EasyChair Preprint

№ 3135

Optimization of Multi-way Join Cost using System R* and SharesSkew

Leela Krishna Chittem and Venkata Subba Reddy Poli

April 8, 2020

Optimization of Multi-way Join Cost using System R* and SharesSkew

Chittem Leela Krishna
Research Scholar
Dept. of CSE
S.V.U. College of Engineering
Tirupati, India
chittem.leelakrishna03@gmail.com

Dr. Poli Venkata Subba Reddy
Professor
Dept. of CSE
S.V.U. College of Engineering
Tirupati, India
vsrpoli@hotmail.com

Abstract—In a distributed environment relations are stored at different sites. To perform algebraic operations such as join, the relations are to be transferred from one site to the other in such a way that the total communication cost is minimized. This paper deals with the problem of computing the transmission cost using two approaches. The first uses System R* algorithm approach when the data is of non-skew nature and the second uses SharesSkew algorithm when the data has skews i.e., same value for a specific join attribute, named as Heavy Hitter(HH). Rules of the two algorithms to be followed for performing join are specified and by illustrating with Banking System, the communication cost is evaluated.

Keywords—Distributed Databases, Communication Cost, System R* algorithm, SharesSkew algorithm, Heavy Hitter

1. Introduction

A database is said to be distributed if its data is stored at different locations[1]. To get access of such data, multiple computers are to be connected through a communication link (channel). To perform algebraic operations on relations stored at different locations such as join, tuples are to be transferred from one location/site to the other in an optimized manner i.e., the cost of transmission is to be minimized.

Data in the relations is said to be skewed in nature if there exists same value for a specific attribute frequently, named as a heavy hitter(HH). To handle skewed data in multi-way joins, the first step is to identify the attributes with heavy hitters and others in a normal manner.

In this paper, we have numerically solved two problems. First, we have evaluated the cost function of joining three relations stored at three different locations. To do this, we made use of the System R* optimization algorithm[1], which assumes the relations participating in the join as the leaf nodes of an unordered tree with a CHOICE operator to choose among various combinations to minimize the total transmission cost.

The second problem deals with the computation of cost for a multi-way join of relations with skewed data using MapReduce[2,8] mechanism. The map phase results in generating key-value pairs for the tuples using a Hash function[8], and the keys are sent to the reducers. If the data is skewed (contain heavy hitters), then the join is decomposed into several residual joins[2]. The cost of transmitting data from mappers to reducers is the communication cost which is to be minimized using SharesSkew[2] algorithm.

The paper is organized as follows: Section 2 provides related work which deal with the two problems mentioned. In Section 3, the rules of system R* algorithm that are to be followed and the formulae to calculate total transmission cost are given. Finally, we have evaluated the total cost of joining three relations at three different sites by considering Banking system as a numerical example. In Section 4, an overview of Shares algorithm is given, followed by the SharesSkew phases and evaluation metrics. At last, the formation of residual joins and the total cost of each join are calculated by taking Banking system as an example.

2. Related Work

In [3], a new approach to identify the map-key with each attribute getting a share to find the reduce process on a large-scale data is given. Once the values of attributes are hashed, the tuples are

replicated to the reducers. The problem of optimizing shares for a fixed number of reducers is also discussed.

The problem of computing a parallel query that is run in multiple servers in one round of communication with two cases is discussed in [4]. First, by considering only the statistical parameters of the database with a skew-free data viz., relation cardinality. Second, data in the relations with skews (heavy hitters). For both cases, upper and lower bounds are also expressed.

In [5], communication cost is evaluated for complex parallel queries involving data reshuffling. Algorithms for evaluating Multi-join query and optimal-communication in distributed environment are also discussed.

Handling skewed data during joins in MapReduce using SkewTune is discussed in [6, 7]. Mitigating skews in real time applications at runtime in a public cloud and through a Graphical User Interface environments are also demonstrated.

3. System R* Optimization

System R* is used in a distributed environment where algebraic operations such as select, project, join, union etc. are to be performed on the relations stored at different locations. R* algorithm performs query evaluations in an optimized manner by introducing a new operator, namely CHOICE[1], which gives a provision for the system to pick an identical copy of a relation among its replicas. Suppose, if we want to perform R⋈S and relation R has three replicas R_1, R_2, R_3 at three different locations and relation S has two replicas S_1, S_2 at different locations, then the join operation can be expressed as CHOICE(R_1, R_2, R_3) ⋈ CHOICE(S_1, S_2) as shown in Fig 1. The cost of transferring the relations R and S can be minimized by picking the replicas near to each other using the CHOICE operator.

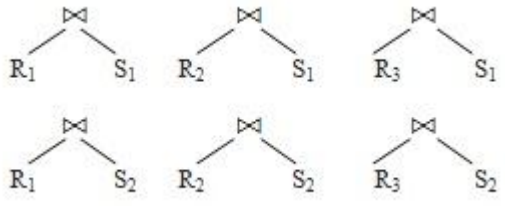

Figure 1 Joins possible with CHOICE operator

A. Description of R* Algorithm

To perform join operation between two relations R and S stored at different locations, R* algorithm follows the below rules:

- Fetch relation R to the site of S and perform join there. The cost will be the size of relation R plus a constant.
- Fetch relation S to the site of R and perform join there. The cost will be the size of relation S plus a constant.
- Fetch relations R and S to a new site and perform join there. The cost will be the sizes of relations R and S plus a constant.
- Perform a lookup of all tuples of relation R. For each tuple in R, find a matching tuple in S and fetch only such tuples to the site of S and join there. The cost will be the product of the size of R and expected tuples to be shipped plus a constant.
- Perform operation similar to the above step with the relations R and S being swapped.

To perform join operation between two relations R and S stored at the same location, the below rules are followed:

- Compute the join at the same site where R and S reside. Since there is no transfer of relations involved, the cost is zero.
- Compute the join at the same site where R and S reside and transfer the result to a new location. The cost will be the size of the result relation plus a constant. Or the relations R and S can be shipped to a new location and perform join there, and the transmission cost will be the sizes of relations R and S plus a constant.

B. Evaluation of Cost Function

When dealing with the relations distributed across various sites, evaluating the total cost using system R* will be equal to the cost of transmission and the cost of computing the algebraic operation. From the rules given above, the cost of fetching a relation R to the new site and perform an operation with other relation residing at the same site will be given as size(R)+c_0, where c_0 is a constant. Similarly, the cost of performing a lookup of the tuples of a relation R with the other relation S for matching tuples will be size(R)[c_0+(size(S)/I)], where c_0 is a constant and I is the image size of the projection of attribute of S onto R∩S. After performing the join

between relations R and S, i.e., R⋈S, the new image size will be [size(R)*size(S)]/max(I(R),I(S))

C. Applying R* Algorithm on an Example

Consider a Bank information system contains three relations Customer(C), Account(A) and Balance(B) with their schemas C(cname,custid), A(custid,acno) and B(acno,bal) and sample data as given in Table 1:

Table 1 Sample data of Banking System

Customer			Account			Balance	
Cname	Custid		Custid	Acno		Acno	Bal
C_1	101		101	1001		2001	1000
C_2	102		102	1002		2002	1000
C_3	103		103	1003		1001	1500
C_4	104		104	1004		1002	1500
			201	2001		…	…
…			202	2001			
			…	…			

Assume the three relations are stored at three different sites L_1,L_2,L_3 respectively, and a join of the three relations C⋈A⋈B is to be performed with the following parameters initialized:

Size(C) = T_C = 10, Size(A) = T_A = 1000, Size(C) = T_B = 100.

Image $I_{Ccustid}$ = 10, Image $I_{Acustid}$ = 20, Image I_{Aacno} = 500, Image I_{Bacno} = 25 and c_0 = 10.

Fig 2 gives all the possible ways of computing a three-way join among the relations C,A and B in the form of unordered trees.

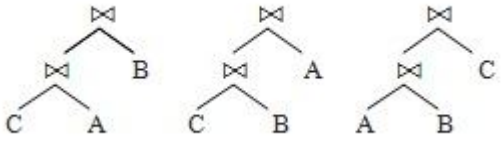

Figure 2 Unordered Tree Structure for 3-way join

We choose the first unordered tree structure to compute the result i.e., calculate C⋈A first and later include B. By considering the rules of R* algorithm, the computation of result at each location is given below:

1. At location L_1, we can either fetch A to L_1 at a cost of c_0+T_A = 10+1000=1010 or perform a lookup of the matching tuples of A with C, at a cost of
   $T_C[c_0+(T_A/I_{Acustid})]$ = 10(10+(1000/20)) = 10(10+50)=600.
   The cost of computing C⋈A at location L_1 will be *min*(1010,600) = 600.
2. Similarly, at location L_2, a similar approach to the above can be followed and the cost of computing the result will be
   min(20,11000) = 20.
3. At location L_3, both the relations C and A are to be fetched at a cost of $2c_0+T_C+T_A$ = 20+10+1000 = 1030.

The new size of the join of T_{CA} = T_C* T_C / max($I_{Ccustid}$, $I_{Acustid}$) = 10*1000/20 = 500 and the image size of C⋈A with the attribute acno will be approximately $3/4^{th}$ of the I_{Aacno}.[] i.e.,

I_{CAacno} = ¾(I_{Aacno}) = ¾(500) = 375.

To compute the result of C⋈A⋈B, the size of the join of three relations will be

T_{CAB} = T_{CA}*T_B / (max (I_{CAacno},I_{Bacno})) = 500 * 100/375 = 133.

Table 2 gives the evaluation strategies involved in calculating C⋈A⋈B in (C⋈A) ⋈B manner. i.e., first by computing (C⋈A) and later joining the result with B. From the table, we can infer that minimum cost of evaluating C⋈A⋈B at location L_1 can be obtained by first computing C⋈A at L_2 and fetching the relations CA, B to the destination. The same inferences can be made at the other locations as well.

The process can be continued by considering other possibilities like joining C with B first and later with A or joining A with B and later with C to obtain minimum cost.

TABLE 2 Total Cost Evaluation Using System R*

C⋈A⋈B site result	C⋈A site result	Approach	Cost (D)	Cost C⋈A (E)	Total Cost (D+E)
L_1	L_1	Fetch B	10+100 = 110	600	710
		Lookup B	500(10+100/25) = 7000	600	7600
	L_2	Fetch CA,B	20+500+100 = 620	20	**640**
	L_3	Fetch CA⋈B	10+133= 143	1030	1173
		Fetch CA, B	20+500+110 = 620	1030	1650
L_2	L_1	Fetch CA, B	20+500+110 = 620	600	1220
	L_2	Fetch B	10+100 = 110	20	**130**
		Lookup B	500(10+100/25) = 7000	20	7020
	L_3	Fetch CA⋈B	10+133 = 143	1030	1173
		Fetch CA, B	20+500+100 = 620	1030	1650
L_3	L_1	Fetch CA	10+500 = 610	600	1210
		Lookup CA	100(10+500/37 5) = 1133	600	1733
	L_2	Fetch CA	10+500 = 510	20	**530**
		Lookup CA	100(10+500/37 5) = 1133	20	1153
	L_3	No Transfer	0	1030	1030

4. SharesSkew Algorithm Using MapReduce

To perform join on two relations R(A,B) and S(B,C) through MapReduce mechanism, several systems implement a two-round algorithm. The first round is to identify the tuples with Heavy Hitters(HH), i.e., same value for a specific attribute and the second round deals with the tuples without HH for the join attribute. The map phase results in producing key-value pairs for each attribute using a hash function[8]. Each key is associated with a reducer, where the tuples are shuffled from mappers to reducers, termed as communication cost.

SharesSkew algorithm is extended from Shares algorithm where the data of the join attribute deal with skews, in the form of heavy hitter(s). The shares algorithm primarily focuses on distributing the data from mappers to *k* reducers. For instance, to perform a 3-way join on R(A,B) ⋈ S(B,C) ⋈T(C,A), a tuple (a,b) of a relation R is sent to $(h_1(a),h_2(b),i)$ reducers. The communication cost will be $r_1x_3+ r_2x_1+ r_3x_2$ with the constraint $x_1x_2x_3=k$, where r_1, r_2, r_3 are the relation sizes and x_1, x_2, x_3 represent shares of the relations respectively. The Cost expression can be minimized using Lagrange's Method[], which results in a minimum communication cost of $3(kr_1r_2r_3)^{1/3}$.

A. Attribute Dominance

If an attribute B appears in all relations where attribute A appears during join operation, then B is said to dominate A. If an attribute is found to be dominated, then its share is treated as 1. For the three relations R(A,B), S(B,C) and T(C,D), if r,s,t are their relation sizes and w,x,y,z are the shares of the attributes respectively, then it is observed that attribute A is dominated by B and D is dominated by C. Hence the share of attributes A and C i.e., w,z will be 1 and the communication cost expression is ry+s+tx with the constraint xy=k.

B. Relation Partition

After obtaining dominating attributes, the next task is to find the HHs for each of the attributes and perform relation partition. For each dominating attribute, partitioning is done by checking whether it has a HH or not. Consider three relations R(A,B) ⋈S(B,E,C) ⋈T(C,D) with attribute B containing 2 HHs b_1,b_2 and attribute C with one HH c_1. Then relation R can be partitioned into 3 pieces i.e., B with b_1, b_2 and others, relation S into 6 pieces and T into 2 pieces. Every partition is named as a Residual Join[2]. The six residual joins possible are given below:

1. All attributes with no heavy hitters (T_). *r* is the tuple count of R with $b\neq b_1$ and $b\neq b_2$, *s* is tuple count of S with $b\neq b_1$, $b\neq b_2$ and $c\neq c_1$ and *t* is the tuple count of T with $c\neq c_1$.
2. Attribute B of type T_{b1} and all other attributes with no heavy hitters (T_). *r* is the tuple count of R with $b=b_1$, *s* is tuple count of S with $b=b_1$ and $c\neq c_1$ and *t* is the tuple count of T with $c\neq c_1$.
3. Attribute B of type T_{b2} and all other attributes with no heavy hitters (T_). *r* is the tuple count of R with $b=b_2$, *s* is tuple count of S with $b=b_2$ and $c\neq c_1$ and *t* is the tuple count of T with $c\neq c_1$.
4. Attribute C of type T_{c1} and all other attributes with no heavy hitters (T_). *r* is the tuple count of R with $b\neq b_1$, $b\neq b_2$, *s* is tuple count of S with $c=c_1$ and *t* is the tuple count of T with $c=c_1$.
5. Attribute B of type T_{b1} and C of type T_{c1} and all other attributes with no heavy hitters (T_). *r* is the tuple count of R with $b=b_1$, *s* is tuple count of S with $b=b_1$ and $c=c_1$ and *t* is the tuple count of T with $c=c_1$.
6. Attribute B of type T_{b2} and C of type T_{c1} and all other attributes with no heavy hitters (T_). *r* is the tuple count of R with $b=b_2$, *s* is

tuple count of S with $b=b_2$ and $c=c_1$ and *t* is the tuple count of T with $c=c_1$.

C. Description of SharesSkew

SharesSkew algorithm follows the below four steps to obtain minimum communication cost during joining of relations with the tuples containing HHs:

Step 1: Form all the possible Residual Joins

Step 2: For every residual join, form a set of keys using a Hash function to compute the share of each attribute in the join.

Step 3: Evaluate the cost expression of each residual join by assigning the share of each attribute with HH equal to one in the generic cost expression.

Step 4: Distribute tuples to the set of keys constructed for every residual join.

D. Evaluation of Cost Function for Residual Joins

For the 3-way join between relations R,S and T given in above example, if r,s,t are the relation sizes and a,b,c,d,e are the shares of the respective attributes, the generic cost expression will be rcde+sad+tabe. The minimum cost expression is to be evaluated for the six residual joins as in below:

1. Since all attributes are normal (no HHs) and attribute A is dominated by B and attributes D and E are dominated by C, the shares of a,d,e is 1. The resultant cost = rc+s+tb.
2. Attribute B has a HH and its share b=1. Attributes D and E are dominated by C, and their shares d,e are also 1. The resultant cost = rc+sa+ta.
3. In 2, B has b_1 as HH and now the HH is b_2, so the resultant cost = rc+sa+ta.
4. Attribute C has a HH and its share c=1. Attributes A and E are dominated by B, and their shares a,e are also 1. The resultant cost = rd+sd+tb.
5. Attributes B and C have HHs and their shares b,c is 1. The resultant cost = rde+sad+tae.
6. The resultant cost is same as that of 5, since B and C have HHs i.e., rde+sad+tae.

E. SharesSkew on Banking System

Consider a bank database consisting of three relations R(A,B), S(B,E,C) and T(C,D) representing Branch(bid,cid), Customer(cid,age,acno) and Account(acno,bal) with the tuples given below. To perform a 3-way join R⋈S⋈T through SharesSkew algorithm, relations with skewness in their tuples is considered as shown in table 3.

TABLE 3 Banking System Relations with Skewed Tuples

R(A,B)

A	B
10	1
11	2
10	**3**
12	**3**
10	**9**
12	**9**

S(B,E,C)

B	E	C
2	24	301
3	27	302
3	24	**300**
3	25	**300**
4	26	**300**
7	25	**300**
3	28	303

T(C,D)

C	D
300	500
300	501
303	600
302	600
301	1000

The join attribute B has two heavy hitters 3 and 9, and C has one heavy hitter 300. A total of six residual joins listed above are possible. Joins are represented from J_1 to J_6.

J_1 considers tuples satisfying B≠3, B≠9 and C≠300. The resultant tuples from table 2 are R(10,1) R(11,2) S(2,24,301) T(303,600) T(302,600) T(301,1000) with relation sizes r=2, s=1, t=3. Attribute shares are b=2 and c=3 with the total cost (rc+s+tb) will be (2*3)+1+(3*2)=13.
J_2 considers R(10,3) R(12,3) S(3,27,302) S(3,28,303) T(303,600) T(302,600) T(301,1000) for B=3, B≠9 and C≠300.

J_4 considers R(10,1) R(11,2) S(4,26,300) S(7,25,300) T(300,500) T(300,501) for B≠3, B≠9 and C=300.
J_5 considers R(10,3) R(12,3) S(3,24,300) S(3,25,300) T(300,500) T(300,501) for B=3, B≠9 and C=300. Joins J_3 and J_6 are not considered for cost evaluation as the size of relation S (s) is zero i.e., there is no value for a join attribute. Table 3 presents share of each attribute and total communication cost computed for all the valid residual joins.

TABLE 3 Total Cost Evaluation Using SharesSkew

Join	Attribute Type	Relation Size	Attribute Share	Cost Expression	Total Cost
J_1	B≠3, B≠9 C≠300	r=2, s=1, t=3	b=2, c=3	rc+s+tb	13
J_2	B=3, B≠9 C≠300	r=2, s=2, t=3	a=2, c=3	rc+sa+ra	16
J_4	B≠3, B≠9 C=300	r=2, s=2, t=2	b=4, c=2	rd+sd+tb	16
J_6	B=3, B≠9 C=300	r=2, s=2, t=2	a=2, d=2, e=3	rde+sad+tae	24

Conclusions

In this paper, we evaluated the communication cost of joining relations at different sites using the System R* algorithm when the data is non-skewed. By considering three Banking system relations Customer, Account and Balance at three locations, we have calculated total cost and showcased the optimized cost among various possible cases. For the data with heavy hitters (skewed), we used SharesSkew algorithm using MapReduce mechanism. The same Banking System relations with skewed data is considered. The total multi-way join is partitioned into several residual joins depending on the shares of the attributes and the total cost is evaluated.

References

[1] Ullman D.J., "Principles of Database Systems", Second Edition, Galgotia Publications, 1984.

[2] Afrati N.F., Stasinopoulos N and Ullman D.J., "SharesSkew: An Algorithm to handle skew for joins in MapReduce", Information Sytems 77, Elsevier, pp. 129-150, 2018.

[3] Afrati N.F and and Ullman D.J., "Optimizing Multiway Joins in Map-Reduce Environment", IEEE Transactions on Knowledge and Data Engineering, 23(9), pp. 1282-1298, 2011.

[4] Beame P., Koutris P and Suciu D., "Skew in Parallel Query Processing", Proc. 33rd ACM SIGMOD Symposium on Principle of Database Systems, USA, June 22-27, pp. 212-223, 2014.

[5] Chu. S, Balazinska M and Suciu D., "From Theory to Practice: Efficient Join Query Evaluation in a Parallel Database System", Proc. 2015 ACM SIDMOD Int'l Conf. on Management of Data, SIGMOD, ACM, 2015.

[6] Kwon Y., Balazinska M., Howe B and Rolia J., "SkewTune: Mitigating Skew in Mapreduce Applications", Proc. 2012 ACM SIDMOD Int'l Conf. on Management of Data, SIGMOD, pp. 25-36, ACM, USA, 2012.

[7] Kwon Y., Balazinska M., Howe B and Rolia J., "SkewTune in Action: Mitigating Skew in Mapreduce Applications", PVLDB, 5(12), pp. 1934-1937, 2012.

[8] Ullman D.J., "Designing Good MapReduce Algorithms", XRDS, 19(1), pp. 30-34, September 2012.