell. This simple addition allows that histogram to answer any rectangular selectivity estimation query in a constant time regardless of its size. This makes an opportunity to build a denser histogram without worrying about the running time of the selectivity estimation query. This binning technique was found very efficient and became popular in many applications that employ the selectivity estimation problem [4, 5, 8, 23]. In the proposed work, we do not propose a new technique for selectivity estimation, rather, we provide an experimental evaluation to compare the relative performance of the existing techniques in the distributed setting.

Experimental studies on selectivity estimation: Research efforts in this category aim to provide an experimental study for selectivity estimation techniques from different categories. Poosala *et al* [2] is the only work that we believe belongs in this category. This work surveys the existing techniques, at that time, for spatial selectivity estimation and experimentally evaluates them. They also proposed their own techniques which were based on using spatial indexes or on the notions of spatial density and spatial skew. This seems to be the only study that tries to compare different kind of techniques, because most of the comparison evaluation studies try to compare similar techniques and then to propose a new method which improves a little bit a previous one.

This paper belongs to the third category where we provide an experimental evaluation to compare several techniques. Any of the techniques listed in the second category can be added to the experimental evaluation. In this work, we focus on three widely used techniques, namely, uniform sampling, uniform binning, and non-uniform binning. We focus on the distributed setting where a preprocessing phase runs in parallel to compute the sample or the histogram, while a single-machine online phase answers selectivity estimation queries based on the result of the preprocessing phase. To the best of our knowledge, this is the first experimental study that evaluates the tradeoff in selectivity estimation in big spatial data.

3 EXPERIMENTAL SETUP

In this section, we describe the foundation of our experimental evaluation. First, we provide a formalization of the selectivity estimation problem in Section 3.1. In Sections 3.2-3.4 we provide the details of the three methods we study in our experiments, namely, sampling, uniform binning, and non-uniform binning. Section 3.5 provide the several parameters that we change and the metrics that we measure in our experiments. Section 3.6 describes the query workload. Finally, Section 3.7 details the cluster setup in terms of hardware and software.

3.1 Problem Definition: Selectivity Estimation

The selectivity estimation problem is to calculate the number of points in a given query range. Given a dataset of points, R, and a query rectangle Q, the output is the number of points in the set R that lie inside the query rectangle Q. While this query can be generalized to arbitrary shapes, this paper focuses on the special case where the input dataset is points. Arbitrarily shaped records are usually approximated as points given that they are small enough. In this query, while approximate answers are acceptable a more accurate result is desired. This problem has been of a huge interest in the past and we believe that it will continue to do so. Despite the huge advance in systems hardware, e.g., CPU and memory, the amount of data is growing faster than Moore's law and a fast and approximate answer will always be useful.

This query is not to be mixed with the range query where the desired output is the set of all records that lie in the query range and where an approximate answer is usually not acceptable. In many cases, the selectivity estimation query runs as a preparation step for the actual range query to estimate the answer size and prepare the system to receive the answer, e.g., allocate the desired memory and disk buffers, or choose an appropriate level of parallelization.

3.2 Sampling Method

The first technique that we consider is sampling. Sampling method is a widely-used technique in many algorithms and applications. Its popularity comes from the simplicity of the implementation that makes it easy to use and applicable to several problems including visualization [25], partitioning [10], and approximate queries [3]. In the case of the selectivity estimation query, a range query is executed on a small sample of the data and the answer is divided by the sampling ratio to obtain an answer for the selectivity estimation problem. We implement this algorithm in Spark in three steps, *sample*, *index*, and *estimate*, as detailed below.

In the *sample* step, we read a sample from the input file in parallel using the Spark built-in functionality. Spark provides two sampling functions, namely `sample` and `takeSample`. The `sample` method takes as input a sampling ratio in the range $[0, 1]$ and decides for each record to be in the sample independently with the given probability. On the other hand, `takeSample` takes as input a fixed number and returns a sample with this size. While we actually need to pick a sample with a fixed size, we chose to use the first function, i.e., `sample`, for two reasons. First, the `sample` function is much faster as it operates completely in parallel and requires no coordination between the nodes while `takeSample` requires all the sub-samples acquired by different machines, to be collected in one central machine to produce the final answer with the given size. Second, `sample` function defines a Spark *transformation* which allows us to combine it with other transformations for parsing and processing the file. `takeSample`, on the other hand, is an *action* and it returns the sampled data to the driver machine which should continue any further required processing.

In the *index* step, we insert the sample data into an in-memory K-d tree index [7] to speed up the in-memory processing of the third step. A k-d tree can be built efficiently in $O(n \log n)$ time if all the points are known in advance, where n is the sample size.

The third *estimate* step runs only when a user submits a query. In this case, the K-d tree index is used to retrieve and count the points in the sample that matches the user query. The returned number is divided by the sampling ratio to produce an estimate for the query results on the entire dataset. For example, if the sampling ratio is 1%, the result of the range query on the sample is multiplied by 100 to produce an estimate for the selectivity estimation query. A range search on the K-d tree tree can be executed in $O(\sqrt{n} + k)$, where k is the number of points in the query region.

3.3 Uniform Binning Method

The sampling method has two major drawbacks that the binning method tries to address. First, it relies on a random number generator which causes it to produce a different answer each time it runs. This can be undesirable in some applications. Second, it has to drop some records during the sampling method which could have been used to improve the answer.

In the uniform binning method, the entire data is scanned and a two-dimensional histogram is generated to store the number of records in each bin. The uniform histogram is similar to an equi-width histogram but for a two- or multi-dimensional space. This method runs in three steps, *histogram*, *prefix-sum*, and *estimate*, as detailed below.

In the histogram step, each machine prepares a two-dimensional histogram that has n-bins where n is computed according to the memory budget, $n = \lfloor B/|b| \rfloor$ where $|b|$ is the size of one bin, typically, 8-bytes to hold a counter. Then, all machines work in parallel by scanning all the records stored on that machine, each record is mapped to a bin in the two-dimensional grid, and the value of the corresponding bin is incremented by one. This is done in parallel on all machines and the results are finally combined together in one machine which computes the final histogram by adding up the corresponding bins from all machines.

The prefix-sum step aims to speed up the selectivity estimation query, by employing the method proposed by Ching-Tien Ho *et al* [13]. This method computes a prefix sum along the two dimensions of the histogram. The result is that the counter in each bin will represent the total number of points in all the bins that are to the top or left of that bin. This allows a selectivity estimation query to run in a constant time regardless of its size as detailed in [5, 13].

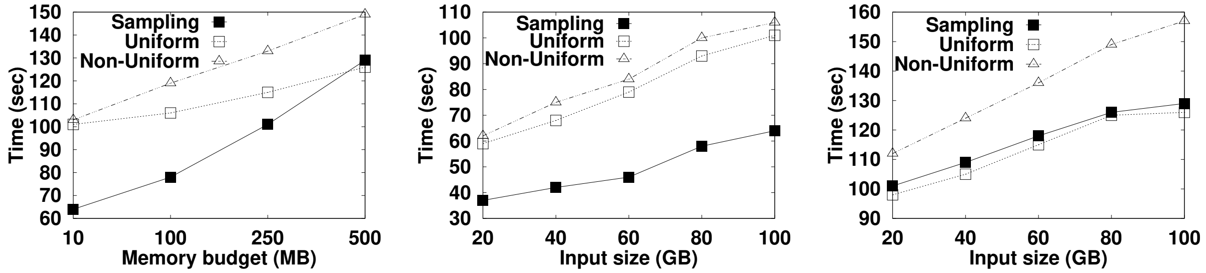
The final *estimate* step runs only when a user submits a selectivity estimation query. First, it maps the four corners of the query rectangle to the histogram to find the corresponding bins. Then, the estimated selectivity is computed by adding up the values in the top-left and bottom-right corners, and subtracting the values in top-right and bottom-left. This simple computation makes use of the prefix-sum to compute the answer in a constant time. We further refine this step by taking a fraction of the value in a cell that partially overlaps the query rectangle proportion to the area of the overlap as compared to the total area of the bin. This is based on the assumption that the data is each grid cell is uniformly distributed.

3.4 Non-uniform Binning Method

If the data is uniformly distributed, the uniform binning method works fine. However, if the data is highly skewed, it is better to use the non-uniform binning method described below.

This technique is very similar to the uniform binning method except that it uses different widths and heights for columns and rows, respectively, in order to equalize the number of objects in each bin. This method is similar to the equi-depth histograms in single dimension. In two-dimensional space, equi-depth means that each column and each row has roughly an equal number of objects, but not necessarily each bin. Actually, the most common case is for each bin to have different number of objects.

To compute the width and height of each column and row, we first read in parallel a sample of the input file on the size of our memory budget (B), as done in the sampling method, and then in a single machine we use this sample to divide the space into columns and rows of equal depth, i.e., equal number of points. Once the widths and heights of columns and rows are computed, the sample is discarded and its memory is reclaimed to be used for the histogram. After that, we use these widths and heights to construct the grid and scan the file again to populate the values of the histogram as done in the uniform binning method. Once the two-dimensional histogram is computed, the estimation step runs



(a) Input size=100 GB **(b) Memory Budget (B)=10 MB** **(c) Memory Budget (B)=500 MB**

Figure 1: Running time of the offline phase

exactly as before. Since the widths and heights are different, to find the cell that corresponds to one corner, we need two binary search operations as one for the row and one for the column. Once the two corners are determined, the logic of the estimation query is exactly as in the uniform binning.

3.5 Experimental Parameters

In our experiments, we study the trade-off between the running time of both phases, offline and online, and the quality of the results. In doing so, we conduct our experiments while varying four parameters, (1) the input data size, (2) the memory budget (B) that is used to hold either the sample of the histogram, (3) the number of machines in the cluster, and (4) the query rectangle selectivity (S) measures as the ratio between the query rectangle area and the input domain area.

For the memory budget (B), we use the values of 10 MB, 100 MB, 250 MB and 500 MB. Memory budgets smaller than 10 MB are unreasonable given the specifications of available commodity machines. Also, as shown in the next section, we found that the trends of the results are stable at the values of 500 MB or higher, thus we decided to report the numbers of up to 500 MB only.

The ratio of the query area to the input area is called the Selectivity Ratio (S). For example, a selectivity ratio equal with 10% means that the query rectangle covers 10% of the total area. In our experiments, we vary the selectivity ratio between the values $\{0.0001\%, 0.001\%, 0.01\%, 0.1\%, 1\%\}$ of total area. Larger values of selectivity ratio are too large to be used in practice by users. Notice that this value does not have to be equal to the actual query selectivity which represents the ratio between the number of selected records and the total number of input records.

3.6 Query Workload

For the input dataset, we use the 100 GB `all_nodes` dataset publicly available at the website of SpatialHadoop [spatialhadoop.cs.umn.edu/datasets.html] which contains around 2.7 billion points covering our whole planet. To test the effect of the input size, we vary the size by taking samples of sizes 20, 40, 60, and 80 GB.

Selectivity estimation queries are determined by picking 20 random sample points from the input dataset. These points act as the center of the query ranges. Then, the size of each query range is determined according to a parameter S which represents the selectivity ratio. For example, if $S = 1\%$, the area of the query rectangle is 1% of the total input area.

3.7 Cluster Setup

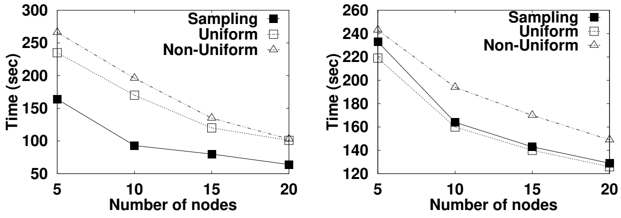
We run the offline phase on a Spark cluster on Amazon Web Services (AWS) with up-to 20 nodes of type '`m3.xlarge`' with four cores, 15 GB of RAM, and 80 GB of SSD storage. We used Spark 2.10, Hadoop 2.7.2, and Java 1.8. The online phase runs on a single machine with Intel Skylake at 2.5GHz with 16 GB of DRAM.

4 EXPERIMENTAL RESULTS

In this section, we present the results of the conducted experimental evaluation. We begin with the analysis of the performance of the offline phase while varying the number of machines, the input size, and the memory budget. We then report the results of the online phase and highlight the tradeoff between the performance and the quality of the selectivity estimation problem.

4.1 Offline Phase

For the offline phase, we measure the overall running time of this step. This helps researchers determine which method is faster and how the input size and the number of machines affect their performance. The general view of the offline phase for each method is presented in Figure 1. In Figure 1a, the 100 GB input dataset is used while the memory budget B is changed from 10 MB to 500 MB. We observe that for small memory budgets the sampling method is the fastest. As we increase the value of B, sampling running time increases faster than the other methods. This is something that we expect, since we have used a K-d tree index to store the sample in memory. The K-d tree index is a necessary step to make the online phase works in an real-time manner. Otherwise, each selectivity estimation query will have to scan the entire sample which is unpractical and would be too slow, especially, if the system is expected to answer multiple queries simultaneously. On the other hand, we notice that uniform binning method scales pretty well and it becomes the faster method at B=500 MB and above. This outcome is justifiable since each record updates the histogram in a constant time, and the prefix sum phase runs in linear time. Lastly, the running time of the non-uniform binning is slower than the others because in the histogram construction phase, each records requires



(a) Mem Budget (B)=10 MB **(b) Mem Budget (B)=500 MB**

Figure 2: Running time of the offline phase as increasing the cluster size

$\log(B)$ work to find the corresponding work since the widths and heights of the columns and rows are variable.

In Figures 1b and 1c, we measure the running time as the input size increases from 20 GB to 100 GB while fixing the memory budget at 10 MB and 500 MB, respectively. As expected, all the algorithms scale nicely with the increasing input size. Interestingly, for a small memory budget (B=10 MB) the sampling method is the fastest while for a larger memory budget (B=500 MB) the uniform binning becomes faster. This suggests the use of sampling for small memory budgets and histograms for larger memory budgets if the running time of the offline phase is of interest.

Finally, since we have conducted our offline phase into a cluster, we could not have omitted to examine if the cluster size in terms of node number effects the methods. It seems that the cluster size, just like the input size, effects the time performance of the offline phase, however it does not change the trend of the methods. Figure 2 provides a view of our experiments as we used different cluster size and different memory budgets.

4.2 Online Phase

For the online phase, we use two measures, the running time of a single selectivity estimation query, and the accuracy of the result as compared to the ground truth. As the online selectivity estimation query runs on a single machine, we only vary the memory budget B and the selection ratio S.

Time performance: Figure 3 illustrates the running time of a single selectivity estimation query as we change the memory budget from 10 MB to 500 MB and the selectivity ratio from 0.0001% to 1%. We notice that the queries of the two binning methods have equal performance in all the cases and this was expected, because the way that a binning query is answered is similar for both cases. Due to the prefix sum that we employ, a selectivity estimation query is answered in constant running time the case of uniform binning, and almost constant running for non-uniform binning. For the case of non-uniform binning, an extra binary search has to be made to find the range of columns that overlap the query range. On the other hand, the sample-based technique has to search the in-memory K-d tree which takes $\log(n+k)$ where k is the result size. Furthermore, as we increase the memory budget, the histogram-based techniques deliver almost the same performance while sampling takes more time with larger values of B as the size of the K-d tree increases.

Looking at the actual numbers, one case easily see that the sampling method is not as scalable as the histogram method. For very small values of S and B, the sampling method is slightly faster, however, as the values of S and B increase, the gap becomes huge. Notice that both of them provide the result in a few milliseconds which makes them both sufficiently responsive for one query. However, if the system expects thousand of queries per second, the performance gap will be huge.

Accuracy: In many applications, the accuracy is the most desirable characteristic of the selectivity estimation query. For each query in the workload, the ground-truth is obtained by running the corresponding range query on the actual data to obtain the correct answer, say Z^*. We run the selectivity estimation query using either the sampling or histogram methods, and we obtain an answer Z. The error is measured as $e = \frac{|Z^*-Z|}{Z^*}$ and the accuracy is $a = 1 - e$.

Figure 4 shows the average accuracy of the selectivity estimation query as we vary both B and S. For each point, we run 20 different queries, all with the same selectivity S, and report the average accuracy. We notice that the sampling method gives a better accuracy for small values of S and B. This suggests the use of the sampling method for selectivity estimation when the memory is limited and the queries are expected to be highly selective. As B and S increase, both binning-based techniques eventually outperform sampling. Furthermore, non-uniform binning is consistently more accurate, even with small difference for large S, than uniform binning which is expected as it has a higher degree of freedom by adjusting the width and height of each column. One last point that we can notice is that, if users desire an accuracy that is almost equal to 100%, then the binning techniques are these that can give such numbers, since sampling seems to have an upper bound on accuracy. And this is because there is always the case of a bad sample, which means that the general average accuracy for the sampling method cannot be considered close to 100%.

5 DISCUSSION

Our experimental evaluation has as target to provide useful guidelines to researchers and developers on how to choose the most suitable technique for each application. Our experimental results can be summarized in four observations. Foremost is that there is no clear winner; no method provides us constantly with the best running time for both phases, as well as the best accuracy at the same time.

In the case where the memory budget is equal to or greater than 500MB, Uniform binning is the best choice for two reasons. First, because its offline phase turns out to be the fastest, and secondly because during the online phase we can have very fast query responses with generally high accuracy, approaching 100% for large S.

We also observed that varying parameters such as cluster size or input size, effects in the same degree the offline phase of any method. This is important because it ensures that we do not need to change our technique for better performance in the event of changing any of the parameters.

Finally, if we are only seeking to have the best accuracy of our results, we must pick any of the two binning methods. This is due to the fact that, largely, binning methods provide us with higher result quality, compared to the sampling method. And of course, as

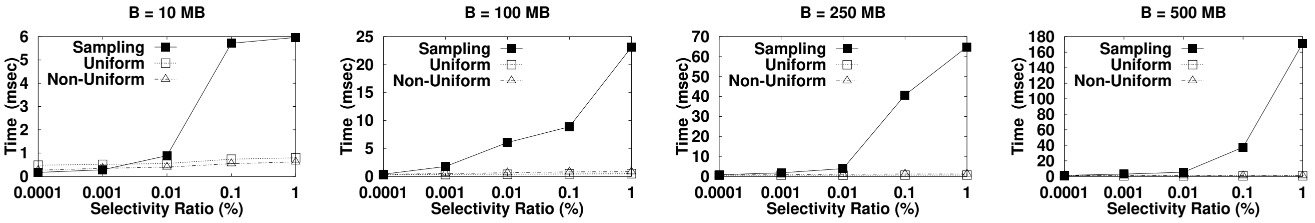


Figure 3: Query running time for different memory budgets

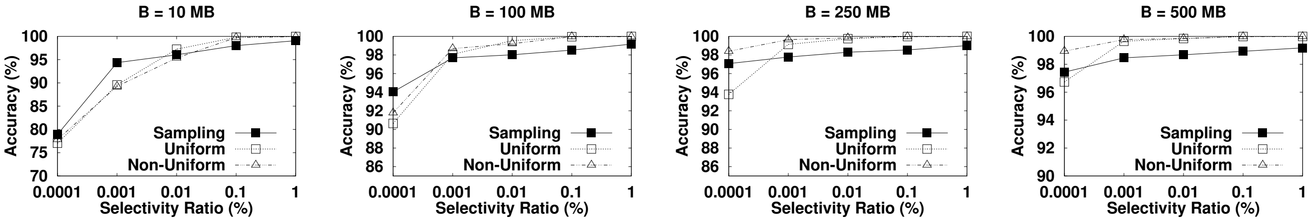


Figure 4: Query accuracy for different memory budgets

we have already discussed, in the case of a bad sample, we suffer a big loss in accuracy.

6 CONCLUSION

We presented an experimental evaluation of three techniques for the selectivity estimation problem on spatial datasets. These techniques represent the two most widely used classes of techniques addressing the selectivity estimation problem, which are sampling and binning. By thorough experiments, we revealed their benefits and weaknesses in terms of time performance and query accuracy, and pointed out when each technique should be used. Our goal for future work is to compare these techniques with datasets which have various formats of spatial data, in order to examine if the trend of the results remains the same.

REFERENCES

[1] Ashraf Aboulnaga and Jeffrey F. Naughton. 2000. Accurate Estimation of the Cost of Spatial Selections. In *ICDE*. San Diego, CA, 123–134.

[2] Swarup Acharya, Viswanath Poosala, and Sridhar Ramaswamy. 1999. Selectivity Estimation in Spatial Databases. In *SIGMOD*. Philadelphia, PA, 13–24.

[3] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *EuroSys*.

[4] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. 1998. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *SIGMOD*. Seattle, WA, 94–105.

[5] Ahmed M. Aly, Ahmed R. Mahmood, Mohamed S. Hassan, Walid G. Aref, Mourad Ouzzani, Hazem Elmeleegy, and Thamir Qadah. 2015. AQWA: Adaptive Query-Workload-Aware Partitioning of Big Spatial Data. *PVLDB* 8, 13 (2015), 2062–2073.

[6] Alberto Belussi and Christos Faloutsos. 1995. Estimating the Selectivity of Spatial Queries Using the Correlation' Fractal Dimension. In *LDB*. 299–310.

[7] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.

[8] Yong-Jin Choi and Chin-Wan Chung. 2002. Selectivity Estimation for Spatio-temporal Queries to Moving Objects. In *SIGMOD*. Madison, WI, 440–451.

[9] Ahmed Eldawy, Louai Alarabi, and Mohamed F. Mokbel. 2015. Spatial Partitioning Techniques in SpatialHadoop. In *PVLDB*. Kohala Coast, HI, 1602–1605.

[10] Ahmed Eldawy and Mohamed F. Mokbel. 2015. SpatialHadoop: A MapReduce Framework for Spatial Data. In *ICDE*. Seoul, South Korea, 1352–1363.

[11] James Faghmous and Vipin Kumar. 2013. *Spatio-Temporal Data Mining for Climate Data: Advances, Challenges, and Opportunities*. Advances in Data Mining, Springer.

[12] Nicolaus Henke, Jacques Bughin, Michael Chui, James Manyika, Tamim Saleh, Bill Wiseman, and Guru Sethupathy. 2016. *The Age of Analytics: Competing in a Data-driven World*. Technical Report. McKinsey Global Institute.

[13] Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, and Ramakrishnan Srikant. 1997. Range Queries in OLAP Data Cubes. In *SIGMOD*. Tucson, AZ, 73–88.

[14] IBM 2017. Bringing Big Data to the Enterprise. (2017).

[15] Yannis E. Ioannidis. 1993. Universality of Serial Histograms. In *LDB*. Dublin, Ireland, 256–267.

[16] Jianfeng Jia, Chen Li, Xi Zhang, Chen Li, Michael Carey, and Simon Su. 2016. Towards Interactive Analytics and Visualization on One Billion Tweets. In *ACM SIGSPATIAL*. Burlingame, California, 85:1–85:4.

[17] Ji Jin, Ning An, and Anand Sivasubramaniam. 2000. Analyzing Range Queries on Spatial Data. In *ICDE*. San Diego, CA, 525–534.

[18] Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. 1990. Practical Selectivity Estimation through Adaptive Sampling. In *SIGMOD*. Atlantic City, NJ, 1–11.

[19] Peng Lu, Gang Chen, Beng Chin Ooi, Hoang Tam Vo, and Sai Wu. 2014. ScalaGiST: Scalable Generalized Search Trees for MapReduce Systems. *PVLDB* 7, 14 (2014), 1797–1808.

[20] Wei Lu, Yanyan Shen, Su Chen, and Beng Chin Ooi. 2012. Efficient Processing of k Nearest Neighbor Joins using MapReduce. *PVLDB* 5, 10 (2012), 1016–1027.

[21] Amr Magdy and others. 2014. Taghreed: A System for Querying, Analyzing, and Visualizing Geotagged Microblogs. In *ACM SIGSPATIAL*. 163–172.

[22] Henry Markram. 2006. The Blue Brain Project. *Nature Reviews Neuroscience* 7, 2 (2006), 153–160.

[23] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. 1998. Wavelet-Based Histograms for Selectivity Estimation. In *SIGMOD*. Seattle, WA, 448–459.

[24] Frank Olken and Doron Rotem. 1993. Sampling from Spatial Databases. In *ICDE*. Vienna, Ausria, 199–208.

[25] Yongjoo Park, Michael J. Cafarella, and Barzan Mozafari. 2016. Visualization-aware Sampling for Very Large Databases. In *ICDE*. Helsinki, Finland, 755–766.

[26] Viswanath Poosala and others. 1996. Improved Histograms for Selectivity Estimation of Range Predicates. In *SIGMOD Record*. 294–305.

[27] Viswanath Poosala and Yannis E. Ioannidis. 1997. Selectivity Estimation Without the Attribute Value Independence Assumption. In *LDB*. Athens, Greece, 486–495.

[28] Jagan Sankaranarayanan and others. 2009. TwitterStand: News in Tweets. In *ACM SIGSPATIAL*. 42–51.

[29] Farhan Tauheed, Laurynas Biveinis, Thomas Heinis, Felix Schürmann, Henry Markram, and Anastasia Ailamaki. 2012. Accelerating Range Queries for Brain Simulations. In *ICDE*. Washington D.C., 941–952.

[30] Michael Vassilakopoulos and Yannis Manolopoulos. 1997. On Sampling Regional Data. *TKDE* 22, 3 (1997), 309–318.

[31] Randall T. Whitman, Michael B. Park, Sarah A. Ambrose, and Erik G. Hoel. 2014. Spatial Indexing and Analytics on Hadoop. In *ACM SIGSPATIAL*. 73–82.