A DECISION SUPPORT SYSTEM FOR DISTRIBUTED DATA ALLOCATION IN SUPPORT OF C³ SYSTEMS

Edward T. Poore, Jr. and Thomas C. Hartrum
Department of Electrical and Computer Engineering
School of Engineering
Air Force Institute of Technology
Wright-Patterson AFB, Dayton, Ohio, 45433

Abstract

The allocation of data in a distributed database system can significantly affect the response time of the system. This is particularly important when the database supports an operational command, control, and communication C³ system. There are several methods for determining where to store the data in a distributed system. These include various optimization algorithms which use linear programming to minimize some cost function, as well as numerous heuristic methods. This paper presents a Decision Support System (DSS) that rank-orders various allocation methods based on the relative importance given to such factors as response times in the communication network, resources needed, and execution time of the allocation algorithm. To demonstrate this DSS, the Army Maneuver Control System (MCS) was simulated using SLAM. Two optimal allocation schemes (using linear programming) and three heuristics were implemented to assign data location among the MCS nodes. The results from these simulations were then input to the DSS, along with several scenarios to provide importance weights to the decision factors. The results show that these weights do affect the choice of allocation scheme, including some scenarios where neither of the optimal techniques were selected.

1 Introduction

The Army Maneuver Control System (MCS) is a distributed command and control system, supporting the operations section (G3/S3) from the Battalion to the Corps level. MCS provides the commanders and operations sections up-to-date battlefield information. The planning cycle of the operations section is time critical. Rapid response time to updates (changes in data) and queries (requests for information) is essential to an efficient planning cycle. The allocation of data in a distributed database affects the response time of updates and queries at each node in the distributed network.

A query can be processed faster if the requested data is available locally at the node where the query originates than if communication to another node is required to access the data. Thus if data is replicated at those nodes where it is likely to be requested, then the overall query response time of the network should be, on the average, lower. However, an update to a replicated data item submitted at one node must be transmitted to all other locations where the data is stored, requiring longer processing time and more network traffic than an update to a single storage location. A cost function can be defined over the network that takes into account the relative weight of the network load due to the updates and queries. An allocation scheme which optimizes the storage location of each item in the database by minimizing this cost function can then be determined. However, algorithms which solve optimal allocation problems for large networks are often cost prohibitive in terms of computation time or special resources required (such as a fast parallel processor). The use of heuristics in assigning data items may reduce this cost of solving the problem, but may result in other than an optimal solution.

It is not known what the best method for allocation of data is for a distributed database such as the MCS. Especially in a dynamic battlefield environment, it may be necessary to re-allocate data on a frequent basis. The problem is to determine the methodology which most efficiently (in terms of computational time and cost of required resources) allocates data items in a way which still meets the performance criteria of the network.

This study proposed a decision function to evaluate allocation schemes for a distributed database system. It was tested with three heuristic allocation schemes and two optimal allocation schemes. A simulation model was used to gather performance measures since a functioning MCS system is not yet available for data collection. The performance measures used in the comparison are average system completion time for an update and query message for both local and remote access to data. The optimal allocation schemes for the two cases of nonredundant and redundant storage are based on a numerical algorithm solution. The heuristic methods were developed in a previous study by Harwood [4], and include total replication of all data, nonredundant storage based on Harwood's least update cost algorithm, and nonredundant storage based on Harwood's greatest query rate algorithm.

2 The Decision Function

There are many factors to be considered in selecting between data allocation methods. The load on the system,
the average system times of the performance criterion, the data storage cost, and the cost to execute the allocation algorithm are all factors which affect the decision of choosing the best method among alternate allocation schemes. A low load on the system means no resource is utilized more than 25 percent. A high load represents the utilization of any one resource in the network approaching but not exceeding 95 percent. A decision function is needed that takes into account all of these factors and assigns a single relative score to each allocation method to aid in a selection choice. The decision function proposed in this paper is

$$C_i = t_1(hA_1[2i + 1, 1] + tA_1[2i + 2, 1]) + t_2(hA_1[2i + 1, 2] + tA_1[2i + 2, 2]) + t_3(hA_1[2i + 1, 3] + tA_1[2i + 2, 3]) + t_4(hA_1[2i + 1, 4] + tA_1[2i + 2, 4])) + sA_2[i + 1, 1] + rA_3[i + 1, 2]$$

where

- $C_i$ is the score for allocation method $i$.
- $t_1$ is the weight for aggregate system response time.
- $t_2$ is the weight for Update Local time.
- $t_3$ is the weight for Update Remote time.
- $t_4$ is the weight for Query Local time.
- $t_5$ is the weight for Query Remote time.
- $h$ is the weight for system at high load.
- $l$ is the weight for system at low load.
- $A_1[i]$ is an array holding data for system criterion. The first row of an allocation method is performance criterion data at low load and row two is data at high load. Column 1 of each row is the average system time for Update Local. Column 2 holds Update Remote time. Column 3 holds Query Local time. Column 4 holds Query Remote time.
- $s$ is the weight for storage cost of allocation method.
- $r$ is the weight for resource cost in terms of computational complexity for execution of the allocation method.
- $A_2[i]$ is an array holding data for system storage cost and resource cost to allocate data in the system.

Prior to executing the decision function to obtain relative scores of each allocation method, performance data and storage cost data must be collected or simulated for each allocation method being considered, and a cost estimation for executing each allocation method must be obtained. The weight or importance of these normalized factors needs to be assigned by the manager or commander. Since actual MCS data was not available, a simulated network was constructed to gather data by running the simulation based on the data allocated by each method.

3 Test Network Design Specification

Since the MCS network topology has not yet been finalized, a test network was devised that was at once similar to that anticipated for MCS and yet small enough to be simulated in the time available. The model represents a possible configuration of an Army division’s three tactical headquarters (nodes 1, 2 and 3) connected to one of the division’s brigade tactical headquarters (nodes 4, 5 and 6). All other loads in this representation were ignored. The input message rates did not reflect any actual known input loads, but were based on anticipated levels. The high rates were based on empirical tuning of the simulation model. All other numbers in the specifications were not based on an actual network, but were representative estimates to form a base network to gather response time data for comparative analysis.

1. The message sizes are uniformly distributed between 250 and 2000 characters.
2. The operator typing time of original messages is 2.00 minutes per 2000 characters.
3. The CPU processing time of messages is exponentially distributed with a mean of 0.0006667 minutes.
4. There are 12 average disk accesses per 2000 character message.
5. Disk access time is 0.0001667 minutes, which includes data transfer time.
6. The operator time to handle routing messages is 0.1 minute.
7. Each link in the network model is full duplex.
8. Message routing is fixed in the network.
9. The conversion factor for transmission time from a message size is 0.0001111 minute per character.
10. Each CPU burst is uniformly distributed.
11. The low and high load arrival rates of update and query messages by data item are based on a given 24 hour period, and are estimated rates.
12. The CPU requires 0.001333 minutes to process the message routing routine.
Duplicate update messages are created for each destination node and routed separately.

The cost of an Update, CU, from node j to node k is 1 times the number of links between the two nodes.

The cost of a Query, CQ, from node j to node k is 2 times the number of links between the two nodes.

The storage cost of placing a data item at a node is 1.

4 Data Allocation Methods

The different allocation methods all considered network transmission costs. Network communication costs include both the cost of update messages and the cost of query messages. The goal of all the allocation methods is to reduce the network transmission cost by reducing either the total query cost, the total update cost, or both. The total transmission cost is defined as a weighted sum of the update and query costs for both the nonrecurrent and redundant cases.

4.1 Transmission Cost Function Defined

Let

- \( i \) = data item.
- \( j \) = demand node (node where message request originates).
- \( k \) = source node (node where data item is stored).
- \( CU_{jk} \) = Cost of Update from node j to node k.
- \( CQ_{jk} \) = Cost of Query from node j to node k.
- \( RU_{ij} \) = Rate of Updates for data item i generated at node j.
- \( RQ_{ij} \) = Rate of Queries for data item i generated at node j.
- \( X_{ik} \) = binary decision variable for storage of data item i at node k. 0 is assigned if data item not stored and 1 is assigned if data item is stored.

The network Update cost is

\[
\sum_i \sum_j \sum_k CU_{jk} RU_{ij} X_{ik} \quad (2)
\]

The network Query cost (nonredundant) case is

\[
\sum_i \sum_j \sum_k CQ_{jk} RQ_{ij} X_{ik} \quad (3)
\]

The network Query cost (redundant) case is

\[
\sum_i \sum_j \sum_k \min(CQ_{jk} RQ_{ij} X_{ik})\text{ for } j \neq k \quad (4)
\]

The total network cost (nonredundant case) from equations 2 and 3 is

\[
\sum_i \sum_j \sum_k (CQ_{jk} RQ_{ij} + CU_{jk} RU_{ij}) X_{ik} \quad (5)
\]

The total network cost (redundant case) from equations 2 and 4 is

\[
\sum_i \sum_j \sum_k (CU_{jk} RU_{ij} X_{ik}) + \sum_i \sum_j \sum_k \min(CQ_{jk} RU_{ij} X_{ik}) \quad (6)
\]

4.2 Optimal Allocations

For the optimal nonredundant case, equation 5 was minimized subject to the constraints that \( X_{ik} \) is a binary integer such that \( \sum_k X_{ik} = 1 \) for each data item i. This means each data item i has to be stored once and only once in the network.

For the optimal redundant case, equation 6 was minimized subject to the same binary constraint as in the nonredundant case. The assignment constraint, \( \sum_k X_{ik} = 1 \), was relaxed to allow for a sum greater than one for each data item i. This means each data item i has to be stored at least once, but may be stored more than once. For this test network, there were 63 combinations of storing a data item in a six node network at least once.

4.3 Heuristic Allocations

Total redundant allocation was a trivial case in the sense that it reduced the total transmission cost to just the update cost, equation 2. Since every demand node j was also the source node, the query cost was zero. However, this heuristic also had the distinction of maximizing the network update cost.

The Least Update Rate nonredundant allocation "seeks to allocate data items to those nodes where the sum of the update rates to that data item from all other nodes is the least" \[4,137\]. Equation 2 was adjusted to find the source node k which minimized \((CU_{jk} RU_{ij} X_{ik})\), where \( k \neq j \). The data item was stored at this node.

The Greatest Query Rate nonredundant allocation attempts to reduce the network transmission cost for each data item by storing the data item at the demand node with the greatest request rate. This would eliminate any query transmission cost at the highest demand node for that data item. This required finding the max value of the \( RQ_{ij} \) term of equation 4 for each data item i and storing the data item at the demand node j for that value.
5 Results

Several intermediate results were calculated leading up to the execution of the final decision model. The results of the various allocation techniques were used as input to the simulation model. The results of the simulation model runs were averaged and normalized for use as input parameters to the decision model. The storage data output of the allocation methods were totaled for each method to determine its overall storage cost. Harwood's computational complexity analysis of different allocation methods [4, 205-209] was used to determine the resource cost (in terms of number of operations) to execute each allocation method. Both the storage and resource costs were normalized and these data were used as parameters in the decision model. Several scenarios of a network and the weights for the parameters were considered. The decision model was executed for each scenario and the results were tabulated.

5.1 Storage Cost

Figure 1 gives a summary of storage allocation and destination nodes for all five allocation methods. The symbol for the allocation method represents the storage of the data at that node.

Figure 1 shows that for the total replication allocation method, data items 1-8 were stored at every node. For the optimal redundant allocation method Figure 1 shows that data item 1 was stored at nodes 3 and 4. It is clear that the various allocation methods did not store all data items at the same nodes.

The storage cost for each allocation method can be easily obtained by summing the number of each type symbol in Figure 1 for each allocation method. Total replication was the most expensive allocation in terms of storage; 8 data items were each stored at six nodes for a total cost of 48. The next most expensive allocation method, in terms of storage, was the optimal redundant with a cost of 17. All other allocations being nonredundant stored each data item only once in the network for a storage cost to each of 8.

5.2 Resource Cost

Harwood's computational complexity analysis of different allocation methods [4, 205-209] was used to determine the resource cost (in terms of number of operations) to execute each allocation method. The only method not analyzed by Harwood was for the optimal nonredundant case (equation 5). This case includes both Update and Query costs of the network. These two costs together equal Harwood's total cost function. "It was determined previously that \((M^2N + MN)\) operations were required for computing total costs" [4, 205]. Figure 2 is a summary of Harwood's analysis and the big "O" cost for a six node network with eight data items.

5.3 Analysis of Simulation Data

After the five allocation methods were run, the resulting data allocations were applied to the simulation. For each allocation the simulation was run with a low system load and a high system load. (The high system load was determined empirically by iteratively increasing the query and update rates until one of the nodes reached the 95 percent utilization point). With the number of runs for each combination being six, there was a total of sixty data points for each system criteria. All the data points for

<table>
<thead>
<tr>
<th>NODE</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA 1</td>
<td>TC</td>
<td>TQ</td>
<td>TOL</td>
<td>TO</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>TCOL</td>
<td>TQ</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>TO</td>
<td>TO</td>
<td>T</td>
<td>T</td>
<td>TQ</td>
<td>TO</td>
</tr>
<tr>
<td>4</td>
<td>TCOL</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>TQ</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>TCOQ</td>
<td>TL</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>TOQ</td>
<td>T</td>
<td>T</td>
<td>TCOQ</td>
<td>TO</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>TL</td>
<td>TO</td>
<td>T</td>
<td>TC</td>
<td>TOQ</td>
<td>T</td>
</tr>
<tr>
<td>8</td>
<td>TL</td>
<td>T</td>
<td>T</td>
<td>TCOQ</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Legend:

T TOTAL REPLICATION
C OPTIMAL NONRENDANT
O OPTIMAL REDUNDANT
L LEAST UPDATE COST
Q GREATEST QUERY RATE

Figure 1: Results of Allocation Methods Storage Allocation

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Big &quot;O&quot;</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Replication</td>
<td>O(M^2N)</td>
<td>288</td>
</tr>
<tr>
<td>Optimal Nonredundant</td>
<td>O(M^2N+MN)</td>
<td>336</td>
</tr>
<tr>
<td>Optimal Redundant</td>
<td>O(2MN)</td>
<td>147456</td>
</tr>
<tr>
<td>Greatest Query Rate</td>
<td>O(MN)</td>
<td>2304</td>
</tr>
<tr>
<td>Least Update Cost</td>
<td>O(M^2N)</td>
<td>2304</td>
</tr>
</tbody>
</table>

Legend:

- M is # of Nodes (six)
- N is # of Data items (eight)

Figure 2: Complexity and Cost of Executing Allocation Methods [4,205-209]
the four system performance measures (response times for local queries, remote queries, local updates, and remote updates) were combined into one file for analysis.

Analysis of variance (ANOVA) was used to test the null hypothesis that there was no effect on the system performance criterion due to the factors of allocation methods, system load, or their interactions. Where the null hypothesis was rejected, multiple-range tests were performed to indicate which means differed significantly. There was sufficient evidence to reject the null hypotheses and accept the alternate hypotheses that the allocation methods and system loads had an effect on system response times.

5.4 Normalize Data for Decision Function

Before the simulation results could be used in the simulation, normalization was required, since the costs associated with system performance, storage, and resource did not all have the same dimensions. Normalizing the data on a scale from 0 to 1 made the decision function dimensionless and eliminated the problem of scaling between different costs or performance criterion. Each variable was normalized to its maximum value. Figure 3a contains the normalized data from the system performance criterion by allocation method and experimental load. Figure 3b is the normalized data for the storage requirement of each allocation method and the resource cost in terms of the number of operations needed to execute the allocation for six nodes and eight data items. The data for normalization of resource costs came from Figure 2. The large magnitude of the number of operations for the optimal redundant case caused the normalized value for all other allocations to be essentially zero.

6 Using the Decision Function

Normalization of the simulation data provided the system variables needed for the decision function of equation 1. The weights for equation 1 have to be determined by the system manager, based on the importance of the different variables. To demonstrate the use of the decision function, consider a specific case of weights based on the following scenario.

Case A The network manager considered response time, storage, and resource cost all equally important, therefore a value of 0.33 was assigned to each of these weights in the decision function. The manager also noted that the network operated just as often at high load as it did at low load, so a value of 0.5 was assigned to these weights. There were as many queries as update messages that flowed through the network and they were both processed as many times locally as they were remotely, so the value of 0.25 was assigned to each of these weights. These values are summarized in the first column of Figure 4. As shown in Figure 5 for Case A, the optimal nonredundant allocation would have been selected for this network as it had the lowest value of 0.33. Altogether, six scenarios were run (A through F). The cases examined were a representative group to investigate the use of the decision function, and were not intended to be an exhaustive set. The weights for all of the scenarios are summarized in Figure 4. The decision function was then executed to find the relative ranking of the allocation methods for each scenario. The results of the decision function for each case are shown in Figure 5. The values of Figure 5 were given a ranking of 1 (best) for the lowest value and 5 (worst) for the highest value and the rankings were summarized by case in Figure 6.

Figure 6 shows that the optimal multiple (redundant) was only ranked 1 once. This can be explained in part by understanding that the term "optimal" was optimal only to the extent that the cost functions of equations 5 and 6 accurately model the network delays of the system (in our case, of the simulation). Actually, equations 5 and 6 reflect

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Load</th>
<th>Average System Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL NODES</td>
<td>HIGH</td>
<td>1.00 1.00 1.00 1.00 0.00</td>
</tr>
<tr>
<td>GREATEST QUERY</td>
<td>LOW</td>
<td>0.00 0.09 0.09 0.09 0.09</td>
</tr>
<tr>
<td>LEAST UPDATE</td>
<td>HIGH</td>
<td>0.83 0.82 0.92 1.00 1.00</td>
</tr>
<tr>
<td>LEAST UPDATE</td>
<td>LOW</td>
<td>0.83 0.82 0.92 1.00 1.00</td>
</tr>
<tr>
<td>OPTIMAL MULT</td>
<td>HIGH</td>
<td>0.97 1.00 1.00 1.00 1.00</td>
</tr>
<tr>
<td>OPTIMAL SNGL</td>
<td>LOW</td>
<td>0.97 0.94 0.99 0.83 0.83</td>
</tr>
<tr>
<td>(b) System Performance Criterion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Storage</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL NODES</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>GREATEST QUERY</td>
<td>0.17</td>
<td>0.00</td>
</tr>
<tr>
<td>LEAST UPDATE</td>
<td>0.17</td>
<td>0.00</td>
</tr>
<tr>
<td>OPTIMAL MULT</td>
<td>0.35</td>
<td>1.00</td>
</tr>
<tr>
<td>OPTIMAL SING</td>
<td>0.17</td>
<td>0.00</td>
</tr>
<tr>
<td>(b) Storage and Resource Cost</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Normalized Data a) System Performance Criterion, b) Storage and Resource Cost

Figure 4: Summary of Scenario’s Weights

\[
\begin{array}{l|llllll}
\text{Weights} & A & B & C & D & E & F \\
\hline
\text{RESPONSE TIME} & 0.33 & 0.70 & 0.70 & 1.00 & 1.00 & 1.00 \\
\text{HIGH} & 0.75 & 0.75 & 0.50 & 0.75 & 0.75 & 0.75 \\
\text{LOW} & 0.75 & 0.25 & 0.25 & 0.50 & 0.75 & 0.75 \\
\text{UPLOC TIME} & 0.25 & 0.50 & 0.50 & 0.25 & 0.25 & 0.25 \\
\text{UEPREM TIME} & 0.25 & 0.50 & 0.50 & 0.25 & 0.25 & 0.25 \\
\text{QYLOC TIME} & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 \\
\text{QYPREM TIME} & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 \\
\text{STORAGE} & 0.33 & 0.20 & 0.20 & 0.00 & 0.00 & 0.00 \\
\text{RESOURCE COST} & 0.33 & 0.10 & 0.10 & 0.00 & 0.00 & 0.00 \\
\end{array}
\]
only the transmission time of the network and not the overall delay. It was, in fact, not knowing the distribution of the network delay that led to the use of the simulation to compare the allocation methods. In addition, the large resource cost of using the optimal redundant allocation acted as a penalty each time the resource factor was heavily weighted.

The large storage cost of the total redundant allocation also acted as a penalty for this method when storage was considered, as in cases A and B. In case C the storage cost penalty was overshadowed by importance of query responses in this network, and that for this allocation there is no network cost associated with queries. Whenever queries were of primary concern, as in cases C and F, or of equal concern with updates and no consideration of storage as in case D, total redundant allocation was the 1st choice.

The ranking of the optimal nonredundant allocation as either 1 or 2 in all cases indicates the efficiency of centrally locating the data items in terms of the network median based on the message load per data item and the low cost of storage associated with this allocation method.

## 7 Summary and Conclusions

This paper analyzed the problem of selecting among different data allocation methods the best one for a distributed database. The selection involved complicated and multiple criteria. There were trade-offs in the different allocation methods criteria, such as faster response times with more storage cost or a more complex allocation method and its associated higher resource cost of computational time. To help a system manager choose the right allocation method, a decision function was proposed. Five allocation schemes were executed, and the resulting allocations used to drive a simulation of a C3 network. Simulation performance parameters were normalized and used in the decision function. Six different scenarios were used to determine the variable weights in the decision function.

On the basis of the simulation study and the analysis of the output of the decision function for the given case scenarios, the following conclusions are drawn:

1. The simulation helped gain greater insight into the allocation problem and the factors to be considered in a selection process.

2. The selected allocation changed, based on which system performance criteria was considered.

3. In some cases the selected allocation changed based on the system load.

4. The selected allocation changed based on the weights assigned to the cost of storage and the computational complexity of each allocation method.

5. The decision function proposed included all the factors which affect the selection of one allocation method over another method. The function allowed for weighting of the system response time criteria and system load.

6. The decision function is network independent. Given system data from an actual distributed database network, the decision function could be applied to aid in the complex decision of which data allocation method would be best for the network. The methodology should scale to larger networks.

7. The optimal allocation methods were optimal only in the area of network transmission time; not in transmission delay time or storage cost.

The allocation of data in a distributed database is a complex problem. Many issues such as replication of data to ensure survivability of a military network or unreliable communication links were not covered in this research. This study addressed multiple factors impacting an allocation scheme such as message response time, storage cost, and resource cost to execute the allocation method. A framework or methodology for relating and evaluating the multiple and sometimes conflicting factors was presented in the form of a decision function. The decision function can be used in future research in evaluating new proposed heuristics in data allocation.
Bibliography


