PERFORMANCE ESTIMATES FOR DISTRIBUTED QUERY PROCESSING

by

M. Murphy

Memorandum No. UCB/ERL M84/30

10 April 1984

PERFORMANCE ESTIMATES FOR DISTRIBUTED QUERY PROCESSING

by

Marguerite Murphy

Memorandum No. UCB/ERL M84/30

10 April 1984

ELECTRONICS RESEARCH LABORATORY

NSF MCS-8211528

PERFORMANCE ESTIMATES FOR DISTRIBUTED QUERY PROCESSING

Marguerite Murphy

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

The introduction of low cost hardware and the realization that physically distributing data can result in higher performance systems has led to the implementation of a number of distributed relational database management systems (D-RDMS). In this paper we present a model of distributed query processing and propose two performance metrics based on this model. These metrics may be used to compare various data distribution strategies and system architectures in an implementation independent manner.

1. Introduction

A Distributed Relational Database Management System (D-RDMS) is implemented on a hardware base consisting of multiple processing units able to work in parallel on a single query. For example, Distributed Ingres [STON76,STON83] is a relational database management system built on a collection of VAX 11/780 and VAX 11/750 processors connected via an Ethernet. Queries originate on one processor, then split into subqueries which execute in parallel on participating machines. In addition, many of the database machine proposals are similar to a D-RDMS in that the processing of an individual query is split into subqueries which execute in parallel on similar processing units. In the Direct [DEWI78] architecture, a collection of PDP-11/23 processors with associated CCD memories are attached to a PDP 11/45 front end via a cross-point switch. Queries are processed by paging data blocks into the CCD memories and examining them in parallel with the 11/23 processors. Lastly, in the CASSM

[SU__75] architecture microprocessors are associated with individual tracks of a fixed head disk attached to a host processor. Data is distributed across the tracks and examined in parallel as it is being read off of the disk. In all of these architectures the data elements are distributed across some number of processing sites thereby permiting parallelism within a single query.

The purpose of this paper is to formulate a model of distributed query processing. The input for the model is a description of the way in which data is distributed across the sites and the amount of data requested by a query. From this information, the expected amount of requested data located on each site is estimated. These estimates are statistical in nature and depend neither on semantic knowledge about the queries nor on a detailed simulation of query processing. The output of the model is the expected number of data elements accessed on each site under a variety of assumptions about the way in which data is stored on individual sites and distributed across sites. Our results provide a measure of the way in which the resource requirements of a query are distributed under various data organizations. Moreover, these results could be used in a more general architectural analysis of data base machines or distributed databases.

This problem has many points in common with the so-called block estimation problem, i.e. given the distribution of tuples to disk blocks and number of tuples requested by a query, estimate the number of blocks touched. [CHRI83] presents a summary of the earlier work of [CARD75], [RIES79], [SILE76] and [YAO_77]. Those studies present equations for the mean number of blocks touched under random or sequential access, assuming the distribution of tuples to blocks is constant and selection is either with or without replacement. [CHRI83] then extends the earlier models to non-constant tuple distributions, assuming that individual tuples are equally likely to be selected and successive

selections are uncorrelated.

In the next section our models of a D-RDMS and distributed query processing are presented along with the parameters used to estimate system workload. The following section presents derivations of equations for the probability that each site is accessed and the mean and variance of the distribution of the number of data units examined on each site.

2. Distributed Query Processing

A D-RDMS can be broken into four main units: the data items themselves, the underlying hardware, the supplementary fast access paths, and the query processing mechanism. Each of these units is described in the following subsections and descriptive parameters defined.

2.1. Data Description

We assume a relational database in which all data appears to the user as fixed format tables, or *relations*. The *cardinality* of a relation is the number of tuples (records) stored in the relation. Each tuple has some number of data fields, or *attributes*. For example, suppose we have a database containing records for 500 professors in a college. Each record has four fields: PNAME, an alphanumeric field containing the name of the professor; PNO, an integer field containing a unique professor identifier; RANK, an integer field containing a code for the professor's current rank; and SALARY, a real valued field containing the professor's salary.

The unit of transfer between secondary storage and memory is one *block*. Each block contains some number of tuples. The exact number depends on the width of the individual tuples, the block size and whether or not the blocks are filled to capacity. Assume that the number of tuples on each block is some small constant, B. In our example, if each record is 200 bytes long, blocks are 1000

bytes long and filled to capacity, then B is equal to 5.

2.2. Distributed Hardware

The hardware is modeled as a collection of N sites interconnected in some manner. The exact nature of this interconnection is not important. Each site has a collection of associated data and a single processing unit. For example, sites in Distributed Ingres would correspond to VAX, sites in DIRECT to 11/23 processors with associated CCD memories and sites in CASSM to microprocessors with associated disk tracks.

2.3. Distribution Criteria and Fast Access Paths

The most rudimentary query processing strategy is to simply scan all records and select those with attributes satisfying the qualification clauses. This is the technique used in the CASSM architecture. More complex systems utilize a variety of indexing strategies to reduce the amount of data which must be examined during query processing. In addition to providing fast access paths, these strategies will group and order the data. This section describes some of the ways in which a relation can be distributed across the N sites and the subrelations organized on each individual site. No data are replicated and all sites are assumed to have identical data organizations.

2.3.1. Distribution Criteria

The partitioning of data across the N sites in the distributed system is defined by *distribution criteria*. Distribution criteria are assumed to be of the form:

f(attribute) = value.1, location = site.i
f(attribute) = value.2, location = site.j
.
.
.
f(attribute) = value.G, location = site.k

where f is some function on the values of an attribute which partitions the data into a collection of sub-relations. For example, one possible distribution criterion for the PROFESSORS relation described above is:

f(PNO) = PNO div 100, location = PNO div 100

where div is integer division. If the 500 professors have PNO values in the range [1,1000], this criterion will partition the professor records into 10 disjoint subsets and allocate those subsets to sites 1 through 10.

The distribution criteria serve two purposes. First, they allocate tuples to sites and second, they group together tuples with equal f(attribute) values. The tuples with a single f(attribute) value will be called a *group*. Define G_i to equal the number of groups on site i. Assume that the number of tuples in each group is discribed by a random variable T with distribution {T}. The distribution criteria, along with a count of the number of tuples actually associated with each value, can be used by the distributed query processing mechanism to restrict the number of sites which are searched for qualifying tuples, as described below.

2.3.2. Global Index

A global index associates f(attribute) values with sites and maintains a count of the number of tuples with a given value stored on each site. A global index, unlike distribution criteria, does not group the data in any particular way. For example, if the PROFESSORS relation is distributed by PNO as described above, a global index on (SALARY div 10) would associate sites with salary

values. The tuples are assumed to be randomly distributed across the various sites. G_i and T are defined as above, where the sum of the G_i's may be greater than G.

2.3.3. Local Indexes

Once the data have been distributed to individual sites, the data is placed in blocks on a secondary storage device. This section describes several indexing techniques which provide fast access paths to tuples containing particular attribute values. A primary index assigns tuples to blocks by clustering and perhaps ordering them by indexing attribute. A secondary index associates attribute values with tuple locations. If no index is present, the tuples are assumed to be randomly located on blocks.

2.3.3.1. Clustering Hash Index

A clustering hash index provides a fast access path to a collection of tuples with equal f(attribute) value where f is any function over all the possible attribute values. The *extent* of the index is the number of distinct f(attribute) values (i.e. the number of clusters). Each block contains tuples from a single cluster and all blocks for a particular cluster are stored together in physical proximity. For example, assume that the professor records on each site are clustered by the first letter of PNAME into 26 clusters. A clustering hash index will provide the address of the first block of each cluster. If there is a distribution criteria or global index with identical f(attribute) function, there will be G_i clusters on site i and T tuples per cluster. If not, define G_i to be the number of clusters on site i and T the number of tuples per cluster.

2.3.3.2. Secondary Index

A secondary index provides a fast access path to individual f(attribute) values. It has no effect on the arrangement of data. For example, suppose the

PROFESSORS relation were stored in a Clustering Hash Index structure on the first letter of PNAME. A secondary index on RANK would associate a set of tuple locations (block addresses) with each RANK value. The extent of this index would be the number of distinct RANK values. The set of locations is assumed to be randomly distributed over secondary storage (i.e. PNAME is uncorrelated with RANK). G_i is the number of distinct f(attribute) values on site i and T is the number of tuples per value, as above.

2.4. Queries and Query Processing

In the relational data model, a query requests a collection of tuples by specifying a qualification which tuples must meet. This section describes a model of simple queries spanning a single relation and the way in which fast access paths are used during query processing.

The queries which will be considered in this paper are of the form:

```
retrieve (tuples)
where f(attribute) = {value.1, ... , value.n}
```

where f is any function over the set of attribute values. The simplest kind of query is one where f is simply the identity function and there is a single qualifying value. For example, to retrieve all professors with PNO equal to 457, the following query could be executed:

```
retrieve (professor tuples)
where PNO = 457
```

A somewhat more complex query might retrieve all professor tuples whose names begin with a letter in the range A through F:

retrieve (professor tuples)
where f(PNAME) = {A,B,C,D,E,F}

and the value of f(PNAME) is the first letter of PNAME. Note that the particular function and values chosen to express a given query are not unique since *every* query can be expressed as the identity function and a (possibly large) collection of values. The *scope* of a query is the number of distinct values in its qualification clause. Note that the scope of a query is dependent on the way in which the query is expressed and is not equivalent to the *selectivity*, or number of qualifying tuples present in the database.

A query is said to *match* an index if the indexing function and the query function are identical and refer to the same attribute. In this case, the scope of the query is the number of distinct groups of data which must be examined using the index. If a global index or distribution criteria are available, they will restrict the number of participating sites to just those actually containing qualifying tuples. If no global index is available, all sites must execute the query. There are several modes in which data can be examined on individual sites. If no matching index is available, the data blocks must be sequentially scanned until all of the qualifying tuples (values) have been located. If a global index is present, the number of such tuples will be known and scanning can be stopped once that number have been encountered. If a matching secondary index is available, the qualifying tuples (values) may be located by only examining at most a number of blocks equal to the scope of the local query. If a matching clustering hash index index is available, the qualifying tuples will be grouped into clusters and the number of *cluster* accesses will be equal to the scope of the query. Table 1 summarizes the processing required for each of the 5 situations. Scope(i) is the number of f(attribute) values *accessed* on site i.

Matching Index	Query Processing	Access Unit
No Matching Index	Scan all	block
Global Index only	Scan for scope(i) units on site i	tuple
Distribution Criteria only	Scan for scope(i) units on site i	tuple
Clustering Hash Index	Randomly access scope(i) units on site i	cluster
Secondary Index	Randomly access scope(i) units on site i	tuple

Table 1. Query Processing

3. Models and Analysis

In this section formulas are derived for the probability that a given site is accessed and the distribution of the number of block accesses per site as a function of the way in which data (tuples) are distributed among sites, groups, clusters and blocks.

3.1. Definitions and Assumptions

Table 2 summarizes the parameters defined above.

A query executes by selecting some number of data units (tuples, clusters, groups) from the total number stored in the database. The first statistical assumption is that these selections are made *independently*. E.g. data units are replaced after each selection and the probability of selecting any given unit at each draw remains constant. This implies that a data unit may be selected twice by a query, however if the database is large with respect to the total number of data units chosen the probability of this is small and the assumption is reason-

Parameter	Definition
N	# Sites
G	# Groups/Clusters
B	# Tuples per Block
{T}	# Tuples per group or cluster
Gi	# groups/clusters on site i, i=1,N

Table 2. Parameters

able.

The second statistical assumption is that all values of a fixed scope are equally likely to be selected by a given query independent of the actual number of tuples with that value. In the absence of any prior knowledge of the expected data access patterns, this is a reasonable assumption.

3.2. Probability that site i is accessed

If neither the distribution criteria nor any global index match the query, all sites are accessed. Otherwise, the number of f(attribute) values on site i is equal to G_i. Let k be the scope of the query. Then the probability that none of the k units independently chosen by the query are on site i is

$$(1-\frac{G_i}{G})^k$$

and hence the probability that site i is chosen is

$$1-(1-\frac{G_i}{G})^k.$$

3.3. Distribution of Random Accesses on Site i

Let the number of blocks randomly accessed on site i be equal to the random variable Y_i and the number of f(attribute) values chosen on each site (i.e. scope(i)) be equal to the random variable X_i. X_i has the binomial distribution with parameter G_i/G and mean $k(G_i/G)$. Let Z be the number of blocks per f(attribute) value. If the data is accessed via a secondary index, Z is distributed as T (i.e. there is one tuple per block). If the data is accessed via a clustering hash index, Z is distributed as T/B (i.e. the tuples are grouped onto blocks). Then Y_i is equal to the sum of X_i units of size Z

$$Y_i = \sum_{1}^{X_i} Z$$

By Wald's equations, Yi has mean equal to $E(X_i)E(Z)$ and variance $Var(Z)E(X_i)$.

3.4. Distribution of Sequential Accesses on Site i

If no local indices are available, the tuple count in the global index or distribution criteria may be used to limit the amount of data to be scanned. Let the number of qualifying tuples on site i be equal to Q_i. Note that Q_i is distributed as Y_i for Z equal to T.

Let V_k equal the number of tuples scanned between the (k-1)st and kth tuple selected by the query. V_k has a geometric distribution with parameter Q_i/C_i, where C_i is the cardinality of the subrelation on site i. Let W_i be the number of tuples examined on site i, where

$$W_i = \sum_{k=1}^{Q_i} V_k.$$

By Wald's equation, $E(W_i) = E(Q_i)E(V_k)$. and hence, the expected number of blocks examined on site i is $E(W_i)/B$, where B is the number of tuples per block, as defined above. The variance is equal to $E(Q_i)V(V_k)/B^2$.

4. Conclusion

In this paper we have presented a general model for distributed processing and derived several performance metrics based on this model. These metrics allow various comparisons to be made among different data distribution strategies by varying the values of C_i and B. In addition, they provide a means of estimating the distributed workload for input into queueing or other architectural models. In evaluating an existing system, these values will be known and well defined. In design work, however, they are not known. Derivations of analogous results assuming that these parameters are random variables from a given probability distribution is a desirable extension.

5. Acknowledgements

I would like to thank my advisor, Mike Stonebraker, for his unending patience in reading earlier drafts and suggesting alternate assumptions and approaches. I would also like to thank Toni Guttman for reading earlier drafts and offering helpful suggestions.

6. References

[CARD75]
A.F. Cardenas, "Analysis and Performance of Inverted Data Base Structures", CACM, Vol. 18, No. 5 (May 1975), pp. 253-263.

[CHRI83]
S. Christodoulakis, "Estimating Block Transfers and Join Sizes", Sigmod Record Vol. 13, No. 4 (Proceedings of SIGMOD 83, San Jose, CA), May 1983.

[DEWI78]
D.J. Dewitt, "DIRECT– A Multiprocessor Organization for Supporting Relational Data Base Management Systems," Proc. 1978 ACM-SIGMOD Conference on Management of Data, Austin, TX May 1978.

[LANG82]
A.M. Langer and A.W. Shum, "The Distribution of Granule Accesses Made by Database Transactions", CACM, Vol. 25, No. 11 (November 1982), pp.831-832.

[POTI80]
D. Potier and Ph. Leblanc, "Analysis of Locking Policies in Database Management Systems", CACM, Vol. 23, No. 10 (October 1980),pp.584-593.

[RIES79]
D. R. Ries, "The Effects of Concurrency Control on Database Management System Performance", PhD. Dissertation, Computer Science Department, U.C. Berkeley, April 1979.

[SILE76]
K. F. Siler, "A Stochastic Evaluation Model for Database Organizations in Data Retrieval Systems", CACM, Vol. 19, No. 2 (February 1976),pp. 84- 95.

[STON76]
Stonebraker, M. et. al., "The Design and Implementation of Ingres", TODS 2,3, September 1976.

[STON83]
M. Stonebraker, J. Woodfill, J. Randstrom, M. Murphy, J. Kalash, M. Carey and K. Arnold, "Performance Analysis of Distributed Data Base Systems", Database Engineering, 1983.

[SU_75]
S.Y.W. Su, and Lipovsky, G.J. "CASSM: A cellular system for very large data bases." Proc. Int. Conf. Very Large Data Bases, Sept. 1975, pp. 456-472.

[YAO_77]
S. B. Yao, "Approximating Block Accesses in Database Organizations", CACM, Vol. 20, No. 4 (April 1977), pp. 260-261.