ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse203952020.pdf
https://doi.org/10.30534/ijatcse/2020/203952020

Computing Iceberg Queries on Map Reduce Framework

Pallam Ravi[1], D Haritha[2]
[1]Schalor KLEF, India, satishpallam@gmail.com
[2]Professor KLEF, India, haritha_donavalli@kluniversity.in

ABSTRACT

To find insights in raw data, Need to extract knowledge from it. knowledge represents with different patterns. These patterns are used in business analysis and data analysis. Frequent item-set is one kind of pattern in data mining. The frequent item-set is a set of co-occurrence item-sets, it support value satisfies the user-specified threshold value. Iceberg queries are also find co-occurrence item-sets, but in it Items(Attributes) are grouped and compute Aggregate values based on this item-set groups. These's value above the threshold are very small(tip of the iceberg).Compute iceberg queries within available memory is difficult because aggregate values are large in number (iceberg) .It needs huge computation and memory requirement, so first find candidate sets then generate results. To find candidate sets need to have read entire data information, in parallel and distributed environment is difficult get that information for computing candidate sets. Because no information about entire data. In proposed algorithms use parallel and distributed environment map-reduce framework to find candidate sets within available memory and reduce memory requirement and computation . Proposed different algorithms named as single reducer (SRIceberg), Multiple reducers (MRIceberg) and Multiple with iterative (IMRIcebrg). From our experiments IMRIcerberg algorithm gives better performance than other**.**

Key words : Iceberg queries, Map-Reduce, Big data and Pattern mining.

1. INTRODUCTION

In Business intelligence, data analytics is one of the field used ,data analytic have set of methods and techniques convert raw data into meaning full information and extract pattern .Extract pattern in data, computing the aggregate values major task in it. Frequent item set is one of the pattern. Frequent item is a set of co-occurrence item set, in which items set satisfies the user specified threshold value, majorly use mini support count. . Iceberg queries are one kind of frequent items set, item are grouped(Attributes) , using frequent item set generate association rules[15], computed based on item group(aggregated), this aggregate values are large in number (Iceberg),find aggregate values above the threshold values, it will be very small (Tip of Iceberg).

Answer the iceberg queries involve compute large number of aggregate values in number(Iceberg) but,it return only 10% of complete aggregate values, which are above the threshold value only , Example of iceberg query so it require huge computation and memory .iceberg queries run on huge data in which domain size(number of aggregate values) is very large, major challenge is cannot store all domain values in available memory at a instance because memory is limited, an example iceberg query show in Example1.
Example1:
Find the students ,collage who got marks above 70% of total marks.
In Example 1 problem represent inSQL query as follows for better understanding.
SELECT *Student,college* FROM *marks_table* GROUP BY *subject_marks, collage* HAVING
SUM(Student_marks)/*total_marks*> 0.70
Student and ,collage are attribute which represent set students and set of college names
SUM is aggregate value,0.70 is threshold value

The algorithm for compute frequent pattern are not efficient for compute iceberg queries .In frequent pattern mining problem represent as : let I is set of singletons(items) $I=\{i_1,i_2,\ldots i_n\}$, Frequent item-sets F { $f= \{i_j,\ldots i_k\} \subseteq I$ $,|F|<=|I|, Support(f)> T\}$, $support(f) = |\{\forall t_l \in T : f \subseteq t_l\}|$,T $\in$ *D, D is data set. the iceberg query represent as* $I=\{A_1,A_2,A_3,B_1,B_2,B_3,C_1,C_2,C_3\}$, Frequent items sets F,{ $F=\{i_A,i_B,i_C\},|i_A|=|i_B|=|i_C|=3,i_A\in\{A_1,A_2,A_3\},i_B\in\{B_1,B_2,B_3\}\}$, $i_C\in\{C_1,C_2,C_3\}$, $|F|<=|I|, Agg(f)> T\}$,The variation of iceberg query , first variation in candidate set generation. in frequent item-set S_n use S_n-1 ,and generate candidate set use set join operation. apply Apriori pruning technique to reduce the candidate set computation. The same set join operation not apply to iceberg query computation for computing candidate set , because the items are grouped illustration in example2

Example2: in frequent item set let $s_2=\{\{i_1,i_2\}\{i_1,i_3\}\}$,compute $s_3 = s_2 \bowtie s_2$, $s_3=\{\{i_1,i_3\}\}$.
in iceberg query item set let $s_2=\{\{A_1,B_2\}\{A_1,B_1\}\}$,
$S_3=\{\{A_1,B_2,C_1\},\{A_1,B_1,C_1\}\{A_1,B_1,C_2\}\{A_1,B_1,C_2\}$ $\{A_1,B_2,C_3\}\{A_1,B_1,C_3\}\}$.
Second variation between frequent item-set and iceberg computation is size of item-set, in frequent item- set size

are 1 to n. But in iceberg query, all item set are equal size .The size of item-set is equals to number of grouped items (Attributes) ,third variation is threshold support count function only, But in iceberg query are aggregate function like SUM,COUNT,AVG.. etc.

.

The general technique for answering iceberg queries is sort the records based on domain values and compute aggregate values ,sorting is takes heavy computation and space. Domain size is very large than available memory. Only 10% of aggregate values in the result set, so huge computation required and many scans over the data set. for example domain size is S in data set , available memory is M so S/M scans are required compute the all aggregate values .reduce scans over the data is primary goal in iceberg query computation.

iceberg cube computation and iceberg query have different goal in cube computation[13[14] the aggregate values shared .

Today data set size growth in volume ,to compute iceberg query ,many scans takes place ,to improve the performance required parallel computation needed, so for no work done in this context, so we use map reduce framework for computing iceberg queries. we proposed algorithm use single reducer (SRIceberg) .second we use Multiple reducer (MRIceberg) and third algorithm Multiple with iterative (IMRIcebrg)

The rest of paper are organized as in section2 discussed relative works, in section3 proposed algorithms, section 4 discussed our experiment results , conclusion in section 5.

2. RELATIVE WORK

For computing aggregate queries sorting and hashing method are used. These methods are not apply to iceberg queries because it have huge domain ,The first iceberg proposed in [1] ,in [1] use coarse count and sampling methods proposed, for avoid false passive and false negative proposed hybribed method. partition method[2] are used for average aggregate values in iceberg query , methods and two bucket state algorithm [3],in [5] dynamic pruning methods proposed for bitmap index data[6]To improve the bitmap map based iceberg query many different strategies like cache based[7], look head pointer[8][9],[10][16][17][18]bit map number proposed proposed,[11] low iceberg queries,[12] Iceberg Querying in Vertical Database, different algorithm which compute pattern like frequent item set, max, closure frequent item sent on map reduce on hadoop. [21] min,max iceberg queries based on value based property, No algorithm use parallel distributed computation for answer the iceberg queries, the map reduce framework support nothing shared architecture, compute in parallel ,it have two phases, one is map and phase, map phase generate key and value pairs ,and reduce phase combine all record which have same key and compute values have

In figure 1 show how map reduce works as mapper generate <key,value> in above <A2B2, 1> <A1B3,2> <A2B1,5> <A2B2, 8> <A1B3,3> <A2B1,9> <A2B2, 6> <A2B1,1> <A1B3,8> <A2B2, 0> <A3B1,3> <A3B1,2> the reducer combine files which have same key final values are <A1B3,13> <A2B1, 6> <A2B2,15> <A3B1, 5> .this general methods is not applicable in case of computing iceberg queries because the number of key are huge so the reducer required huge memory and sorting and shuffling in map reducer frameworks take heavy computation

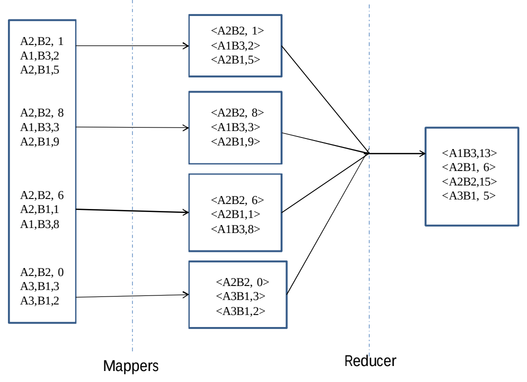

Figure 1:Map-Reduce Framework

[10] in proposed method computing frequent item set by Apriori based pruning strategies as we discussed in introduction the iceberg query have some variation ,we cannot apply same methods to iceberg query computations

With single reducer is bottleneck for computing Iceberg queries on map reducer, because in pattern mining algorithm many unique key ,value pair (k,v) generated, it require computational cost and memory requirement. to avoid this problem we proposed multiple reducer based algorithm, we allocate one Attribute keys to a single reducer by this we can reduce memory requirement of each reducer

With multiple reducer ,same key will pass to different reducer ,the information in data lost, it leads huge number of key received by reducer ,to avoid this some set of key are allocated a reducer, based one attribute items we allocate to each reducer.

To reduce number of unique key analyzed by each reducer, we allocating all reducer to one attribute items ,let attributes A,B,C . A={A1,A2,…A1000} B={B1,B2…..B1000} and C={C1,C2,….C1000} are item set for each attributes, a iceberg query based attribute (A ,B,C) ,total pattern are 1000*1000*1000=10^9, for example Attribute A item are allocate to all reducer , for first reducer allocating 100 item so that reducer receive on 10^7 key only, for second also same allocating 100 item ,it receive 10^7 key only, and so on.

our proposed three new algorithms for compute the iceberg queries, first algorithm use single reducer (SRIceberg) .second we use Multiple reducer (MRIceberg) and third algorithm Multiple with iterative (IMRIcebrg)

3. PROPOSED METHOD

First we proposed algorithm 1 called Single Reducer Iceberg(SRIceberg) ,in map reduce frame work , mapper phase generate <key ,value> pair and sent it reducer, the reducer combine all same key ,in this all keys are have same no of item sets means same length ,each item belongs to one attributes

Algorithm1: Single Reducer Iceberg (SRIceberg) Algorithm

```
Begin procedure SRIcebergMapper(tl)
 //p item-sets in tl,Agg is value
Emit(p,Agg);
End procedure

Begin procedure SRIcebergReducer(p,(Agg($P_1$), Agg($P_2$), Agg($P_3$)….. Agg($P_m$))
Aggvalue=0;
For all Agg € ,(Agg($P_1$), Agg($P_2$), Agg($P_3$)….. Agg($P_m$))
Aggvalue=Aggvalue+Agg
End for
Emit (P,Aggvalue)
End procedure
```

In figure 1 shows single reducer ,the mapper find aggregate values like sum it require huge amount of memory to reduce that memory ,we proposed Algorithm2: Multiple Reducer Iceberg (MRIceberg) Algorithm ,in which we allocate items of single attribute to each reducer by this the memory requirement of each reducer is reduced

It shown in figure 2 ,in that attribute A items are allocated to each reducer ,reduce one A1,two A2,reduce take A3 mean the reducer will compute aggregate value which key is prefix with allocated item ,A1 will take care about <A1B3,13>,second reducer <A2B1, 6> <A2B2,15> and third reducer about <A3B1, 5>,so in single reducer in figure 1 reducer required 4 memory counter needed ,with multi reducer one, second and third needs only1,2 and 1 counter respectively.

Algorithm 2: Multiple Reducer Iceberg (MRIceberg) Algorithm

```
Begin procedure SRIcebergMapper(tl)
 //p item-sets in tl,Agg is value
Emit(p,Agg);
End procedure

Begin procedure SRIcebergReducer(p,(Agg($P_1$), Agg($P_2$), Agg($P_3$)….. Agg($P_m$))
C={set of item allocation reducer}
P prefix € C;
Aggvalue=0;
For all Agg € ,(Agg($P_1$), Agg($P_2$), Agg($P_3$)….. Agg($P_m$))
Aggvalue=Aggvalue+Agg
End for
Emit (P,Aggvalue)
End procedure
```

With Multiple Reducer Iceberg (MRIceberg) Algorithm ,its checks all possible items set aggregate values ,and its need sorting and shuffles ,there is no pruning ,for pruning we proposed Algorithm3: Iterative Multiple Reducer Iceberg (IMRIceberg) Algorithm in which map reducer work with multiple iteration each iteration the map key size will increment ,it start with key size with one end with no of Attributes,

For each iteration ,each aggregate value is compared with threshold value ,if it is below threshold value keep it infrequent item set, this infrequent item set will used next mapper for generating key, it does not generate key with prefix it is in infrequent set, it will refreshed each iteration

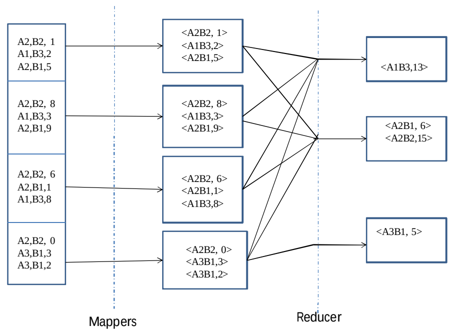

Figure 2:Map-Reducer with Multi Reducer

In figure 3 shows that first iteration in which it find the A3 is infrequent because it aggregate value below threshold value (7),it show in dashed in reducer, second iteration the mapper not generate key with A3 as prefix, it shows in figure 4

Algorithm 3: Iterative Multiple Reducer Iceberg (IMRIceberg)

```
AlgorithmBegin procedure SRIcebergMapper(tl,s)
 //p item-sets in tl,Agg is value
I={ }
Emit(p,Agg);
End procedure

Begin procedure SRIcebergReducer(p,(Agg($P_1$), Agg($P_2$), Agg($P_3$)….. Agg($P_m$))
C={set of item allocation reducer}
P prefix € C;
Aggvalue=0;
For all Agg € ,(Agg($P_1$), Agg($P_2$), Agg($P_3$)….. Agg($P_m$))
Aggvalue=Aggvalue+Agg
End for
Emit (P,Aggvalue)
End procedure
```

After second iteration produce final results show in fig In next section we will discussed about experiment results

Algorithm 4: MINMAX Multi reducer Iceberg (MMMIceberg)

```
Algorithm Begin procedureMMMIcebergMapper(tl,s)
//p item-sets in tl,Agg is value
I={ }
If(Agg>Threshold)
        Emit(p,Agg);
End procedure
```

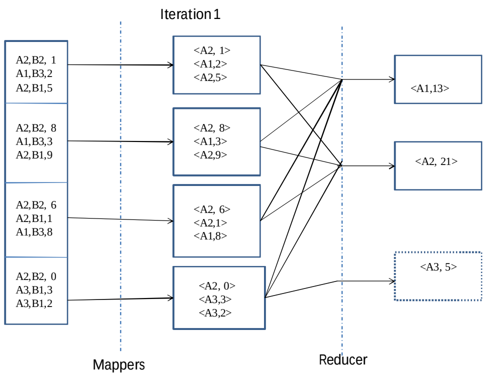

Figure 3: Iteration-1Itterative Map Reducer

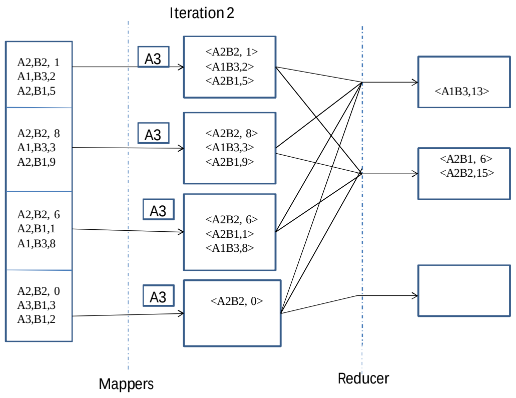

Figure 4: Iteration-1Itterative Map-Reducer

4. EXPERIMENT RESULTS

The performance of algorithms are studied with different data volume ,the main objective of this study as follows
1. Study the computation for variant algorithms on map reduce frame work
2. Impact of attribute order based on cardinality
3.Study performance our algorithm with different threshold values
4.Study performance of min and Max aggregate function in iceberg queries
The experiment is conducted on environment with 16 GB ram and i7 8 th intel processor. The software configuration is established with hadoop 3.0 cluster with single node. The experiment is conducted on various specifications with data records ranging from 1 million to 10 million records with attributes with cardinalities of 10,5,4,3. The Hadoop setup has different configurations with combinations such as single reducer and mapper, multiple mapper and reducer. Each experiment time has been recorded with precision and accuracy by conducting in suitable environment
in fig 5 shows that iterative algorithm give good performance that other, in single Reducer (SRiceberg) , mapper generate huge keys ,which we need shuffles and sort takes the computation time, with MRiceberg queries the key are distributed among the reduce so no of sort key per reducer is minimized so the performance was improved. in IMRIceberg queries takes advantage of multi reducer ,in it each iteration prune the candidates based upon threshold values, so the performance is improved

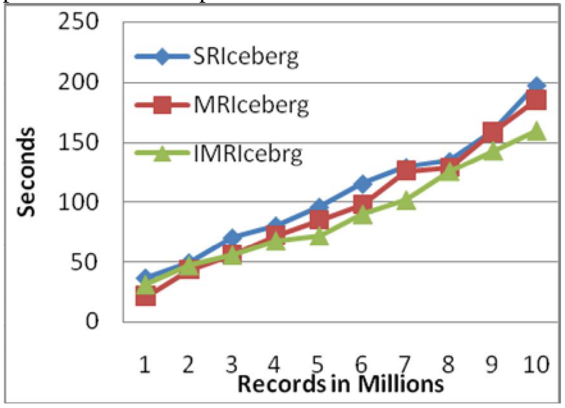

Figure 5: Performances of our proposed algorithm

in figure 5 show that performances of our proposed algorithm with different threshold values, IMRIceberg is give high performance because it is threshold based pruning ,

in figure 6 show attributes cardinality influence the performance of IMRIceberg in it ascending order give better then descending order, because no of key generation are less compared to descending order.

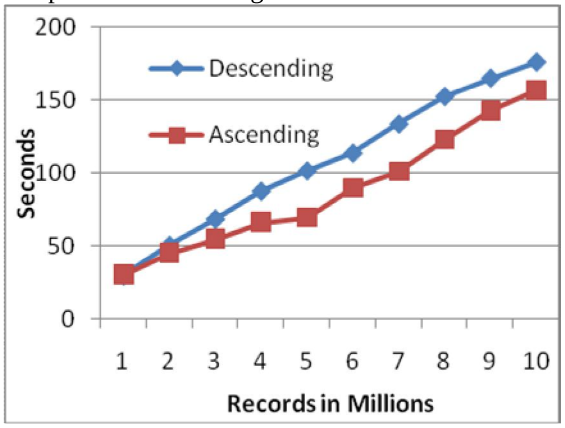

Figure 6: Performances of IMRIceberg algorithm with Attribute ordering

In our experiment we use SUM aggregate function ,our first two algorithm can used other all aggregate values also but in IRMiceberg algorithm only applicable for anti-monotone

aggregate functions only, it not works for non anti-monotone functions like AVG,
For MIN and MAX aggregate function the pruning will done in mapper only ,the mapper generate key only when that aggregate value is satisfy the threshold values, Our algorithm will work with multi nodes in a cluster and multi mapper also
.

5. CONCLUSION

with huge volume data ,Generate candidate set for iceberg query computing with the parallel and distributed map-reduce framework is suitable because it have map and reducer phases. multi reducer and iterative manner allocation strategies to reduce memory requirement to allocate the key to reducer ,the effective way of allocate key s is decreasing order of their cardinality of attributes. For MIN and MAX aggregate function the pruning will done in maper only ,the mapper generate key only when that aggregate value is satisfy the threshold values. Our algorithm will work with multi nodes in a cluster and multi mapper also. The future study is improve the AVG aggregation function iceberg query performance with map-reduce framework..
.

REFERENCES

1. M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J.D. Ullman **Computing Iceberg Queries Efficiently** Proc. Int'l Conf.VLDB, pp. 299-310, 1998
2. Bae and S. Lee **Partitioning Algorithms for the Computation of Average Iceberg Queries**," Proceeding.Sec Int'l Conf. DaWaK , 2000.
3. Pallam Ravi,D.Haritha **Average iceberg queries computation with state buckets counter**". SSRG International Journal of Engineering Trends and Technology, 2020, 68(8), pp. 53-57.
4. Pallam Ravi, D.Haritha **A Survey: Computing Iceberg Queries**", IJET 7(2.7).
5. B.He et al **Efficent computing Iceberg queries using compresed bitmap index** IEEE Trans' On Know And Data Eng', 2012
6. V shanker et al **Cache Based Evaluation of Iceberg Queries** "ICCCT-2014
7. V Shanker et al **Effective Iceberg Query Evaluation by Deferring Push and Pop Operations**, IJAC Vol.36, Issue.2.2015
8. V Shanker et al, **Answering Iceberg Queries Efficiently Using Check Point Mechanism**,IJAC , Vol.46, Issu.2,2015
9. Pallam Ravi and D Haritha **Computing Iceberg Queries Having Non Anti Monotone Constrains With Bit Map Number**, JATIT,Vol. 8. No. 2 -- 2016
10. K.P. Leela et al **On Incorporating Iceberg Queries in Query Processors**, Proc. Intl Conf. DASFAA, pp. 431-442, 2004.
11. Y.Cui and W.Perrizo **Aggregate Function Computation and Iceberg Query-ing in Vertical Database**", CTA, 2006
12. 4. 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.
13. K.S. Beyer and Ramakrishnan R, **Bottom-Up Computation of Sparse and Iceberg CUBEs**, Proc. ACM SIGMOD Int"l Conffeence. Management of Data, pp. 359-370, 1999.
14. S. Agarwal et al **On the Computation of Multidimensional Aggregates** Proc. Int'l Confference. VLDB , pp. 506-521, 1996.
15. R. Agrawal et al **Mining Association Rules between Sets of Items in Large Databases**," Proc. ACM SIGMOD Int'l Conference Mgmnt of Data, pp. 207-216, 1993.
16. G. Antoshenkov **Byte-Aligned Bitmap Compression**, Proceeding.Conf.Data Compression PP. 476, 1995.
17. K.S. Beyer and R. Ramakrishnan, "**Bottom-Up Computation of Sparse and Iceberg CUBEs**," Proceeding. ACM SIGMOD Int'l Conference. Mgmnt of Data, pp. 359-370, 1999.
18. C.Y. Chan and Y.E. Ioannidis, "**Bitmap Index Design and Evaluation**," Proc. ACM SIGMOD Int'l Conf. Management of Data, 1998.
19. F. Delie`ge and T.B. Pedersen, "**Position List Word Aligned Hybrid: Optimizing Space and Performance for Compressed Bitmaps**," Proc. Int'l Conf. Extending Database Technology (EDBT), pp. 228-239, 2010.
20. Lakshmi L,**A Comparative Study of Navigation Techniques and Information Retrieval Algorithms for Web Mining,IJATCSE,vol 8(1.3) pp 10-14**
21. Pallam Ravi and D.Haritha **Efficient computation of min & max iceberg queries using value based property**.Journal of Engineering Science and Technology Review, 2019, 12(6), pp. 202-207