Average Iceberg Queries Computation With State Buckets Counter

Pallam Ravi[1], D. Haritha[2]

[1]scholar K L E F
[2]Professor in CSE ,K L E F University

[1]satishpallam@gmail.com

Abstract: *in limited memory environment computing Aggregate values needs many scan of data ,to avoid these scans use apriori property for computing anti-monotone iceberg queries but with efficient use bucket counter reduce scans for computing non anti-monotone iceberg queries ,till now all algorithms use single state Bucket counters, which suffers massive counter checking for candidates ,we propose two state Bucket counter which reduce counter checking, we conduct experiment on POP algorithm .*

Keywords — *Iceberg queries, Bitmap Index, Aggregate Function, Value-based Property.*

I. INTRODUCTION

In data analytics and data mining operation needs aggregate values ,these are handle with larger unique value called domain size, for computing aggregate value for each unique value need large amount of memory need ,for getting information from aggregate value which satisfy threshold value ,finding such kind of unique value called iceberg queries

the small set of domain values are produce as resultant ,it called tip of iceberg queries(ICQ), equaling to 10% of domain values

ICQ are handle large amount of data ,domain size is greater than available counter ,aggregate values computation over attributes , small set records produces as resultant set, and apply user threshold on aggregate values ,its needs huge computation and many data scans needed.

ICQ are use in many application such as data mining, embedded system which have limited memory, information retrieval

The aggregate function like COUNT, MIN, SUM STDIV ,MAX, and AVERGE are used in iceberg queries(ICQ), these classified as anti- monotone and non anti-monotone functions anti monotone aggregation function are MAX , MIN, SUM and COUNT. AVERAGE and STDIV are non anti monotone aggregation functions .use of anti-monotone property it reduce computation in candidate generation but not reduce in non anti monotone aggregation function, it needs many data scans need. Challenge is reduce data scans for computing non anti-monotone ICQ

average iceberg query is computing AVG aggregation . The general form of average iceberg query(ICQ)

SELECT $A_1,A_2 \ldots A_n$, *AVG(rest) FROM D*

GROUP BY $A_1,A_2 \ldots A_n$

HAVING AVG(rest) > Thres

Where D is data set which contain $A_1,A_2 \ldots A_n$ *,rest attributes , Thres is threshold value*

General method to Answering Average iceberg queries is sort the data with respective target attributes values, sorting take many scans and swapping its in efficient,

.the other method is allocating one counter buckers for each unique target value, in this method no of counter bucket need equals, but ICQ computed with limited memory so memory is not available as required ,the first work on average ICQ in [1] it use partition methods namely POP and BOP.

Partition methods POP work as fill bucket with data and remove which those are not satisfied threshold value, it reparative until no more remove counter from bucket, it suffers with many scans and check of counter buckets it explain in section 2 ,we proposed two state bucket to eliminate scans of counter buckets

Related work on ICQ in section2 ,in section 3 explain two state bucket, in section 4 explain experiment and dataset used .

II. RELATED WORK

The first work average ICQ based on partition based algorithm called POP and BOP [], ICQ proposed in [1] coarse count and sampling methods it give false negative

For anti monotone ICBQ are efficiently compute using Bitmap Indexes[2]in [6],in[6] use dynamic pruning to avoid mass empty BIT-WISE AND operations, the different author are improve the performance are in [8] cache based, in [9] checkpoint, in[11]look head pointer , in [13 work with distributed system and in [12]work on vertical datasets ,in[10] proposed bitmap number to sort the targeted attributes.

The algorithm used for iceberg cubs[5][14] and database queries[4] are not used for ICBQ .because have its own goals, iceberg cubs algorithm optimize use of memory where as ICBQ algorithms are reduced computational time.

Partition methods POP work as fill bucket with data and remove which those are not satisfied threshold value, it reparative until no more remove counter from bucket, it suffers with many scans and check of counter buckets it explain

With example1 ,we proposed two state bucket to eliminate scans of counter buckets

with one state bucket it suffers massive counter checking for candidate selection and average computation it will explain with an example 1

Example 1:let R is a relation with target attributes A, B and C, threshold values is 10 and Max counter in bucket is 3

	A	B	C
1	A1	B1	12
2	A1	B2	9
3	A1	B1	11
4	A2	B1	13
5	A2	B2	9
6	A1	B2	8
7	A1	B1	12
8	A2	B1	5

Relation R

Counter allocate for each new unique record ,bucket becomes

	Value	count
A1B1	23	2
A1B2	9	1
A2B1	13	1

Table 1.a.Counter Buckets values

Bucket are fill up third record scan then remove counter which average value is below threshold value, so A1 B2 Is removed, allocation counter for A2B2 then bucket is full

	Value	count
A1B1	23	2
A2B1	13	1
A2B2	9	1

Table 1.b.Counter Buckets values

For remove counter need to read all three counter and calculate its average value its ,to void all counter checking and calculating average value we proposed a two stage counter bucket

Above problem with solve with new way checking using theorem 1 average threshold that explain in example 2

Theorem 1: Checking average values of $a_1, a_2 \ldots a_n$ with T

$$F = \sum_{i=0}^{n} (a_i - T) \qquad (1)$$

F > 0 *average values of* a1, a2 …. an above T

F < 0 *average values of* a1, a2 …. an below T

Example2: With use of example 1 problem, checking average value of A1B1 its allocate bucket with A1B1 2 (12-10) for record 1 for record 3 update it as A1B1 2+1 (11-10) ,it becomes 3, finally check it is above zero are not ,so avoid maintain counter

With use of theorem 1 bucket becomes table it eliminate A2B1 and A2B2 without check the A1B1

	Value
A1B1	5
A2B1	-2
A2B2	-1

Table 2. two state Counter Buckets

III. TWO STATE COUNTER BUCKETS

In POP algorithm the buckets operations are New counter, Update and remove that explain in fig 1, In this remove operation did with respective all counter in bucket to avoid this two state bucket counter proposed ,in state one have new counter ,remove update and change state, the new operation change state is added and differ in remove operation, the remove operation remove all counter which average value below threshold and update the counter which value greater than threshold with *value/count – threshold.*

In state 2 new counter ,remove and update operation are differ with state 1 and one state bucket operation, in new counter bucket allocate with single values only i.e *<value - threshold>* when value greater than threshold, where as in state1 without checking value with threshold, it create with two values i.e *<value ,1>*, Update operation add value with value-threshold and remove operation done with respective single counter if that counter have less than zero then remove it

State 1:

New counter*:*

if no Counter in Bucker for a Record then

Allocate New Counter with <value ,1>

Remove*:*

if |counter buckets| >=Max_counter then

for all counter update its value with its value/count - threshold

for all counter its value < Threshold

Remove counter from bucket

Update*: if a record have counter in bucket*

Add value of counter with record value and increment count

Change state: *After remove, change state to state 2*

State 2:

New counter:

if no Counter in Bucker for a Record and its value > Threshold then Allocate New Counter with <value - threshold>

Remove*: if counter value <0 then*

Remove counter from bucket

Update*: if a record have counter in bucket*

Add value of counter with record value-threshold

Change state: *After buckets is full, change state to state1*

In state 1 &2 in bucket operation flow represent in fig 5 & 6 respectively , in state 1 first its check for new counter allocation ,if already have counter for it update it for record not able to create new counter perform remove operation then change state to 2 .

In state 2,first check for new bucket allocation if it have counter for it update it and perform remove operation, change state operation done when no counter for a record

Algorithm: state 1 Bucket operation

```
If(!New counter){
    if(!Update){
        if(Remove){
            Change state 2
        }
    }
}
```

Algorithm: state 2 Bucket operation

```
If(!New counter){
    if(Update){
```

*if(**Remove**){*

}

}

}else{

Change state 1

}

One State Counter Buckets:

New counter*: if no Counter in Bucker for a Record then*

Allocate New Counter with <value ,1>

Remove*:*

if |counter buckets| >=Max_counters then

for all counter its value < Threshold

Remove counter from bucket

Update*: if a record have counter in bucket*

Add value of counter with record value and increment count

In POP algorithm have two scans namely first_scan and Second_Scan , our two state bucket used in first_scan only , in second_scan differs the record update ,it did as Update operation in state 2, in first_ scan bucket in state1 New Counter and Update operation are not differ only differ in remove operations

Algorithm: Two State Bucket POP algorithm(statesPOP)

First Scan: start with Bucket in state 1

*In State2 :**change state** start second scan*

*Second scan: perform State2 :**Update***

Print counters in Bucket which > 0

Domain Ration	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
uniform	118	2571	1900	2000	2062	2631	3229	4753	5231
Normal	3	42	325	755	153	208	122	193	170

Table3:No of counter Buckets Scans

IV. EXPERIMENT

We conduct experiment on 4GB RAM and windows 7 operation system, we use two synthesized data set with different distribution namely normal and uniform distribution of target attributes

for this experiment with 100,000,000 records, about 2.1 GB. The distribution of target attributes values , domain size and max average value as follow

Dataset 1: Target attributes in normal distribution and domain size 220000, min and max values are 0 and 999000 respectively

Dataset 2: Target attributes are in uniform distribution and domain size 1,000,000 , min and max values are -19000and 21000 respectively

Domain Ratio: It is the ratio of domain size and no of counter buckets ,Domain ratio >=1.0 indicates sufficient counter bucket available for all possible target values(domain size),if it is <1.0 indicates insufficient counter buckets

To evaluate Performance of **statesPOP** ,we did an experiment with respective of execution time , In this experiment keep threshold value constant with changing of domain ratio, threshold value of 700,000 for data set1and 14000 for dataset2,the experiments reveals the **statesPOP** gives better performance shows in fig 1 &2. Due to reduce scans of counter buckets shown in table 1

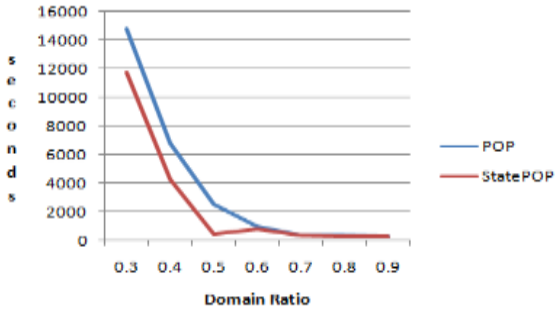

Fig 1 .Execution Time with Dataset 1

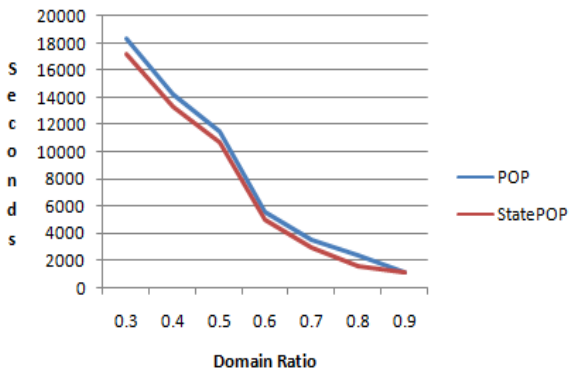

Fig 2. Execution Time with Dataset 2

IV. CONCLUSION

Our theorem for efficient checking of average value with threshold used in our two sate counter buckets for eliminate rescans of entire counter buckets for remove counter which are below threshold value due this reduce computation time for average iceberg queries.

References

[1] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J.D. Ullman, "*Computing Iceberg Queries Efficiently*", Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 299-310, 1998.

[2] C.Y. Chan and Y.E. Ioannidis, "*Bitmap Index Design and Evaluation*", Proc. ACM SIGMOD Intl Conf. Management of Data, 1998.

[3] J.Bae and S.Lee. "*Partitioning algorithms for the computation of average iceberg queries*." Proc. Second Intl Conf. Data Warehousing and Knowl-edge Discovery (DaWaK), pp. 276-286, 2000.

[4] K.P. Leela, P.M. Tolani, and J.R. Haritsa, "*On Incorporating Iceberg Queries in Query Processors*", Proc. Intl Conf. Data-base Systems for Advances Applications (DASFAA), pp. 431-442, 2004.

[5] J. Han, J. Pei, G. Dong, and K. Wang, "*Efficient Computation of Iceberg Cubes with Complex Measures*", Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 1-12, 2001.

[6] B.He,H-I.Hsia,Z.Liu,Y.Huang and Y.Chen "*Efficent computing Iceberg queries using compresed bitmap index*" IEEE TRANSACTION ON KNOWLEDGE AND DATA ENGINEERING,2012'

[7] Vuppu shanker et al ."*Effective Iceberg Query Evaluation* by Deferring Push and Pop Operations,IJAC, ISSN:2051-0845, Vol.36, Issue.2.2015

[8] Vuppu shanker et al," *Cache Based Evaluation of Iceberg Queries*" ICCCT-2014 IEEE,2014

[9] Vuppu shanker et al, Answering Iceberg Queries Efficiently Using Check Point Mechanism,IJAC , Vol.46, Issue.2,2015

[10] Pallam Ravi et al "*COMPUTING ICEBERG QUERIES HAVING NON ANTI MONOTONE CONSTRAINS WITH BIT MAP NUMBER*", JATIT,Vol. 8. No. 2 – 2016

[11] Kale Sarika Prakash et al,"*Tracking Pointer and Look Ahead Matching Strategy to Evaluate Iceberg Query"*.JCS,2017

[12] Y.Cui, W.Perrizo "*Aggregate Function Computation and Iceberg Query-ing in Vertical Database*",Computers and Their Applications,2006

[13] Vuppu shanker et al "*Efficient iceberg evaluation in Distributed databases by Developing Deferred Strategies*",2016

[14] K.S. Beyer and R. Ramakrishnan, "*Bottom-Up Computation of Sparse and Iceberg CUBEs*", Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 359-370, 1999.

Pallam Ravi, Research scholar in KL university, Assistant professor in Dept. of Computer Science & Engineering Anurag Group of institutions, he reived B.Tech in Computer Science & Information Technology from JNTU Hyderabad, M.Tech in Software Engineering from JNTU Hyderabad University, he present research on data mining

Dr. D. Haritha, Professor in Dept. of Computer Science & Engineering is a Committed Academician and Active Researcher having 8 years of teaching experience at KL University. She Received B.Tech in Computer Science & Engineering from JNTU Hyderabad, M.Tech in Computer Science & Engineering from Acharya Nagarjuna University, Ph.D from Acharya Nagarjuna University in the area of Software Reliability. She has 14 research publications in various referred International Journals. She is a life member of CSI. Currently 4 research scholars are pursuing Ph.D. degree under her guidance. Her areas of interest are Software Reliability, Data Analytics and Text Mining.