An Efficient Heuristic Algorithm for Reliable Non Distributed Join Processing in Distributed Databases

Vivek Kumar
Department of Computer Science,
Gurukula Kangri Vishwavidyalaya, Haridwar-249404

Prof. Vinod Kumar
Department of Computer Science,
Gurukula Kangri Vishwavidyalaya, Haridwar-249404

Abstract - In computing systems, the quality, in terms of assurance with some degree in its operations, is desired. This aspect of quality plays a greater role in distributed computing systems where multiple elements (processors, links etc.), with varied degree of capabilities, are involved. In the case of Distributed Database Systems, many measures at different levels are included. Keeping in view of assurance, the data units are replicated and transaction management protocols are designed. Reliability factor has also been included in deciding the strategy for the allocation of operations for performing the required query. This paper explores the reliability aspect when it is achieved through the use of redundant processors and communication links among the sites of the distributed database system and allocation of operation of distributed query with mission reliability as the objective. We have suggested a heuristic algorithm in contrast to [1] which has increased the computational efficiency relying on the fact that the addition and multiplication operations are more efficient than the exponential function calls and the product operation on the results of these calls. Programs are designed to implement both the original and the enhanced algorithms. Several experiments (simulated) are conducted and the results show a very considerable enhancement obtained by applying the present heuristic along with the concept of more efficient operations. A total of six different sets of queries have been considered for simulation. Each set has been explored for six different ranges of execution cost over 200 randomly generated sets, thus, resulting into a total of 7200 simulations. The total gain achieved in terms of number of nodes of the search tree is 84.22%. The execution time difference could not be clearly notioned below 5 joins. However, the approximated total gain in this regard is obtained as 53.84%.

Key Words: Distributed Databases, Query processing, Heuristic Algorithms, Reliable Task Allocation.

I. INTRODUCTION

The advent of telecommunication era and the constant development of hardware and network structures have encouraged the decentralization of data while increasing the needs to access information from different sites. A distributed database system is a collection of sites on a common high-bandwidth network [2]. Logically, data belongs to the same system but physically it is spread over the sites of the network, making the distribution invisible to the user [3]. Each site is an autonomous database with its processing capability and data storage capacity. The advantage of the distribution resides in achieving availability, modularity, performance, and reliability. The reliability of a distributed system depends not only on the reliability of the communication network, but also on the processing node reliability and the distribution of the resources in the network [4]. A number of approaches and modeling techniques have been suggested for the reliable allocation of resources among the sites [5, 6, 7, 8, and 9]. To improve the system reliability, hardware redundancies can be introduced. When the system hardware configuration is fixed, the system reliability mainly depends on the allocation of resources. The problem of reliable allocation of join operations of distributed database system, main resource of distributed query processing, using processor and communication link redundancies, has been discussed in [1]. This paper re-explores the problem of reliable join processing and suggests a more efficient heuristic algorithm.

II. NOTATION AND ASSUMPTIONS

1. Acronyms

RDS	redundant distributed system
ETM	execution time matrix
ITC	inter-task communication
AM	allocation matrix
FV	fragmentation vector
MJP	multiple-join problem

2. Notation

B(.)	Boolean function: B(True) =1 ; B(false) = 0
r	redundancy level of RDS
T	set of tasks to be executed
t_i	task i in T
m	number of Joins in the query
P	set of processing nodes in RDS
n	number of processing nodes in P
P_k	processing node k in P
p_k	processor at P_k
λ_k	failure rate of p_k
e_{ik}	ETM for module t_i running on p_k
c_{ij}	ITC cost between t_i & t_j during the mission; measured in some quantity of data units
l_{pq}	communication link connecting P_p & P_q
w_{pq}	transmission rate of link l_{pq}
λ'_{pq}	failure rate of link l_{pq}
L_{st}	communication path between Ps & P_t; it contains at least one link
X	m x n binary matrix corresponding to a task assignment
x_{ik}	B(t_i is assigned to p_k): element of X
Q	binary path matrix
Q(p,q,k,l)	B(l_{pq} belongs to L_{kl})
C	inter-task communication cost: a function of X
c(i,j,p,q)	c_{ij} . B(c_{ij} uses l_{pq})
R_k(T,X)	reliability of p_k: probability that p_k is operational for executions of tasks assigned to it under assignment X during the mission

$r_{pq}(T,X)$ reliability of l_{pq}: probability that l_{pq} is operational for performing all ITC that use l_{pq} under assignment X during the mission

$R^{(r)}(T,X)$ reliability of the distributed-computer system

3. Assumptions

- The underlying RDS is heterogeneous.
- The network topology of the RDS is free from cycles.
- The processors or communication links have two states, namely operational or failed, with constant failure rates.
- Units can fail independently.
- Task-allocation implies Join allocation.
- The software is error free.
- The cost includes only transmission costs.

III. DATABASE DISTRIBUTION AND JOIN ALLOCATION

Each file of the distributed database is a set of homogenous records those are not necessarily stored at a common node. A file can be logically partitioned into disjoint subsets of records called fragments [10]. The size of the file is the number of bytes of the file. The size of each fragment is given by a fragmentation vector, depicted as number of bytes of the fragment of the file. Each fragment can be allocated at one of the nodes of the network. An allocation matrix [11] defines the allocation of the fragments of a file.
A Join operation consists of finding those records of two different files that possess some matching property. In a distributed database, the join can be performed in two ways namely, Non-distributed Join and Distributed Join.
The Non-distributed join, considered in this paper, produces a result file that is not distributed. For estimating the size of the result file, statistical-profile estimation methods [12] can be used. An operator tree representational approach is used to depict a global query, consisting of several joins. To perform join operations some fragments of files are transferred from their location site to the execution site.

IV. RELIABLE ALLOCATION OF JOINS

This paper extends the model III of [4] in which explicit reliability expression, in terms of system parameters and mission reliability, has been formulated where a mission is a continuous time interval that is sufficiently long for unit failures to become a concern.

Under task assignment X, the reliability of p_k, for the execution of the tasks assigned to it during the mission [4], is:

1. Reliability Expression Formulation

$$R_k(T,X) = \exp\left(-\lambda_k \sum_{i=1}^{m} x_{ik} e_{ik}\right) \quad (1)$$

The reliability of link l_{pq} is

$$r_{pq}(T,X) = \exp\left(-\lambda'_{pq} \sum_{j=1}^{m-1} \sum_{i=j+1}^{m} c(i,j,p,q)/w_{pq} \right) \tag{2}$$

$$c(i,j,p,q) = \sum_{k=1}^{n} \sum_{l \neq k} Q(p,q,k,l) x_{ik} x_{jl} c_{ij} \tag{3}$$

Thus, the reliability of the system with double redundancy is given by

$$R^{(2)}(T,X) = \prod_{k=1}^{n} R_k(T,X)(2 - R_k(T,X)) \prod_{l_{pq}} r_{pq}(T,X)(2 - r_{pq}(T,X)) \tag{4}$$

2. Reliability Approximation

For $0 \le x_i < 1, i = 1,2\cdots,$ by a Taylor series expansion we have:

$$e^{-x_i}(2 - e^{-x_i}) = 1 - x_i^2 + x_i^3 - \cdots \approx 1 - x_i^2 \tag{5}$$

$$\prod_{i=1}^{n}(1 - x_i^2) = 1 - \sum_{i=1}^{n} x_i^2 + \sum_{i \neq j} x_i^2 x_j^2 - \cdots \approx 1 - \sum_{i=1}^{n} x_i^2 \tag{6}$$

$$\prod_{i=1}^{n} e^{-x_i}(2 - e^{-x_i}) \approx \prod_{i=1}^{n}(1 - x_i^2) \approx 1 - \sum_{i=1}^{n} x_i^2 \tag{7}$$

Using (7) for approximating (4), we have

$$\begin{aligned} R^{(2)}(T,X) &= \prod_{k=1}^{n} R_k(T,X)(2 - R_k(T,X)) \prod_{l_{pq}} r_{pq}(T,X)(2 - r_{pq}(T,X)) \\ &\approx 1 - COST(T,X) \end{aligned} \tag{8}$$

$$COST(T,X) = \sum_{k=1}^{n} (\lambda_k \sum_{i=1}^{m} x_{ik} e_{ik})^2 + \sum_{l_{pq}} (\lambda'_{pq} \sum_{j=1}^{m-1} \sum_{i=j+1}^{m} c(i,j,p,q)/w_{pq})^2 \tag{9}$$

The error of the approximation is:

$$ERROR \approx \sum_{k=1}^{n} (\lambda_k \sum_{i=1}^{m} x_{ik} e_{ik})^3 + \sum_{l_{pq}} (\lambda'_{pq} \sum_{j=1}^{m-1} \sum_{i=j+1}^{m} c(i,j,p,q)/w_{pq})^3 \tag{10}$$

In order to maximize the approximate reliability (8), it is sufficient to minimize COST(T,X), which is nothing but the approximated unreliability.

V. THE PROPOSED MODEL AND ALGORITHM

The algorithm assumes redundancy at both processors and communication link of the underlying distributed system. It finds an assignment of joins to the sites of the distributed database system and approximated system un-reliability cost is minimized for finding the reliable execution nodes.

1. Model

Minimize: COST(T,X)

Subject to: $x_{ik} = 0$ or 1,

For all i,k: $1 \leq i \leq m$, $1 \leq k \leq n$

$$\sum_{k=1}^{n} x_{ik} = 1.$$

This optimization problem is converted to a state-space search-tree problem. For finding the goal node, algorithm uses a heuristic function f to order nodes for expansion [4]. At each expansion, only the node with minimum cost of unreliability is expanded. This function is defined as the sum of two components:
i.e. for a node ν corresponding to an assignment X_{ν}^{s} with $t_1, t_2, ..., t_s$ assigned

$$f(\nu) \equiv g(\nu) + h(\nu) \tag{11}$$

where, the function g is a measure of the cost of getting from the initial state to the current node

$$g(\nu) = COST(T, X_{\nu}^{s}) \tag{12}$$

and the function h is an estimate of the additional cost of getting from the current node to a goal state. While considering a node of search tree, for expansion, the heuristic used in the algorithm focuses only on recently generated nodes, at each level, in contrast to all unexpanded nodes considered in [1].

2. Algorithm

1. Order m joins according to join precedence in the query tree.
2. Initialize the assignment matrix.
3. Consider the first or next join j of m joins. If all the joins have been considered, calculate and record the cost of transmission for this complete assignment and go to 7.
4. Create a new list, called NODES.
5. For each processor k of n

If the execution cost for j on k is finite
Then create a new node with this incomplete join assignment up to join j and insert it into NODES list after calculating the corresponding f-value.

1. Choose node from NODES with the least f-value and populate assignment matrix with the corresponding processor assignment for join j. Go to 3.
2. Output the assignment with its cost of transmission and its system reliability.
3. Stop.

VI. SIMULATION RESULTS AND DISCUSSION

The developed algorithm and the algorithm given in [1] have been coded in C. A total of 7200 simulations were subjected to the codes of both the algorithms implemented on Pentium III Personal Computer using Windows XP based Microsoft Visual C++ 6.0 compiler. To estimate the time factor, the clock() function of WINDOWS API is used.

As observed from Table 1, the overall reduction in the total number of nodes generated, as compared to algorithm [1] is 84.42%. Table 1 also shows the comparison of execution times of the implemented algorithms. Though the values, given in this table, are higher than the theoretical expectations, inspired by the mathematical formulation of the problem, yet these can be considered as good estimates of the time. The variation in the values of execution time from the theoretical expectations can be attributed to the limitation of the hardware clock of the system, clock() function of WINDOWS API and hardware implementation of exponential and multiplication operations. However, it can be noticed from Table 1 that the total averaged time reduction achieved through our algorithm is 53.85% as compared to algorithm [1]. These results have been shown graphically in Fig. 1 and Fig. 2 respectively.

The proposed algorithm and the algorithm [1] have also been subjected to 7200 simulations for six values of joins and various ranges of the values of execution cost to observe variations in reliabilities of join allocation. These results are presented in Table 2. The first column of this table contains number of joins. The second column consists of various values of upper bounds of the entries of ETM. For example, a value 15 in the second column, for 3 joins, indicates that for the 200 simulations, considered for 3 joins, 200 execution time matrices were randomly generated whose elements were less than or equal to 15. Column 3 of Table 2 lists the largest variations in the reliability of allocation of distributed joins, encountered during the 200 simulations corresponding to the different ranges of the entries of ETM and a given number of joins. For example, the value 0.000001007 is the maximum variation in the reliability values, out of 200 simulations run for 200 different values of ETM having elemental value less than or equal to 15 for 3 join problems. From this table, one can observe that the variations in the reliability values of distributed join allocation, ranging from 0.000001007 to 0.000035088, are negligible as compared to the achieved computational gain (i.e. 53.85 %).

VII. CONCLUSION

For the non-distributed join operations, required by a global query in a distributed database, a mathematical model is formulated and a heuristic algorithm is designed. While considering a node of search tree, for expansion, the heuristic used in the algorithm focuses only on recently generated nodes, at each level, in contrast to all unexpanded nodes considered in [1]. The allocation problem of join operations is addressed with the consideration of reliability of the system and the communication cost required for transmitting the fragments. Remarkable gain has been achieved in the average number of generated nodes as well as execution time. The proposed approach can be implemented as a part of the optimizer of any database system along with the earlier one [1] and the selection of the algorithm could be problem specific. If the level of criticality is almost 100% then earlier algorithm [1] can be chosen and if it is about 99.99% then the proposed algorithm can be chosen with 53.85% gain on the execution time.

REFERENCES

[1] A. K. Verma and M. T. Tamhankar, “Reliability Based Optimal Task Allocation in Distributed Database Management Systems,” IEEE Trans. Reliability, vol. 46, no. 4, pp. 452–459, 1997.

[2] B. Karwin, “Interbase Server Configuration and Optimization”, Borland Developer’s Conference, 1996.

[3] D. Chatziantoniou, D. and K. Ross, “GroupWise Processing of Relational Queries”, in Proceedings of the 1997 VLDB Conference, pp. 476-485, 1997.

[4] S. M. Shatz and J.-P. Wang, “Models and Algorithms for Reliability oriented Task Allocation in Redundant Distributed Computer Systems,” IEEE Trans. Reliability, vol. 38, no. 1, pp. 16–26, 1989.

[5] S. Srinivasan and N. K. Jha, “Safety and Reliability Driven Task Allocation in Distributed Systems,” IEEE Trans. on Parallel and Distributed Systems, vol. 10, no. 3, pp. 238–251, 1999.

[6] S. Hariri and C. S. Raghavendra, “Distributed Functions Allocation for Reliability and Delay Optimization,” in Proc. IEEE/ACM 1986 Fall Joint Computer Conf.,Dallas, 1985, pp. 344–352.

[7] D. J. Chen and T.-H. Huang, "Reliability Analysis of Distributed Systems Based on Fast Reliability Algorithm," IEEE Trans. on Parallel and Distributed Systems, vol. 3, no. 2, pp. 139–154, 1992.

[8] G.-J. Hwang and S.-S. Tseng, "A Heuristic Task Assignment Algorithm to Maximize Reliability of a Distributed System," IEEE Trans. on Reliability, vol. 42, no. 9, pp. 408–415, 1993.

[9] V. Kumar and K. K. Aggarwal, "Petri Net Modeling and Reliability Evaluation of Distributed Processing Systems," Reliability Engineering and System Safety, vol. 41, pp. 167–176, 1993.

[10] S. Ceri and G. Pelgatti, "Allocation of Operations in Distributed Database Access," IEEE Trans. Computers, vol. C-31, no. 2, pp. 119–129, 1982.

[11] S. Ceri and G. Pelagatti, Distributed Databases: Principles and Systems. McGraw Hill, 1985.

[12] M. V. Mannino, P. Chu, and T. Sager, "Statistical Profile Estimation in Database Systems," ACM Computing Surveys, vol. 20, no. 3, pp. 191–221, 1988.

Table 1: Results Comparison Table

Number of Joins	Average Number of Comparison Nodes		Average Execution Time Order	
	Algorithm [1]	Proposed Algorithm	Algorithm [1]	Proposed Algorithm
3	14	11	0.001	0.001
4	27	15	0.001	0.001
5	52	19	0.001	0.001
6	102	23	0.002	0.001
7	199	26	0.003	0.001
8	402	30	0.005	0.001
Total	**796**	**124**	**0.013**	**0.006**

Table 2: Variations in Reliability of Joins as compared to [1]

No. of Joins	Upper value of range of ETM	Variation in Reliability values
3	15	0.000001007
	20	0.000001665
	25	0.000003840
	30	0.000002909
	35	0.000007268
4	15	0.000003232
	20	0.000005994
	25	0.000007379
	30	0.000008434
	35	0.000014322
5	15	0.000003054
	20	0.000011018
	25	0.000010291
	30	0.000010467
	35	0.000027790
6	15	0.000004466
	20	0.000016647
	25	0.000011309
	30	0.000014804

	35	0.000021966
7	15	0.000005789
	20	0.000012904
	25	0.000010476
	30	0.000013140
	35	0.000035088
8	15	0.000007412
	20	0.000008947
	25	0.000011223
	30	0.000026819
	35	0.000025157

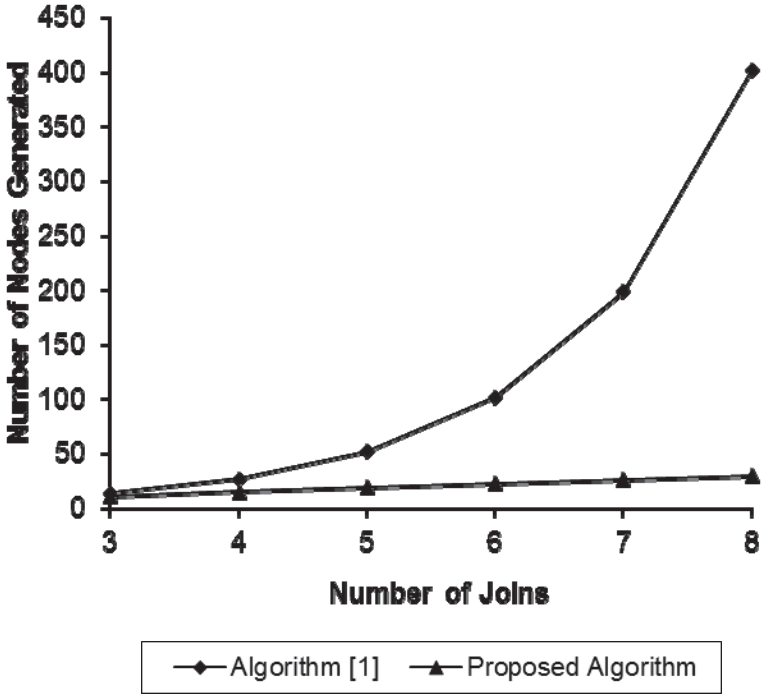

Figure 1: Comparison of Number of Nodes Generated

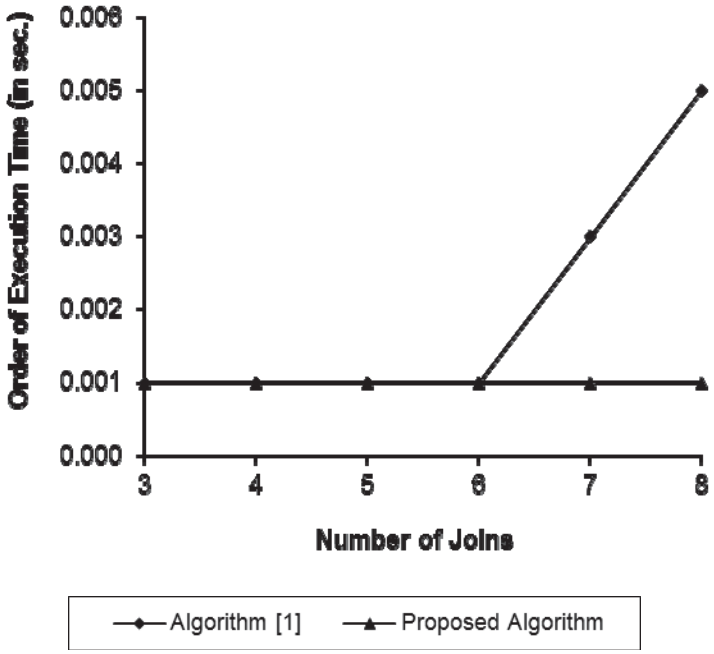

Figure 2: Execution Time Comparison