International Journal of Advances in Engineering Research http://www.ijaer.com
(IJAER) 2012, Vol. No. 3, Issue No. VI, June **ISSN: 2231-5152**

EFFICIENT ALGORITHMS FOR DISTRIBUTED QUERIES

Radha Krishna Rambola , Dr. Nesar Ahmad & Alok Deepak
The Tilka Manjhi Bhagalpur University, Bhagalpur
[Email: rambola@rediffmail.com, nesar_bgp@yahoo.co.in,alokdeepak78@rediffmail.com]

ABSTRACT

The efficiency of processing strategies for queries in a distributed database is critical for system performance. Methods are studied to minimize the response time and total time for distributed queries. a new algorithms is presented to derive processing strategy for, simple / complex queries .
The algorithms comprises of four aspects to minimize response time and total time ,these algorithms may derive optimal solution in all conditions. The proposed algorithms consist of query scheduling , two level indexing of databases in the distributed environments , appropriate logical and relational operation to get the desired information ,appropriate management and administration .
Keywords:- *computer network , database ,distributed database system , distributed processing strategy , heuristics algorithms , query processing , relational data model, system modeling.*

INTRODUCTION

In a distributed database , we have the ability to decentralized data that are most heavily used by end users at geographically dispersed location and , at the same time , to combine data from different source by means of queries . The decentralization of data will result in better response times and, if multiple copies are used , in a more reliable system. The retrieval of data from different sites in a network is known as distributed query processing .the difference between query processing in centralized database and a distributed database is the potential for decomposing a query into sub queries which can be processed in parallel , and their intermediate result can be sent in parallel to the required computer . Finding an efficient way of processing a query is processed inefficiently , it is not only takes long time before the end user gets his answer ,but it might also decrease the performance of the whole system because of network congestion . We will investigate two optimization objectives: the minimization of response time and of total time. Which of these two objectives is better for a specific system dependence upon the system characteristics?

Distributed query processing has received a great deal of attention [15], [19]. The initial research in this area was done by Wong [24]. He proposed an optimization method based on greedy heuristic that produces efficient , but not necessarily optimal query processing

strategy . An enhance version of this method is implemented in the SDD-1 system [5] . Epstein ET. Al [9] developed an algorithms based on the query optimization technique of decomposition [23]. This algorithm is implemented in distributed INGRESS database system. Performance studies of these algorithms are reported in [10]. a dynamic optimization algorithms for the POLYPHEM E system has been proposed by Toan [17] further research on dynamic optimization algorithms has been done by Work by chu and hurley [7] define the solution space of feasible processing strategy for distributed queries . they presented an optimal ,although inherently exponential , optimization algorithms .the use of semi join operation has led to the development of full reduction method of processing a distributed query [6],[11],[25] ,[4] . These methods are applicable for special class of queries known as tree queries. Pelagatti and Schreiber [18] use an integer programming technique to minimize cost in distributed query processing. Kershberg ET. Al [16] apply query optimization to a database allocated on a star network and study the system performance.
For a special class of simple queries , hevner and yao developed algorithms parallel and serial [12] that find strategies with , respectively , minimum response time and total time . In [14], they extended these algorithms to algorithms G that processes general distributed queries. Apers [1] showed that these algorithms had some serious drawback. its complexity for worst case queries does not have a polynomial bound .also ,the analysis of the quality of the derived processing strategies is difficult, both hevner [13] and apers [2] recognize these problems and developed improved algorithms .

In this paper we have presented the new approach to overcome the time latency factor

This paper is organized as follows. in section II , we will briefly repeat the query processing model described in [13], three version of algorithms ,for response time and total time ,are presented and analyzed in section III.

DEFINITION AND DISTRIBUTED SYSTEM MODEL

A distributed database system is characterized by the distribution of the data. for this research , a distributed system is a collection of independent computer interconnected vi point to point communication lines. Each computer , known as a node in the network , has a processing capability , a data storage capability , a data storage capability , and is capable of operating autonomously in the system. Each node contains a version of a distributed DBMS.
To optimize the time latency in data access from data bases, it is proposed to maintain the two levels of indexes in virtual memory as per the need of access requirement, which is over and above conventional data base access strategies. The algorithms for the same are also presented and its performance is also compared.
The database is viewed logically in the relational model [8]. the database is allocated across system nodes in units of relation (without loss of generality ,relation fragments may be consider ed as relation for distribution). The relation distribution allows a general manner of

redundancy. the only allocation constraint is that all data must be either locally or globally accessible from any system node . the distribution of data on the network is invisible to the user.

We assume that we know the following information about for relation.

For each relation Ri,i=1,2,3,…….,m,

ni number of tuples

α number of attribute

si size (e.g. in byte)

for each attribute dij, j=1,2,3…….,αi of relation Ri

pij: selectivity

bij size (e.g. in byte) of the data item in attribute dij the selectivity pij of attribute dij is defined as the number of different values occurring in the attribute divided by the number of all possible values of the attribute . Thus , $0 \leq pij \leq 1$.

The process a query in a relational database , we only need the operations restriction , projection , and join [8] . in a distributed database , we may need the operation restriction ,projection, and join [8] in distributed database , we may need to compute joins on relation which are located at different sites . instead of computing these joins immediately , we will first reduce the size of the relation wherever possible by restriction and projections. A relation ,which is one of the operands of a join , can be made smaller by deleting the tuples that can not play a important role in join . an operation called semi join [4] delete these tuples of the relation .if relation Ri has a semi join with attribute dki on attribute dij then the parameters of a relation Ri are changed in the following way:

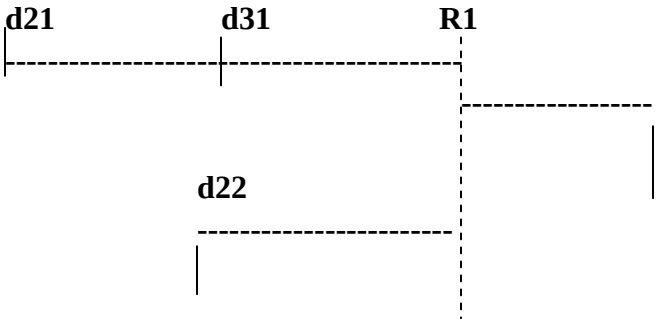
Si ←si*pkl,

Pij ←pij*pkl,

bij ←bij*pkl,

the selectivity and size of only the joining attribute are reduce because of an assumption of attribute independence within each relation in our model.

To compute a semi join ,the unique values of joining attribute of one relation are sent to the other relation it is cheaper to compute this semi join will be computed after the reduce relations have arrived by concatenating matching tuples on the joining attributes . the data transmission of the reduce relation to the query computer form a scheduler for this relation . an example of a scheduler for relation Ri can be seen below.



Attribute d21 is sent to attribute d31. A semi join is performed on relation R3. The reduce d31 attribute is sent to relation R1 in parallel with attribute d22. Further relational operations reduce the size of relation R1. Finally, the reduced relation R1 is sent to the result node

It is also possible to construct a schedule for an attribute instead of a relation .A schedule for an attribute is constructed to send the attribute to a relation upon which a semi join will be computed for example , attribute d31 in the above schedule.

We assume that the transmission cost of a the data is the same between any two computer and is a linear function of the size of data . this function will be denoted by $c(X) \leftarrow co+c_1X$ where X is the amount of data transmitted . the response time of a schedule is the time elapsed between the start of the first transmission and the time at which the relation arrive at the required c attribute relation is the minimum response time among all possible schedules for this relation . the total time of a schedule is the sum of the costs of all transmissions required in the schedule.

The incoming selectivity of a schedule for a relation is the product of selectivity of all the attribute in the schedule excluding the attribute of the relation .

A distribution strategy for a query consists of the schedule for all the relations which do not reside in the result node and are used in the query . with the assumption that data transmission cost are significantly greater than the local processing cost in the system ,the cost of processing a query is determined by the transmission cost in the distributed strategy .another implicit assumption that we make through out this paper is that the query processing strategy is run on the a dedicated system in order to achieve minimum execution times. dynamic system factors such as communication line contention and subsequent queuing delay are not considered in our static query optimization algorithms.

In distributed database , it is ,in general , better to do local processing first because it reduces the amount of data to be transmitted . with local processing , we mean the computation of restriction , projection and semi-join between relations that reside in the same node . after the initial processing each node that has query data will be consider to contain only one integrated relation . The relation at each node remains distinct (i.e. No Cartesian product relation is formed). However ,by reformatting the query so that each node becomes a variable ,the distribution aspect of the query are emphasized [24].without loss of generality , this assumption provides a distribution abstraction to the problem and simplifies the understanding of our following algorithms

Initial local processing result in the following parameter:

M: number of relation in reaming queries

α_i number of attribute in relation Ri

β number of inter nodal joining attribute in relation Ri

in [12] and [14] ,hevner and yao introduced and investigated algorithms, parallel and serial, which respectively ,compute minimum response and total time schedule for simple queries .queries are called simple if , after initial local processing , the relation contain only one attribute: the joining attribute thus , $\alpha_{i=}\ \beta_{i=1}$ for i =1,2,…….,m.

Algorithms parallel
1) Order relation Ri such that s1≤s2≤……≤sm
2) Consider each relation R_i in ascending order of size
3) for each relation R_j(j<i) , construct a schedule to R_i that consist of the parallel transmission of the relation R_j and all schedules of relation R_k(k<j).select the schedule with minimum response time

ALGORITHMS SERIAL
1) Order relation R_i such that s1≤s2≤…..≤sm.
2) if no relations are at the result node, then select strategy
R1→ R2→…….Rn → result node
Or else if Rr is a relation at the result node ,then select strategy:
R1→ R2→…….Rr→…….Rn → or
R1→ R2→…….Rr-1 →Rr+1→…Rn → Rr
Select the one with minimum total time .the complexity of algorithms parallel is $O(m^2)$ and that of algorithms serial is $O(m \log_{g2} m)$ where m is the number of required relation in the query [14]

ALGORITHMS GENERAL

A general query is characterized by $\alpha_{i\geq} \beta_{i\geq 1}$ for i=1 ,2,3……,m . This means that a relation can contain more than one joining attribute. let α represent the number of joining attribute in a query . Therefore , such a relation can be reducing in size by semi-join on different joining attributes. To illustrate our query optimization methods, we will use a database consisting of the following relations:
PARTS (P#, PNAME)
ON-ORDER(S#, P#, QTY)
S-P-J(S#, P#, J#).
The query represented by fig 1 is " list the p# ,pname and total quantity for all parts that are currently on order from suppliers who supply that part to job 10 to 20 ."
In this query , there are two joining attribute , p# and s#. Assume that each relation is located at different node and that the result is required at fourth node. After performing the restriction on s-p-j relation, the required attribute in each relation are projected. The resulting size and selectivity parameters are given in table I
Let C(X) = 20+X
The response time and total time costs of different query processing strategy for this query will be compared for the response time total time versions algorithms general.
A simple way of processing a query is to perform initial local processing , followed directly by the transmission of all remaining data to the result . this will be called the initial feasible solution(IFS)

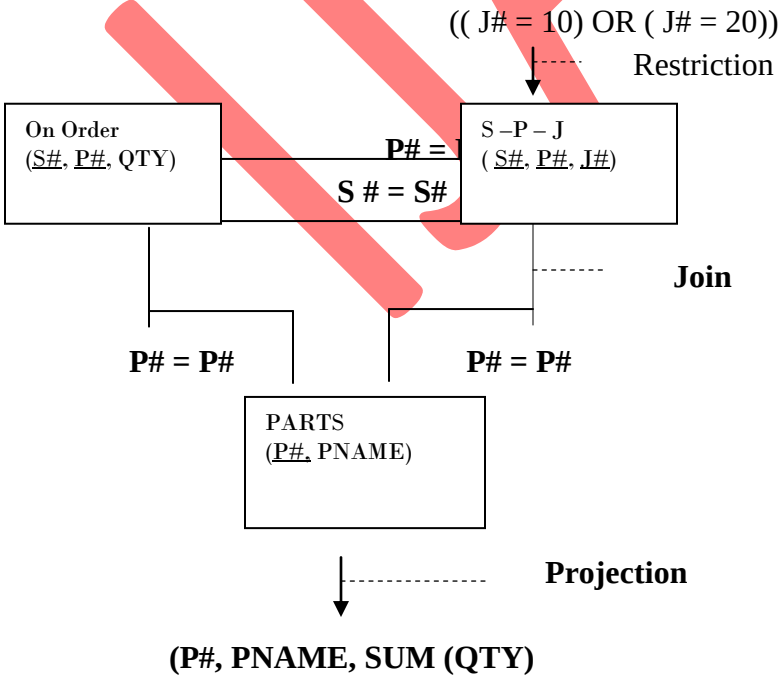
R1 1020
R1: ON-ORDER | ----------
2020
R2: S-P-J | ----------
3020
R1: ON-ORDER | ----------

Response time =3020 .Total Time=6060

We now present an outline of algorithms general. minimization of response time and total time is done by three different version of the algorithms , which are discussed in the next three sections.

Algorithms general

1) Do all initial local processing and two level indexing are to be
2) Generate candidate relation schedule . Isolate each of the joining attribute, and consider each to define a simple query with an undefined result node.
a) To minimize response time, apply algorithms parallel to each simple query. Save all candidate schedules for integration in step 3
b)to further minimize response time are any other kind of bottleneck the monitoring and synchronization aspects have been introduced to derive the optimal performance as well as minimize the response time.
The monitoring and synchronization strategy based on message passing approach in all queries in a distributed database environments



Relation Ri	Size Si	bi1	pi1	bi2	pi2
R1:on order	1000	400	0.4	100	0.2
R2:s-p-j	2000	400	0.4	450	0.9
R3:parts	3000	900	0.9	-	-

c) To minimize total time, apply algorithms serial to each simple query. This result is one schedule per simple query. From these schedules, the candidate schedule for each joining attribute is extracted. Consider joining attribute dij. Its candidate schedule is identical to the schedule produced by algorithms serial, applied to simple query in which dij occur , up to the transmission of dij. All transmission after that is deleted from the schedule.

3).integrate the candidate schedule : for each relation Ri , the candidate schedule are integrated to form a processing schedule for Ri . The integration is done by the procedure response for response time minimization and by procedure total or procedure collective for total time minimization.

4).Remove schedule redundancies: eliminate relation schedule for relations which have been transmitted in the schedule of another relation.

Algorithms general derive query processing strategy for either response time or total time minimization by using the procedure response, total and collective.

A. Response Time version

To minimize the response time of a relation Rk , we have to test whether transmitting an attribute dij, to Rk is cost beneficial . therefore we ,have to know how long it takes to get dij to the site where R_k is located . Algorithms parallel derive minimum response time schedule for joining attributes. In fact, there is no procedural difference between computing the minimum response time schedule for joining attributes. whether they are sent to a result node or the node where Rk is located . therefore , in step2 of algorithms general, algorithms parallel is applied to each joining attribute for each . All these candidate schedule are saved for integration by procedure response

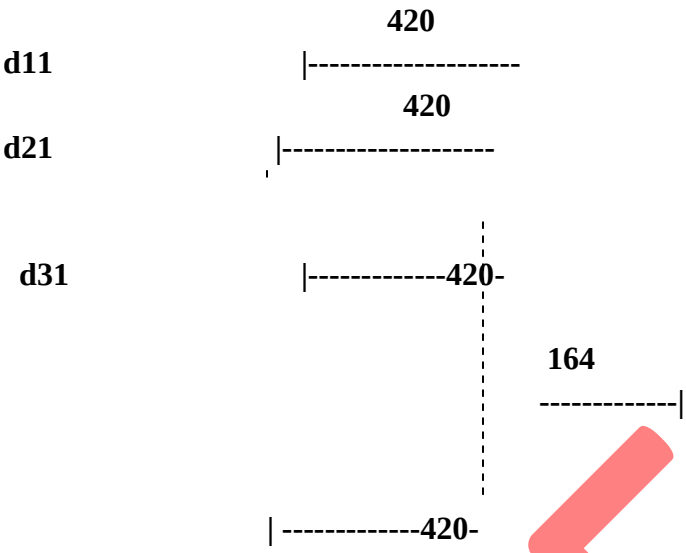
Procedure Response

1) candidate α schedule ordering : for each relation Ri, order the candidate schedule on joining attribute dig=1,2……., α in ascending order of arrival time. Let ARTl denote the arrival time of candidate schedule cache- (for the dij joining attribute not in Ri, disregard the corresponding candidate date schedule)

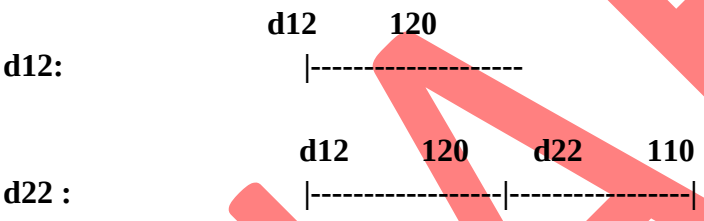
2). Schedule integration for each candidate schedule Cache in ascending order , construct an integrated schedule for Ri that consist of the parallel transmission of Cache and all CSCHk with k<l . Select the integrated schedule with minimum response time.

Applying algorithms general to the previous example ,after the initial processing , two simple queries are formed , one having dij = P# as a common joining attribute and the

other having d12=S3 . Running algorithms parallel on the P# query , the resulting candidate schedule are



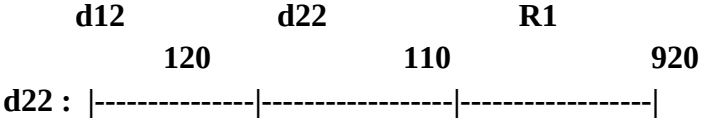
Applying algorithms PARRELEL to the S# query, the candidate schedule are



the construction of the schedule for R1 will be given in detail .in step 1 procedure Response , the schedule of attribute that can be applied to relation R1 are ordered on their arrival time in the node where R1 is located . this gives the following result

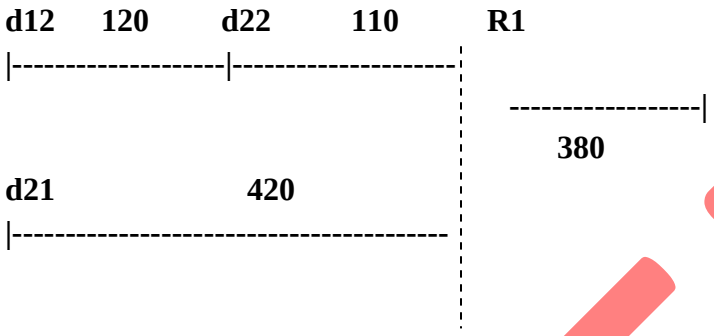
Attribute(dk)	Arrival Time ART_k
d22	330
d21	420
d31	584

In step 2 , for each of these attributes dk ,an integrated schedule for R1 is constructed , constructed , consisting of the parallel transmission of all attribute having an arrival time less than or equal to ARTk. The following three integrated schedule are.



Response time=C(100)+C(0.2*450)+C(0.9*1000)
=120+110+920
=1150

d21:



Response Time= C(400)+C(0.9*0.4*1000)
=420+380
=800

d31:

Response time=C(400)+C(0.4*0.4*900)+C(0.9*0.4*0.9*1000)
=420+164+344

From these three schedules for R1 and the initial feasible solution for R1, the schedule with the minimum response time is chosen. This is the second schedule and its response time is 800

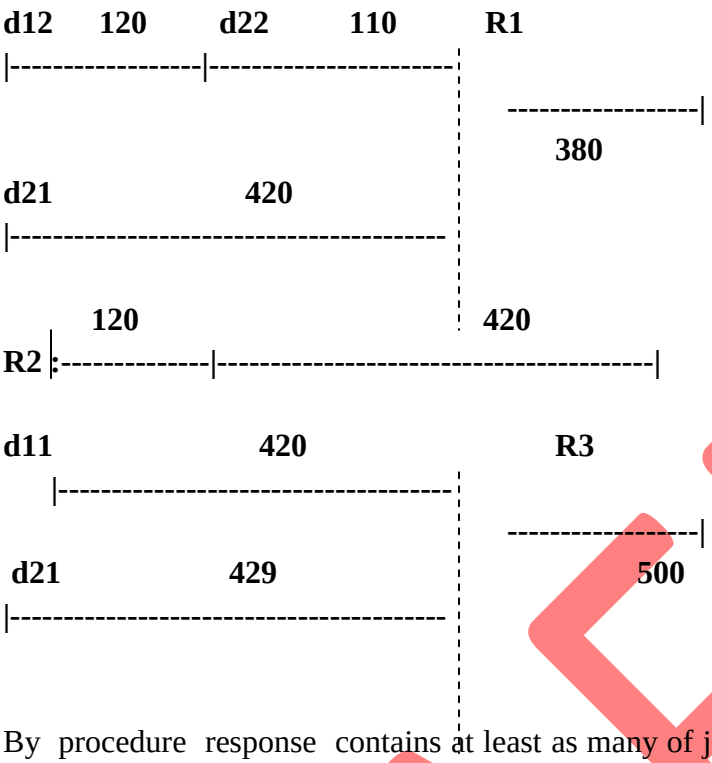
The schedule for R2 and R3 are constructed in a similar way. Algorithms general (response time) query processing strategy for the example query is a compression to the response time of the IFS shows a considerable cost reduction
Procedure RESPONSE is given the minimum response time schedule of all joining attributes. . In step 2 of algorithms general algorithms parallel is applied to different simple queries. therefore the minimum response time schedule for one common joining attribute are given in order of response time . this means that putting the candidate schedule in order of arrival time will take at most O(αm log2 α)
Which is the merging complexity?
In step 2 of algorithms GENERAL algorithms General , algorithms Parallel is applied times and its complexity is o(m2).however the cost of applying procedure response for every relation Ri is O((αm ^{2}log2 α) . therefore ,the complexity of algorithms general is O((αm ^{2}log2 α)

Now we will investigate the quality of the derived schedule the schedule produce by algorithms parallel for the joining attribute have a minimum response time .from this fact , we now can prove that each relation schedule has minimum response time and ,consequently that the total query processing strategy has a minimum response time . Implicit within our proofs, we require the model assumption of attribute independence within each relation. Initial studies indicate that the effect of this assumption on the performance of the resulting query processing strategy may not be significant.

Theorem 1 procedure RESPONSE derive a minimum response time integrated schedule for Ri

Proof: the candidate schedule used in procedure response is all minimum response time schedule because of the optimality of algorithms parrelel [13] . procedure response puts these candidate schedule in ascending order of arrival time less than or equal to the arrival time of a certain CSCHk. We will show that no other integrated schedules need to be considered.

Assume that we are given a minimum response time schedule for Ri this schedule contain the transmission of some joining attribute which arrive last (i.e. has the greatest value of ARTk in the schedule). The corresponding integrated schedule (based on ARTk as the largest ART value) considered by procedure response contain at least R1: ON-ORDER



By procedure response contains at least as many of joiningattribut as the previous schedule and therefore , its selectivity is at least as small. Hence its response time must be less than or equal to that of the minimum response time schedule.

Theorem 2: algorithms general (response time) derives a minimum response time processing strategy for any distributed query.

Proof: the response time of a processing strategy is the maximum response time of the relation schedule in the strategy. Theorem 1 showed that these relation schedule have minimum response time. Hence the theorem follows.

B. total time version

The motivation for using the minimization of the total time as the objective of algorithms general comes from the use of the algorithms in a multiprocessing environments . Minimizing response time leads to an increased number of parrel data transmissions in the query processing strategy. in multi processing system under moderate to heavy loads ,these extra transmission may lead to significant queuing delays and synchronization delays , delays which may cause poor query response time . by minimizing the total time in a query processing strategy ,fewer transmission will be included and improved actual response times may result in certain system environments.

The candidate schedule produced in step 2 of algorithms general (total time) look very much like the schedules produced by algorithms serial . Algorithms serial produce minimum total time schedule for simple queries. In its optimally proof [14] , it is shown that parallel

transmission of common joining attribute can be avoided . Therefore , we will do the same in algorithm general (total time). Only parrelel transmission of different joining attribute is allowed. this means that in constructing a schedule for relation Ri , we will consider only one candidate schedule per joining attribute is considered .

In [1] , it was shown that it is not sufficient to just look at the candidate schedule produced in step 2 of algorithms general. For every candidate schedule to relation Ri containing a transmission of a joining attribute from the same relation Ri we have to add another candidate schedule , namely ,one without the transmission of this joining attribute. This is necessary since the size of relation Ri cannot be reduce by the selectivity of its own attribute. Thus, this data transmission in the incoming schedule may not be cost beneficial. among this extended set of candidate schedules ,we select the schedule which minimize the total time of transmitting Ri if only one joining attribute is considered . This selected schedule for relation Ri considering joining attribute dij will be called BEST ij.

We define SLT ij to be the accumulated attribute n selectivity of the BEST ij candidate schedule into Ri. Note that j may take on values from 1 to only if the common joining attribute dij appear in Ri. Procedure Total

1) Adding candidate schedules. For each relation Ri and each candidate schedule Cache, do the following

If this schedule contain a transmission of joining attribute of Ri , say dij ,then add another candidate schedule which is the same as Cache except that the transmission of dij is deleted .

2. Select the best candidate schedule. for each relation Ri and for each joining attribute dij(j=1,2,3……, α) select the candidate schedule which minimize total time for transmitting Ri if only the joining attribute are considered which can be joined with dij

3.candidate schedule ordering .for each relation Ri order the candidate schedule Best ij on joining attributes

Dij,j=1,2,…,α so that ART i1+c(s1*SLTi1)≤…. ≤ARTi α +C(si*SLT i α)(for the joining attribute not in Ri, disregard Best ij) ART ij denotes the arrival time of the Best ij schedule.

4). Schedule integration : for each best ij in ascending order of j , construct an integrated schedule to Ri that consist of the parallel transmission of candidate schedule Best ij and all schedule Bestik where K<j . Select the integrated schedule that result in the minimum total values

$$TOTTi=\sum_{K=1}^{I}\left[ARTik+C(si*\prod_{k=1}^{j}SLTik)\right]$$

Applying algorithms general (total time) to the example , two simple queries are formed on the following joining attributes , di1 = P# and di2=S# in the example query .in step 2 of algorithms general (total time) , the following serial candidate schedule are formed

For P#

```
        d11       420
d11:    |-----------|
        d11    420            180
d21     |------------|--------------|
        d11    420            180          164
d31     |------------|--------------|--------------|
```

For S #

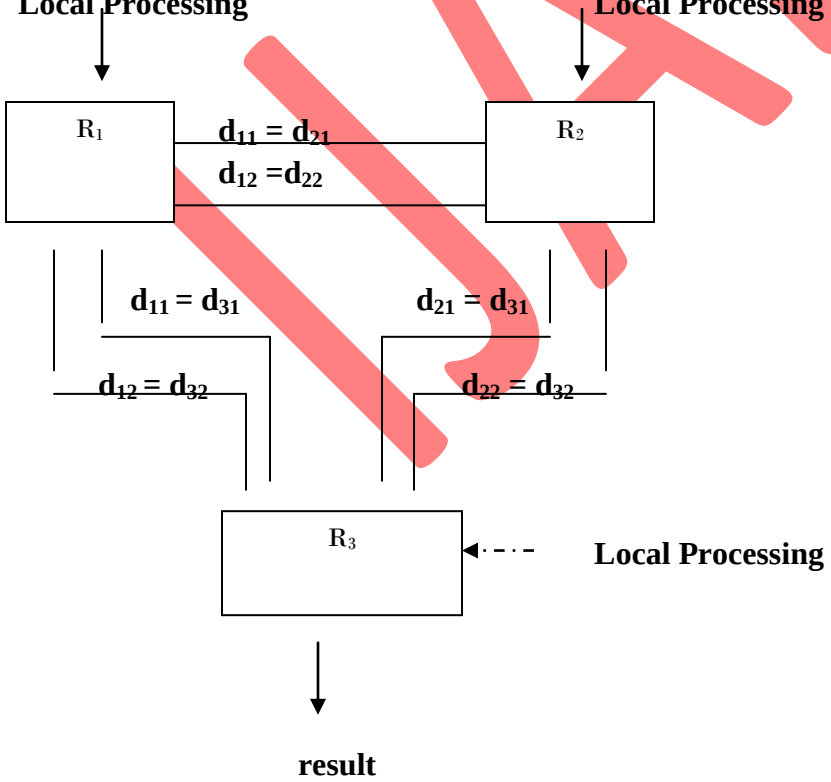
```
        d12   120
d12:    |-----------|

        d12    120   d22        110
d22     |------------|--------------|
```

Again the construction of the schedule for r1 will be discussed in detail. We will treat the two attribute of R1 in turn.

Attribute d11

In step 1 pf procedure total , the following two schedules are added to the above schedule for P# . the first b one is obtained by deleting the transmission od d11 from the schedule for d21 ,and the second one by deleting d11 from the schedule for d31:



d21: d21 420
|----------|

d21 420 380
d31 |----------|----------|

Thus ,there are five schedule that only contain transmission of the P# attribute . Four of them will be tested . The transmission of d11 to R1 is meaningless as a schedule.

Relation	Size (si)	bi1	pi1	bi1	pi2
R1	1000	300	0.6	300	0.25
R2	2000	100	0.2	900	0.75
R3	2000	400	0.8	600	0.5

D21:
d11 420 d21 180 R1 420
|----------|---------|----------|

Total time =C(400)+C(0.4*400)+C(0.4*1000)
= 420+180+420
= 1020

d31:
d11 420 d21 180 d31 164 R1 380
|----------|----------|---------|----------|

Total time= C(400)+C(0.4*400)+C(0.4*0.4*900)+ C(0.4*0.9*1000)
=420+180+164+380
=1144

$d^1$21: 420 R1 420
|----------|----------|

Total time= C(400)+C(0.4*400
= 420+420
=840

$D^1$31:
d21 420 d31 380 R1 380
|----------|----------|----------|

Total time=C(400)+C(0.4*400)+C(0.4*900)+ C(0.4*0.9*1000)
=420+380+380
=1180

Because the schedule of $d^1$21 has the smallest total time ,it is chosen as the Best 11 schedule.

Attribute d12

In step 2 of procedure TOTAL ,only one schedule is added to the above schedule for attribute S#.

d22 470

$d^1$22: |----------|

Each of the schedule for s# is applied to R1 .Again , transmitting d12 to R1 is not considered .

d12 120 d22 110 R1 920

|----------|----------|----------|

Total time =C(100)+C(0.2*450)+C(0.9*1000)

=120+110+920

=1150

$D^1$22:

d22 470 R1 920

|----------|----------|

Total time= C(450)+C(0.9*1000)

=470+920

=1390

Because the schedule of d22 has the smallest total time, it is chosen as the Best 12 schedule In step 3 and 4 of procedure TOTAL , the above obtained Best1j schedule are ordered on their total time ,and the following two integrated schedule are constructed :

d21 420 R1 420

|----------|----------|

d21 R1

|----------

----------|

d12 120 d22 110 380

|----------|----------

The first of these two has the smallest total time ,it is chosen as the solution of algorithms general for R1.

The schedule for R2 and R3 are constructed in a similar way .the algorithms general (total time) query processing strategy for the example query is

R1:ON_ORDER

d21 420 R1 420
|----------|----------|

R2: S-P-J
d12 120 R2 420
|----------|----------|

R3: PARTS
d11 420 d21 R3 180 500
|----------|---------|----------|

Response time =1100, Total Time=2480

The total time of this strategy is considerably smaller than the total time of the IFS . also the response time of its strategy for the example query is not much larger than the response time of the strategy produced by the response time version of algorithms general , although it only tries to minimize total time .
Algorithms general (total time) has a slightly better worst case complexity than the response time version ( ).again ,assume that a general query require data from m relations , and all m relation are joined on joining attribute . in step 2 ,algorithms serial is applied to each simple query . the complexity of this is O( ) because the joining attribute have to be ordered by size.
The complexity of the procedure Total is ( ) . in steps 1 no more than O( ) candidate schedule are added . this means that for every relation , the procedure has to determine the Best ij schedule among O( ) candidate schedules .hence the cost of step 2 is O( ) .this means that for an arbitrary general distributed query , algorithms general (total time) has a [processing complexity no worse than O( ).
The quality of the resulting query processing strategy is much harder to analyze than those for minimizing response time . the Bestij schedules were shown to be the best possible schedule for minimizing the total transmission time of relation Ri if only one common joining attribute is considered [1] .however , the optimality is lost during the integration of the different Best ij schedules. In [13] , it was shown that finding the

minimum total time schedule is equivalent to a problem which is proved to be NP - hard.

C. Handling redundant data transmissions

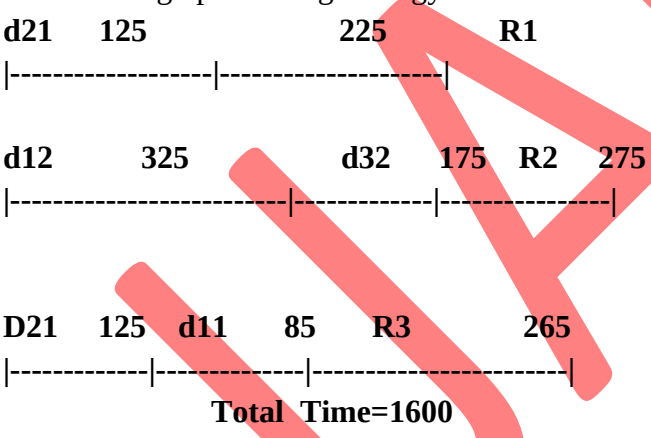
A major reason why the query processing strategy derived by the algorithms GENERAL (total time) is not optimal is that procedure Total does not consider the existence of redundant data transmission in separate relation schedules .(note that such redundant transmission between relation schedules do not effect the minimum response time strategy of section III-A). this is because algorithms general (response time) minimizing the response time of each relation schedule separately

This example illustrates the benefit of redundant data transmission . a database contains three relations R1,R2, and R3. the query represented by fig 2 is entered in a distributed database system wherein each relation is located at a separate node and the result of the query is required at a different node.

The size and selectivity's for each dij , after local processing , are given in table II. Assume the cost function for the system to be c(x)= 25+x

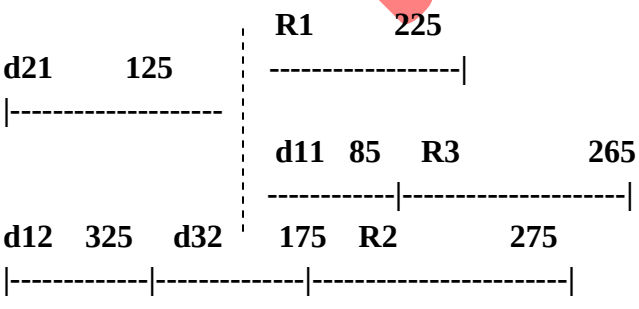
By applying algorithms general (total time) to this example query (the derivation appears in the appendix)

The resulting processing strategy is



Total Time=1600

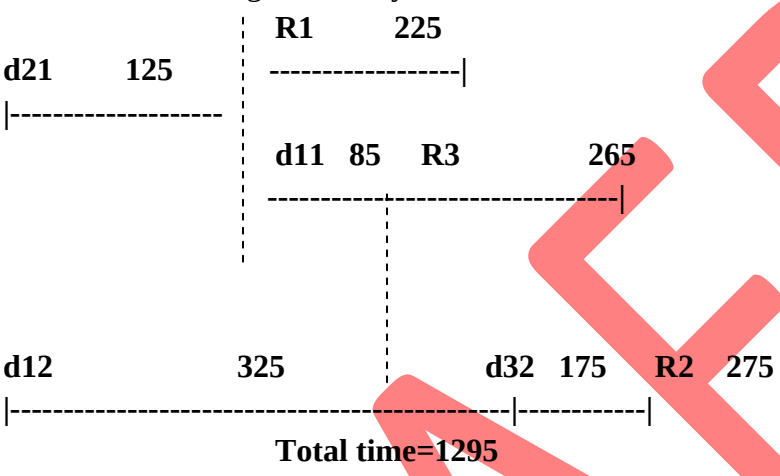
The first redundant transmission one can clearly recognize is the transmission of the attribute d21 towards R1 .this transmission will physically take place only once , so its cost (125) should be accounted for only once as well . we might visualize this removal as follows



Total Time=1475

Apart from this type of direct transmission redundancy , it is possible to discern a somewhat more complicated type .it might happen that the transmission of dij towards the node of Rk is part of some schedule , while it does not occur in the schedule of Rk itself .Because the value of this attribute dij are available for a semi-join will actually take place ,and that the schedule must be synchronized . this could lead to an increased response time . in our example , we can also detect such case.

In the R2 schedule , d12 is sent to d32 . so d12 is available for R3 ,although it is not part of the R3 schedule . the final transmission of R3 will only cost c(0.6*0.25*0.2*2000) =85 instead of 265 .again we try to illustrate this :



Total time=1295

(Note that r3 can be sent at time t= 325, instead of at time t=210)

As these aspects are not considered during the selection of the final schedule, a few beneficial strategy will not be found by algorithms general (total time) this is mainly due to the characteristics that a best schedule is separately chosen for each relation ,without regarding the collective benefit . it means that for some Rk , a schedule might be just too expensive and thus rejected , while it had many transmission in common with selected schedule of other relations ,and thus would have been a very good choice for the strategy as a whole .

In order to test the above statements , we developed an alternative version of algorithms general (total time)called algorithms general (collective),completely based on redundant transmission .it is simpler in that it constructs only one basic strategy for the entire query , after which a few variation are tried out. The schedule of basic strategy includes as many semi- join as possible for each relation. This tends towards a large number of redundant transmissions. the investments for such an extended integrated schedule are shared ,as it were , by all Rk.the strategy is then perturbed by subsequently trying to drop schedule components (i.e. linear part of an integrated schedule for some actually stands for the semi-join on one particular attribute ,from the entire strategy .

Algorithms General (collective) perform step 2b) in algorithms general , the candidate schedule integration is performed in step 3 by the following procedure.

Procedure collective

1) Select candidate schedule. for each relation Ri and for each joining attribute dij(j=1,2,3…….) , select the minimum cost candidate schedule that contain the transmission of all components of attribute j with selectivity <1

2).build processing strategy .for each relation Ri define the schedule to be the parallel transmission of all dij candidate schedule to Ri

3) test variation of strategy .using a removal heuristic , derive new strategy by removing the most costly data transmission . Compare the total time cost of the new and old strategy. Maintain the less costly strategy .continue testing until no cost benefit can be obtained.

Consider the state of relation R1, R2 ,and R3 in a table III after local processing for the query in fig .3 the transmission cost function is c(x)=10+x

The schedule components for the three attribute are

d31 160 d21 70
|----------|----------|
d32 190 d22 154
|----------|----------|
d13 40 d23 19
|----------|----------|

Leading to the basic strategy (integration as much as possible)

d13 40 d23 19 R1 310
|----------|----------|----------|

d31 160 R2
|----------

d32 190 55
|----------|----------|

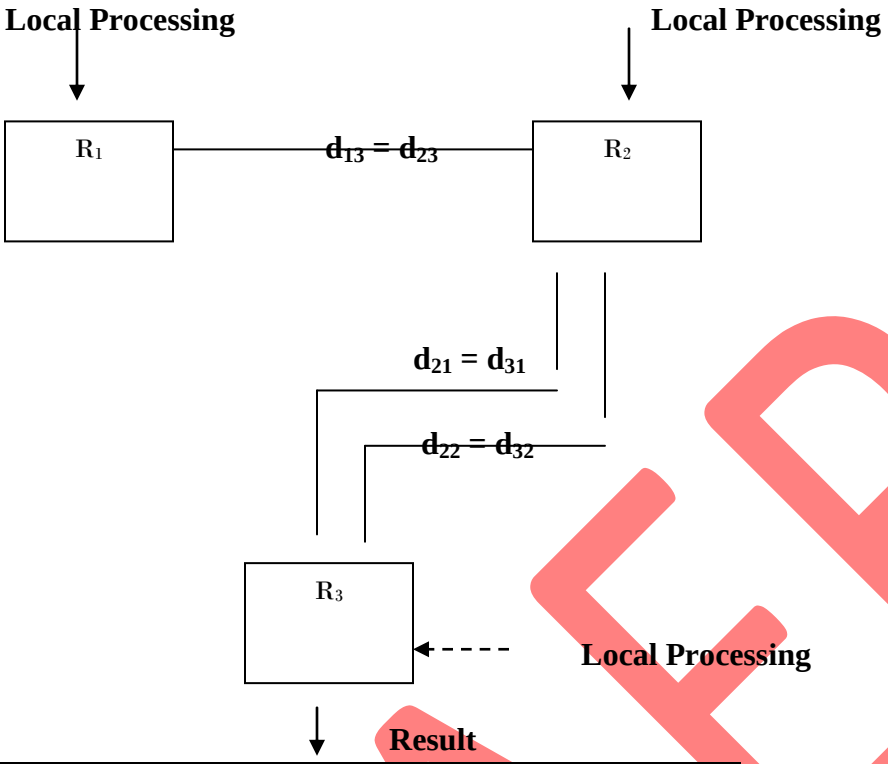
d13 **
|----------

d21 R1
|----------

----------|

d32 ** d22 154 410
|----------|----------

Total time=1408



Relation Ri	Size si	bi1	si1	bi2	si2	bi3	si3
R1	1000	-	-	-	-	30	0.1
R2	2500	200	0.4	240	0.8	90	0.3
R3	1250	150	0.3	180	0.6	-	-

Here (** stand for transmission , which are already accounted for)

During the variation trials , it turn out that it is beneficial to drop the component of attribute 2 from the schedule for R3 . this saves 154 time unit ,while the cost for the transmission of R3 will increase : c(0.4 *1250)= 510. Together this means a gain of 54. in the second variation round , it is found that it better to leave out the component of attribute 2 at all relations.

Thus , the final solution of the collective algorithms for this example is

d13 40 d23 19 R1 310
|----------|----------|----------|

d31 160 R2
|----------

----------|

85

d21 **
|----------

D31 ** d21 70 R3 510
|----------|----------|----------|

Total Time=1194

Algorithms General (collective) has the same worst case complexity of procedure collective is O(αm2) . in step 1 , for every relation (m possibilities) and for every joining attribute (α possibilities), the desired candidate schedule must be selected from m possible schedule . the optimization of step 3 is a greedy heuristics procedure with linear order.

For a significant number of queries , the collective version of algorithms General will produce a smaller total time strategy than will the total time version . such improvement is due to the reorganization and inclusion of redundant data transmission among separate relation schedule in the overall query processing strategy .

CONCLUSION

We claim algorithms General to be an efficient algorithms of polynomial complexity that derive close to optimal query processing strategy on distributed system . the algorithms was designed as a straightforward extension of the processing tactics found optimal for simple queries in algorithms parallel and algorithms serial.

There are two primary version of algorithms general to minimize response time of a processing strategy ,parallel data transmission are emphasized by the use of algorithms parallel and procedure response .algorithms general (total time) can be proved to derive minimum response time strategy under the assumption of attribute independence within query relation . to minimize the total time of a processing strategy ,serial time transmission are emphasized by the use of algorithms serial and procedure TOTAL in algorithms general (total time)

Reorganization the existence of identical data transmission in different relation schedule may lead to further reduction in the total time of a query processing strategy . we develop a third version of algorithms general (collective) that uses algorithms serial and procedure collective to produce strategy with increase data transmission redundancy among schedule . in many cases , the total time of these strategy is less than the total time of strategy produced by algorithms General (Total Time)

Algorithms General can be applied to any general distributed query environments. it is relatively simple to program and has the added flexibility that all version can be

implemented together . then , depending upon run -time factors such as system load or query complexity , the optimization objective can be changed by a simple switch in the program.

REFERENCES

1. P.M.G.Apers, "critique on and improvement of hevener and Yao's distributed query processing algorithms G,"vrije unive. Amsterdam,The Netherlands,IR 48,feb,1979
2. Distributed query processing: minimum response time schedules for relation," Amsterdam,The Netherlands,IR 50,mar,1979
3. C.Baldissera , G.Bracchi,and s.Ceri,"A query processing strategy for distributed databases ,"proc ,EURP-IFIP79,1979,PP-667-678
4. P.A.Bernstein and D.W.Chiu,"using semi-join to solve relational queries ,"J.Ass.comput.Mach..,vol28,jan.1981
5. P.A.Bernstein ,N,Goodman,EWong,C.L.Reeve,and J.Rothin ,"Query processing in a system for distributed database(SDD-1),"ACM Trans.Database Syst..,vol.6,Dec1981
6. D.Chiu.and Y.Ho,"A methodology for interpreting tree queries into optimal semi-join expressions,"in Proc .ACM SIGMOD conf,santa monica,CA ,1980
7. W.W.chu and P.Hurley ,"A model for optimal processing for distributed database ,"in proc .18th IEEE compcon,spring,1979,pp.116-122
8. E.F.codd ,"A relational model of data for large shared data bank,"communAss.comput.mach..,vol.13,pp.377-387,june 1970
9. R.Epstein ,M.R.stonebreker ,and,E.wong,"Distributed query processing in relational data base system ,"in proc,Acm-SIGMOD,may1978,pp,169-180
10. R. Epstein M.R.stonebreker ,"analysis of distributed database processing strategy ,"in Proc .Int.conf.very large Data bases Montreal,P.Q..,canada1980
11. N.G Goodman and o.shmueli,"Hierarchies of database state reductions ,"Aiken computation Lab, Harvard Univer…,Cambridge, M.A, Tech.Rep.TR-18-80:also, ACM Trans.Data base syst..,to be published
12. G.Msacco and S.B. Yao ,"Query optimization in distributed database system,"in advance in computer ,vol.21.NewYork:Academic,1982
13. P.G.selinger,M.M.Astrahan,D.D.Chamberlin,R.A.Lorieand T.G. Price ,"Access path selection in a relational database management system ,'in proc .ACM SIGMOD Conf..,Boston,M.A,1979