www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issues 6 June 2016, Page No. 16860-16866

Integration Of Big Data And Cloud Computing To Detect Black Money Check Rotation With Range Aggregate Queries

K.Nithiya[1],S.Balaji[2]

[1]M.E. Student, Department of CSE, Parisutham Institute of Technology and Science, Tamil Nadu, India
[2]Asst. Professor, Department of CSE, Parisutham Institute of Technology and Science, Tamil Nadu, India
[1]nithyak045@gmail.com,[2]balagicse@gmail.com

Abstract— **A Cloud is expanding from application aggregation and sharing to data aggregation and utilization. To make use of data tens of terabytes and tens of beta bytes of data to be handled. These massive amounts of data are called as a big data. Range-aggregate queries are to apply a certain aggregate function on all tuples within given query ranges. Fast RAQ first divides big data into independent partitions with a balanced partitioning algorithm, and then generates a local estimation sketch for each partition. When a range-aggregate query request arrives, Fast RAQ obtains the result directly by summarizing local estimates from all partitions & Collective Results are provided. Data Mining can process only Structured Data only. Big Data Approach is spoken all over the Paper. They insist of Three Tier Architecture, 1. Big Data implementation in Multi System Approach, 2. Application Deployment - Banking / Insurance. 3. Extraction of Useful information from Unstructured Data. We implement this Project for Banking Domain. There will be Two Major Departments. 1. Bank Server for Adding New Clients and maintaining their Accounts. Every User while Registration has to provide their aadhar card as a ID Proof to create Account in any Bank. 2. Accounts Monitoring Sever will monitor every users and their Account Status in different Banks. This Server will retrieve users who maintain & Transact more than Rs. 50,000 / Annum in all 3 Accounts in different Banks using the same ID Proof. Map & Reduce is achieved.**

Keywords:Balanced partition, big data, multidimensional histogram, range-aggregate query.

I.INTRODUCTION

The Internet and in various news media, can we summarize all types of opinions in different media in a real-time fashion, including updated, cross-referenced discussions by critics? This type of summarization program is an excellent example for Big Data processing, as the information comes from multiple, heterogeneous, autonomous sources and the
evolving relationships, and keeps growing. Along with the above example, the era of Big Data has arrived Every day, 2.5 quintillion bytes of data are created and 90 percent of the data in the world today were produced within the past two years Our capability for data generation has never been so powerful and enormous ever since the invention of the information technology in the early 19th century. As another example, on 4 October 2012, the first presidential debate between President Barack Obama and Governor Mitt Romney triggered more than 10 million tweets within 2 hours Among all these tweets, the specific moments that generated the most discussions actually revealed the public interests, such as the discussions about medicare and vouchers. Such online discussions provide a new means to sense the public interests and generate feedback in real time, and are mostly appealing compared to generic media, such as radio or TV broadcasting. Another example is Flickr, a public picture sharing site, which received 1.8 million photos per day, on average, from February to March 2012 assuming the size of each photo is 2 megabytes (MB), this requires 3.6 terabytes (TB) storage every single day. Indeed, as an old saying states: ―a picture is worth a thousand words,‖ the billions of pictures on Flicker are a treasure tank for us to explore the human society, social events, public affairs, disasters, and so on, only if we have the power to harness the enormous amount of data. The above examples demonstrate the rise of Big Data applications where data collection has grown tremendously and is beyond the ability of commonly used software tools to capture, manage, and process within a ―tolerable elapsed time.‖ The most fundamental challenge for Big Data applications is to explore the large volumes of

data and extract useful information or knowledge for future actions. In many situations, the knowledge extraction process has to be very efficient and close to real time because storing all observed data is nearly infeasible. For example, the square kilometer array (SKA) in radio astronomy consists of 1,000 to 1,500 15-meter dishes in a central 5-km area. It provides 100 times more sensitive vision than any existing radio telescopes, answering fundamental questions about the Universe. However, with a 40 gigabytes (GB)/second data volume, the data generated from the SKA are exceptionally large. Although researchers have confirmed that interesting patterns, such as transient radio anomalies can be discovered from the SKA data, existing methods can only work in an offline fashion and are incapable of handling this Big Data scenario in real time. As a result, the unprecedented data volumes require an effective data analysis and prediction platform to achieve fast response and real-time classification for such Big Data.

1.1. CHARACTERISTICS OF BIG DATA

Volume – The quantity of data that is generated is very important in this context. It is the size of the data which determines the value and potential of the data under consideration and whether it can actually be considered as Big

Data or not. The name _Big Data_ itself contains a term which is related to size and hence the characteristic.

Variety - The next aspect of Big Data is its variety. This means that the category to which Big Data belongs to is also a very essential fact that needs to be known by the data analysts. This helps the people, who are closely analysing the data and are associated with it, to effectively use the data to their advantage and thus upholding the importance of the Big Data.

Velocity - The term _velocity_ in the context refers to the speed

of generation of data or how fast the data is generated and processed to meet the demands and the challenges which lie ahead in the path of growth and development.

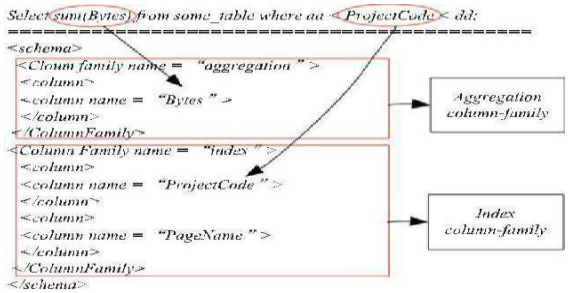
Variability - This is a factor which can be a problem for those who analyse the data. This refers to the inconsistency which can be shown by the data at times, thus hampering the process of being able to handle and manage the data effectively.

Veracity - The quality of the data being captured can vary greatly. Accuracy of analysis depends on the veracity of the source data.

Complexity - Data management can become a very complex process, especially when large volumes of data come from multiple sources. These data need to be linked, connected and correlated in order to be able to grasp the information that is supposed to be conveyed by these data. This situation is therefore, termed as the _complexity_ of Big Data.

1.2 OUR CONTRIBUTIONS

In this paper, we propose FastRAQ—a new approximate answering approach that acquires accurate estimations quickly for range-aggregate queries in big data environments.



FastRAQ first divides big data into independent partitions with a balanced partitioning algorithm, and then generates a local estimation sketch for each partition. When a range-aggregate query request arrives, FastRAQ obtains the result directly by summarizing local estimates from all partitions.

The balanced partitioning algorithm works with a strati-fied sampling model. It divides all data into different groups with regard to their attribute values of interest, and further separates each group into multiple partitions according to the current data distributions and the number of available servers. The algorithm can bound the sample errors in each partition, and can balance the number of records adaptively among servers when the data distribution and/or the num-ber of servers changes.

II OVERVIEW OF THE FAST RAQ APPROACH

PROBLEM STATEMENT

We consider the range-aggregate problem in big data environments, where data sets are stored in distributed servers. An aggregate function operates on selected ranges, which are contiguous on multiple domains of the attribute values.InFastRAQ; the attribute values can be numeric or alphabetic. One example of the range-aggregate problem is shown as follows:

Select exp(AggColumn), other ColName where

li<ColNamei<li opr

lj<ColNamej<ljopr

In the above query, exp is an aggregate function such as SUM or COUNT; Agg Column is the dimension of the aggregate operation; li <ColNamei<li and lj<ColNamej<lj are the dimensions of ranges queries; opr is a logical operator including AND and OR logical operations. In the following discussion, AggColumn is called Aggregation-Column,ColNamei and ColNamej are called Index-Columns.The cost of distributed range-aggregate queries primarily includes two parts. i.e., the cost of network communication and the cost of local files scanning. The first cost is produced by data transmission and synchronization for aggregate operations when the selected files are stored in different servers.

6.Counti hðCEmergedÞ; ==h is a function of cardinality estimation.

The second cost is produced by scanning local files to search the selected tuples.When the size of a data set increases continuously, the two types of cost will also increase dramatically. Only when the two types of cost are minimized, can we obtain faster final range-aggregate queries results in big data environments. Range-aggregate query statement is In FastRAQ, we divide numerical value space of an aggregation-column into different groups, and maintain an estimation sketch in each group to limit relative estimated errors of range-aggregate paradigm.

When a new record is coming, it is first sent onto a partition in the light of current data distributions and the number of available servers. In each partition, the sample and the histogram are updated respectively by the attribute values of the incoming record. When a query request arrives, it is delivered into each partition.We first build cardinality estimator (CE) for the queried range from the histogram in each partition. Then we calculate the estimate value in each partition, which is the product of the sample and the estimated cardinality from the estimator.

Algorithm 1. FastRAQuering(Q)

1.Input: Q; Q: select sum(AggColumn)

2.otherColname where li1<ColNamei<ColNamej< ColNamei < li2 from the local histogram

3. let CEi be the estimator of the ith dimensions;

4: Compute the cardinality estimator of range lj1 < ColNamej < lj2 from the local histogram, and let CEj be the estimator of the jth dimensions;

5: Merge the estimators CEi and CEj by the logical operator Opr, and compute the merged cardinality estimator CEmerged;

$counter_{PID}$ þ 1;

is the number of record;

sum_{PID} þ N;

//N is value of aggregation attribute from R;

$Sample_{PID}$ $sum_{k;l;m;r}$=$counter_{PID}$; 6: RID HashðPID; $counter_{PID}$Þ;

//RID is the unique record identifier for R;

7: Send R to partition PID;

8: return PID.

III SYSTEM ANALYSIS

PARTITIONING TECHNIQUES

Partitioning is a process of assigning each record in a largetable to a smaller table based on the value of a particular field in a record. It has been used in data center networks to improve manageability and availability of big data .The partitioning step has become a key determinant in dataanalysis to boost the query processing performance .All of these works enable each partition to be processed independently and more efficiently. Stratified sampling is a method of sampling from independent groups of a population, and selecting sample in each group to improve the representativeness of the sample by reducing sampling error. We build our partitioning algorithm based on the idea of stratified sampling to make the maximum relative error under a threshold in each partition. At the same time, the sum of the local result from each partition can also achieve satisfied accuracy for any ad-hoc range-aggregate queries. We first divide the value of numerical space into different groups and subdivide each group into different partitions according to the number of available servers.

The partition algorithm can be expressed as follows for data sets R:Partitioning (R)=(g; p)=(Ve; random ;[Vr]) where the number of a partition p in a group g, is a random number in ;Vr , and Ve is a group identifier (GID) for the group g. The stratified sampling is a method to subdivide the numerical value space into independent intervals with a batch of alogarithm functions, and each interval stands for a group. When the number of logarithm functions is fixed, an arbitrary natural integer N can be mapped into a unique group g.

PROPOSED SYSTEM

Fast RAQ first divides big data into independent

partitions with a balanced partitioning algorithm, and then generates a local estimation sketch for each partition. When a range-aggregate query request arrives, Fast RAQ obtains the result directly by summarizing local estimates from all partitions & Collective Results are provided. We deploy Big data for Banking Domain in this Project. User_s banking data is partitioned into multiple Tuples and stored in different sets of Database. We have designed an Application to track multiple accounts maintained in different banks of the same user and their Transaction details. This process helps in finding out Black money Holders so that Government can track them

IV.EXPERIMENTAL RESULTS

The range-aggregate query problem has been studied by Sharathkumar and Gupta [20] and Malensek [21] in computational geometry and geographic information systems (GIS). Our work is primary focused on the approximated rangeaggregate query for real-time data analysis in OLAP. Ho et al. was the first to present Prefix-Sum Cube approach to solving the numeric data cube aggregation [4] problems in OLAP. The essential idea of PC is to pre-compute prefix sums of cells in the data cube, which then can be used to answer range-aggregate queries at run-time. However, the updates to the prefix sums are proportional to the size of the data cube. Liang et al. [6] proposed a dynamic data cube for range-aggregate queries to improve the update cost, and it still costs OðNd 3Þ time for each update, where d is the number of dimensions of the data cube and n is the number of distinct tuples at each dimension. The prefix sum approaches are suitable for the data which is static or rarely updated. For big data environments, new data sets arrive continuously, and the up-to-date information is what the analysts need. The PC and other heuristic pre-computing approaches are not applicable in such applications. An important approximate answering approach called Online Aggregation was proposed to speed range-aggregate queries on larger data sets [7].

OLA has been widely studied in relational databases [8] and the current cloud and streaming systems [9], [10]. Some studies about OLA have also been conducted on Hadoop and MapReduce [10], [11], [12]. The OLA is a class of methods to provide early returns with estimated confidence intervals continuously.

Evaluation Methodology

The framework of FastRAQ includes four types of servers: learning server, load server, query server, and storage serv-ers. The learning server fetches a certain amount of data set

\

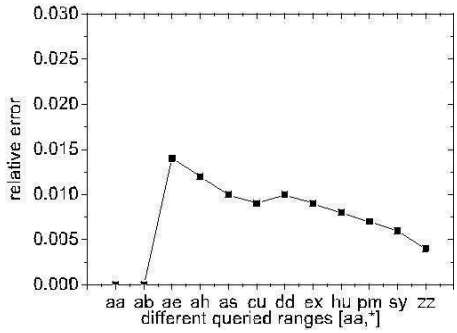


Fig. 1. The relative errors in different queried ranges.

Fig.2. Performance comparisons for count queries with eight days log files.

to learn data distributions, builds histogram and partition vectors for all partitions, and then dispatches them to other servers. The load servers receive online data sets, and deliver them to specified storage servers. The query server receives user_s query request, and sends it to all storage servers. The storage servers keep RC-Tree for each partition, and respond the request independently.

In the experiments, we analyze the pagecount traffic sta-tistics files of Wikipedia [19]. We construct a table contain-ing four columns. We set projectcode and pagename columns as index columns, bytes field as aggregation-col-umn. The FastRAQ stores four months of the traffic files which includes 960 GB of uncompressed data.

We first analyze the relative error in different queried examples. We use the traffic log files from Wikipedia in eight days. We set random variables in the queried examples and calculate the relative errors of different examples. The query example is ―select sumðbytesÞ from pagecounts where projectcode 2 ð aa ; Þ ‖, where _*_ is a random variable string changed from _aa_ to _zz_. The relative errors in different que-ried examples are shown in Fig. 5. We just present the values of _*_ on the X axis. When the _*_ equals to _aa_ and _ab_, the rel-ative errors are equal to zero. The results are calculated by scanning the log files of the two edge-buckets. When the _*_ grows larger, the relative error increases slightly. The rela-tive errors are nearly constant when the _*_ equals to _cu_, _dd_ and _ex_. In our experiment, we use ð aa ; dd Þ as our queried examples in following evaluations.

The examples of range-aggregate queries include count and sum queries, and aggregate functions on union queries. The queried examples are shown below:

Count query: Select countð Þ from pagecounts where projectcode 2 ð aa ; dd Þ;

Sum query: Select sumðbytesÞ from pagecounts where projectcode 2 ð aa ; dd Þ.

DOI: 10.18535/ijecs/v5i6.23

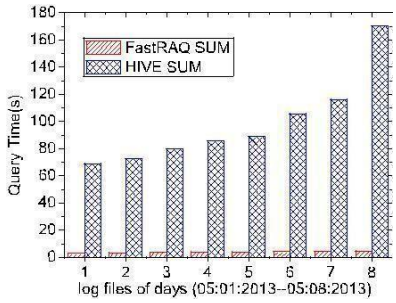


Fig. 3. Performance comparisons for sum queries with eight days log files.

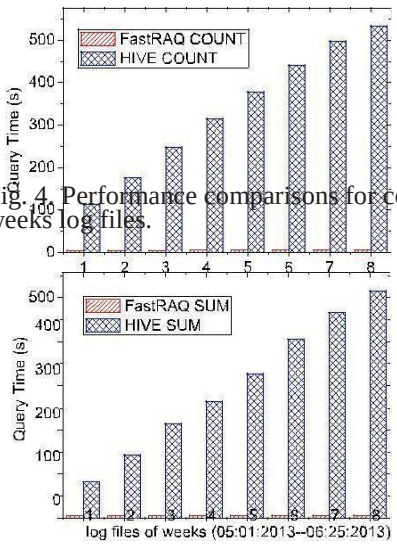


Fig. 4. Performance comparisons for count queries with eight weeks log files.

PERFORMANCE OF UNION OF SET QUERY

Due to the fact that it needs to scan and merge massive duplicated tuples in union of set queries, we primarily focus our testings in union of set range-aggregate queries. The performance comparisons of union query in the two sys-tems are presented.

Hive predicts if the values of the two index-columns satisfy the union statement in memory. It occupies most of time to fetch tuples from disk files to memory, thus the query time does not change much from single index-column statement to union of two index-columns statements. In Fas-tRAQ, different index-columns of queried ranges can be searched in parallel in the RC-Tree. The overhead of union statements is to merge estimators from different index-col-umns. The merging overhead is negligible. Thus the query times of the two approaches are nearly the same .

RELATIVE ERRORS

Hive obtains exact query result, and its relative error of que-ried result is 0. As discussed in Algorithm 7, it does not lead to extra errors into the estimate when we merge estimators of different queried dimensions. Thus the estimated relative errors of the union queries in multiple index-columns are the same as the errors in single index-column queries. We discuss the detailed relative errors of the range-aggregate queries.

V RELATED WORK

The range-aggregate query problem has been studied by Sharathkumar and Gupta [20] and Malensek [21] in computational geometry and geographic information systems (GIS). Our work is primary focused on the approximated range-aggregate query for real-time data analysis in OLAP. Ho et al. was the first to present Prefix-Sum Cube approach to solving the numeric data cube aggregation [4] problems in OLAP. The essential idea of PC is to pre-compute prefix sums of cells in the data cube, which then can be used to answer range-aggregate queries at run-time. However, the updates to the prefix sums are proportional to the size of the data cube. Liang et al. [6] proposed a dynamic data cube for range-aggregate queries to improve the update cost, and time for each update, where d is the number of dimensions of the data cube and n is the number of distinct tuples at each dimension. The prefix sum approaches are suitable for the data which is static or rarely updated. For big data environments, new data sets arrive continuously, and the up-to-date information is what the analysts need. The PC and other heuristic pre-computing approaches are not applicable in such applications.

An important approximate answering approach called Online Aggregation was proposed to speed range-aggregate queries on larger data sets [7]. OLA has been widely studied in relational databases [8] and the current cloud and stream-ing systems [9], [10]. Some studies about OLA have also been conducted on Hadoop and MapReduce [10], [11], [12]. The estimated confidence intervals continuously. As more data is processed, the estimate is progressively refined and the confidence interval is narrowed until the satisfied accuracy is obtained. But OLA can not respond with acceptable accu-racy within desired time period, which is significantly important on the analysis of trend for ad-hoc queries.

Our work is related to two approximate answering methods: sampling and histogram. Sampling is an important

TABLE 5

Storage Overhead of RC-Tree Index with 1-4 Months Log Files

log files of 1-4 months	1 M	2 M	3 M	4 M
RC-Trees data volume (GB)	7.2	8.1	8.6	8.9
the volume ratio	0.031	0.017	0.012	0.009

technique for processing of aggregate queries at run time. The sampling for massive data sets includes two types: row-level sampling and block-level sampling [22]. The work in [22] analyzed the impact of block-level sampling on statistic estimation for histogram, and proposed the corresponding estimators with block-level samplings. Haas and K€onig€ [23] proposed a new sampling scheme, which combines the row-level and page-level samplings in the field of relational DBMS. Data sampling is also well used in the field of distributed and streaming environments [24], [25]. Histogram is another important technique for selectivity estimation. A series of alterative techniques were presented in other articles to provide better selectivity estimation than the original equi-width method. The multi-dimensional histograms were also widely studied by researchers. The problem is more challenging since it was shown that optimal splitting even in two dimensions is NP-hard [26]. The hTree [27] and mHist [28] are the typical works to support multi-dimensional selectivity estimation. While the current works are shown that it is quite expensive to generate a multi-dimensional histogram. FastRAQ combines sampling, histogram and data partition approaches together to generate satisfied estimations in big data environments. All of the above techniques are designed for distributed range-aggregate queries paradigm, and it achieves better performance on both query and update processing in big data environments.

VI CONCLUSIONS AND FUTURE WORK

In this paper, we propose FastRAQ—a new approximate answering approach that acquires accurate estimations quickly for range-aggregate queries in big data environ-ments. FastRAQ has Oð1Þ time complexity for data updates and Oðp _BN Þ time complexity for ad-hoc range-aggregate queries. If the ratio of edge-bucket cardinality (h_0) is small enough, FastRAQ even has Oð1Þ time complexity for range-aggregate queries.

We believe that FastRAQ provides a good starting point for developing real-time answering methods for big data analysis. There are also some interesting directions for our future work. First, FastRAQ can solve the 1:n format range-aggregate queries problem, i.e., there is one aggregation col-umn and n index columns in a record. We plan to investigate how our solution can be extended to the case of m:n format problem, i.e., there are m aggregation columns and n index columns in a same record. Second, FastRAQ is now running in homogeneous environments. We will further explore how FastRAQ can be applied in heterogeneous context or even as a tool to boost the performance of data analysis in DBaas.

REFERENCES

[1] P. Mika and G. Tummarello, ―Web semantics in the clouds,‖ IEEE Intell. Syst., vol. 23, no. 5, pp. 82–87, Sep./Oct. 2008.

[2] T. Preis, H. S. Moat, and E. H. Stanley, ―Quantifying trading behavior in financial markets using Google trends,‖ Sci. Rep., vol. 3, p. 1684, 2013.

[3] H. Choi and H. Varian, ―Predicting the present with Google trends,‖ Econ. Rec., vol. 88, no. s1, pp. 2–9, 2012.

[4] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant,, ―Range queries in OLAP data cubes,‖ ACM SIGMOD Rec., vol. 26, no. 2, pp. 73–88, 1997.

[5] G. Mishne, J. Dalton, Z. Li, A. Sharma, and J. Lin, ―Fast data in the era of big data: Twitter_s real-time related query suggestion architecture,‖ in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2013, pp. 1147– 1158.

[6] W. Liang, H. Wang, and M. E. Orlowska, ―Range queries in dynamic OLAP data cubes,‖ Data Knowl. Eng., vol. 34, no. 1, pp. 21–38, Jul. 2000.

[7] J. M. Hellerstein, P. J. Haas, and H. J. Wang, ―Online aggregation,‖ ACM SIGMOD Rec., vol. 26, no. 2, 1997, pp. 171–182.

[8] P. J. Haas and J. M. Hellerstein, ―Ripple joins for online aggregation,‖ in ACM SIGMOD Rec., vol. 28, no. 2, pp. 287–298, 1999.

[9] E. Zeitler and T. Risch, ―Massive scale-out of expensive continu-ous queries,‖ Proc. VLDB Endowment, vol. 4, no. 11, pp. 1181–1188, 2011.

[10] N. Pansare, V. Borkar, C. Jermaine, and T. Condie, ―Online aggregation for large MapReduce jobs,‖ Proc. VLDB Endowment, vol. 4, no. 11, pp. 1135–1145, 2011.

[11] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Tal-bot, K. Elmeleegy, and R. Sears, ―Online aggregation and continu-ous query support in MapReduce,‖ in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 1115–1118.

[12] Y. Shi, X. Meng, F. Wang, and Y. Gan, ―You can stop early with cola: Online processing of aggregate queries in the cloud,‖ in Proc. 21st ACM Int. Conf. Inf. Know. Manage., 2012, pp. 1223–1232.

[13] K. Bilal, M. Manzano, S. Khan, E. Calle, K. Li, and A. Zomaya, ―On the characterization of the structural robustness of data center networks,‖ IEEE Trans. Cloud Comput., vol. 1, no. 1, pp. 64–77, Jan.– Jun. 2013.

[14] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, ―Integrity for join queries in the cloud,‖ IEEE Trans. Cloud Comput., vol. 1, no. 2, pp. 187–200, Jul.–Dec. 2013.

[15] S. Heule, M. Nunkesser, and A. Hall, ―Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation

[16] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier, ―Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm,‖ in Proc. Int. Conf. Anal. Algorithms, 2008, pp. 127–146.

[17] [Online]. Available: http://research.neustar.biz/2012/12/17/hll-intersections-2/, 2012.

[18] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, ―Hive—a petabyte scale data warehouse using Hadoop,‖ in Proc. IEEE 26th Int. Conf. Data Eng., 2010, pp. 996–1005.

[19] D. Mituzas. Page view statistics for wikimedia projects. (2013). [Online]. Available: http://dumps.wikimedia.org/other/page-counts-raw/

[20] R. Sharathkumar and P. Gupta, ―Range-aggregate proximity quer-ies,‖ IIIT Hyderabad, Telangana 500032, India, Tech. Rep. IIIT/ TR/2007/80, 2007.

[21] M. Malensek, S. Pallickara, and S. Pallickara, ―Polygon-based query evaluation over geospatial data using distributed hash tables,‖ in Proc. IEEE/ACM 6th Int. Conf. Utility Cloud Comput., 2013, pp. 219–226.

[22] S. Chaudhuri, G. Das, and U. Srivastava, ―Effective use of block-level sampling in statistics estimation,‖ in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2004, pp. 287–298.

[23] P. J. Haas and C. Konig,€ ―A bi-level bernoulli scheme for database sampling,‖ in Proc. ACM SIGMOD Int. Conf. Manage. Data, ACM, 2004, pp. 275–286.

[24] S. Wu, S. Jiang, B. C. Ooi, and K.-L. Tan, ―Distributed online aggregations,‖ Proc. VLDB Endowment, vol. 2, no. 1, pp. 443–454, Aug. 2009.

[25] E. Cohen, G. Cormode, and N. Duffield, ―Structure-aware sam-pling: Flexible and accurate summarization,‖ Proc. VLDB Endow-ment, vol. 4, no. 11, pp. 819–830, 2011.

[26] S. Muthukrishnan, V. Poosala, and T. Suel, ―On rectangular partitionings in two dimensions: Algorithms, complexity and applications,‖ in Proc. 7th Int. Conf. Database Theory, 1999, pp. 236–256.

[27] M. Muralikrishna and D. J. DeWitt, ―Equi-depth multidimen-sional histograms,‖ ACM SIGMOD Rec., vol. 17, no. 3, 1988, pp. 28–36.

[28] V. Poosala and Y. E. Ioannidis, ―Selectivity estimation without the attribute value independence assumption,‖ in Proc. 23rd Int. Conf. Very Large Data Bases, 1997, vol. 97, pp. 486–495.

DOI: 10.18535/ijecs/v5i6.23