1983

An Analytical Model for Evaluating Database Update Schemes

Kathryn C. Kinsley
katekinsley@yahoo.com

AN ANALYTICAL MODEL FOR EVALUATING DATABASE UPDATE SCHEMES

by

Kathryn C. Kinsley

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science at the University of Central Florida Orlando, Florida

April 1983

Major Professor: Dr. Charles E. Hughes

Abstract

A methodology is presented for evaluating the performance of database update schemes. The methodology uses the M/Hr/1 queueing model as a basis for this analysis and makes use of the history of how data is used in the database. Parameters have been introduced which can be set based on the characteristics of a specific system. These include update to retrieval ratio, average file size, overhead, block size and the expected number of items in the database.

The analysis is specifically directed toward the support of derived data within the relational model. Three support methods are analyzed. These are first examined in a central database system. The analysis is then extended in order to measure performance in a distributed system. Because concurrency is a major problem in a distributive system, the support of derived data is analyzed with respect to three distributive concurrency control techniques -- master/slave, distributed and synchronized.

In addition to its use as a performance predictor, the development of the methodology serves to demonstrate how queueing theory may be used to investigate other related database problems. This is an important benefit due to the lack of fundamental results in the area of using queueing theory to analyze database performance.

ACKNOWLEDGEMENTS

The author wishes to thank her adviser, Dr. Charles E. Hughes for his direction of this work. It has been a pleasure to work with him. His patience and insight is appreciated.

In addition, the author wishes to thank her research committee, namely, Drs. Mostafa Bassiouni, Ron Dutton, Ratan Guha and Brian Petrasko, each of whom contributed to the quality and direction of this dissertation. A special thanks is due to Dr. Terry J. Frederick, head of the Computer Science Department, who has provided support and encouragement throughout the author's graduate academic career.

And of course, the author expresses her appreciation to her husband for his understanding and encouragement throughout this endeavor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

A	is the arrival rate to a node
b	is the block size, indicating the number of ddr tuples accessed in one I/O operation to the external device that stores ddrs
C	is the service rate for a lock
C2	is the square of the coefficient of variation
ddr	is a dynamic derived relation
k	is the culling factor
ki	is the culling factor for the ith ddr
n	is the average number of ddrs per defining relation
N	is the number of nodes in the distributive database
P	is the probability that an arrival is a ddr retrieval
pdf	is the probability density function
pi	is the probability that an arrival is a ddr retrieval for the ith ddr (p1+p2+...+pn=P)
Pd	is the probability that a defining relation will reside locally
Pl	is the probability that an update will be local
Q	is the probability that an arrival is an update, Q+P=1
q	is the queue length
R	is the utilization factor (R=X*A)
SUM1	is (k1+k2+...+kn)/b
SUM2	is (k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2
1/U	is the service time it takes to do one update

ui	is the service rate to retrieve the ith ddr
X	is the mean service time at the node
X2	is the second moment of the probability density function (pdf)
W	is the average wait time at the node

CHAPTER I

INTRODUCTION

This chapter discusses the objectives of the dissertation, introduces basic terminology and outlines the organization of the presentation.

A. Objective

The major objective of this dissertation is to present a methodology for evaluating database update schemes. The methodology uses the M/Hr/1 queueing model as a basis for this analysis. The M/Hr/1 model was chosen because it allows for several classes of customers, each with a different service time, to be serviced by one processor. This characteristic is consistent with an update/retrieval system in which different actions can occur; however, only one action can occur at a time and each of these actions can have a different service time.

The methodology also makes use of the history of how data is used in the database. The major historical parameter is the update to retrieval ratio. This parameter is essential in determining the performance of specific update schemes. Additional parameters have been introduced which can be set based on the characteristics of a specific system. These include average file size, overhead and the expected number of items in the database.

The analysis is specifically directed toward the support of derived data within the relational model. Three support methods are analyzed. These are first examined in a central database. The analysis is then extended in order to measure performance in a distributed system.

Because concurrency is a major problem in a distributive system, the support of derived data in this environment is analyzed with respect to three concurrency control techniques -- master/slave, distributed and synchronized. The distributive database model uses parameters that reflect the amount of nonlocal activity and the cost of transmitting data into the networks. The parameters can be tuned to reflect the characteristics of a specific system.

In addition to its use as a performance predictor, the development of the model presented here serves to demonstrate how queueing theory may be used to investigate other related database problems. This is an important benefit due to the lack of fundamental results in the area of using queueing theory to measure database performance.

B. Database Termonology

In a relational database, the data can be thought of as a collection of tables called relations. The tables are made up of columns, which are called attributes, and rows which are called tuples. More formally, given a collection of sets Dl, D2, ..., Dn (not necessarily distinct), R is a relation on these n sets if it is a set of ordered n-tuples dl,d2,...dn such that dl belongs to Dl, ..., dn belongs to Dn (Date 1979).

Most relational database systems provide the facility for supporting dynamic derived relations (ddrs). A dynamic derived relation is defined from existing relations and reflects updates made to its defining relation(s). In other words, it is a "dynamic window" of the database in that when an update is made to any one of its defining relations, the dynamic derived relation is automatically updated in accordance with its definition.

A dynamic derived relation and its defining relations are shown in Figure 1. In this case, since X is defined as AUB, the union of A and B, any insertion into A or B would also be inserted into X. Any deletion from A would be deleted from X if the tuple was not in B, and similarly for a deletion from B.

A

NO	LT
1	a
2	b
3	c
4	d

B

NO	LT
2	b
3	c
5	e
6	f

X

NO	LT
1	a
2	b
3	c
4	d
5	e
6	f

Figure 1. An example of a dynamic derived relation X formed by AUB.

There are two basic approaches to the support of dynamic derived relations -- potential form and actual form. Implementation using potential form involves describing the dynamic derived relation so that it can be generated when needed. This description could use access paths or a formula of the dynamic derived relation. The dynamic derived relation is constructed each time it is involved in a query. This method has the obvious advantage of saving memory space; however, the cost of generating the dynamic relation may occur many times because the same result may have to be computed again and again.

Implementation using actual form involves physically storing each dynamic derived relation. Therefore, the relation "actually" exists. Any updates made to the defining relation must be explicitly reflected in the ddr. An advantage of using actual form is that, because the dynamic derived relation physically exists, the derived results do not have to be regenerated each time the ddr is needed. However, more storage space is used to store the ddrs and the reflection of defining relation updates is sometimes difficult and time-consuming.

C. Queueing Theory Terminology

Queueing theory has been used as a common tool for analyzing computer system performance. The characteristic that a job arrives in the system and waits for service in a queue is consistent with the queueing theory modelling process. In general, the queue is FIFO; however, other queueing disciplines can be used.

The notation which is used in this dissertation to specify the queueing model is the widely used A/B/n where A and B represent the interarrival and service time probability density, respectively. The probability density can be one of three types: M, G or D which are characterized by exponential (Markov), arbitrary (general) and deterministic probability density, respectively. The last value in the notation, n, indicates the number of servers. Thus, an M/M/l queueing model is a system with one server and exponential interarrival-time and service-time distribution.

The M/M/l model is the simplest of the queueing systems and, yet, is nontrivial enough to model many different systems. The exponential density function has the unique property that the service time remaining for a customer is not affected by the amount of service time the customer has previously accumulated, i.e., it has no memory. This memoryless property allows the model to be described totally in terms of how many customers are in the system.

A state-transition diagram for an M/M/1 system is shown in Figure 2. Customers enter the system at arrival rate A and are serviced at service rate U. Each time a customer arrives, the system moves to the next state, and each time a customer is serviced and leaves the system, the system moves back to the previous state. Therefore, the entire system is described in terms of the number of cutomers in the system.

By finding the probability of being in each state, we have found the probability of having one customer in the system, two customers in the system and so on. Once these probabilities have been computed, the expected number of customers and expected wait time can be found, as well as other statistics.

The M/G/1 queue has Markov arrivals and uses a single processor; however, the service time is general in nature. That is, the service time is not necessarily of the memoryless type. The state description, therefore, must consider expended service time as well as the number of customers in the system. In order to deal with this in a manner consistent with the M/M/1 analysis, we examine selected points in time, i.e., state-transitions occur at customer departure instants and the state variable is the number of customers left behind by the departing customer. Thus, the M/G/1 queue can be analyzed to yield the expected number of customers in the system and the mean wait time.

This dissertation makes use of a special case of the general service time, the M/Hr/1 queue, which is shown in Figure 3. The M/Hr/1 queue has Markov arrivals and uses a single processor; however, the system allows for r parallel stages. Each additional stage increases the variability of the service time and produces a service time probability density that is referred to as "hyperexponential".

When a customer enters the service facility, he will proceed to stage one with probability P1, state two with probability P2, ... , stage r with probability Pr, where P1+P2+...+Pr=1. The customer will then be serviced according to the service time indicated for that stage and depart the system. At that time, another customer can enter the service facility. By specifying which stage within the service facility the customer occupies, we can analyze the system with respect to average service time and mean wait time (Kleinrock 1975).

D. Organization of Presentation

This dissertation has been organized into five chapters. Chapter I (the current chapter) presents the objectives of the thesis and the terminology used in the area of database and queueing theory. Chapter II discusses

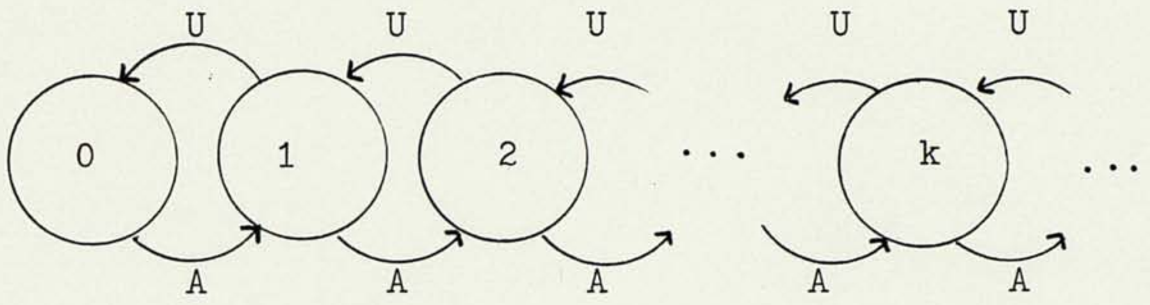


Figure 2. An M/M/1 state-transition diagram.

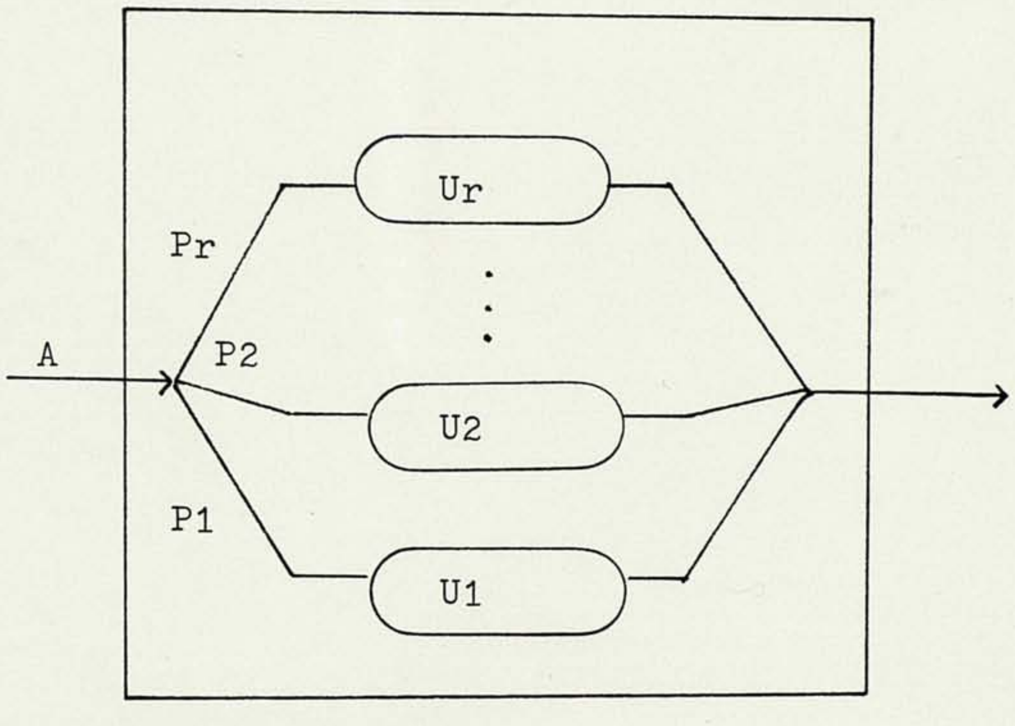


Figure 3. The M/Hr/1 model.

previous research with respect to derived data and formal analysis of database performance in both centralized and distributive databases. Criteria for designing a distributive database are also discussed in this chapter.

All research contributions of this dissertation are found in Chapters III and IV. The formal queueing theory analysis begins with Chapter III. The characteristics to be modelled for each ddr support method are discussed and a specific queueing model is chosen. The support methods are then analyzed in a central database system.

Chapter IV extends the analysis in Chapter III to the distributive database system. Information flow is incorporated in order to find the response time for a transaction. The ddr support methods within three distributive concurrency control methods are analyzed: master/slave, distributed control and synchronized control. Graphical comparisons of both ddr support methods and the concurrency control systems are made.

Chapter V discusses the conclusions that can be drawn from the results of this dissertation. The performance of the ddr support methods and the concurrency control methods are compared. The unique characteristics of the methodology used to analyze the performance of the schemes are outlined. Areas of future work are indicated.

CHAPTER II

SURVEY OF THE LITERATURE

In this chapter, important references regarding dynamic derived relations and storage and retrieval performance in both central and distributed database systems are presented.

A. Dynamic Derived Relations

Background

In 1971 the report of the CODASYL group of the Data Base Task Group defined derived data as data that is derived procedurally from related data items instead of being explicitly stored and directly retrieved (CODASYL 1971). The relational model, introduced by Codd (1970), extended this concept by introducing derived relations. Derived relations have been used to support users' views, integrity constraints (Stonebraker 1975) and access control (Eswaran 1975).

There are two types of derived relations--static derived relations and dynamic derived relations (Chamberlin 1975). A static derived relation becomes independent of the relation(s) used in its definition immediately after it is populated. Once populated, static derived relations can be explicitly updated by insert, delete and modify operations. Hence, immediately after their creation, static derived relations behave exactly like base relations (Wiederhold 1977). On the other hand, a dynamic derived relation automatically reflects changes made to the relation(s) used in its definition. In other words, it is a "dynamic window" of the database in that, when an update is made to its defining relation(s), the dynamic derived relation is implicitly updated in accordance with its definition.

There are two basic methods of implementing dynamic derived relations--the actual results method and the potential results method. Implementation using the potential results method involves storing a formula or access paths to represent a dynamic derived relation. Hence, the dynamic derived relation does not physically exist but can be generated from the stored formula when it is needed. At this time, updates to the defining relation(s) are automatically reflected in the newly generated dynamic derived relation. System R and INGRES are two examples of relational systems which use the potential results method (Astrahan 1976, Stonebraker 1975, Stonebraker 1976).

Implementation using the actual results method involves physically storing each dynamic derived relation.

Within this implementation technique, if a relation used in the definition of a dynamic derived relation is changed, a procedure must be invoked to immediately update the dynamic derived relation. Therefore, in maintaining dynamic derived relations in actual form, an explicit update to a base relation causes implicit updates to all dynamic derived relations which the base relation defines. An advantage of using actual form is that, because the dynamic derived relation physically exists, the derived results do not have to be regenerated each time the dynamic derived relation is needed. The RAQUEL II DBMS and, to some extent, IS/1-PRTV are examples of systems which use the actual results method (Dutton 1978, MacDonald 1975, Todd 1976).

Although Kim (1979) states that the potential method of supporting dynamic derived relations is better for most applications, this strategy is particularly vulnerable when a ddr is frequently accessed and the recalculation involves a large amount of tuples. Unfortunately, no formal analysis has been done to validate or invalidate Kim's or any other such claim.

Implementation of Dynamic Derived Relations

Three algorithms for the support of dynamic derived relations are shown in Figures 4, 5 and 7.

```
INPUT.  A dynamic derived relation definition

OUTPUT. A dynamic derived relation in actual form

METHOD. This procedure is called to calculate a ddr
         when the ddr is queried.


PROCEDURE POTENTIAL (ddr definition)
BEGIN
   recalculate according to the definition
END; (* POTENTIAL *)
```

Figure 4. Potential form algorithm for the support of ddrs.

```
INPUT.  A base relation

OUTPUT. All dynamic derived relations defined by the
          base relation in updated form.

METHOD. This procedure updates all the ddrs defined by
          a base relation, whenever the base relation
          is updated.

PROCEDURE IMMEDIATE-ACTUAL (updated base relation)
BEGIN
  FOR all ddrs defined by the base relation DO
      Apply update schemes of Figure 6
  END; (* IMMEDIATE-ACTUAL *)
```

Figure 5. Actual form algorithm which immediately updates ddrs when a defining relation is updated.

	Definition of X	Operation	Insertion Update
1.	AUB	union	XUI
2.	A∩B	intersection	XU(A∩I)
3.	B-A	difference	XU(I-A)
4.	A-B	difference	X-I
5.	B(list)	projection	XUI(list)
6.	B(qual)	selection	XUI(qual)
7.	A*B	join	XU(A*I)

(a) Insertion Updates for Dynamic Derived Relations

	Definition of X	Operation	Deletion Update
1.	AUB	union	X-(D-A)
2.	A∩B	intersection	X-D
3.	B-A	difference	X-D
4.	A-B	difference	XU(A∩D)
5.	B(list)	projection	X=(B-D)(list)
6.	B(qual)	selection	X-D
7.	A*B	join	X-(A*D)

(b) Deletion Updates for Dynamic Derived Relations

Figure 6. Updates schemes for dynamic derived relation using actual results method (Kinsley and Driscoll 1979). Dynamic derived relation X is defined by relation B (and, if needed, A). Relation B has been updated. Relation I holds the tuple inserved in relation B. Relation D holds the tuple deleted from relation B. Join refers to natural join on a key.

INPUT. A dynamic derived relation

OUTPUT. A dynamic derived relation in updated form

METHOD. This procedure updates the ddr whenever the ddr is retrieved. Until that time, update tuples are collected in a differential file. A check is made to see if the tuple has been previously entered into the defining relation's update tables. If it has, and the transaction types are the same (i.e. insertion or deletion), no action occurs. If the transaction types do not match, the tuple is deleted from that update table and inserted into the update table that matches its transaction type. If the type is not in the defining relation's update tables, it is inserted into the corresponding update table. When all tuples have been checked, the update schemes of Figure 6 are invoked, using the I and D tables.

```
PROCEDURE DEFERRED-ACTUAL (ddr)
BEGIN
  FOR all tuples used to update the defining relation DO
    IF tuple already a member of defining relation's
        update files THEN
     IF update file type is same as tuple type THEN
        no action
     ELSE
        BEGIN
         delete tuple from update file;
         insert tuple into update file with its type
        END
   ELSE (*TUPLE IS NOT IN UPDATE FILES*)
     insert tuple into update file with its type
 END; (* END OF BUILDING I AND D TABLES *)
 apply update schemes of Figure 6
 END; (*DEFERRED-ACTUAL*)
```

Figure 7. Deferred actual algorithm.

The algorithm in Figure 4 maintains ddrs in potential form and thus must be recalculated whenever a dynamic derived relation is used in a query. The algorithm in Figure 5 supports the ddr in actual form. Each update to a base relation causes all dynamic derived relations defined by the updated base relation to also be updated. The algorithms of Figures 5 and 7 both use the update schemes shown in Figure 6. These were first developed by Kinsley and Driscoll (1979). Related, independent work has been done by Osman (1979). The basis on which these schemes were developed is as follows.

The procedure for inserting a tuple into a relation can be represented as the union of the relation to be updated and the relation formed from the tuple to be inserted (Date 1979). For instance, if relation B was to be updated by inserting a tuple contained in relation I, the update could be represented as $B \cup I$, where the relation formed by $B \cup I$ would be the updated B and would replace the "old" B.

Now assume X is a ddr formed by $A \cap B$. If relation B were updated by the insertion tuple which forms relation I, then X can be implicitly updated by creating $A \cap (B \cup I)$ where $B \cup I$ is the updated B and $A \cap (B \cup I)$ is the updated X. However, the expression $A \cap (B \cup I)$ is equivalent to $(A \cap B) \cup (A \cap I)$. Therefore, $X \cup (A \cap I)$ is the updated X. Using this approach, a scheme has been developed for insertion of a tuple into a dynamic derived relation under the standard set of database operations. These insertion schemes are shown in Figure 6a. Each scheme takes advantage of the fact that X physically exists in actual form.

The operation of deleting a tuple from a relation can be represented as the relative complement of the relation to be updated and a relation formed from the tuple to be deleted. For instance, if relation B were to be updated by deleting the tuple contained in relation D, the update could be represented as B-D, where the relation formed by B-D would be the updated B.

Now assume X is a dynamic derived relation formed by AUB. If relation B were updated by the deletion tuple which forms relation D, then X can be updated by creating AU(B-D) where B-D is the updated B and AU(B-D) is the updated X. However, the expression AU(B-D) is equivalent to (AUB)-(D-A). Therefore, X-(D-A) is the updated X. Using this approach, a scheme has been developed for deletion of a tuple from a dynamic derived relation under every operation, except projection. As before, each deletion scheme takes advantage of the fact that X physically exists in actual form.

Modifying an existing tuple can be implemented as a deletion followed by an insertion. Therefore, all tuple update operations -- insertion, deletion and modification -- can be represented by using the insertion and deletion update methods just mentioned. An insertion update table (I) and a deletion upate table (D) must exist for each relation used to define a dynamic derived relation.

The algorithm presented in Figure 7 also maintains dynamic derived relations in actual form; however, it defers updates until the specific dynamic derived relation is queried. Updates are collected in differential files (Severance 1976) according to the algorithm, and the update schemes of Figure 6 are used to update the ddr. In this case, differential files I and D may hold more than one tuple. Also, the algorithm may reduce the total number of tuples involved in the actual update due to a culling process that is carried out when update tuples are added to the differential files. The same tuple used in more than one update is reduced to only the last update before the database is accessed.

In summary, very little research has been done with respect to the best way to support ddrs. Three algorithms have been presented in this subsection - one using the potential method and two using the actual method. The proposed research will analyze the cost/performance of these three methods.

B. Formal Analysis of Database Performance in a Centralized Environment

File Organization Analysis

Much of the research regarding database performance has been directed at different types of file organization. No specific database model is specified.

Lowe (1968) examines memory utilization and retrieval performance for inverted file with adjustable bucket size and linked list file organizations. The analysis makes use of Ziph's law which involves f, the frequency of occurrence of distinct items appearing in printed English text. Given N distinct index items and a unique number j $(1<=j<=N)$, Ziph's law states that the probability that an index item selected by the user of the system is the jth index is $p(j)-0.1/j$. The analysis of memory utilization and average response time is found for f and p having uniform distribution, p uniform and f having Ziph's distribution, and both having Ziph's distribution. The cases of Ziph's distribution and uniform distribution tend to represent the extremes in database retrieval. It was found that, with an appropriate assumption for f and p, a value for the bucket size can be selected and a reasonable decision concerning packing more than one list into single buckets can be made.

Collmeyer and Shemer (1970) investigate retrieval performance for spatial index (index records within the index are stored in physical order according to key), calculated index (hashing), and tabular index (multilevel index where each level has index records stored as logically sequential). The investigation focuses on the number of blocks transported from secondary storage to main memory in the course of the index search. No attempt is made to determine the effect of queueing delays. The mean time to retrieve a record is expressed in terms of the file organization parameter, characteristics of the storage device and search and overflow strategy used. A case study is presented. It was found that retrieval via hash was generally faster, but hashing is not well suited for sequential processing. Multilevel indexing is not good for insertion/deletion activity because one change may affect all levels of the index. Retrieval time for spatial index organization increases at a faster rate than the other organizations as the size of the file increases.

Cardenas (1973) discusses the factors which affect file organization performance and presents a tool to help in file structure selection based on the specific usage of the system. The parameters are database characteristics (e.g. number of records, record length, etc.), user requirements (e.g. frequency of query, frequency of update, mode of retrieval, etc.) and device specifications (e.g. block length, average access time, etc.). A model was designed which simulated three file techniques- binary tree, inverted file and multilist--on six existing databases. It was found that a combination of characteristics can cause a great deal of difference in performance based on the file technique used. This study was one of the few to incorporate database usage into its evaluation. Cardenas also noted that file organizations which do well in retrieval of data do not perform well in updating and vice versa. Further analysis on performance of inverted list file organization is presented by Cardenas (1975). He analyzes the inverted file organization within levels 2 and 3 of the DIAM (Direct Independent Access Model) hierarchy. Levels 2 and 3 involve the physical view and the mapping from the conceptual to the physical view.

Siler (1976) presents a stochastic model based on a statistical representation of the database. He measures data retrieval performance associated with a set of user queries, a particular database organization and a given database. Monte Carlo techniques are used to measure performance by estimating access frame movements and disk rotations in the stochastic represention of the database. The author compares inverted list, threaded list (records with the same key are linked together) and cellular list (a pointer exists to each bucket with at least one record in the specified key; all records in each bucket with the same key are linked together). Queries of different complexity are examined. It was found that when the query complexity increases, the pure inverted list organization significantly improves performance.

Yao (1977) presents a general model for physical database organization. This model has a hierarchical form with attributes at the highest level followed by keywords, accession lists and virtual records in descending order. He develops general cost formulas using the model and demonstrates how these costs are valid for inverted lists, threaded lists, cellular, indexed sequential and B-tree organization.

Performance Analysis Based on a Specific Database Model

Analysis based on the three database models--hierarchical, network and relational--are presented here. In most cases, file organizations at the physical level are considered with respect to a specific model. Performance in the relational model is based not only on the method of representing and referencing the database, but also on how the query is evaluated. For this reason, query optimization plays an important role in performance analysis of relational operations. However, research involving query optimization will not be discussed here except in conjunction with access methods and file organizations at the physical level.

Ghosh and Tuel (1976) and Lavenberg (1976) investigate performance based on the hierarchical model. Both papers are based on the IMS (Information Management System) DBMS , an IBM system which is one of the most well-known hierarchical systems. Ghosh examines six data manipulation commands of IMS in conjunction with three IMS access methods. IMS update commands of insertion, replacement and deletion are not considered. On analysis, it was found that the residual error does not have a symmetric distribution; therefore, the model did not account for all the factors affecting access time of the given IMS database. Further examination suggested that the interaction of the segment type and logical view are the important controllable factors which affect access time.

Lavenberg and Shedler (1976) present a queueing model based on an embedded finite state semi-Markov process. The IMS DBMS call is represented as a sequence of processor and I/O unit services. The processors are as follows:

1. determine the access path
2. search buffer for the block containing the access path
3. find the path segment within the path block
   determine if another path segment needs to be accessed
4. perform activities associated with block transfer

5. process target segment
6. I/O transfer

Database calls can compete concurrently for I/O and processor units. A priority rule exists to manage the contention. I/O and processor units can perform in parallel. Expected service time and expected cumulative process time are computed. This paper develops one of the few queueing models for hierarchical database systems.

Teory and Oberlander (1978) develop an analytical model for the evaluation of network database systems. Specifically, they use the IDS (Integrated Data Store) DBMS which closely follows the CODASYL network model. Input to the model are the database characteristics (definition of master and detail records, page size, record lengths, etc.), workload specification (an application program) and hardware characteristics. The model produces main storage and secondary storage, I/O and CPU time requirements. It does not incorporate queueing delays into the model.

Several performance analyses have been done involving relational operations. Gotleib (1975) analyzed the complexity of the natural join operation with respect to memory utilization, I/O and CPU time. This research is directed toward the calculation only, with no consideration of storing the results or the definition. The analysis is assumed to be done in isolation with no contention for I/O and processors.

Blasgen and Eswaran (1977) primarily consider access time in their analysis of join, projection and selection. Four methods for evaluating the query are examined based on how the data is represented and referenced in the database. The four methods are indexing on the join columns, sorting both relations, multiple passes and use of key and key indices. Cases involving the existence (or nonexistence) of certain indices and clustering are explored. It was found that in different situations, different methods performed well and no one best method existed.

Yao (1979) presents a query optimizer which computes the cost for seven optimization techniques. The author compares access methods and takes into account the storage structure of the database.

Menon and Hsiao (1981) develop a G/G/1 queueing model to analyze a VLSI implemented equijoin algorithm of their own design.

In summary, while a great deal of research has been devoted to analyzing database performance, very little of this research has been directed toward measuring queueing delays. In a real world situation, this is an important aspect of any performance measurement.

C. Distributive Databases

Background

A distributive database is a database that is not stored in its entirety in one physical location, rather it is spread across a network of dispersed (but interconnected) computers (Enslow 1978). Any or all of the database could be distributed - the data itself, the description of the data and the database called the schema, and/or the programs that run the DBMS could be dispersed.

There are three major advantages to a Distributive Database Management System (dDBMS) (Rothnie 1975). First, dDBMSs are reliable. Because a dDBMS is constructed from multiple computers at multiple sites, it is not susceptible to total failure when one computer breaks down. Second, accessibility is improved over a system which has geographically dispersed users and a centralized DBMS because it is possible to store data where it is most frequently accessed. Finally, there is the ability to handle database growth and large database management more efficiently since large databases can be supported as several moderate sized DBMSs at different sites. If the database grows larger, another site can simply be added.

The distributive property of a dDBMS gives importance to certain design factors. One design factor which needs to be taken into consideration, if the data is to be distributed, is how to split the files. Typically, files are split according to geographical or functional considerations. For example, if an organization's branches are geographically dispersed, where each branch serves a unique set of customers, the obvious solution is to place customer files at the location of the respective site. A functional distribution might be a business which has personnel accounts payable, accounts receivable, inventory and order entry files used by the appropriate department only. A solution would be to assign a processor and file to each department.

Another consideration is whether or not to allow redundancy in the data. In a system where a file is used quite often by two different sites, the solution might be to store the file at both sites. Redundancy could also be advantageous in a Point-of-Sales (POS) environment where a central node holds a complete set of files and each local node maintains files that are relevant to its site. Update files could then be transmitted when the system is not busy. As an example, a credit balance could be kept at a store for those customers associated with the store, making credit checks local. No redundancy simplifies the reliability problem if one site fails and no other copy exists (Rothnie 1977).

Directory (also called schema) distribution is another design factor to consider. The directory is needed for parsing queries, optimizing access of the data and determining access privileges. The goal is to minimize storage costs as well as processing and communications for directory and data requests. There are three basic methods for handling directory distribution in a dDBMS -- centralized, distributed and local. In a centralized directory, the directory for the entire database is located at one processing node. All requests, including local requests, must go through this node. This may slow down response time in a system that is geographically dispersed over large areas. In a distributed directory, each node has an entire copy of the directory. In such a system, response time is improved but the cost of storing the directory in its entirety at each node may be prohibitive in any system where the database is large. A compromise to the above two methods is a system of local directories. A copy of that portion of the directory which pertains to the local node is stored at that node. A centralized directory may be used in conjunction with local directories. Although this resolves local requests locally, all external requests must still be routed through the central node.

Another design factor which needs to be considered is how to distribute the programs of the dDBMS. Situations may exist where programs must operate on an entire file rather than a small set of records in the file. In such a case, it may be feasible to locate the program at the node where the file exists. An example of this is a system with a centralized directory (seen as a file itself) and the program which updates the directory.

Two alternatives exist for distributing the programs. The Data Base Administrator (DBA) could either store the programs at a centralized node and then send copies of the programs to nodes which need them, or store the programs at nodes which use them heavily and then have them dispatched to other nodes as needed. In the first case, all requests must go through the central node which may slow response time. In the second, some method must exist which keeps track of what programs are stored where. This may introduce another directory and the corresponding overhead to maintain it. If the dispersment is such that all programs of the dDBMS are stored at each node, the problem of what is stored where is solved; however, this may not be feasible in a system where the programs are large and a limited amount of storage exists.

The last area for consideration is how to design the communications network to support interleaved message flow. Characteristics of communication media vary. ETHERNET, for example, provides a high band, low delay transmission that is suitable for short range dDBMS. ARPANET uses a packet switching network with a lower bandwith and higher delay. It is better suited for message sending over longer ranges. A point to point leased circuit configuration has a lower bandwidth and a lower delay. The requirements of the dDBMS must be considered in choosing the communications network.

Concurrency Controls in Distributive Databases

One of the most important considerations in the design of a shared database system is concurrency control. In a distributive database system, where files used in processing one query may be dispersed among several sites, the complexity of the concurrency problem is increased.

Concurrency is a situation where more than one user is allowed to access into an area of the database at one time. While such a situation can improve throughput, unrestrained concurrency can threaten the integrity of the database. To illustrate this, suppose that two users are trying to update a record consisting of SS#, NAME, ADDRESS AND PHONE. Each user reads the specified record. The first user then changes the address and writes back to the file, followed by the second user who changes the phone number and writes back to the file. After this series of events, the record's phone number update, but not the address update, will be reflected.

To avoid such interleaving of updates, transactions are used. A transaction is a series of commands issued by one user which is treated as one indivisible unit. Thus, in the previous example, the first user's read and update commands would be one transaction and the second user's read and update commands would be a second transaction. The first user to issue a transaction would lock the required record (or some other unit of the database containing the needed record), perform the commands in the transaction, unlock the record and allow a waiting transaction to proceed. This locking protocol is called two-phase locking (Bernstein 1979).

Distributive systems concurrency control can be divided into two basic groups -- centralized control and decentralized control. In a centralized concurrency control system, any conflicting read or write requests are resolved at a central site. In a decentralized control system, each site has its own concurrency control. Both types presently exist.

The centralized concurrency control makes decisions regarding conflicts based on the total picture of what is happening in the database. However, this type of system can involve high costs for the communication overhead, especially if the deadlock detector is several thousand miles away.

In a decentralized concurrency control system, communication is kept to a minimum; however, a site may abort a transaction simply because it does not have enough information to proceed. Thus each system has drawbacks and advantages.

Systems Using Centralized Concurrency Control

A centralized concurrency control system is presented by Menasce and Muntz (1979). This system strives to minimize the disadvantages of the centralized system by resolving conflicts at a site "near" to the sites involved, if possible. Two types of controllers exist--leaf controllers (LK), which are for database components local to the site, and non-leaf controllers (NLK), which resolve conflicts.

This system makes use of a wait-for graph which depicts when transactions are waiting for one or more other transactions to release a lock. A global wait-for graph is maintained at the primary site, while each LK maintains a subset of the graph consisting of transactions involved with the site's local resources. In addition, output parts and input parts are depicted in the local wait-for graphs showing where a transaction came from and where it is going next.

When a conflict occurs, it is sent to the NLK which is the parent of the transaction and LK. If this node cannot resolve the conflict, the next level up the tree is employed. The conflict may be propagated all the way to the root before a resolution is possible. Fortunately, this is not usually necessary. Deadlock is detected by finding a cycle in the wait-for graph.

A distributed version of INGRES is presented by Stonebraker (1979). Conflict resolution is done by one machine dubbed "THE SNOOP". A local concurrency controller (CC) exists at each site and receives parts of a transaction involving its local resources from the site where the transaction originated. If deadlock is detected at one site, the CC picks a transaction, backs it out locally and sends a message to the site where the transaction originated, which in turn notifies the other sites executing parts of the transaction.

If deadlock involves several sites, the transaction is in a wait state. The CC sends the name of the waiting transaction and the transaction causing the wait, to the SNOOP, which uses a wait-for graph to determine if a deadlock exists. If no deadlock is detected, the CC sets a flag to indicate that the transaction is committed.

To minimize downtime, if a site should crash, duplicate copies of database components are kept at different sites. This, of course, introduces the problem of maintaining these additional copies. One copy among the up-sites is designated as the primary copy. Only the primary copy is locked. When a site commits a transaction, it directs the system to update the copies.

Systems Using Decentralized Concurrency Control

Two decentralized concurrency controls, WAIT-DIE and WOUND-WAIT, are discussed by Rosenkrantz, Stearns and Lewis (1978). In both, a unique number is assigned in ascending order to each transaction. This number is used to indicate the age of the transaction. In the WAIT-DIE system, if the requestor causing the conflict is older than the locking transaction, the requestor waits, or else the requestor is aborted, rolled back and restarted.

In the WOUND-WAIT system, if the requestor is younger, he waits. If the requestor is older and the transaction has not yet terminated at the site of origin, it is aborted, rolled back and restarted. If the transaction has terminated at the site of origin, no action occurs.

Although no deadlocks can occur under these systems, unnecessary restarts can occur. To improve on this, the authors allow transactions involving only local relations to be handled in a different manner, thus eliminating unnecessary local starts.

A decentralized concurrency control system using local locking and time stamps to update replicated databases is described by Gardarin and Wesley (1979). A time stamp holds the time a transaction entered the system. In this case, the time stamp is local to the site and also includes the site number. Each site maintains a lock table for each transaction which lists all resources requested.

Transaction execution proceeds as follows: First, the lock table and time stamp are sent to the sites with stored copies of the resource. At each site, the local resource is locked if no transaction with an older time stamp has locked the resource, unless that lock has been guaranteed. All younger transactions waiting for the same resource are aborted. A transaction never waits on more than one transaction per resource.

If all copies of a resource can be locked, the transaction is committed. A message is sent to the site to prevent cancellation by another transaction with an older time stamp. The site of origin then sends a message to the other sites involved to perform the update and unlock their copies of the database. If a transaction is aborted, the site that aborted sends a message to all involved sites, which unlock their copies. The site of origin performs a restart.

This system uses a wait-for graph to determine cycles. Duplicate files are maintained to help with recovery. A site that was down recovers by using differential files to update an old copy of the file.

A voting algorithm is used for concurrency control by Thomas (1979). Each site contains a total copy of the database; however, the author states that the sites could contain subsets of the database instead. When a transaction enters the system, a time stamp is created that consists of the local time, a site identifier and a transaction identifier. Update requests consisting of the updated value, a list of resources and their time stamps are sent to the sites involved.

A site votes to reject if one or more of the variables to be modified are older than the site of origin's variable. It votes to accept if the variables are current and no pending requests conflict. It votes to pass if the variables are current but there is a conflict with a request of higher priority. It defers voting if the variables are younger than the site of origin variables. When the site of origin has a majority of accept votes and no reject votes, it performs the operation and notifies all other sites to do so as well.

A decentralized concurrency control for a relational system, SDD-1 is described by Bernstein (1980), Hammer and

Shipman (1980), and Rosenkrantz and Hunt (1980). This system uses preanalysis to determine the amount of serialization needed. Each transaction is assigned a time stamp from a local logical clock with an appended site identifier. All write messages carry the time stamp and, if the time stamp is younger than the original data item, the transaction is not executed on that copy of the file. Only time stamps on WRITE transactions are stored.

The system uses a conflict graph to determine if transactions need to be serialized. This graph is not analyzed at run time but when the database is set up. This analysis produces transaction classes. Let Tw be a transaction write of a record and Tr be a transaction read of the same record. The following serialization rules apply:

1. if Tw precedes and conflicts with Tr, then Tw must precede Tr in serialization
2. if Tr precedes and conflicts with Tw, then Tr must precede Tw
3. if Twi conflicts with Tw, then Twi and Tw must be executed in their time stamp order

Locking is invoked only when a cycle is formed. Each Tr contains a list of classes with which it may conflict. The Tr is deferred until all earlier Tw commands in the specified class are processed. If a Tw in the specified class with a later time stamp has already been processed, the Tr is rejected.

The decentralized concurrency control introduced by Stearns and Rosenkrantz (1981) makes use of time stamps and allows a user to read values after a write transaction has started to update an entity but before the entity has been written back to the file. This is called the "before value". The authors state that this is desirable since in a traditional locking system a transaction might be aborted and, in that case, the requestor would ultimately receive the before value after waiting a period of time for a transaction to release a lock. Two times for committment are discussed, based on the following terms:

1. a transaction is said to be terminated when it is finished processing at all sites and has made a request to make its write values permanent

2. a transaction is said to be committed when it is terminated and cannot be rolled back

3. a transaction is said to be closed when its write values become official and it cannot be rolled back

A scheme for committing at termination requires that no older transaction receive a before value of a younger transaction and no younger transaction can write to an entity read by an older, nonterminated transaction. In this scheme, readers can receive after values of older transactions that have not yet terminated. If commitment is done when the system is closed, a younger transaction must wait to read the after value of the older transaction after it has closed.

Synchronized Concurrency Control

Synchronized concurrency control involves the use of a daisy chain (a ring) formed of all nodes. Updates travel around the ring to the required nodes. This synchronization insures consistency.

Ellis (1977) presents an algorithm for updating totally replicated databases which use a ring. Each database has three states, idle, passive and active. An idle database is not locked. An active database is locked and in the process of performing an update. A passive database is locked and knows that an update is coming. When an update is requested, it must move along the ring until it receives passive locks from every node and returns to the originating node. At that time, the originating node receives an active lock and is updated. The other nodes involved are then signaled to perform the update. This is done by sending the update around the daisy chain once again.

A synchronized control algorithm which uses a ring and can be used for both replicated and partitioned databases is presented by LeLann (1978). A particular message called a token, which serves the purpose of ticketing allocation requests, circulates the virtual ring of nodes. When a node receives the token, all updates for files residing at that node are taken from the token and updates for files residing at other nodes are appended to the token and sent to the next node in the ring. This ordering insures synchronized updating and thus eliminates conflicts in concurrently accessing the same item.

D. Formal Analysis of Database Performance in Distributive Database Systems

Much of the performance analysis in distributed database systems can be found in the areas of performance with respect to central vs. distributed controllers, allocation of resources and updating replicated files.

File Directory

Chu (1976) develops a queueing model to study the performance of file directory systems. Three directory placements are considered: a single master directory, an extended master directory (one master directory exists but each node has its own directory; if the needed information is not in the local file, the master file must be accessed) and multiple master directories (one copy of the master directory per cluster or node). Arrivals at the input queue of the directory represent the queries and updates generated by all the computers in the system. Network topology, communication costs, storage cost and code translation are considered. The model is M/D/1. It was found that for low update rates, the distributed file has lower operating cost as compared with the central directory systems. If the update rate is greater than 15% of the query rate, the centralized system performs the best. For update rates less than 1 to 5% of the query rate, the extended central directory performs the best.

Updating Totally Replicated Databases

Denola (1981) presents a M/M/1 queueing model to measure system performance for a totally replicated distributed database in a centralized management system, a master/slave system (where the nodes handle updates locally but updates to other nodes must be handled through the master) and synchronized (where nodes involved in the update are connected in a ring and must be visited in a predetermined order.) Arrivals to the queue consist of updates and retrievals. It was found that the master/slave system has a better average response time than the central system; which is reasonable since the local updates are handled locally and the only contention for the master node occurs when a nonlocal update is needed. The synchronized model does not outperform the other two systems until they have become totally saturated.

Gardarin and Chu (1979) present a deadlock-free algorithm for updating replicated databases in a distributed system. The algorithm locks tables locally and provides time stamps to detect conflicts at replicated sites. Performance comparisons between the voting and the centralized locking algorithm are analyzed in terms of the number of messages required to perform an update and the volume of control messages required to perform the update to multiple copies. No queueing analysis is done. The proposed algorithm performed better than the voting algorithm for more than two sites with respect to the number of messages per update. The centralized algorithm tended to perform better than either method with respect to control messages where the number of updates plus one per transaction is less than the number of sites in the network. The author points out, however, that the centralized management has response time problems.

Garcia-Molina (1979) presents a M/G/1 queueing model to measure system performance for totally replicated distributed database systems. Centralized control, distributed voting control and Ellis ring control are analyzed. The author considers conflicts as well as specific query situations. It was found that the centralized control algorithm performed best in most cases.

Gelenbe and Sevcik (1978) study concurrency issues in totally replicated databases. They develop a precise definition of coherence (how many different states exist in the dDBMS at a given time) and promptness (distance of nodes from the reference copy) and examine two control tecnhiques with respect to these two performance measures.

Optimal File Allocation

Chen (1973) develops a M/M/k queuing model to analyze the three types of file allocation problems:

1. minimize response time without regard to storage cost
2. minimize storage cost and meet a mean system response time criterion
3. minimize storage cost and meet an individual response time criterion for each file

It is proved that the optimal allocation strategy for problems 1 and 2 is to allocate files that are more frequently accessed to faster storage devices. No optimal strategy was found for type 3 problems. However, it was proved that the optimal allocation policy is not always the same as the other two. Although not directed toward distributed databases specifically, the author points out that his results can easily be applied to a distributed system.

Both Mahmoud and Riordon (1976) and Coffman, Gelenbe and Plateau (1981) develop queueing models to determine how to allocate copies of database files among the network nodes. Mahmoud examines file allocation and communication link capacities simultaneously. He develops an objective function for optimal allocation and finds that its solution falls into the class of nonlinear integer programming techniques. Since a solution is cumbersome, he uses decomposition to find a solution which has a relatively low cost.

Coffman examines the allocation of resources based on maximizing the read throughput in the presence of update arrivals. An optimum number of copies is found. The authors then examine the problem with respect to the Ellis ring algorithm (1977).

In summary, while queueing theory can be found in distributed database performance analysis, there is no prior research in the area of the performance of different support techniques for dynamic derived relations. This is clearly an important topic in a distributed system, since the support of user views requires some type of implementation of dynamic derived relations. In these distributed systems, the potential technique is quite costly since the ddr needs to be recalculated and transmitted every time a user refers to it. But, the actual technique is similarly hindered by the need to regenerate the ddr each time an update is performed on one of the base relations. Intuitively, the deferred-actual has a high probability, under certain circumstances, of outperforming either of the other methods. Our analysis verifies this.

CHAPTER III

EVALUATING DDR SUPPORT METHODS IN A CENTRALIZED SYSTEM

This chapter presents a methodology for evaluating database update schemes in a central database environment. Specifically, three ddr support methods are analyzed. The chapter discusses the characteristics of the three support methods and illustrates how these characteristics and the history of data usage in the database can be faithfully represented using an M/Hr/1 model.

A. Characteristics of the Three Update Methods

The purpose of this section is to discuss characteristics of the three update methods. The three algorithms shown in Figures 4, 5 and 7 will be referred to as potential, immediate-actual and deferred-actual methods, respectively. The immediate-actual method will be altered slightly for simplicity and uniformity in the modelling process.

The immediate-actual has been changed so that all ddrs defined by its defining relation are updated when one ddr is retrieved. Until that time, update tuples are stored in a differential file. No culling will occur and the immediate-actual method will not have overhead associated with maintaining this file. This differs slightly from the original algorithm which updates all ddrs when a defining relation is updated; however, the amount of work done is the same since we do not consider any advantage that might arise from the batching of these updates. This modification allows us to model the immediate-actual in a way consistent with the deferred actual and potential methods because now all maintenance work is done at ddr retrieval time.

Each of the three ddr support methods has differing characteristics with respect to how the ddr is retrieved, what ddrs are updated, and what storage requirements exist. A ddr retrieval in the immediate-actual method involves applying all the defining relation update tuples collected since the last ddr retrieval to the ddr.

The deferred-actual method also makes use of the defining relation update tuples since the last ddr retrieval. Updates are collected in differential files until the ddr is retrieved. This ddr support method may reduce the total number of tuples involved in the actual retrieval due to a culling process that is carried out when update tuples are added to the differential files. The same tuple used in more than one update is reduced to only the last update before the database is accessed. When tuples are used as a collection (or batch) of elements, there is the possibility that two updates might take less than twice one update time. Therefore, batching may also reduce the total work to be done.

The potential method totally reconstructs the ddr each time it is retrieved. The advantage of batching tuples for the ddr retrieval is also present in this case.

With respect to what ddrs are updated, both the potential and the deferred-actual update only the ddr involved in the query. The immediate-actual updates all ddrs defined by the base relation that has been updated.

Different storage requirements exist in that both the immediate-actual and the deferred-actual store the ddr in actual form. The deferred-actual incurs an additional cost for storing the differential files. The potential method stores only the formula of or access paths to the ddr so that it can be recalculated. In the analysis performed in this dissertation, storage requirements are modelled in the distributive database case.

B. Using the History of Database Retrievals and Updates to Determine the Average Service Time

This section illustrates how the history of data usage in a database can be used to determine needed parameters for the system. Specifically, we examine the update to retrieval ratio.

The first parameter to determine is the average service time to update a ddr. Let the probability of a ddr retrieval be P and the probability of a base relation update be Q. Under both actual storage methods, the amount of work done to retrieve a ddr is based on the number of updates which have occurred between two ddr retrievals. In the immediate-actual, this is the total number of tuples used to update the base relation between two ddr retrievals. The deferred-actual allows for culling if the same tuple is used to update the base relation more than once. Thus, the number of tuples used in the ddr retrieval may be less than or equal to the total number of tuples used to update the base relation.

Whenever a ddr is retrieved, tuples used to update the base relations which define the ddr are incorporated into the ddr retrieval. Since the number of tuples used in a ddr retrieval depends on the number of updates that have occurred, the service time for a ddr retrieval is

$$k * (Q * A / P * A) * (1 / U) = k * Q / (P * U)$$

where A is the arrival rate and 1/U is the average time to perform one update. When updates are batched, there is the possibility that two updates might take less than twice one update time. This will be incorporated into the model later but for now we will not consider the advantage of batching updates.

The value of k ($0<k<=1$) represents a culling factor. This factor is needed to analyze the difference between the two actual methods. In the immediate-actual, no culling occurs, while in the deferred-actual method, duplicate tuples are culled. If k is one, no duplicates are recognized. As k approaches zero, the ratio of distinct tuples to the total number of updates also approaches zero. The product of k and the total number of tuples used to update the base relation, ($Q*A/P*A$), represents the number of update tuples used in the ddr retrieval.

To illustrate this, let $k=1$ (no culling) and let there be m times as many updates as ddr retrieval arrivals. The state-transition diagram in Figure 8 represents this case with infinite update service rates. Arrivals are of two types, base relation updates and ddr retrievals. All arrivals are considered to be Markov with arrival rates for updates and retrievals being $m*A$ and A, respectively. Base relation updates are represented with respect to ddr retrievals.

States S0,S1,S2,... represent the number of updates that have occurred to the base relation, since the last ddr retrieval. A ddr retrieval arrival will immediately empty the system, that is, all tuples used to update the base

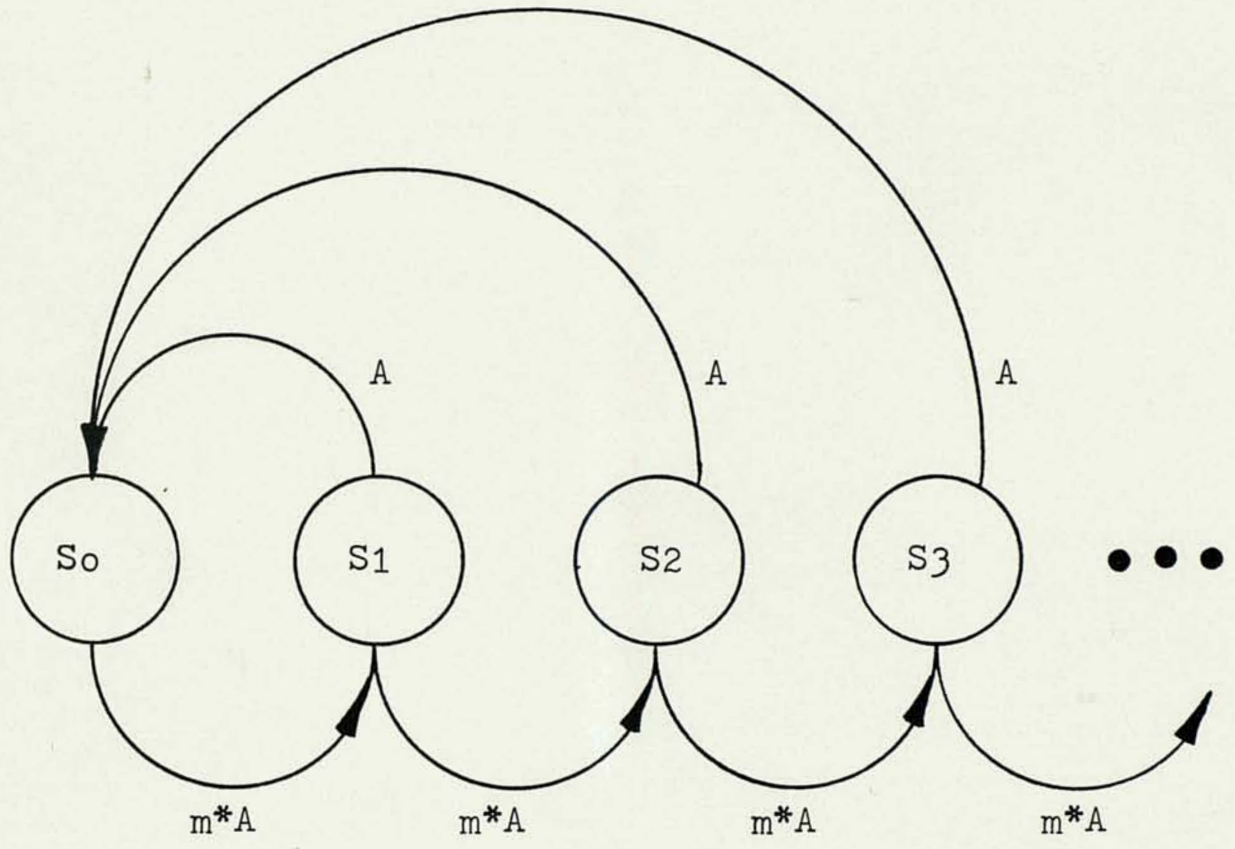


Figure 8. A state-transition diagram representing update and retrieval arrivals. The update arrivals are m times more than the retrieval arrivals.

relation will be used in the ddr retrieval and the collection of update tuples will begin again. As stated previously, update arrivals occur at the rate m*A and retrieval arrivals occur at rate A. The probability of being in state Sj (j>0) is

$$PROB(S_j) = (((m * A)/ (A + m*A))^{j}) * PROB(S_o)$$

and the probability that no tuples are in the system, state So, is

$$PROB(S_o)=1/(m+1)$$

The expected number of customers (tuples) is:

$$\sum_{i=1}^{\infty} i * PROB(S_i)=(1/(m+1)) * ((m*A/(A+m*A))^{1} + 2*(m*A/(A+m*A))^{2}+...) = m$$

which is the ratio of the update arrival rate to the retrieval arrival rate. (See appendix 1 for detailed calculatons.) Therefore, by determining the frequency of update and retrieval calls, probabilities can be computed which can be used to determine the average retrieval and update times.

C. The M/G/1 Model

As its notation indicates, the M/G/1 queue is a single server system with Markov arrivals and arbitrary service times. Within this system, a customer can be required to go through r parallel stages which have different service times (called M/Hr/1), or any combination of parallel and serial stages. This type of model lends itself well to representing a system which has updates and retrieval arrivals and where each type of arrival requires a different service time.

A 2-stage parallel server is shown in Figure 9 and its state-transition diagram is shown in Figure 10. Note that the first arrival proceeds to one of the parallel states based on the probability that the arrival will be of a specific type. This is because there is no one in the queue and, therefore, execution can begin immediately. After this point, all customers are queued, and when the customer being serviced has finished, then and only then can a queued customer enter for service. This is consistent with the update/retrieval system, where two different actions can occur, however, only one action can occur at a time and each of these has a different service time.

The following notation will be used throughout the the remainder of this dissertation:

A	is the arrival rate to a node
b	is the block size, indicating the number of ddr tuples accessed in one I/O operation to the external device that stores ddrs
k	is the culling factor
Ki	is the culling factor for the ith ddr

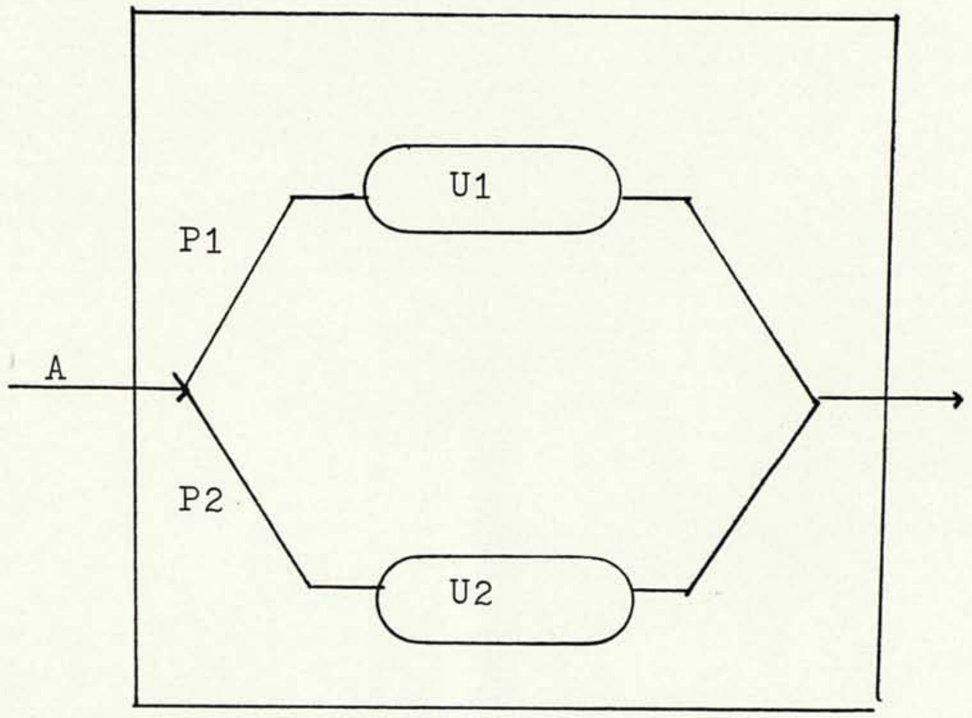


Figure 9. The M/H2/1 model.

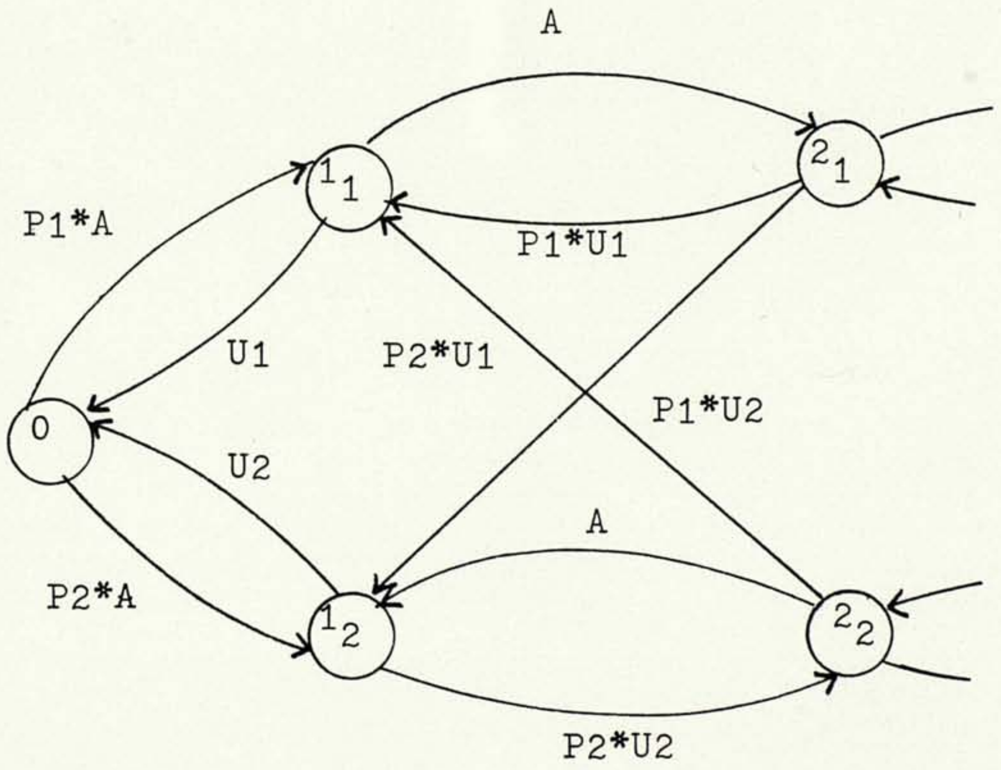


Figure 10. A state-transition diagram for the M/H2/1 model.

n	is the average number of ddrs per defining relation
P	is the probability that an arrival is a ddr retrieval
pi	is the probability that an arrival is a ddr retrieval for the ith ddr (p1+p2+...+pn=P)
Q	is the probability that an arrival is an update Q+P=1
1/U	is the service time it takes to do one update
ui	is the service rate to retrieve the ith ddr
X	is the mean service time at the node
pdf	is the probability density function
X2	is the second moment of the probability density function (pdf)
C2	is the square of the coefficient of variation
R	is the utilization factor (R=X*A)
q	is the expected queue length at the node
W	is the average wait time at the node

The formulas for the M/Hr/1 model which are used to analyze the individual nodes are as follows (Kleinrock 1975):

X= p1/u1 + p2/u2 + ... + pr/ur

X2= 2 * (p1/u1**2 + p2/u2**2 + ... + pr/ur**2)

C2= (X2/X**2)-1

q= R + (R**2 ((1+C2)/(2*(1-R)))

W= q/A

Refer to appendix 2 for a simplification of these formulas.

D. Using the M/Hr/1 Model to Represent the Immediate Actual Method

The M/Hr/1 model representing the immediate-actual method, which updates all ddrs whenever one ddr is retrieved, is shown in Figure 11. Arrivals are of two types, updates and retrievals. When an arrival occurs, the probability of the arrival being a retrieval is P. The probability of it being an update is Q. Note that Q+P=1. As noted in section B, the retrieval service time for one ddr is

$$k * (Q * A / P * A) * (1 / U) = k * Q / (P * U)$$

Since all ddrs are updated whenever one ddr is retrieved, and there are n ddrs, this service time should be multiplied by n. Also, in this method, there is no culling so the culling factor, k, is 1. Batching is not considered in this case due to the fact that the original version of the algorithm does not batch updates. Therefore, the retrieval service time is

$$(n * Q) / (P * U)$$

As an illustration, let the probability of a ddr retrieval arrival be P=.5 and the probability of an update be Q=.5 and let there exist two ddrs. The above formula indicates that the expected service time for each retrieval is (2*.5)/(.5*U) or 2*(1/U). That is, we can expect two updates for each retrieval. To see how this might come

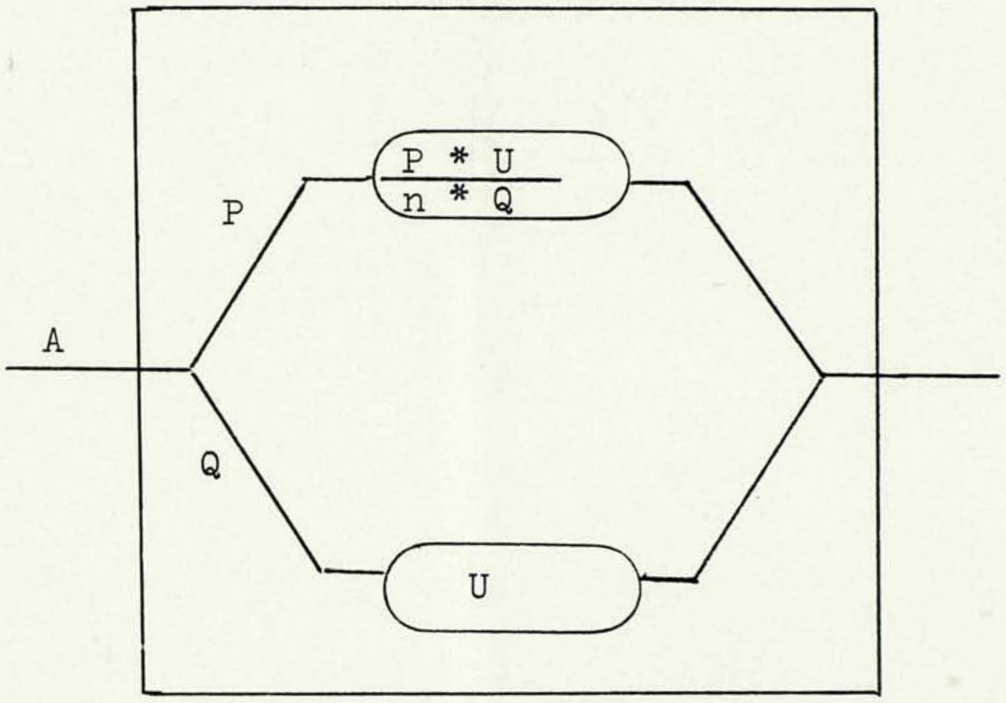


Figure 11. M/H2/1 model for the immediate-actual method.

about, consider the following sequence, where an update arrival is denoted by u and a ddr retrieval arrival by d:

u d u d u d u d u d

Since each ddr retrieval updates all ddrs when one is retrieved, each retrieval event will cause both ddrs to be updated. In the preceding sequence, there were five retrievals. One tuple was collected between each of these retrievals and, therefore, there were a total of ten update tuples used. Thus, the average number of updates per retrievals was 2.

The wait time, W, is

$$W = \frac{Q}{U} * \left((n+1) + \frac{A * (Q*n**2+P)}{P * (U-A*Q*(n+1))} \right)$$

The detailed calculations for the parameters can be found in the appendix 2a. Note that for R, the utilization factor, less than one, U is greater than A*Q*(n+1). Therefore, neither W nor q, the expected queue length can become negative.

E. Using the M/Hr/1 Model to Represent the Deferred Actual Update Method

The M/Hr/1 model for the deferred-actual update method is shown in Figure 12. In this method, only the ddr requested in the ddr retrieval is updated. Also, based on the algorithm, certain culling is done when the same tuple is used to update a base relation more than once. Note

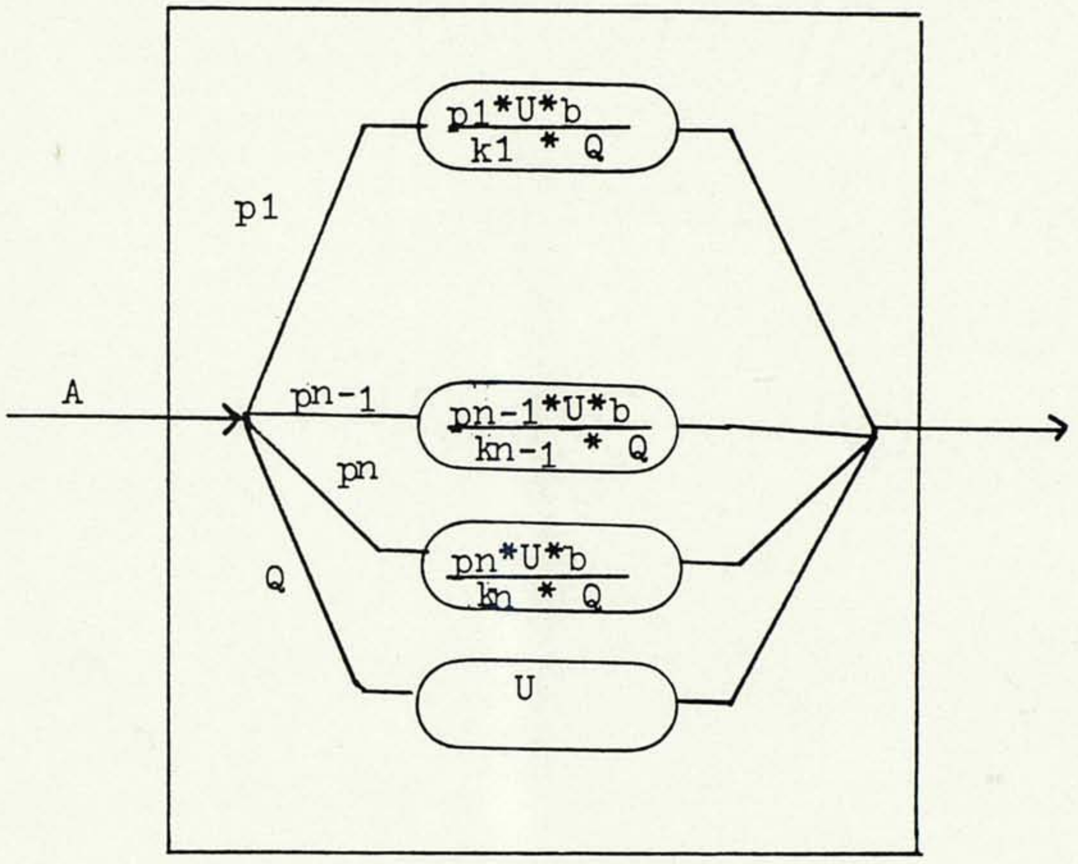


Figure 12. M/Hn+1/1 model for the deferred-actual method.

that there are now n+1 parallel stages, one for each ddr and one for a base relation update. The probability of a retrieval occurring for the ith ddr is now pi where p1+p2+...+pn=P. Using the service time for one ddr retrieval that was developed previously, we find the retrieval service time for the ith ddr to be

(ki*Q)/(pi*U)

where ki is the appropriate culling factor for the ith ddr.

Using the previous illustration, where the probabilities of a ddr retrieval and update arrival are both .5 and the number of ddrs is two, let one ddr have a probability of .4 and the other ddr have a probability of .1 of being retrieved. Representing the ddr with probability of .4 as d1 and the ddr with probability of .1 as d2, we might have the following arrival sequence with the number of tuples used in the retrieval listed below it

```
arrival: u d1 u d1 u d1 u d1 u d2 u d1 u d1 u d1 u d1 u d2
tuples:    1    1    1    1    5    2    1    1    1    1
```

Once the system has stabilized, the total number of tuples used to update the ddrs is again ten and the average is two updates per retrieval. This coincides with the result predicted by the above formula. The average service time to retrieve ddr1 is .5/(.4*U) or 1.25*(1/U) per retrieval. The average service time to retrieve ddr2 is

.5/(.1*U) or 5*(1/U). Since there are four retrieval arrivals of ddr1 in this sequence and one retrieval of ddr2, we have

$$4*1.25*(1/U) + 1*5/(1/U) = 10*(1/U)$$

Which is the same as the immediate-actual method. Therefore, the number of tuples used in the ddr retrieval is the same for both actual methods if no culling or batching is done.

When updates are batched, as in the deferred-actual method, there is a possibility that two updates might take less than twice one update time. We must represent the advantage of batching in order to accurately measure the performance of the deferred-actual method.

Batching can be expressed in terms of a block size, b. The block size is determined by how many tuples can be fetched from an external device at one time. Therefore, the expected ddr retrieval service time is

$$k*Q/(pi*b*U)$$

where k*Q/pi is the expected number of update tuples used to update the ddr and 1/b is the batching factor based on the block size.

Using the formulas presented in section C, we find the following:

```
          Q*(SUM1+1)          A * (Q**2*(SUM2+1)/U**2)
W=      ------------ + ------------------------------------
              U                 1- A*Q*(SUM1+1)/U
```

where SUM1=(k1+k2+...+kn)/b

and SUM2=(k1**2/p1+k2**2/p2+...+kn**2/pn)/b

See appendix 2b for detailed calculations. Note that if we assume that all pi's are equal, that is pi=P/n, that all ki=1 and that the block size is 1, then

k1+k2+...+kn=n

and hence, SUM1=n and SUM2=n**2/P. This then gives an expected wait time of

```
    Q*(n+1)       A*(Q**2(n**2/P + 1)/U**2)
    -------- + ---------------------------
       U          1 - A*Q*(n+1)/U
```

using simple albebra, we see that the formula reduces to

```
             Q                 A * (Q*n**2+P)
     W =  --- * ( (n+1) + ------------------ )
             U               P * (U-A*Q*(n+1))
```

which is equivalent to the immediate-actual method. The major advantage of the deferred-actual method is when culling exists and batching is incorporated.

F. Using the M/Hr/1 Model to Represent the Potential Method

The potential method requires total reconstruction of the ddr each time it is retrieved. Therefore, the potential method is dependent upon the size of the defining relation. Let S represent the average size (in tuples) of a defining relation in the database. As in the deferred-actual method, we introduce the batching factor 1/b where b is the block size. Then the ddr retrieval time will be

$$S/(U*b)$$

The M/Hr/1 model for the potential method is shown in Figure 13.

Using the formulas presented in section C, we find that the wait time is

$$W = \left(\frac{P*S}{U*b} + \frac{Q}{U} \right) + \frac{A * (P*S**2/(U*b)**2 + Q/U**2)}{1 - A * (P*S/(U*b) + Q/U)},$$

Again, the detailed calculations are in appendix 2c.

Note that the immediate-actual and deferred-actual methods are equivalent to the potential method if no batching occurs, Q*n/P=S, and ki is 1 for all i.

G. Incorporating Overhead into the Actual-Deferred Method

The actual-deferred method incurs overhead from which the other two methods are exempt. The overhead involves maintaining the insertion and deletion tuples in the differential files. The cost of updating the differential files should be proportional to how many tuples are in the file. Since we know the service time to update a defining relation, 1/U, we can express the service time to update the differential file as a factor of this. Note that the

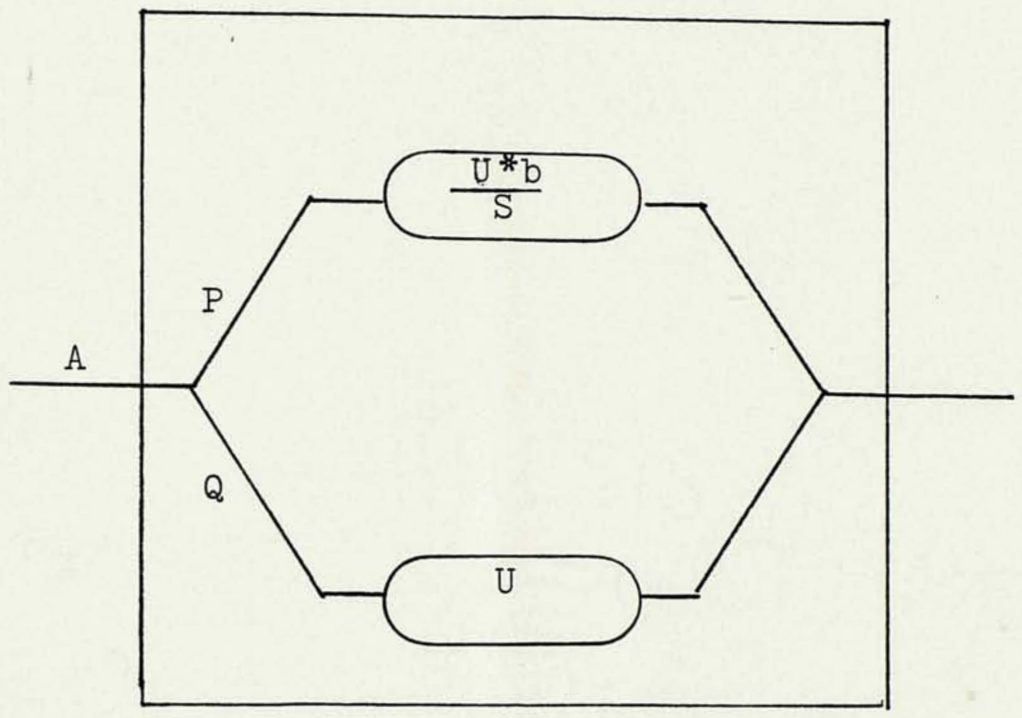


Figure 13. M/H2/1 model for potential method.

update to defining relation ratio will always be less than or equal to one. The deferred-actual requires that the join be done only on the key and, therefore, a join can not cause the derived relation to be larger than any of its defining relations. The other relational operations yield a derived relation that is always smaller than or the same size as its defining relations.

Let Z be a factor which represents the ratio of the size of a ddr to the sum of the sizes of its defining relations. Then let

1/U	represent the service time to update a defining relation
1/U+Z/U	represent the service time to update a defining relation and to insert it into the insertion or deletion table

Although it seems obvious that the ratio of updates to the actual defining relation size should be a parameter in the differential file update function, there may be other overhead involved in maintaining the file which have not been considered. We will, therefore, introduce another factor Y such that 0<=Y<=1/Z, which can be adjusted to represent a particular system. Using the Y and Z parameters, the service time for an update becomes

$$\frac{1}{U} + \frac{Y*Z}{U}$$

Note that when Y is zero, there is no overhead and when Y is

1/Z, the cost to update the differential file is equivalent to the cost of updating the defining relation.

H. Comparison of Performance of the Three DDR Support Methods

In this section, comparisons of the performance of the algorithms are presented. Because there exists no standard data from a "typical" database, we have no way of knowing what is the norm or the exception in the way of data usage. For this reason, we can not state emphatically that under "typical" data usage, one ddr support method is best. We have, instead, provided parameters which can be set based on a specific system. This provides us with the flexibility of considering special cases while allowing the specialization necessary for a useful performance analysis tool.

In deriving the average service time, queue length and wait time for the three ddr support methods, we indicated that by varying certain parameters, one method might reduce to the same performance as another method. One of the more important parameters was the size of the ddr with respect to its defining relation. It was found that the size affects both the update/relation size ratio and the deferred method's overhead execution time parameter.

Another important parameter is the number of ddrs per defining relation. This was found to affect both the deferred-actual and the immediate-actual methods. The batching parameter affects both the deferred-actual and potential methods. Finally, the culling factor for the deferred update method is another important parameter which affects the deferred-actual service time.

The special cases presented vary the parameters we have discussed. They are considered to be special cases because they cause the different update algorithms to either be indistinguishable or to have extremely different performance characteristics. These cases are discussed and then presented graphically. After considering these special cases, we examine a situation with none of these limitations.

The graphical representations are presented with respect to the arrival rate and the response time. Although, typically, queueing model performance is presented by varying the utilization factor, we have not done so. This is because such a presentation is, in this case, misleading. The service times vary with such extremes under these special cases, that, in order to set the systems to a specific utilization factor, the arrival rates differed greatly. This was not obvious when plotting the performance by varying the utilization factor. We have, therefore, presented the special cases by varying the arrival rate.

The graph in Figure 14 shows response time as a function of the arrival rate. Using the equal scale on both axis of the graph causes the data to plot as almost indistinguishable lines. To more clearly illustrate the differences in various factors as the arrival rate increases, the other graphs in this dissertation will use unequal scales.

Case A: Vary the ratio of updates to defining relation tuples. The number of ddrs and block size is 1 and there is no culling. This case will cause both of the actual update methods to behave in the same way. The potential method, however, will vary greatly with respect to the ratio of updates to relation size. This is shown in Figure 15. This seems logical in view of the fact that the amount of tuples used in reconstructing the ddr depends on this parameter.

Case B: Vary the number of ddrs. There is no culling, the block size is one and the ratio of updates to relation size is one. In this case the immediate-actual and the deferred-actual methods vary greatly. In the immediate-actual method this is due to the fact that all ddrs must be updated whenever one ddr is retrieved. The greater the number of ddrs, the more that must be updated. In the deferred-actual method, since no culling occurs and

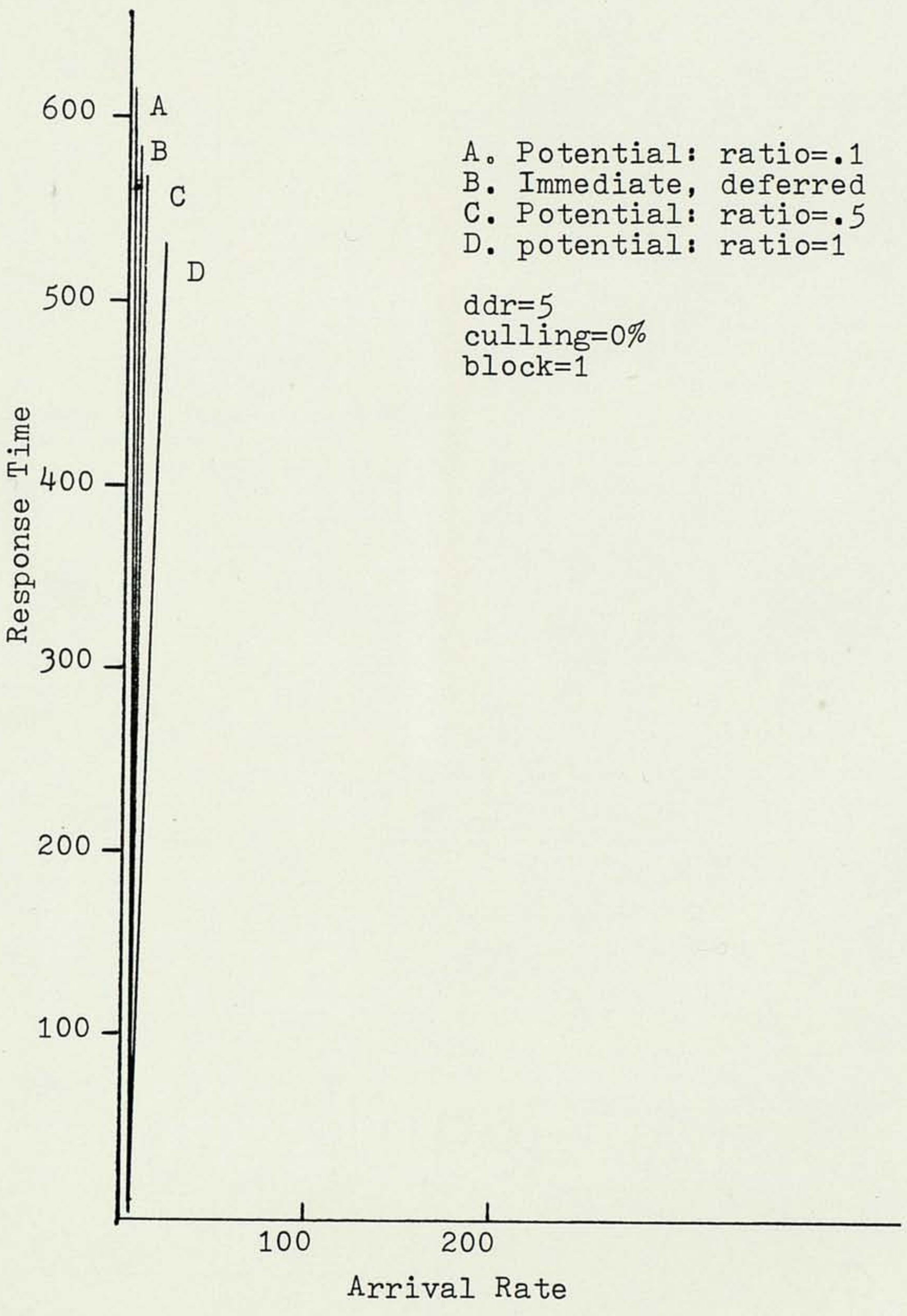


Figure 14. Vary update/relation size (equal scale).

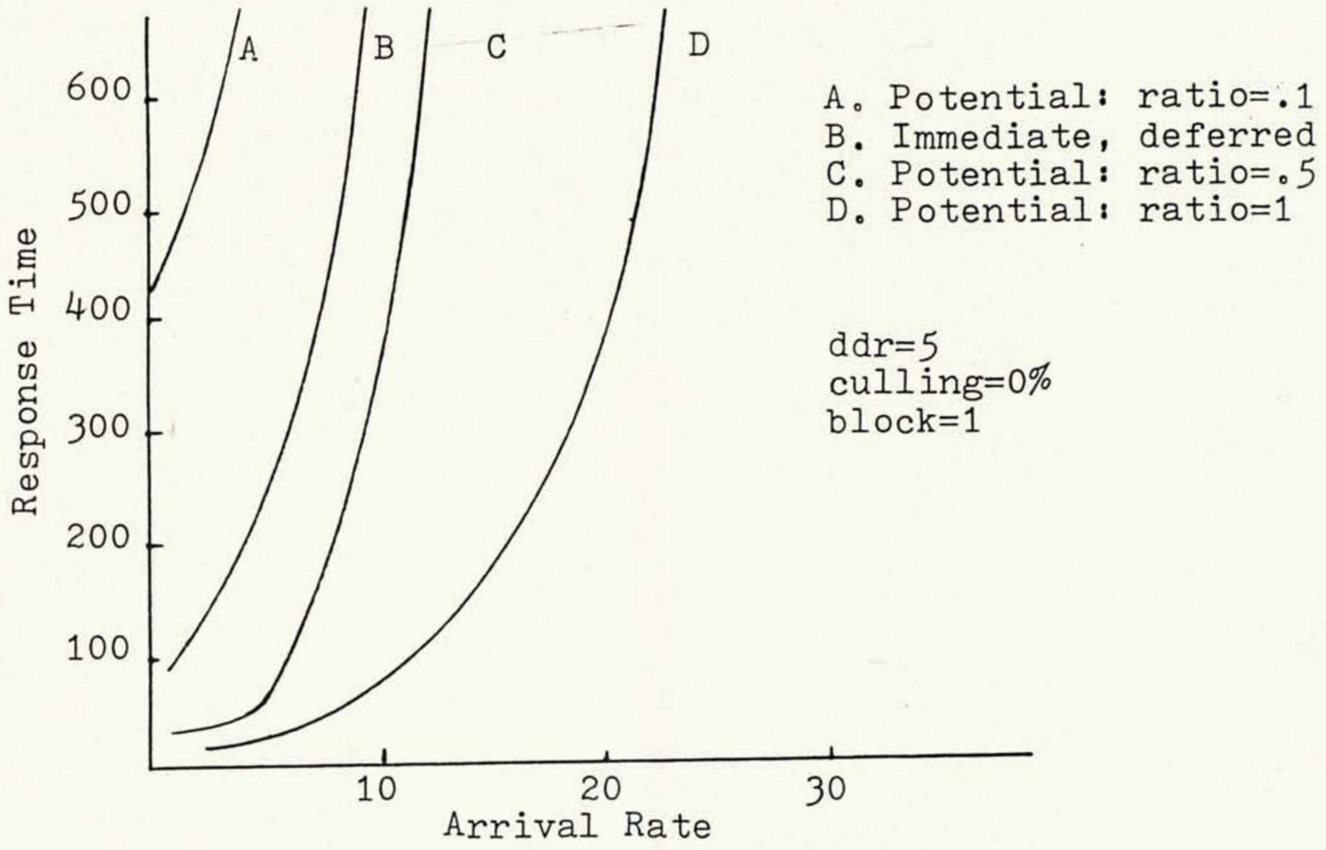


Figure 15. Vary update to relation size.

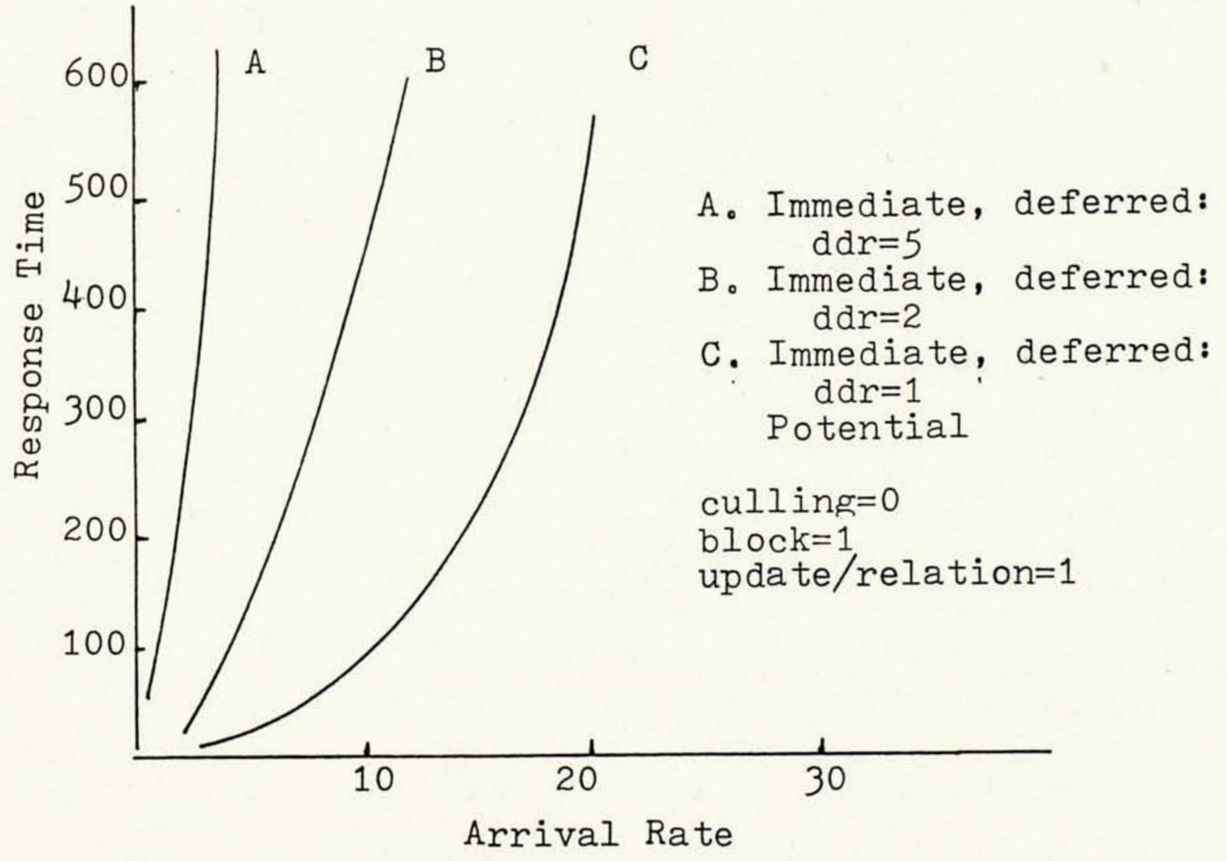


Figure 16. Vary number of ddrs.

no advantage is allowed from batching, it performs the same as in immediate actual. This is shown in Figure 16.

Case C: Vary the culling factor. The number of ddrs is one and the ratio of updates to defining relation size is one. This case will cause the deferred actual to vary, with worst case occurring when there is no culling. At this point the performance is equal to the other two methods. This is shown in Figure 17.

Case D: Vary overhead. This case will cause the deferred-actual to vary with worst case occurring when the cost of updating a ddr is equal to the cost of updating a defining relation and best case when there is no overhead. This is shown in Figure 18.

Case E: Vary the bucket size. This case will cause the performance of the deferred-actual and potential methods to vary greatly. Note that because the update to relation size ratio is one and there are two ddrs, the potential method has better performance than the other two methods. This is shown in Figure 19.

Primary Factors Affecting the Performance of Each Method

As we have shown, the performance of each method is based on certain parameters. With regard to the potential method it is the ratio of updates to defining relation size and the blocking factor. With regard to the immediate

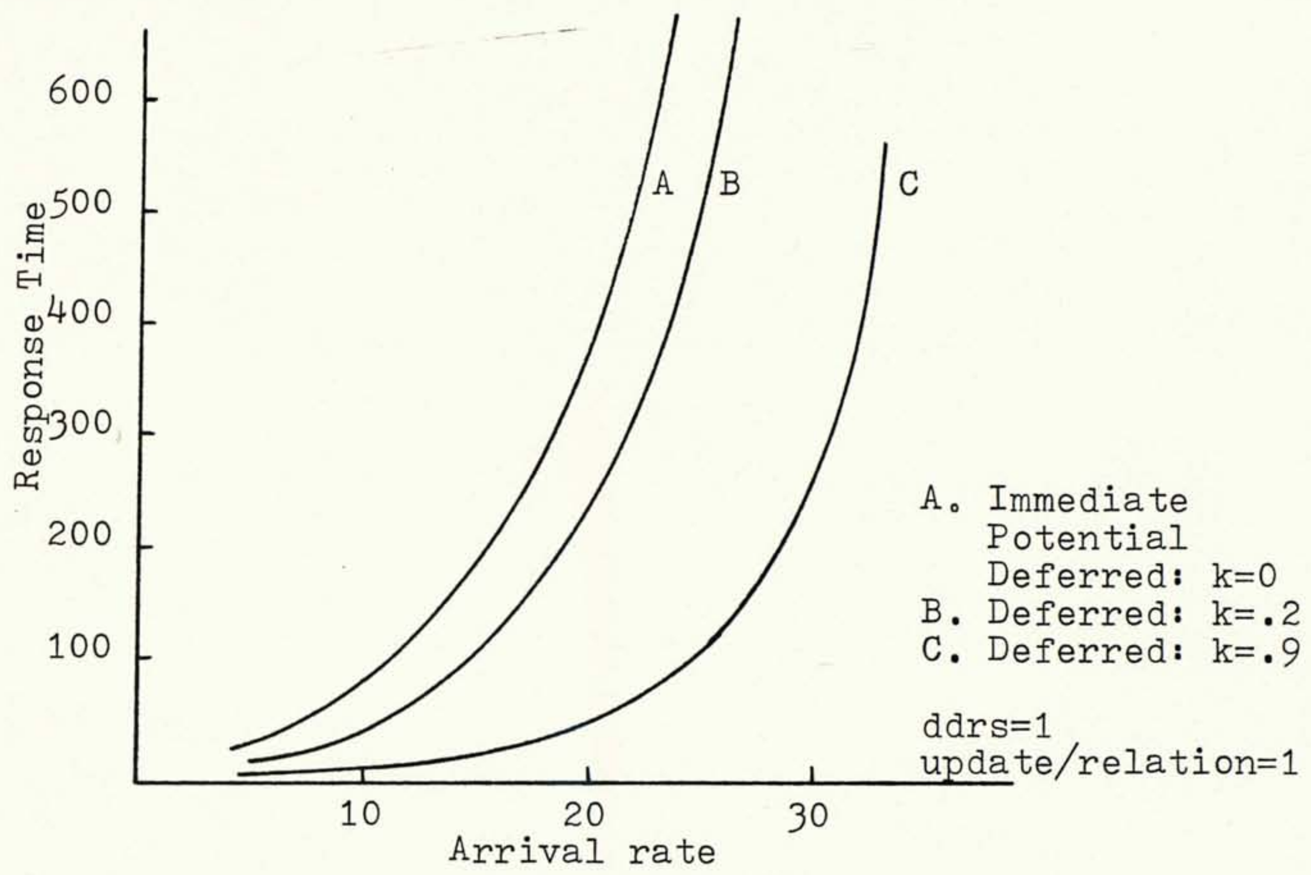


Figure 17. Vary culling factor (k).

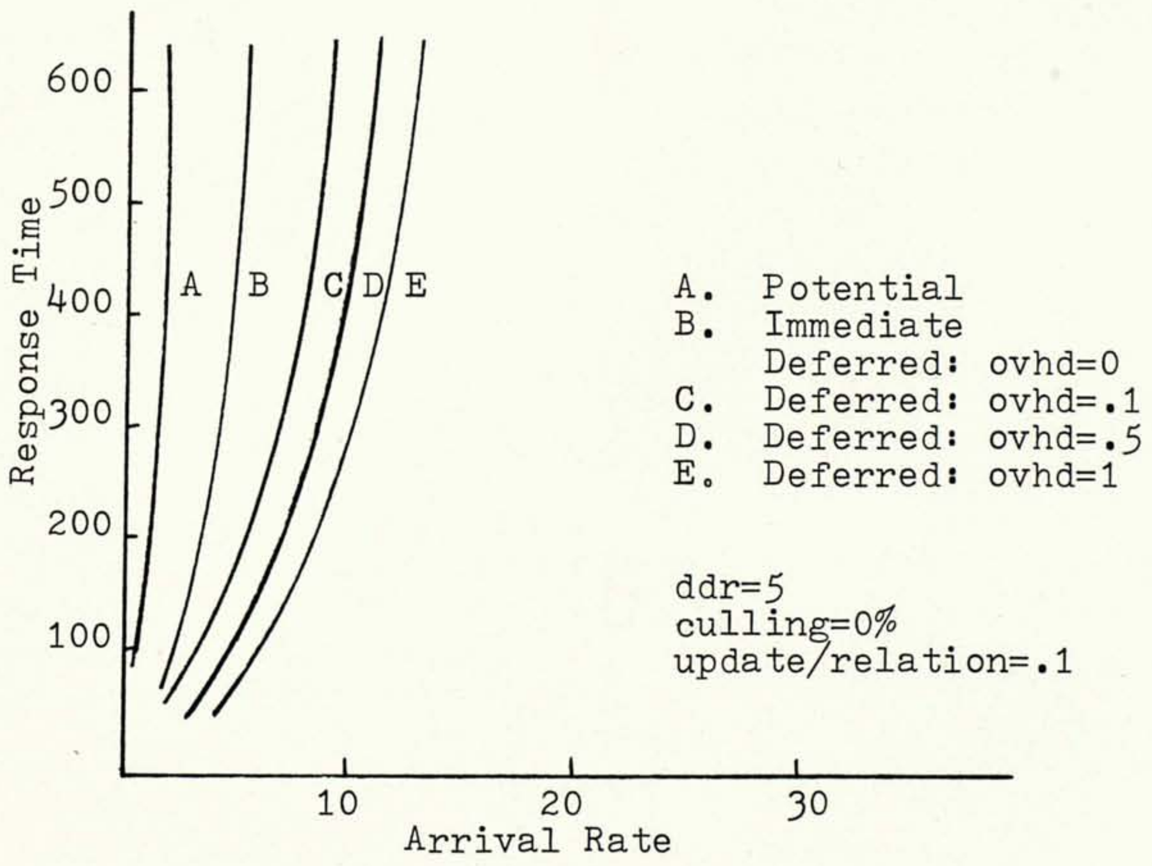


Figure 18. Vary overhead (ovhd).

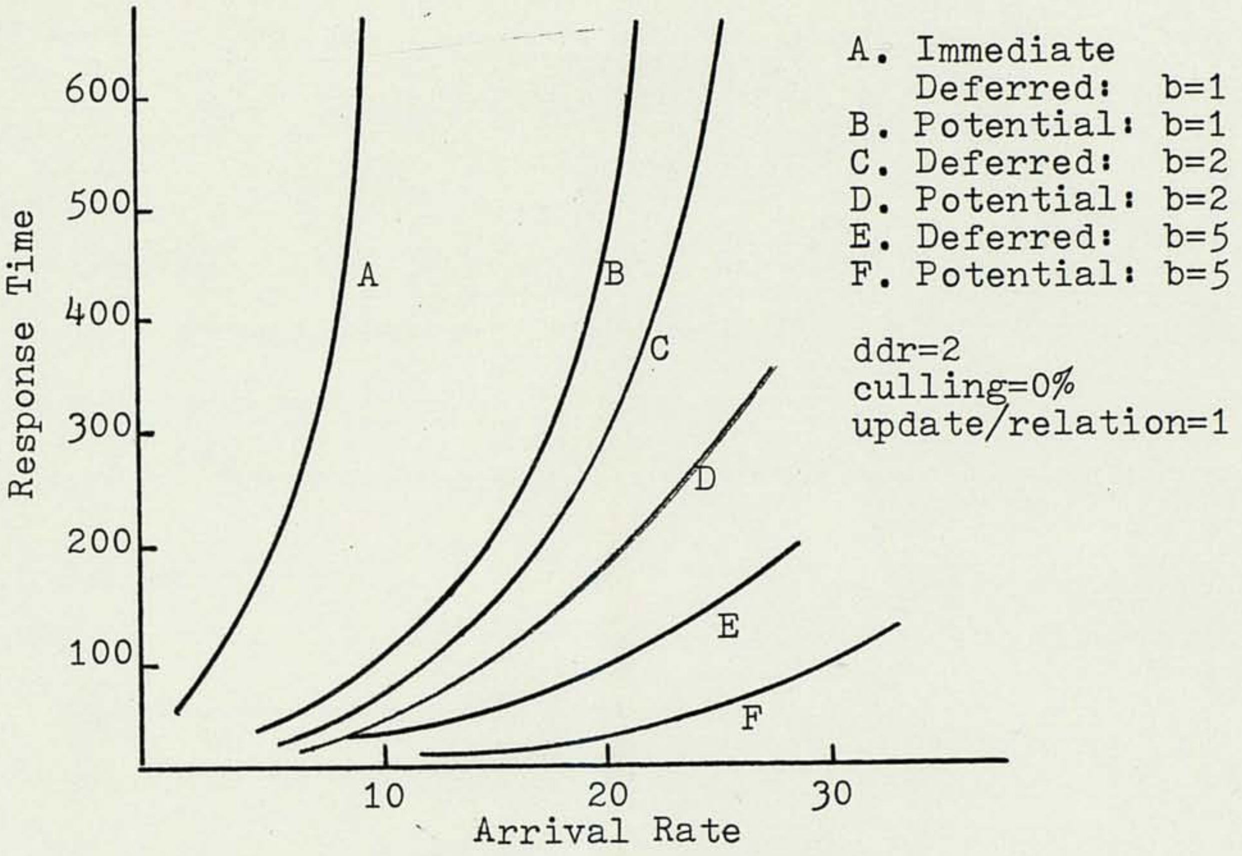


Figure 19 . Vary the block size (b).

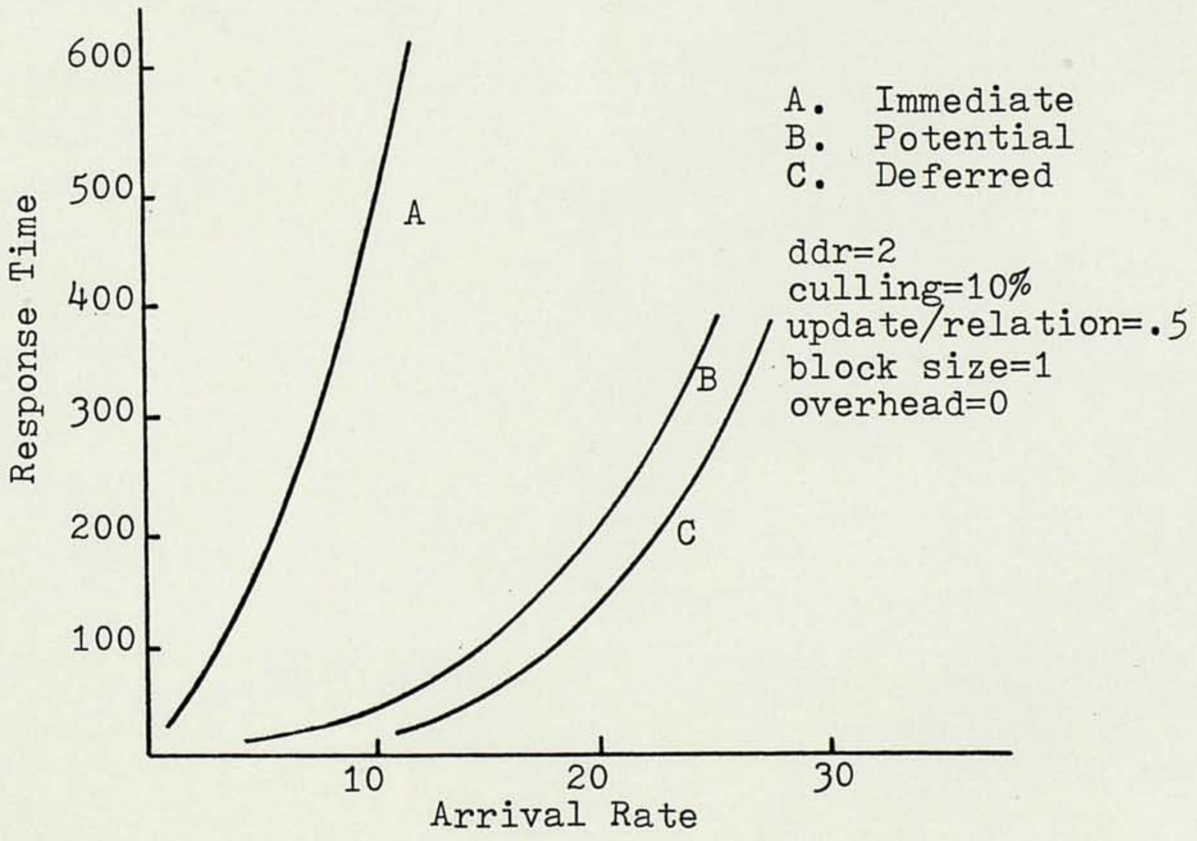


Figure 20. Nonspecial case.

actual, the main performance parameter is the number of ddrs. The deferred method also depends on this parameter as well as the culling, blocking and overhead parameters. In general, the deferred-actual method will perform better than the immediate-actual method if batching and culling are considered.

The deferred method performs well as compared to the potential method when the update to relation size ratio and the number of ddrs is low. Special cases such as no culling, extreme overhead and the existence of many ddrs contribute to poor response time for the deferred strategy.

Figure 18 presents a case closer to the medium of the parameter ranges. The number of ddrs is two, a ddr is considered to be one half the size of its defining relation and culling occurs 10% of the time. The block size is 3.

CHAPTER IV

ANALYSIS OF DDR SUPPORT METHODS IN A DISTRIBUTIVE ENVIRONMENT

As stated previously, one of the most important design considerations in a dDBMS environment is concurrency control. This section presents the analysis of the 3 ddr support schemes within three distributive concurrency approaches -- centralized, distributed and synchronized ring. The following notation is introduced in this chapter:

C	is the service rate for a lock
Pd	is the probability that a defining relation is not stored at the same node as its ddr
Pl	is the probability that a nonlocal ddr is requested
N	is the number of nodes in the network

A. Master/Slave System of Centralized Control

Figure 21 shows the information flow of a Master/Slave centralized control system. All updates must be sent to the master to lock the affected item before an update can proceed. Figure 22 shows the master and slave information flow. The slave sends all updates immediately to the master. Retrievals for files at other nodes are also sent

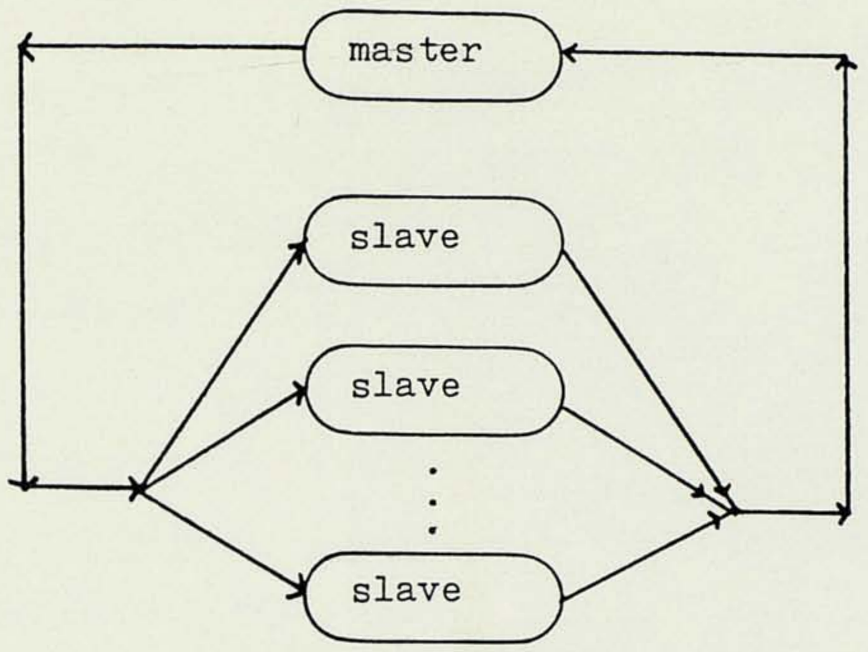
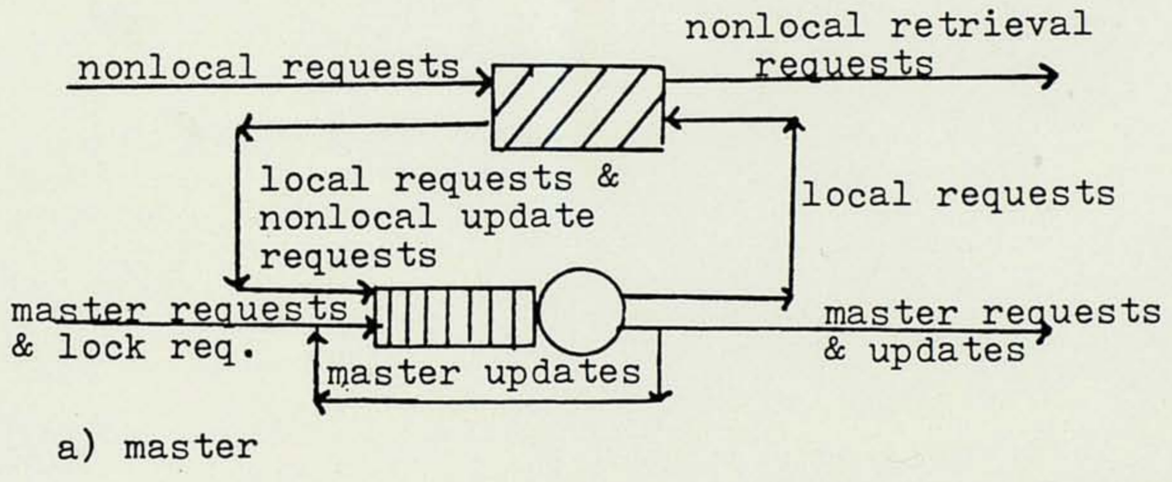


Figure 21. Master/Slave flow among the nodes


nonlocal requests
nonlocal retrieval requests
local requests & nonlocal update requests
local requests
master requests & lock req.
master requests & updates
master updates
a) master

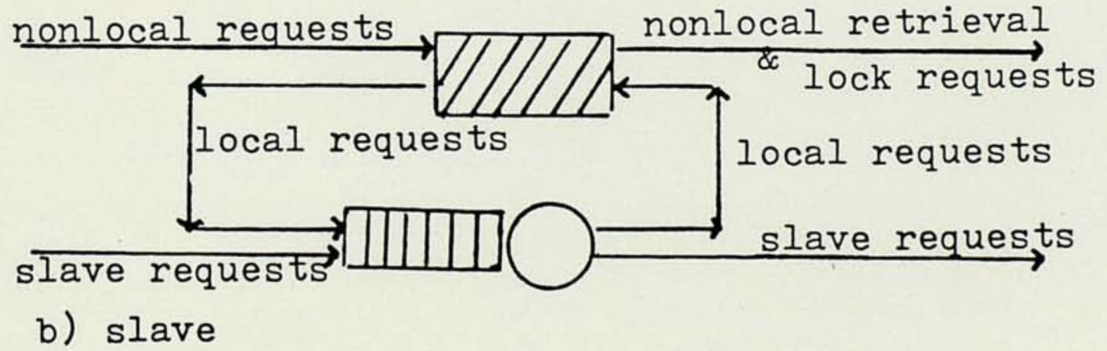


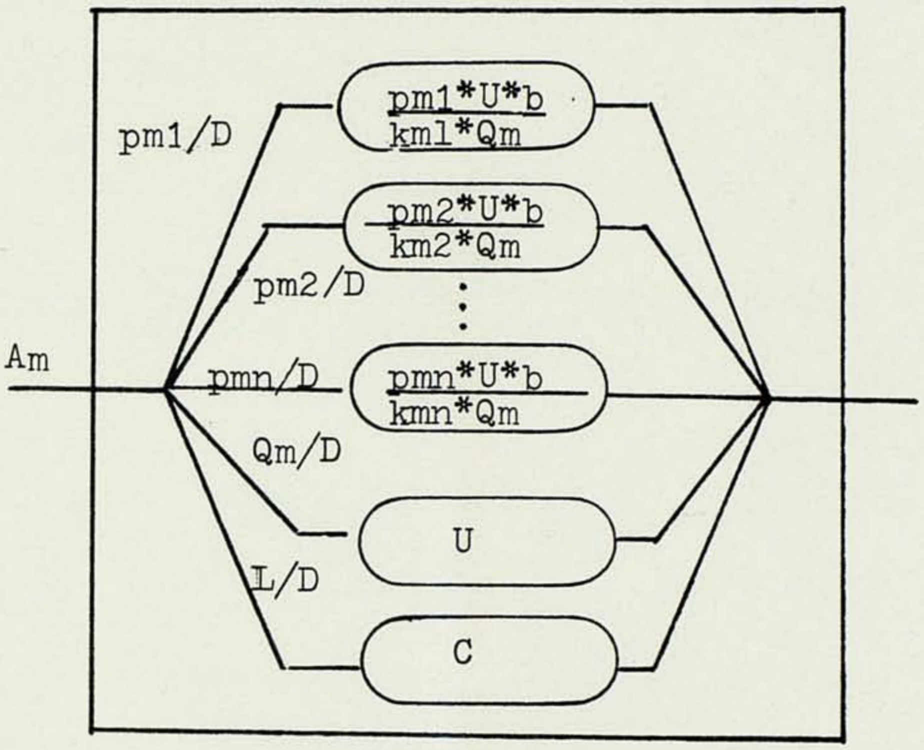
Figure 22. Information flow for individual node in master/slave system.

immediately to the appropriate node. Local retrievals are sent to the queue from the node's terminals. From outside the node, arrivals are occurring from the master which grants update privileges, as well as from other nodes requesting retrieval of local files. These external retrieval requests then result in data being sent to the requesting nodes. Local requests, once fulfilled, are returned to the terminals.

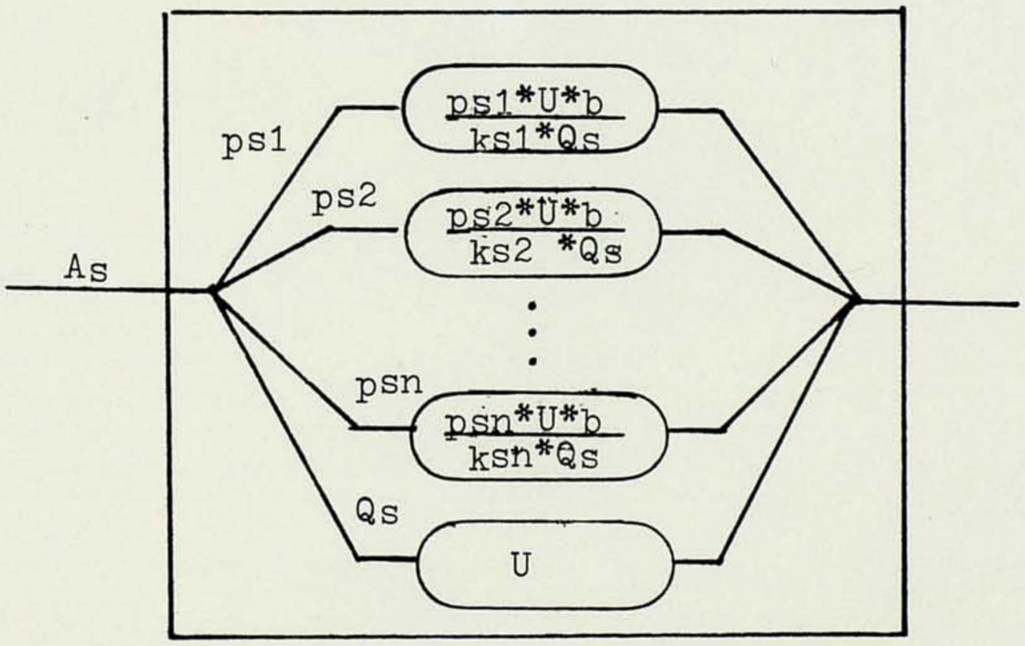
The master receives system requests for locks, requests from other nodes for retrieval of the master's files and update and retrieval requests from its terminals. Notice that since items to be updated at the master must also receive a lock before an update can proceed, its updates must cycle through the system twice. Terminal requests for other files are sent immediately to the appropriate node and, after processing update and retrieval requests from other nodes for the master's files, the files are then sent to the requesting nodes.

Deferred Actual Method

Figure 23 shows an M/Hr/1 model for the master and slave nodes in the deferred-actual system. Here, only a ddr which has been requested is updated. Due to the algorithm, culling may occur which may cause fewer updates to be processed than may have actually occurred. The slave



a) Master node for deferred-actual method



b) Slave node for deferred-actual method

Figure 23. Master/Slave model for deferred actual method.

node is the same as in the central database. The master node, on the other hand, must obtain locks for all updates. Therefore, a service rate of C has been introduced for this task. Because the master/slave system must grant lock requests for all nodes and the probability of an update request occurring is Q, the probability of a lock request occurring at the master node, assuming no conflicts, is

$$L=Q/D$$

where 1/D, a normalizing factor (D=Qm+Pm+L), has been introduced so that the sum of the probabilities entering the service queue is one. Conflicting update requests will be considered in a later section.

Using the formulas in Chapter III, section B, the average wait time Wm and Ws (for master and slave respectively) are shown below. Detailed calculations can be found in appendix 3a.

$$Wm = \left(\frac{Qm * SUM1m}{D*U} + \frac{Qm}{D*U} + \frac{L}{D*C} \right) + \frac{Am*\left(\frac{Qm**2*SUM2m}{D*U**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2} \right)}{1-Am*\left(\frac{Qm*SUM1m}{D*U} + \frac{Qm}{D*U} + \frac{L}{D*C} \right)}$$

where SUM1m=(km1+km2+...+kmn)/b and

SUM2m=(km1**2/pm1+km2**2/pm2+...+kmn**2/pmn)/b**2

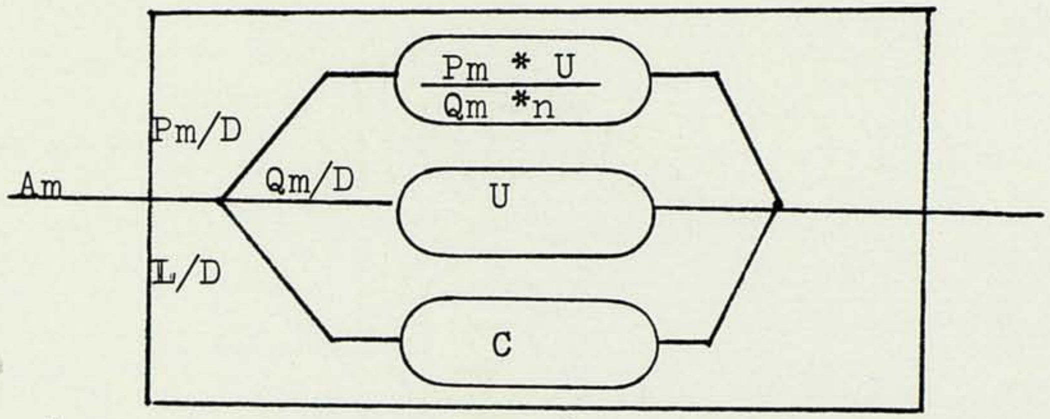
$$Ws = \left(\frac{Qs*Sum1s}{U} + \frac{Qs}{U}\right) + \frac{As * \left(\frac{Qs**2 * SUM2s}{U**2} + \frac{Qs}{U**2}\right)}{1 - As\left(\frac{Qs * SUM1s}{U} + \frac{Qs}{U}\right)}$$

where SUM1s=(ks1+ks2+...+ksn)/b and

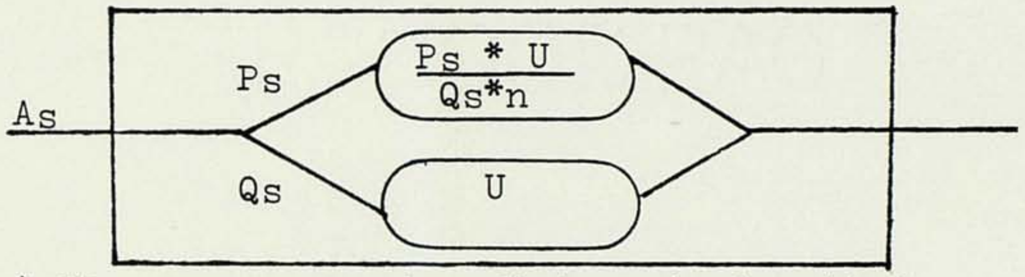
SUM2s=(ks1**2/ps1+ks2**2/ps2+...+ksn**2/psn)/b**2

Immediate Actual Method

Figure 24 shows the master and slave nodes for the immediate actual method. In this algorithm, updates are collected as they occur to defining relations. At retrieval time for one ddr, all appropriate ddrs are updated. The slave node is the same as in the central database. The master has an added service stage for granting locks. The values of Wm and Ws are shown below. Detailed calculations can be found in appendix 3b.

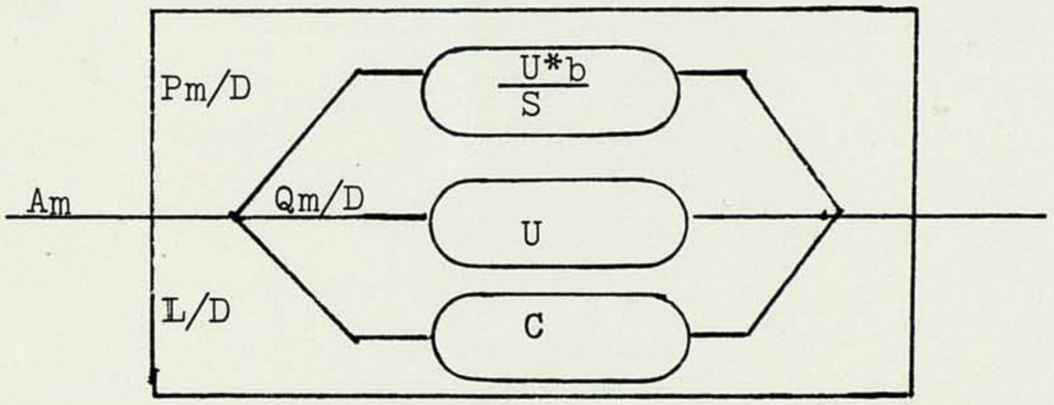


a) Master node for immediate-actual method

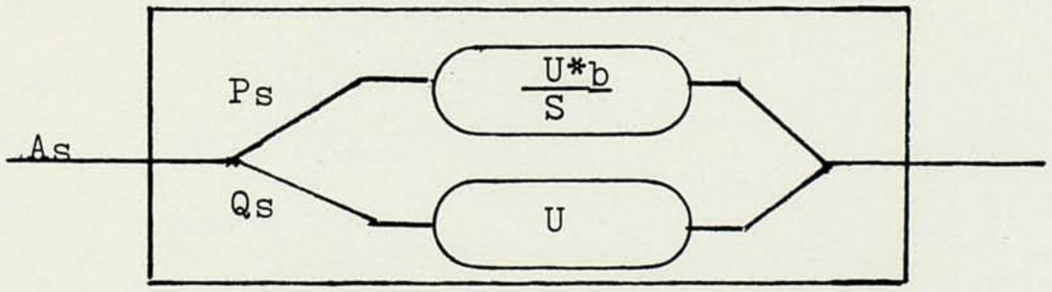


b) Slave node for immediate-actual method

Figure 24. Master/Slave model for immediate-actual method.



a) Master node for potential method



b) Slave node for potential method

Figure 25. Master/Slave model for potential method.

$$Wm = \frac{Qm \ast (n+1)}{D \ast U} + \frac{L}{D \ast C}$$

$$+ \frac{Am \ast \left(\frac{(Qm \ast n)^{2}}{D \ast Pm \ast U^{2}} + \frac{Qm}{D \ast U^{2}} + \frac{L}{D \ast C^{2}} \right)}{1 - Am \ast \left(\frac{Qm \ast (n+1)}{D \ast U} + \frac{L}{D \ast C} \right)}$$

$$Ws = \frac{Qs \ast (n+1)}{U}$$

$$+ \frac{As \ast \left(\frac{(Qs \ast n)^{2}}{Ps \ast U^{2}} + \frac{Qs}{U^{2}} \right)}{1 - As \ast \left(\frac{Qs \ast (n+1)}{U} \right)}$$

Potential Method

Figure 25 shows the M/G/1 Model for the potential method. In this method, whenever a ddr is retrieved, it is totally reconstructed. Therefore, the service time is based on the average size of a defining relation in the system.

This value, S, is based on a variable Pu which represents a ration of the average number of updates between arrivals compared to the average size of a defining relation. Specifically, S can be computed from Q/(P*Pu), i.e. the average number of updates to retrievals divided by the factor representing the percentage that the average number of updates is to the size of the relation.

As in the actual ddr support methods, the slave node is the same as in the central database. The master has an added service stage for granting locks. The values for Wm and Ws are shown below. Detailed calculations are in appendix 3c.

$$Wm = \frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}$$

$$+ \frac{Am*\left(\frac{Pm*S**2}{D*(U*b)**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2}\right.}{1 - Am*\left(\frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}\right)}$$

$$Ws = \frac{Ps*S}{U*b} + \frac{Qs}{U} + \frac{As*\left(\frac{Ps*S**2}{(U*b)**2} + \frac{Qs}{U**2}\right)}{1 - As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right)}$$

Response Time for a Transaction

Response time is based on several variables. First of all, there is the percentage of local requests as opposed to nonlocal requests. It is the case that files should be requested most often by the local terminals. If not, some optimization probably needs to be done to accomplish this. Although this seems logical, there is no real data provided on how often this is the case. For this reason, the variable Pl has been introduced in order to represent the percentage of local requests. 1-Pl then represents the percentage of nonlocal requests. This can be varied to model a specific system in order to predict its expected performance.

This analysis assumes that a ddr is stored at the same node as one of its defining relations (perhaps the largest) and that a ddr is defined by at most two base relations. This is not an unreasonable assumption, since in any of the three methods, storing a ddr away from its base relation in an active update environment would add greatly to the update time.

There may be times, however, when the second defining relation is not stored with its ddr and thus some transmission must occur to update or construct the ddr. Here again, no data can be found on how often this might occur, so we introduce another variable, Pd, which represents the percentage of ddr retrieval arrivals which may involve a defining relation at another node. This can be set to different values in order to analyze real situations that might arise in the system being modelled.

As stated previously, the probability that a transaction is a ddr retrieval is P and the probability of a base relation update is Q (P+Q=1). Let there be N slaves and 1 master. Then, in a properly balanced system, the average number of updates and retrievals per node will be

$$A*P/(N+1) + A*Q/(N+1) = A/(N+1)$$

In addition, the master node must examine all updates to see if a lock can be obtained. If we assume that the arrivals are distributed evenly among the nodes, the probability that a request occurs to a slave node is N/(N+1) and that a request occurs to the master node is 1/(N+1).

In order to determine the response time for a transaction, we must look at the probability of an event happening and the wait time and transmission time involved. In this case, T represents the transmission time to any node for one tuple or lock. We are representing an Ethernet-like system with variable packet sizes so that several packet transmissions for one transaction do not have to be dealt with. This is adequate to model the performance of the various update schemes in a general sense.

The response time for a transaction in a Master/Slave system can be divided into ten distinct events. One event is obtaining a lock update for a slave node. This involves the probability that the node is a slave and that the request is an update, Q*N/(N+1). The time involved is the wait at the master node to obtain the lock, Wm, and the transmission to and from the master, 2*T. Therefore, the probable wait time contributed by this event is

$$Q*(N/(N+1))*(Wm+2*T) \qquad (1)$$

A second event is obtaining an update lock for a master. This analysis is the same as above, except that there is no transmission time involved and the probability of being a master node is 1/(N+1). Therefore, the contribution to the expected wait time as a result of this event is

$$Q*(1/(N+1))*Wm \qquad (2)$$

The third and fourth events involve local and nonlocal slave update requests. The probability of a local slave update request occurring is the probability that an update occurs to a slave and is local, Pl*Q*(N/(N+1)). The time involved is the wait time at the slave to do the update, Ws. Therefore, the response time for a local slave update is

$$Pl*Q*(N/(N+1)) * Ws \qquad (3)$$

Similarly, the probability of a nonlocal slave update request is (1-Pl)*Q*(N/(N+1)). The time is the wait time for the update at the slave node and the transmission time for the update. Therefore, we have

$$(1-Pl)*(Q*N/(N+1)*(Ws+T)) \qquad (4)$$

The fifth and sixth events involve local and nonlocal master update requests. The probability of a local master update request occurring is the probability that an update occurs to a master and is local, Pl*Q*(1/(N+1)). The time involved is the wait time at the master to do the update, Wm. Therefore, the response time contribution for a local master update is

$$Pl*Q*(1/(N+1)) * Wm \qquad (5)$$

Similarly, the probability of a nonlocal master update request is (1-Pl)*Q*(1/(N+1)). The time is the wait time for the update at the master node and the transmission time for the update. Therefore, we have

(1-Pl)*(Q*1/(N+1)*(Wm+T) (6)

The seventh and eighth events involve local and nonlocal ddr retrieval requests at the master. The probability of the local ddr retrieval request occurring at the master is P*Pl*1/(N+1). The time involved to perform this action is the wait time at the master node, Wm. If one of the ddr defining relations is not local (with probability Pd) we must send the portion of the local defining relation to the nonlocal defining relation's node, perform the update computation at that node, and send the resulting update file back to the originating node. The needed computation and resulting number of tuples transmitted, depends on the ddr support method used. Therefore, we will let the number of tuples transmitted be T' and will illustrate what T' will be for each method in the individual ddr support sections that follow. This gives us, then

Pl*P/(N+1)*(Wm+Pd*(Ws+2*T*T') (7)

Similarly, for nonlocal master retrievals, the analysis is the same except that a request must be sent to the node where the ddr resides and the ddr must be sent back. Since S is the average size of relations in the database, we will assume that the ddr is of size S. Therefore, we have

$$(1-Pl)*P/(N+1) * (Wm+(S+1)*T+Pd*(Ws+2*T*T')) \tag{8}$$

Finally, events nine and ten are for local and nonlocal slave ddr retrievals. The probability of a local retrieval occurring to a slave node is Pl*P*N/(N+1). The time involves waiting at the slave node for the retrieval, Ws, and, if a defining relation is not local, the same time is involved as with the master node. Therefore, we have

$$Pl*P*N/(N+1) * (Ws+Pd*(Ws+2*T*T')) \tag{9}$$

The analysis is similar to the master for a nonlocal retrieval as applied to the slave node. Therefore, we have

$$(1-Pl)*P*N/(N+1) * (Ws+Pd*(Ws+2*T*T')) \tag{10}$$

Combining events (1) through (10), we have a total expected response time for a transaction of

$$\begin{aligned} RT= Q*Wm + Ws/(N+1)+Wm*N/(N+1) + P*Pd*(Ws+2*T*T') +\\ P*(S+1)*T*(1-Pl) + Q*(N/(N+1)*2*T*+(1-Pl)*T)\end{aligned}$$

This response time is consistent with an intuitive view, for as the number of nodes approaches infinity, it is more and more likely that requests will occur from nonlocal sources and, thus, the likelihood of a transmission approaches one. Also, increasing the number of nodes is likely to diminish the wait time for a particular node because the work will be dispersed. Also, if the number of slave nodes is zero, the response time reduces to Q*Wm+Wm, which means that for any arrival, there is a wait at the node, and if the arrival is an update, there is a second wait for the lock.

The rest of this section involves the particular ddr support methods and how the transmission time differs for each.

In the deferred update scheme, only the retrieved ddr is updated. The defining relation updates are collected in the differential file. Since there are n ddrs and P is the probability of a ddr retrieval, the probability of a specific ddr being requested is on the average of P/n. Therefore, an average of Q/(P/n) or n*Q/P updates are collected between ddr retrievals. Because of culling, there may be less than n*Q/P updates involved. If the culling factor is k, the average number of tuples transmitted will be

$$T'=n*Q*k/P$$

The immediate actual causes any ddr retrieval to update all ddrs which have the same defining relations. Thus, if a defining relation is not stored with the ddr, the differential file update table must be transmitted to the ddr to be used in the update. Since no culling occurs, these files would have an average of Q/P tuples. Also, since all ddrs are updated when one is retrieved, this transmission may occur several times. We will assume that no more than one ddr defining relation is not local to the node. Therefore, the transmission time is

$$T'=n*Q/P$$

Note that if there were no culling in the deferred-actual method, the transmission times are the same.

The potential update scheme involves reconstructing the ddr whenever it is retrieved. Therefore, each time a ddr is retrieved, the entire defining relation is involved. If a ddr involves a nonlocal defining relation, the entire relation must be transmitted. Since S is based on the average relation size, transmission involves the request for the defining relation file (considered to be equivalent to transmitting one tuple) plus transmitting the file to the requesting node. Therefore, we have

$$T'=(S+1)/2$$

As noted before, a variable Pu has been introduced to represent the factor of what percentage the average number of updates is to the average relation size. By varying Pu, P and Q, different values for S can be obtained.

Introducing Conflict

This research is studying the effect on performance of ddr support methods under specific concurrency control schemes. We must, therefore, introduce the cost of conflict. The purpose of this section is to consider this cost. We will assume that no request will conflict more than once. Also, this conflict eventually is resolved without rollback. In other words, no deadlock occurs.

The first value needed is how long an average transaction holds a lock. In the master/slave system, we must consider two lock types-- a lock held by a file at the master and a lock held by a file at a slave, Lm and Ls. Then we have

Lm=	t(obtain a lock)	1/C
	+ t(perform the update)	1/U
Ls=	t(get to master)	T
	+ t(obtain the lock)	1/C
	+ t(perform update)	1/U
	+ t(get back to slave)	T

The average time a lock is held is

$$Lt = 1/C + 1/U + (1-P1)*T/(N+1) + 2*N*T/(N+1)$$

The average time an update has to wait for a lock is

$$Lt/2$$

if we assume that the conflicting update arrives at a random point in time.

If there are M items in the database, the probability that two updates vie for the same item is

$$1/M$$

Using Little's result which states that the average number of customers (in our case update requests waiting for a lock) is equal to the product of the arrival rate of customers to the system and the average time spent in the system, we find the average number of updates in the system holding a lock to be

$$J=Q*A*Lt/2$$

Therefore, Pw, the probability that an update has to wait for a lock is

$$Pw=J/M$$

So, for any arriving update lock request at the master node, if we assume that a conflict occurs at most once, the lock requests will arrive at a rate

$$L=Q+PwQ$$

i.e. the update lock arrivals plus the number of updates that are locked out and must try again. We assume that they will not be locked out a second time.

Comparison of the Performance of the Three Methods

As in the centralized model, certain parameter settings cause the algorithms to act in a similar manner. In a distributive system, transmission time is especially important. The three ddr support schemes differ in transmission time based on the number of tuples needed for a ddr retrieval when servicing nonlocal requests or when a defining relation is not found locally. In the potential method, transmission time is the product of the average defining relation size and the time it takes to transfer one tuple. The immediate-actual method depends on the update/retrieval arrival rate ratio and the number of ddrs. The deferred-actual depends on the culling factor, the number of ddrs and the update retrieval ratio. Blocking is an important parameter for both the potential and deferred-actual methods. The fact that these were also important parameters in the centralized model reinforces the importance of selecting the correct ddr support method based on a specific system usage.

Figure 26 illustrates the case where one ddr exists, there is no culling and the percentage of updates to relation size is one half. Here, as in the centralized case, the immediate-actual and deferred-actual methods' performance are equal. If overhead was not zero, the deferred-actual method's performance would be slightly worse. No advantage is gained from the deferred-actual method over the immediate-actual due to no culling and a block size of one. The potential method's response time is poorer as the update to relation size ratio decreases. This is consistent with the centralized model.

Figure 27 is the case where the number of ddrs is varied and the update/relation size ratio is 1. As in the

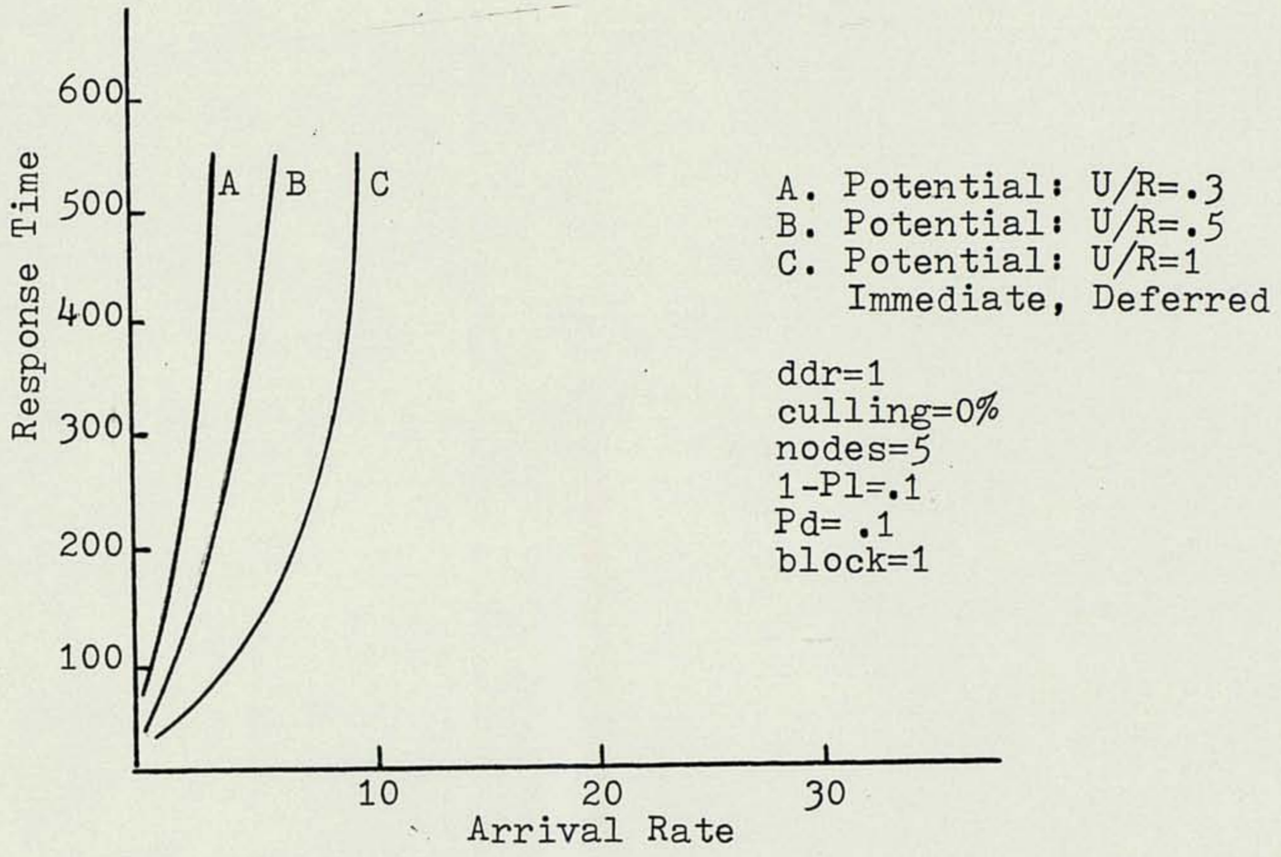


Figure 26. Master/slave: vary update to relation size ratio (U/R).

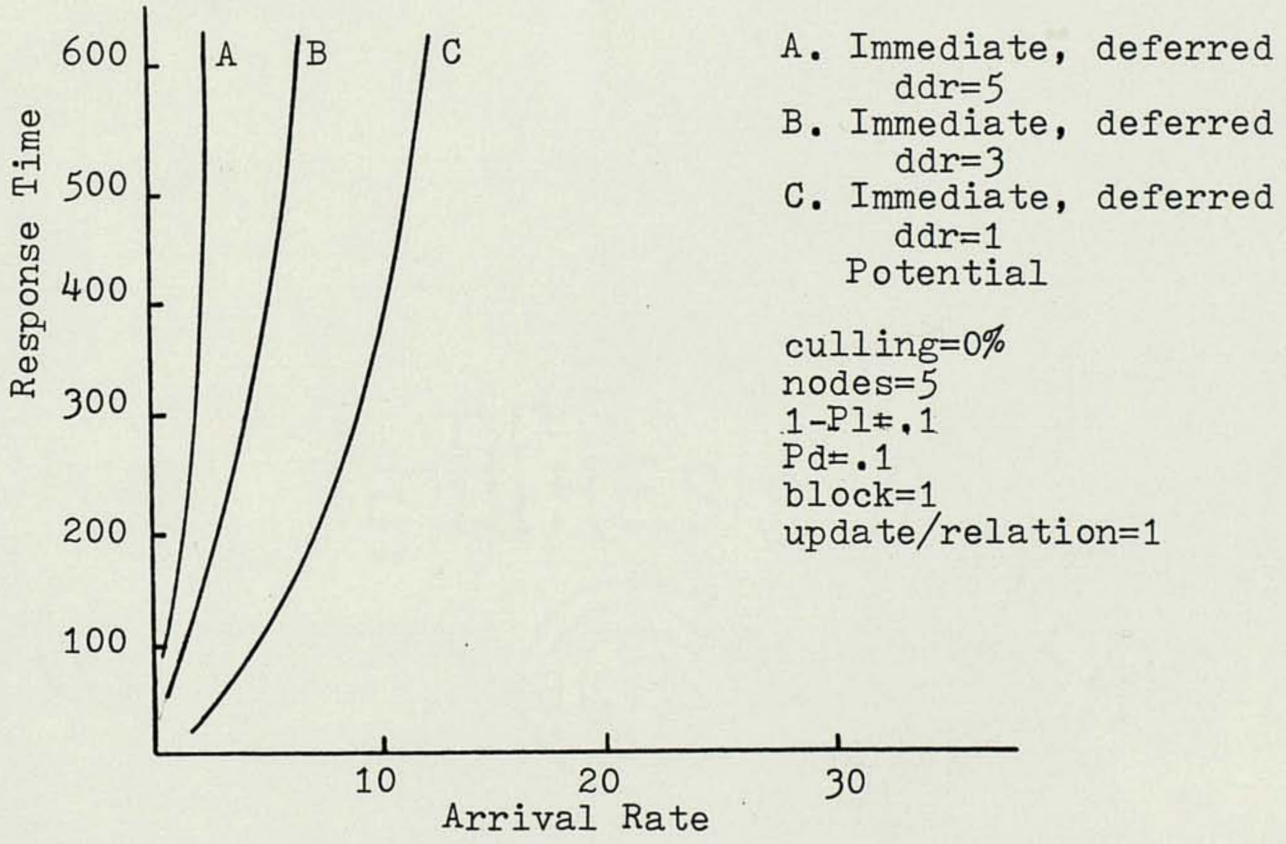


Figure 27. Master/slave: vary the number of ddrs.

centralized model, the immediate-actual and deferred-actual methods are equal in their performance with poorer response time as the number of ddrs increases. Here again, no advantage is reflected in the deferred-actual method because no culling is done and the block size is one. Also, there is no overhead reflected in this analysis.

Figure 28 illustrates the results of varying the culling. The immediate-actual and deferred-actual methods are equal in performance due to the update to relation size setting of one. The deferred-actual response time gets improves as the percent of culling increases.

Figure 29 illustrates the results of varying the block size. Response time for the deferred-actual and potential methods improves as the block size increases.

B. Distributed Concurrency Control

In the distributed control system, each node resolves its own conflicts. Thus all nodes are essentially the same and are similar to the master node presented in section A. The only difference is that nodes do not get all lock requests, but only those local to the node. Figure 30 shows the individual node flow. Updates and retrievals for files at other nodes are sent to those nodes immediately. Arrivals at the node consists of local requests from a node's terminals and requests from other nodes for retrievals and updates to its local files.

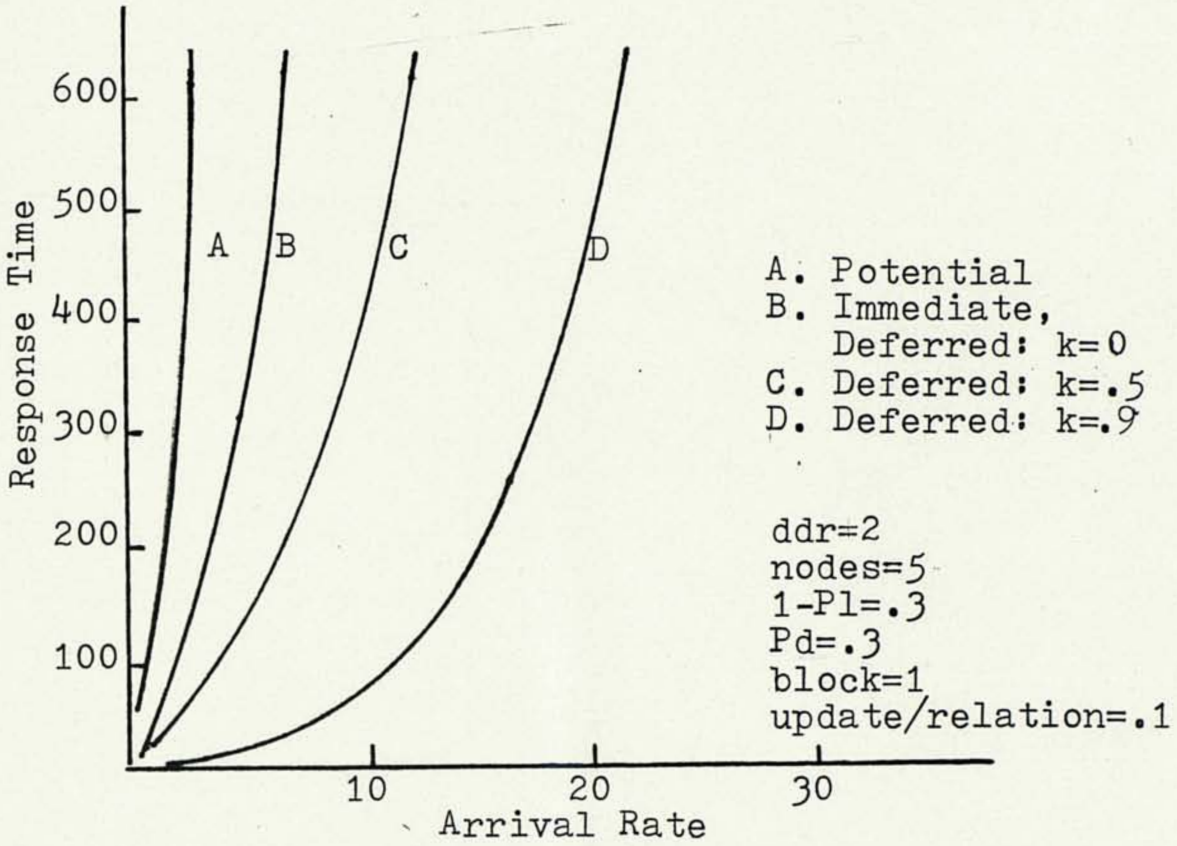


Figure 28. Master/slave: vary culling (k).

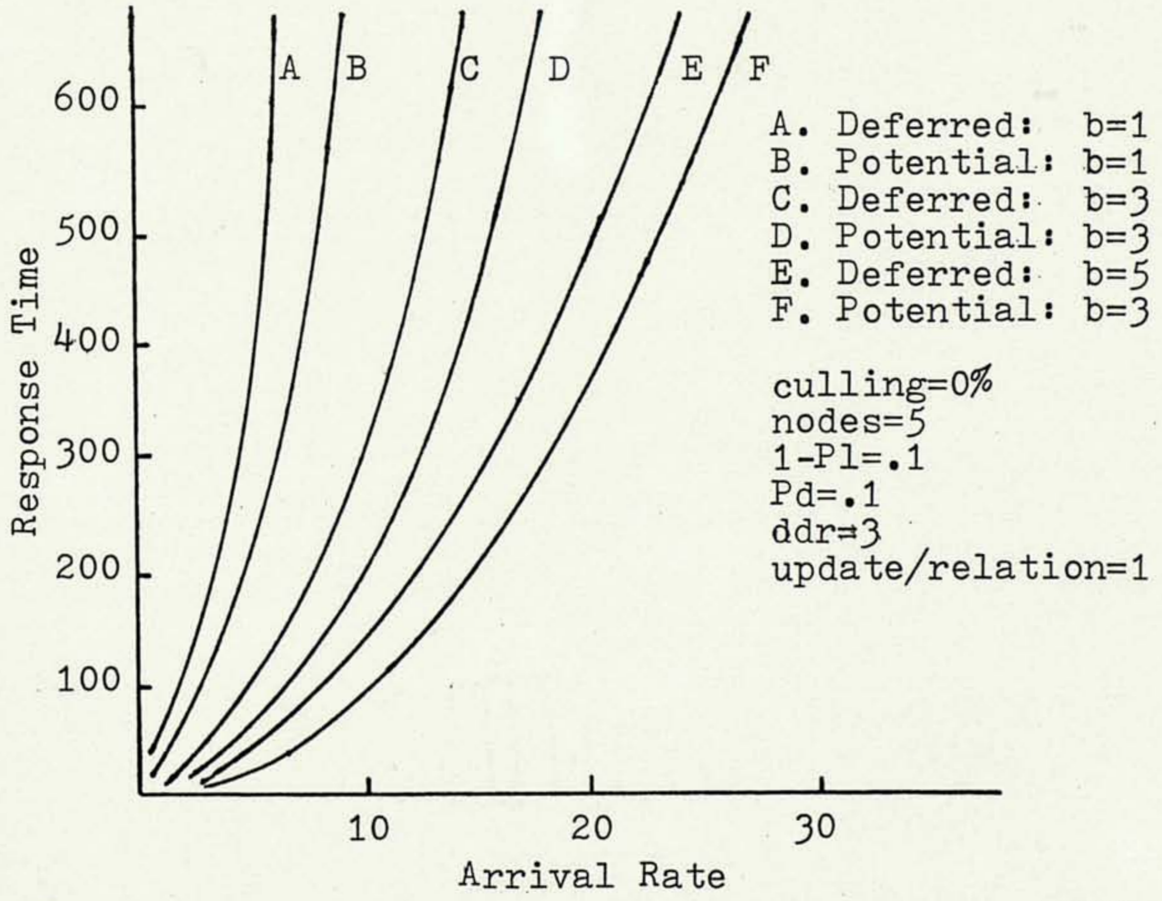


Figure 29. Master/slave: vary the block size (b).

Acknowledgements, in the case of updates, and the ddrs in the case of retrievals are sent back to the requesting node or local terminal. Figures 31, 32 and 33 show the internal node model for each ddr support method. As in the case of master/slave, a normalizing factor of 1/D has been introduced where D=P+L+Q.

The analysis for the distributed concurrency control follows the same arguments as those for the master node in the master/slave concurrency control system with the exception that each node in the distributed system handles locks to only local files. With respect to conflict, the average time a lock is held is

$$Lt = 1/C + 1/U + (1-Pl)*T$$

and the probability of a lock request arrival is

$$L/D$$

where

$$L=Q+Pw*Q$$

and Pw=Q*A*Lt/(2*M).

Based on the discussion for the master node of the master/slave system, the following values for the wait time W are shown below for each ddr support scheme. Detailed calculations can be found in Appendix 4a, 4b, and 4c.

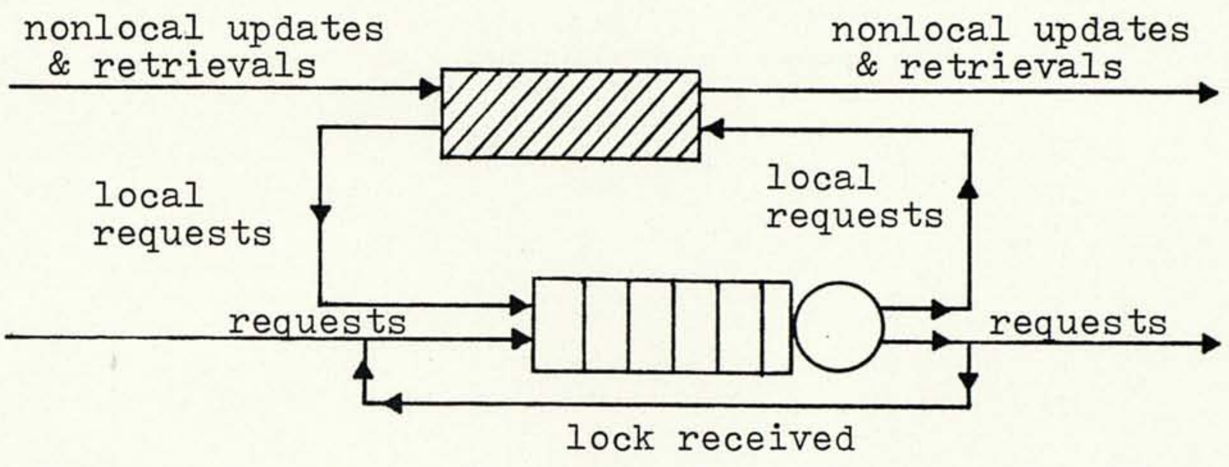


Figure 30. Internal information flow for distributed control.

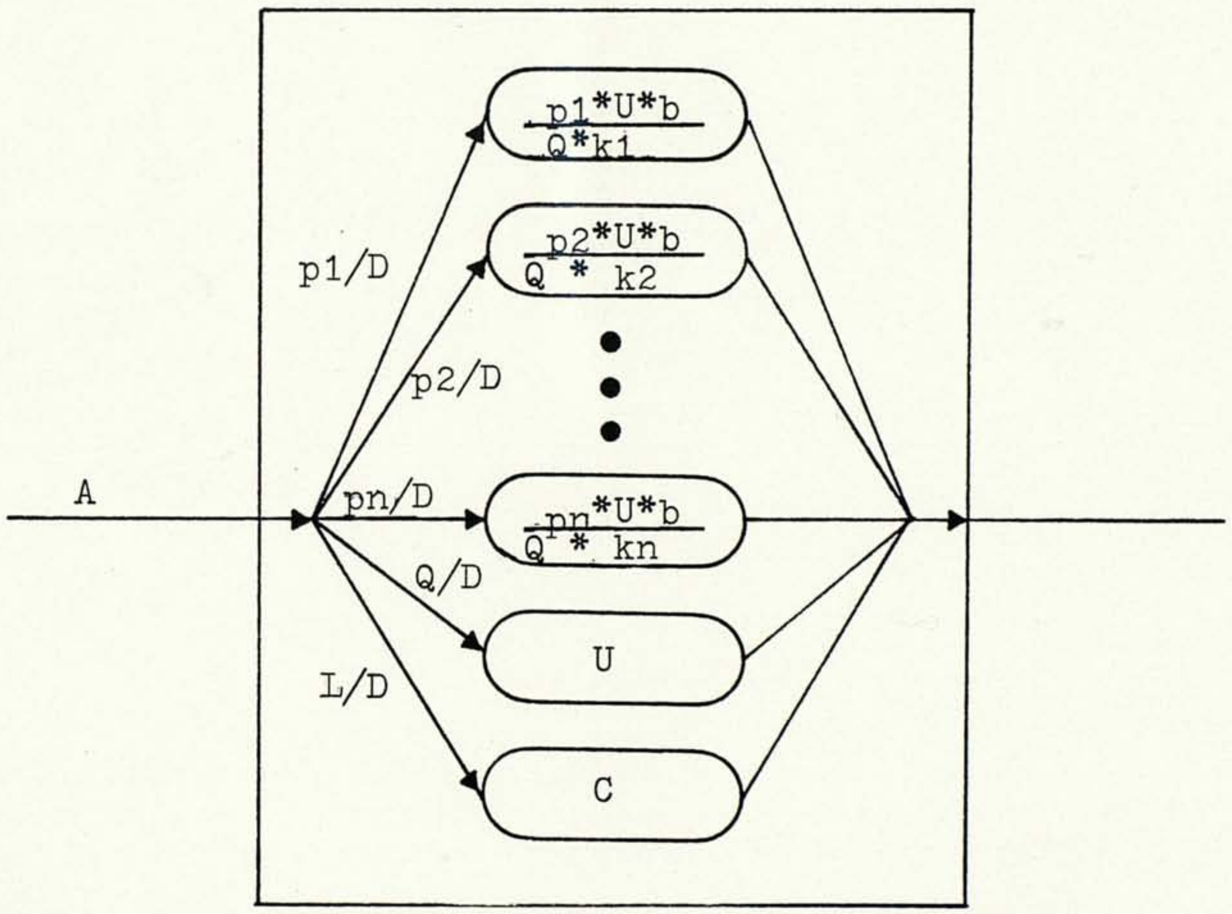


Figure 31. Distributed model for deferred-actual method.

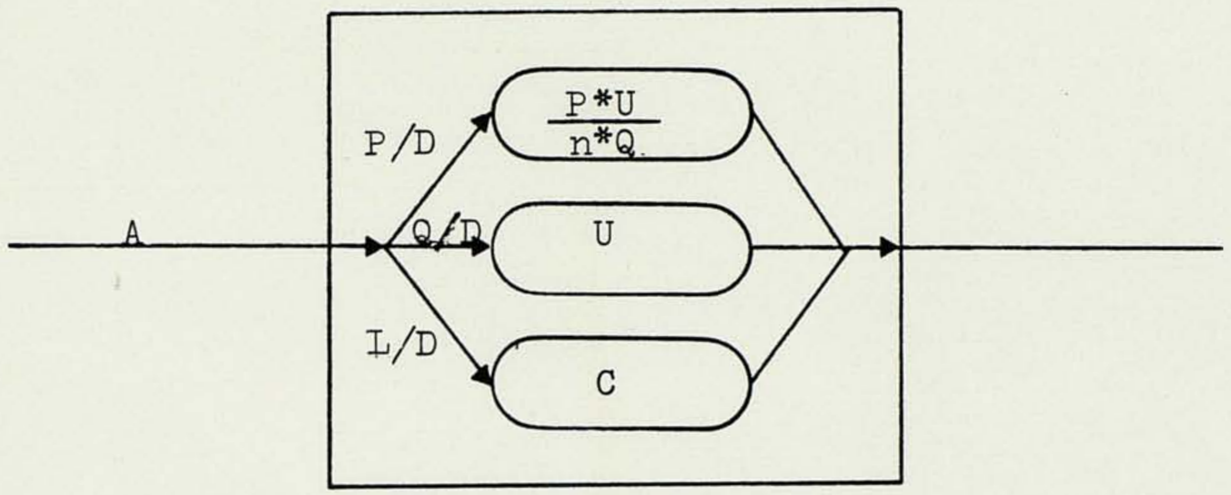


Figure 32. Distributed model for immediate-actual method.

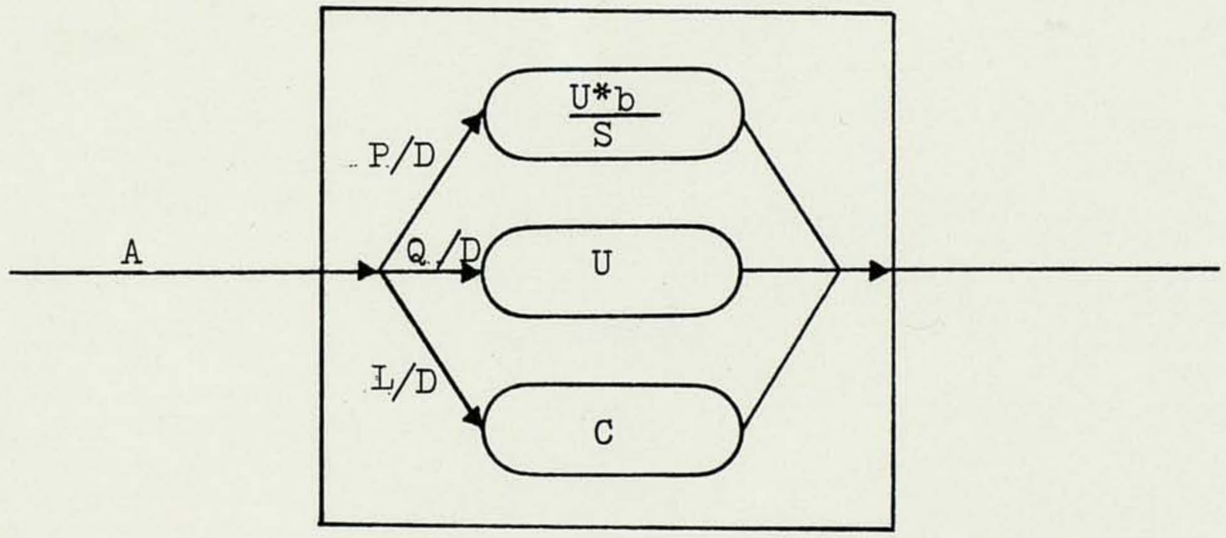


Figure 33. Distributed model for potential method.

Deferred-actual ddr support method:

$$W = \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right) + \frac{A * \left(\frac{Q ** 2 * SUM2}{D*U**2} + \frac{Q}{D*U**2} + \frac{L}{D*C**2} \right)}{1 - A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right)}$$

where SUM1 =(k1+k2+...+kn)/b and

SUM2=(k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2

The immediate-actual ddr support method:

$$W = \frac{Q * (n+1)}{D*U} + \frac{L}{D*C}$$

$$+ \frac{A * \left(\frac{(Q * n)**2}{D*P * U**2} + \frac{Q}{D*U**2} + \frac{L}{D*C**2} \right)}{1 - A * \left(\frac{Q * (n+1)}{D*U} + \frac{L}{D*C} \right)}$$

The potential ddr support method:

$$W = \frac{P * S}{D * U * b} + \frac{Q}{D * U} + \frac{L}{D * C}$$

$$+ \frac{A * \left(\dfrac{P * S^{2}}{D * (U * b)^{2}} + \dfrac{Q}{D * U^{2}} + \dfrac{L}{D * C^{2}}\right)}{1 - A * \left(\dfrac{P * S}{D * U * b} + \dfrac{Q}{D * U} + \dfrac{L}{D * C}\right)}$$

Response Time for a Transaction

Because the distributed control method resolves updates at each node and does not have to depend on one node for this purpose, the analysis of the response time for a transaction is less complex than the master/slave system. Once again, probabilities Pd and 1-Pl for a nonlocal defining relation and nonlocal updates, respectively, apply. The response time for a transaction is determined by finding the probability of an event happening and the wait time and transmission time involved.

One type of transmission which may occur is a request for a local update. The probability of a local update occurring is the product of the probability of an update occurring, Q, and the probability that it is local, Pl. The transaction must wait at the node twice, once for the lock and once for the update. Therefore, we have

$$Q*Pl*2*W \qquad (11)$$

A second type of transaction which might occur is a request for a nonlocal update. This involves the probability that the arrival is an update, Q, and is nonlocal, (1-Pl). The wait time involves the wait for the lock and the update, and the transmission time to and from the requested node. Therefore, we have

$$Q*(1-Pl)*(2*W+2*T) \qquad (12)$$

A third type of transaction is a local retrieval. This involves the probability that an arrival is a ddr retrieval, P, and the probability that it is local, Pl. The time involved is the wait time at the node to do the retrieval, W. If one of the defining relations in not local (with probability Pd) we must send either a message or the portion of the local defining relation to the nonlocal defining relation, and back. The needed computation and resulting number of tuples transmitted, depends on the ddr support method. Therefore, we will let the number of tuples transmitted be T'. The value of T' under each ddr support method is the same as T' under the master/slave system. This gives us then

$$P*Pl*(W+Pd*(W+2*T*T')) \qquad (13)$$

Finally, a transaction might be a request for a nonlocal ddr retrieval. The probability of a retrieval being nonlocal is (1-Pl). Since the ddr retrieval is nonlocal, a request for the ddr must be sent and the ddr must be returned. The average size relation in the database is S, and, therefore, the transmission to get the ddr is (S+1)*T. The analysis is the same as the local ddr retrieval in the case of a nonlocal defining relation. Therefore, we have

$$P*(1-Pl)*(W+(S+1)*T+Pd*(W+2*T*T')) \qquad (14)$$

Combining events (11) through (14), we have a total response time of

$$RT = Q*W + W + P*(Pd*(W+2*T*T')+(1-Pl)*(S+1)*T)) + Q*(1-Pl)*2*T$$

Note that if there were only one node, the last two terms would be zero because Pd and (1-Pl) are zero. This would leave Q*W+W as the response time. Here, as in the master/slave case, this is correct because an arrival would cause a wait at the node, and if the arrival were an update, an additional wait for a lock would be needed.

Comparison of the Performance of the Three ddr Support Methods Under Distributed Control

As in the master/slave system, the transmission is a large factor influencing the performance of the algorithm. Here again, the transmission time is based on how many tuples are being transfered. Therefore, this analysis depends heavily on the update/retrieval ratio and the culling and blocking factor. Figures 34 through 37 show the special cases discussed previously which make the algorithms behave in similar manners. As these figures indicate, when the number of ddrs is one and the update to retrieval ratio is one, the immediate and potential methods perform better than the deferred method. As the culling and block size increases and the update to defining relation size decreases, the deferred method performs significantly better.

C. Synchronized Management

In synchronized management, nodes form a ring in order to synchronize the updates. All records are appended to a token which continuously cycles the ring. When a token arrives at a node, the updates relating to that node are taken off the token and updates initiated by that node are appended. Therefore, updates are arriving in bulk at each node. This section begins by analyzing a simplified version of this algorithm and then proceeds to a more realistic representation of the algorithm.

A Simplified Analysis

In this analysis, we make the simplifying assumption that all retrieval and update arrivals are bulk, with all updates performed before retrievals are done. Note that

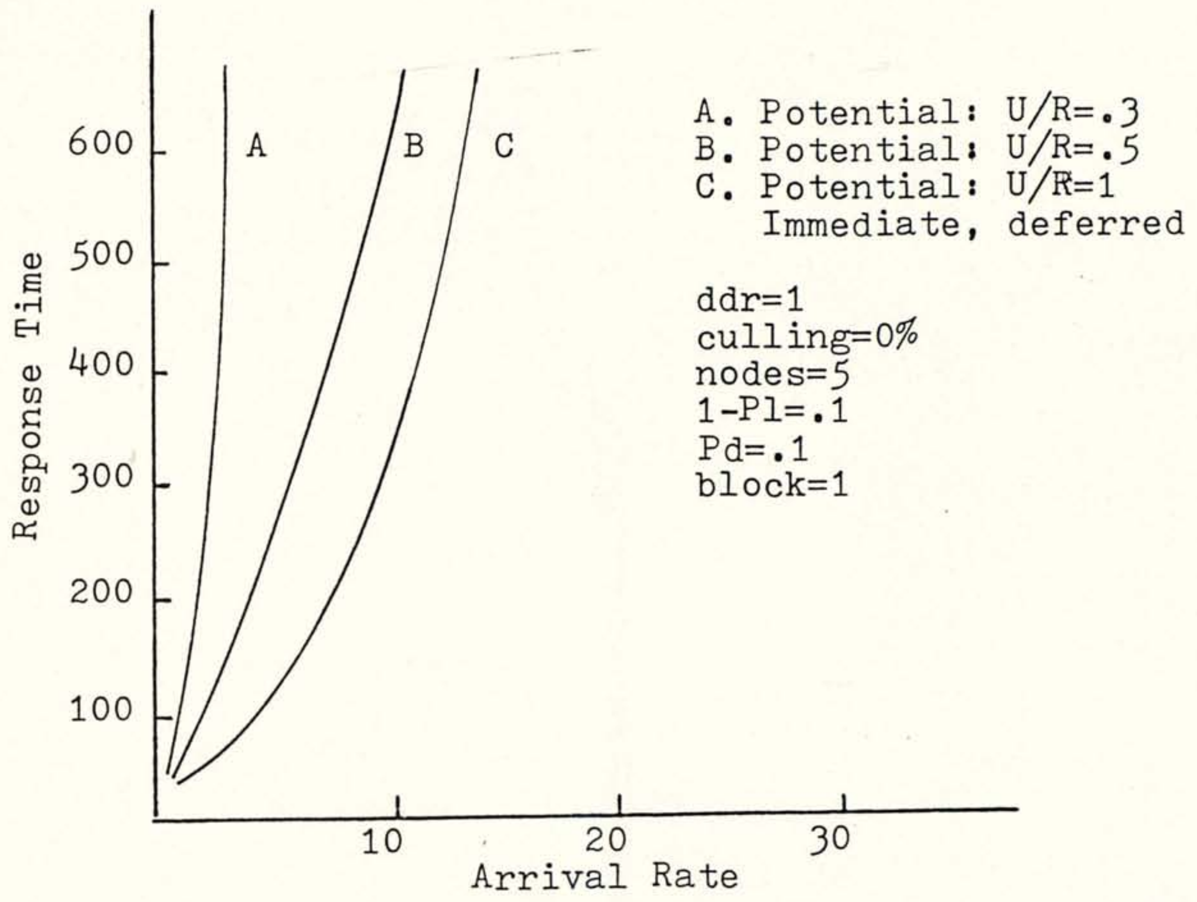


Figure 34. Distributed: vary update to relation size ratio (U/R).

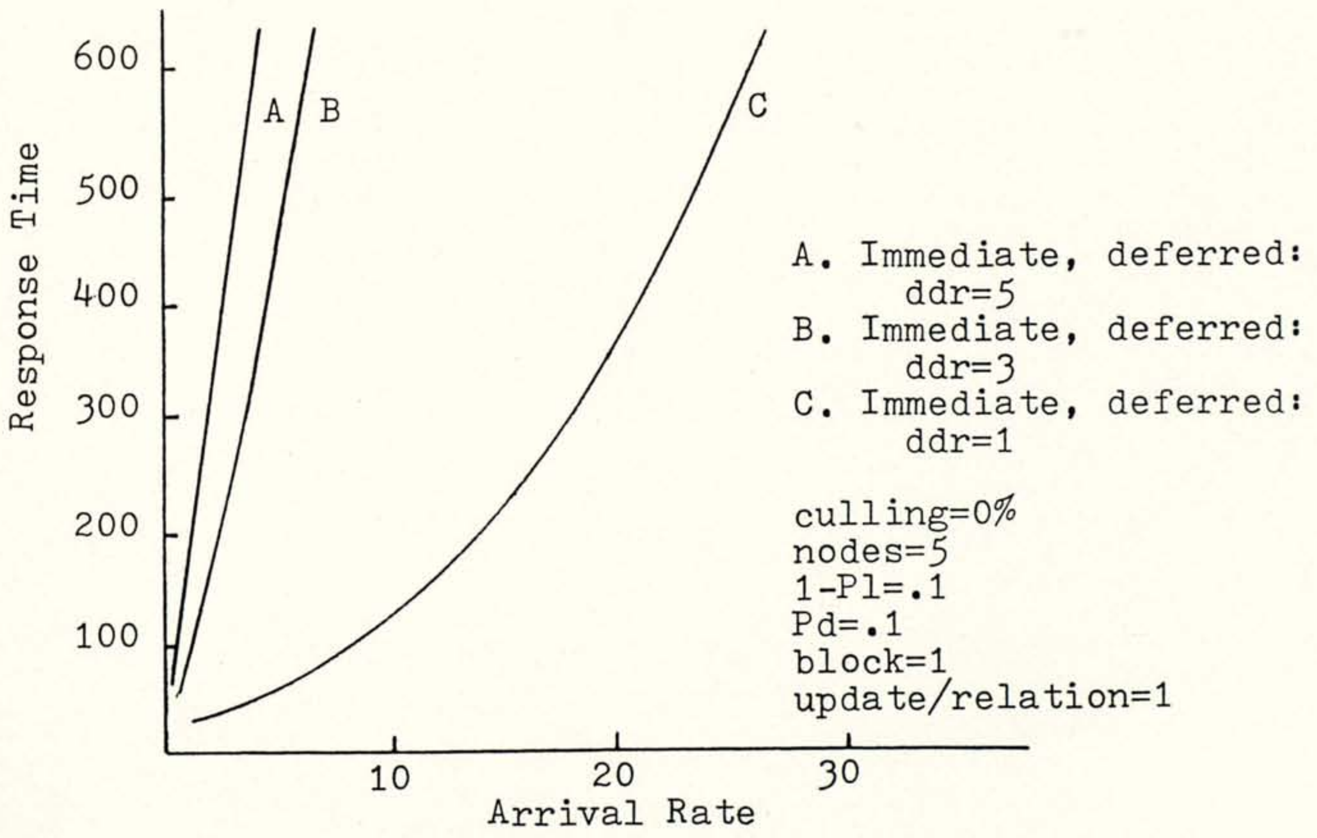


Figure 35. Distributed: vary the number of ddrs.

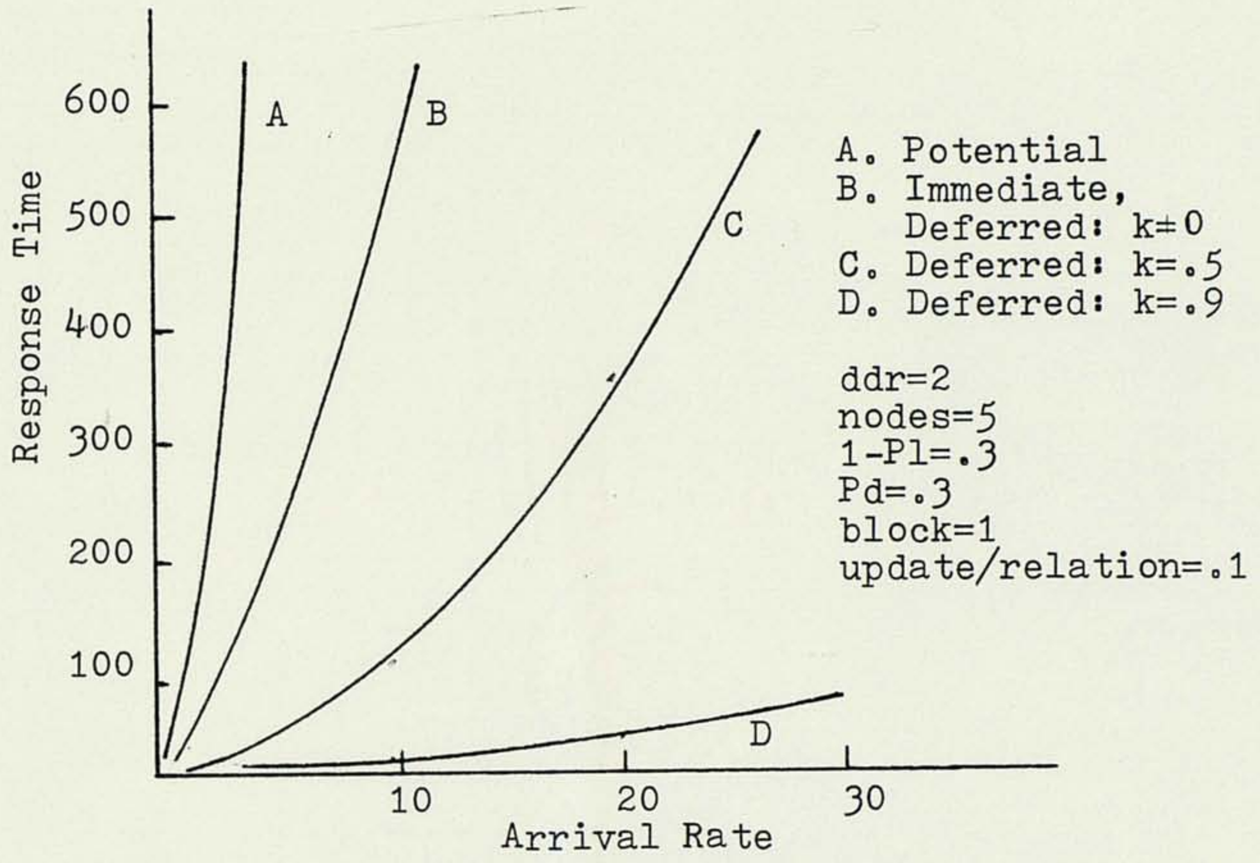


Figure 36. Distributed: vary culling (k).

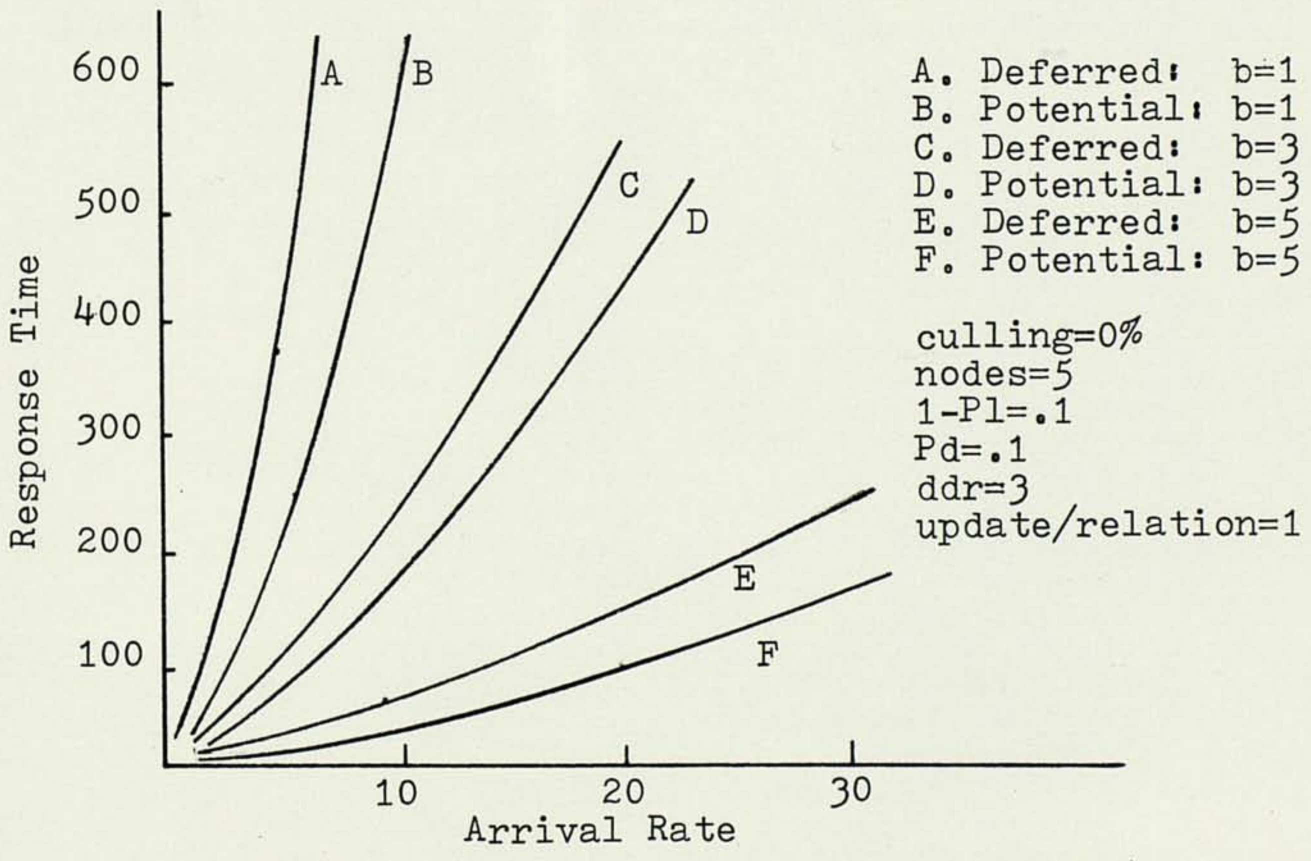


Figure 37. Distributed: vary the block size (b).

this is no longer an M/G/1 model because the arrivals are not Markov. Also, we assume that the amount of time it takes a node to remove its updates/retrievals from the token and append nonlocal updates/retrievals to the token and send it to the next node in the ring is negligible.

The average wait time for the token to reach the appropriate node is one half the time it takes to go around the ring. Let N be the number of nodes in the ring and T be the transmission time from node to node, then the approximate wait time for the token is

$$(N*T)/2$$

If the average update arrival rate is Q*A and the average retrieval arrival rate is P*A, then the average number of updates which arrive with the token is

$$(Q*A*N*T)/2$$

and the average number of ddr retrievals which arrive with the token is

$$(P*A*N*T)/2$$

Since no processing occurs until the token arrives, the average wait time for an update is one half the time it takes to service all updates plus the average time around the ring, i.e.

$$(1/2)*(Q*A*N*T/(U*2)) + N*T/2 = (N*T/2)+(Q*A/(2*U)+1)$$

Similarly, average response time for retrieval is the time it takes to service all updates plus the time it takes to service half of the retrievals. In the immediate-actual method, the average service time for a retrieval involves updating all ddrs when one ddr is retrieved,

$$\frac{N*T}{2}\left(\frac{Q*A*(n/2+1)}{U} + 1\right)$$

The deferred-actual updates only the ddr to be retrieved and uses culling as discussed in Chapter 3. Therefore, we have

$$\frac{N*T}{2}\left(\frac{Q*A*(1+SUM1/2)}{U} + 1\right)$$

where SUM1=(k1+k2+...+kn)/b.

The potential method must totally reconstruct the ddr when it is retrieved. Therefore we have

$$\frac{N*T}{2}\left(\frac{Q*A + P*A*S/2}{U}+1\right)$$

A More Realistic Analysis of the Synchronized Method

The previous analysis for the synchronized method was unnecessarily restrictive with regard to retrievals. Specifically, there is no reason to delay all retrievals until the token arrives. The analysis in this section will remove this restriction and allow retrievals to arrive in a

Markov distribution. The algorithm can then be analyzed using the M/G/1 model and, thus, produce results which can be compared with the other two methods which were analyzed using the M/G/1 model. It is unlikely that the synchronized algorithm could perform on a par with the other two algorithms using the analysis just completed.

Figure 38 shows the individual node flow and figures 39, 40 and 41 show the M/G/1 model for each node under the three ddr support schemes. Local retrievals are serviced locally while nonlocal retrievals are sent to the appropriate node. Updates issued locally for local files are held until the token arrives at which time it enters the execution queue as a bulk arrival with the other transactions that update files at this node. Updates issued locally for files at other nodes are sent to the appropriate node and held there for the token. When the token arrives, these updates are performed just as though they had been issued locally.

Let Q*A be the rate at which updates arrive to the system and P*A be the arrival rate for retrievals. Since N*T/2 is the average wait for the token to get around the ring, we can compute the average number of update requests that arrive with the token

$$Q*A*N*T/2$$

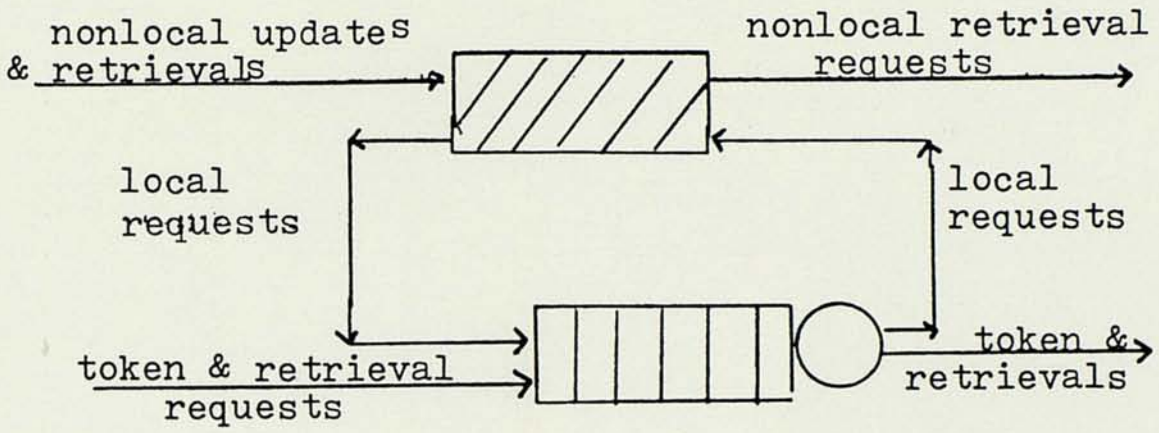


Figure 38. Internal information flow for synchronized

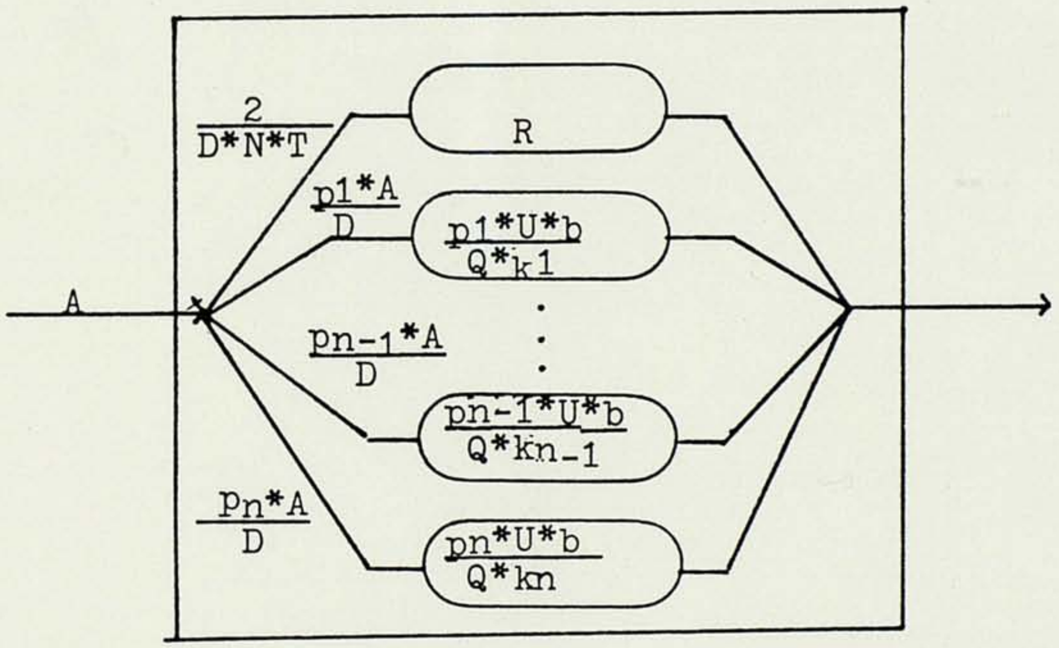


Figure 39. M/G/1 model for deferred actual in synchronized control.

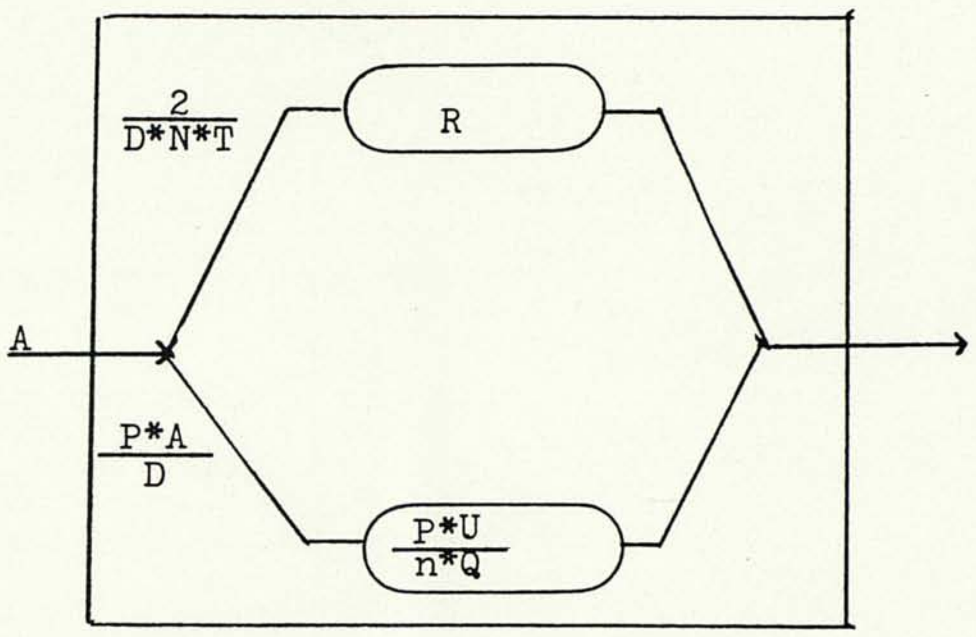


Figure 40. M/G/1 model for immediate-actual in synchronized control.

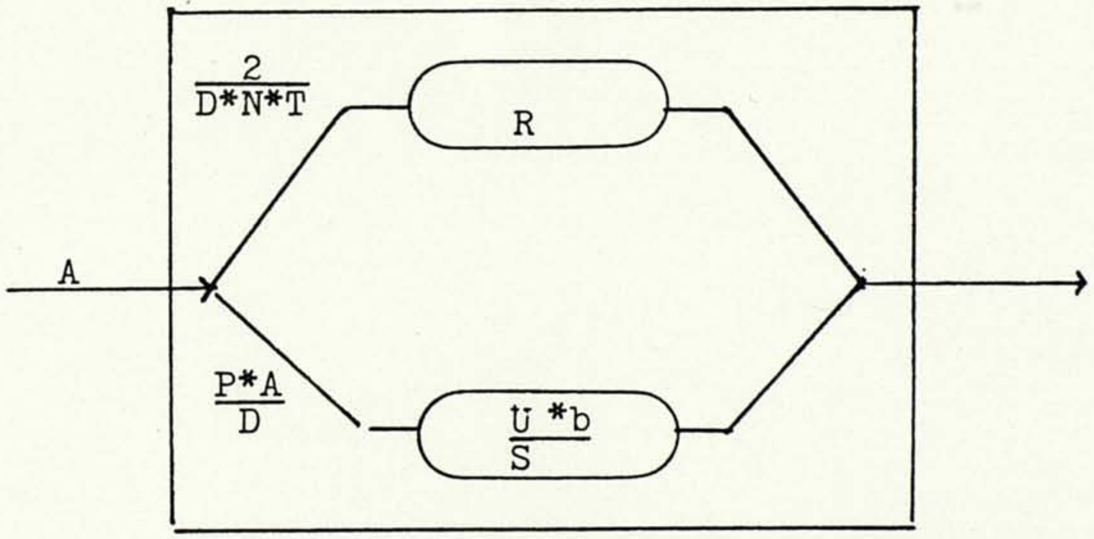


Figure 41. M/G/1 model for potential method in synchronized control.

The token arrival rate is 2/(N*T). Thus the total arrival rate at each node is

$$P*A+2/(N*T)$$

As in the master/slave and distributed control, 1/D is a normalizing factor. The average wait time for each ddr support method is shown below. Details are in appendix 5.

Deferred-actual ddr support method:

$$W = \frac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U} + \frac{\dfrac{A*(2*(R*U)**2 + N*T*A*SUM2*Q**2)}{D*N*T*U**2}}{1 - A * \left(\dfrac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U} \right)}$$

where SUM1=(k1+k2+...+kn)/b and

SUM2=(k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2.

Immediate-actual ddr support method:

$$W = \frac{2*R*U + N*T*A*Q*n}{D*N*T*U} + \frac{\dfrac{A*(2*P*(R*U)**2 + N*T*A*(Q*n)**2)}{D*N*T*P*U**2}}{1 - A * \left(\dfrac{2*R*U + N*T*A*Q*n}{D*N*T*U} \right)}$$

Potential ddr support method:

$$W = \frac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b} + \frac{\dfrac{A*(2*(R*U*b)**2 + N*T*A*P*S**2)}{D*N*T*(U*b)**2}}{1 - A * \left(\dfrac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b} \right)}$$

Response Time for A Transaction

In the synchronized control, updates are serialized by issuing tickets from a token that is sent around the ring. No conflicts can occur because of this serialization. The ddr retrieval arrivals are Markov.

In order to determine the response time for a transaction, we determine the probability of an event occurring and the wait time and transmission time involved. The response time for a transaction in the synchronized system can be divided into three distinct events. One event is an update. There is no distinction between a local and nonlocal updte in the synchronized system due to the fact that all must wait for the token and, thus, the transmission has occurred by passing the token. This has been incorporated in the service time. Therefore, for an update transaction we have

$$Q*W \tag{15}$$

A second transaction type is a local retrieval. Here, as in the previous sections, a defining relation may be stored at a node other than the node where the ddr is stored. Therefore, we must include this probability, Pd, and the wait and transmission time involved. As in the previous sections, we introduce a term T' as the transmission to and from the nonlocal node. This is dependent on the ddr support method used. We have, then

$$P*(Pl(W+Pd*(W+Q*T*T'))) \tag{16}$$

The last transaction type involves a nonlocal retrieval. This case is similar to the previous case except that the nonlocal ddr must be requested and retrieved. We use S, the average relation size, to determine this. We have then

$$P*((1-Pl)*(W+(S+1)*T+Pd*(W+2*T*T'))) \tag{17}$$

Combining (15) through (17), we find that the response time is

$$RT=Q*W+P*(W+Pd*(W+2*T*T')+((1-Pl)*(S+1)*T)))$$

The value of T' for each ddr support method is the same as that discussed under the master/slave system.

Comparison of the Three ddr Support Methods in Synchronized Concurrency Control

The special cases discussed in sections A and B are shown in Figures 42 through 45. These results are consistent with those in the other sections.

E. Comparison of the Three Concurrency Control Methods

Performance of the deferred-actual ddr support method is shown with respect to the three concurrency schemes in Figure 46. Both the synchronized and distributed concurrency control schemes have better performance than the master/slave system. In the synchronized method, response time is better or as good as the distributed concurrenty control scheme as long as the system utilization is low. As the arrival rate increases, response time degrades quickly for the synchronized control while the distributed control does not suffer as greatly and has a generally lower response time than the synchronized method after this point.

Previous research into these methods has predicted that the master/slave should perform the best. In all cases, this was not expected nor did it seem reasonable. However, most of this prior research did not consider conflicts. This must be considered if the synchronized method is to be at all competitive. It is the "no rollback" aspect of the synchronized method that makes it

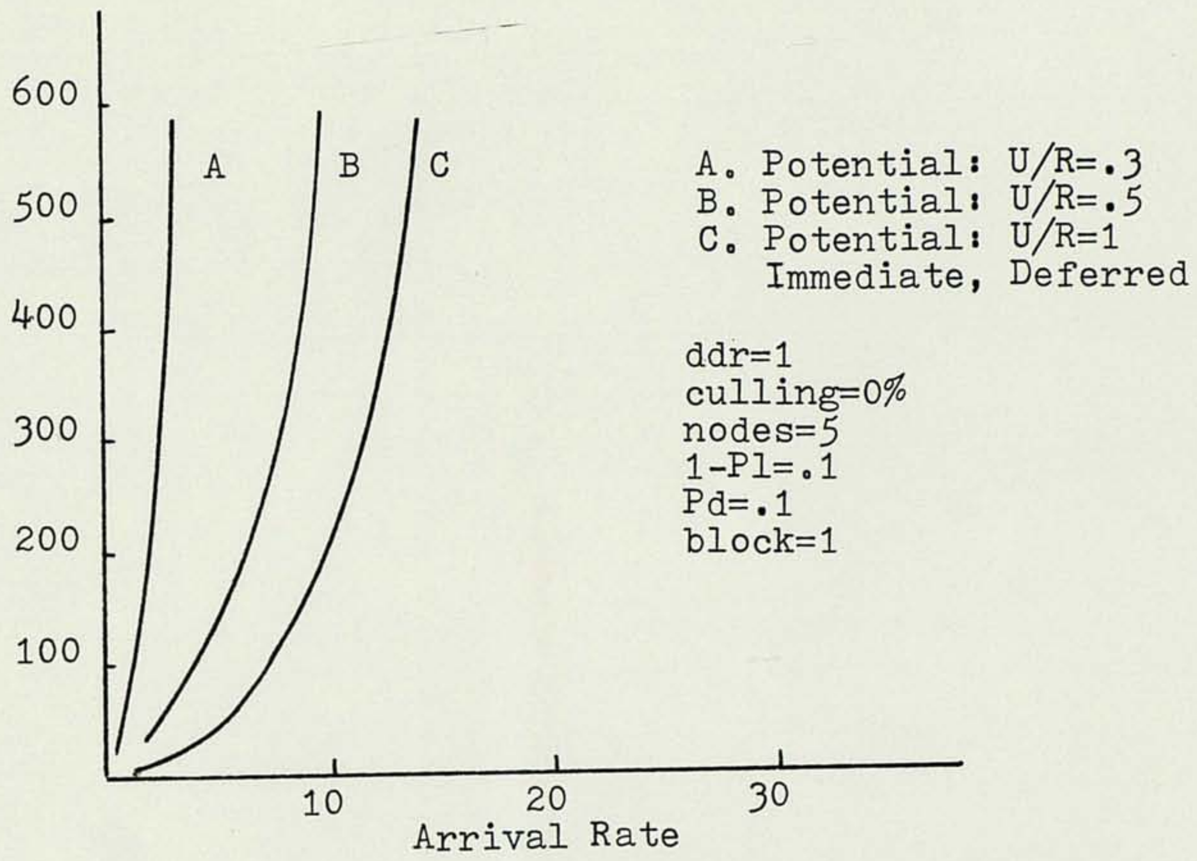


Figure 42. Synchronized: vary update to relation size ratio (U/R).

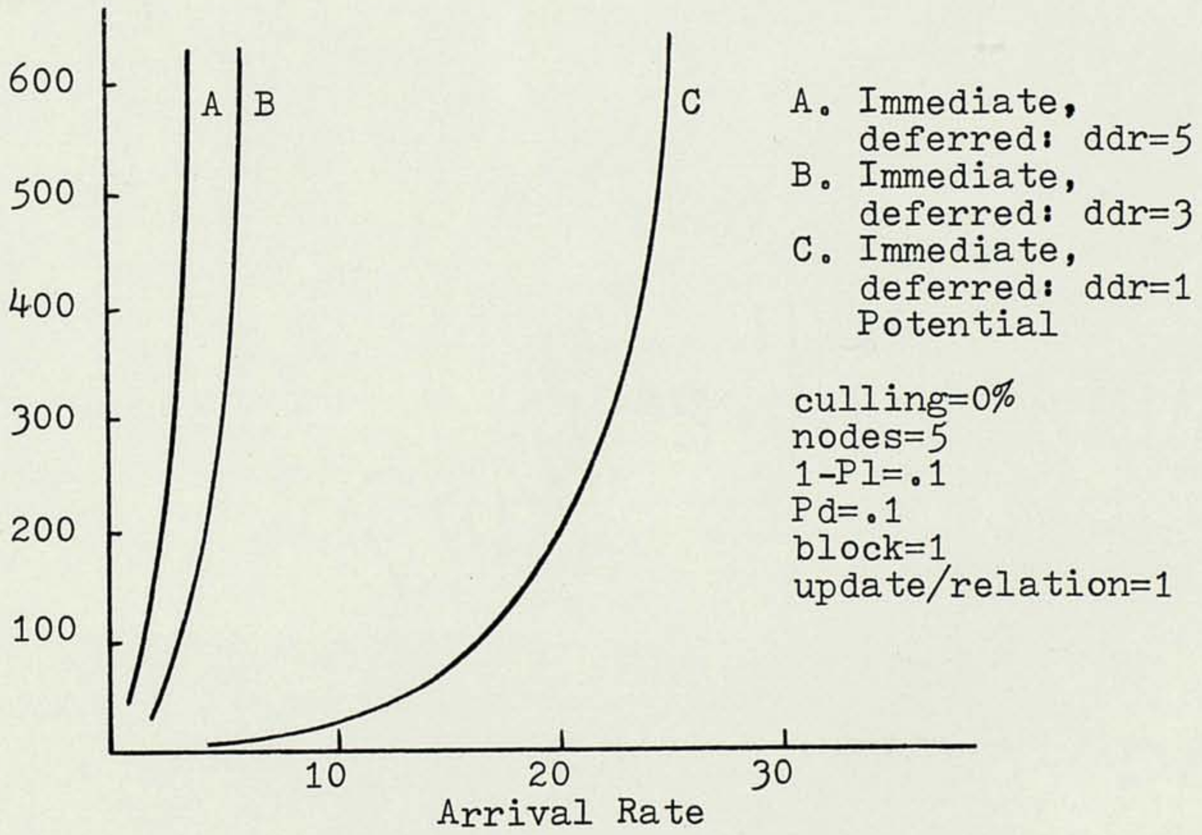


Figure 43. Synchronized: vary the number of ddrs.

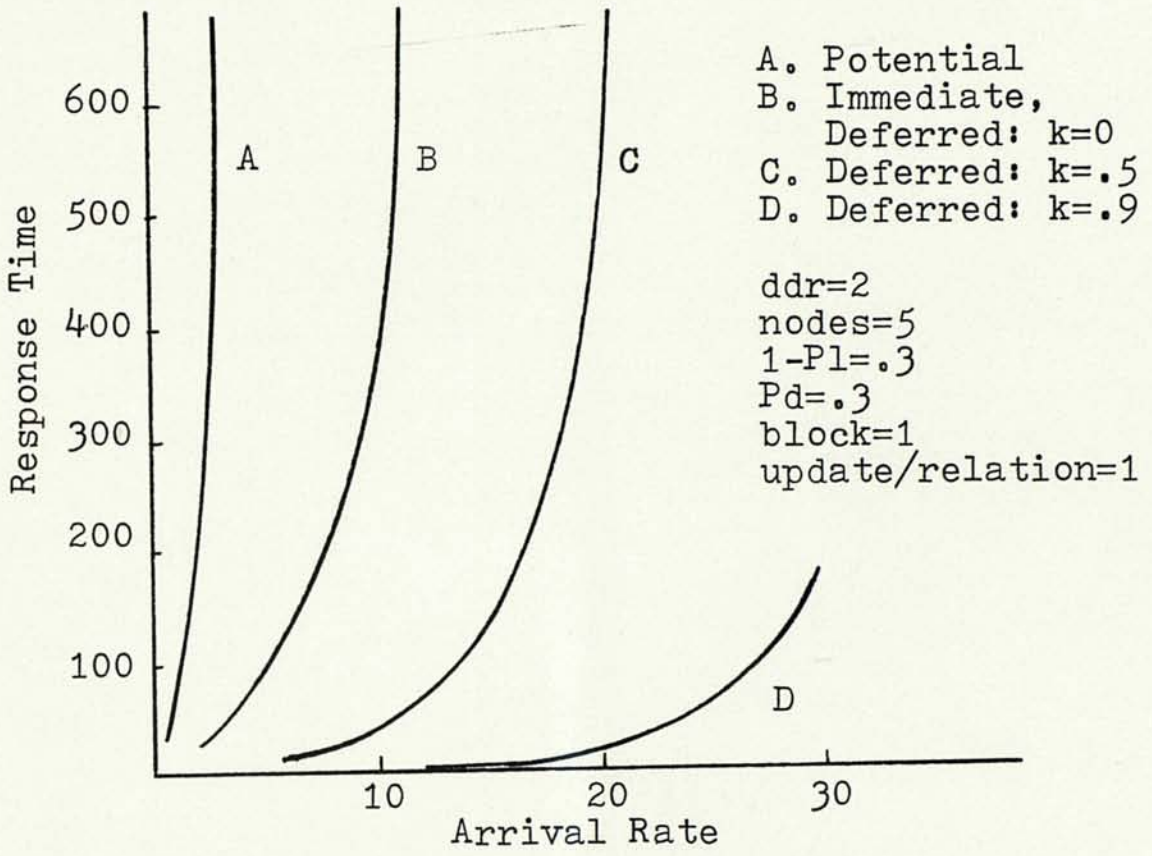


Figure 44. Synchronized: vary culling (k).

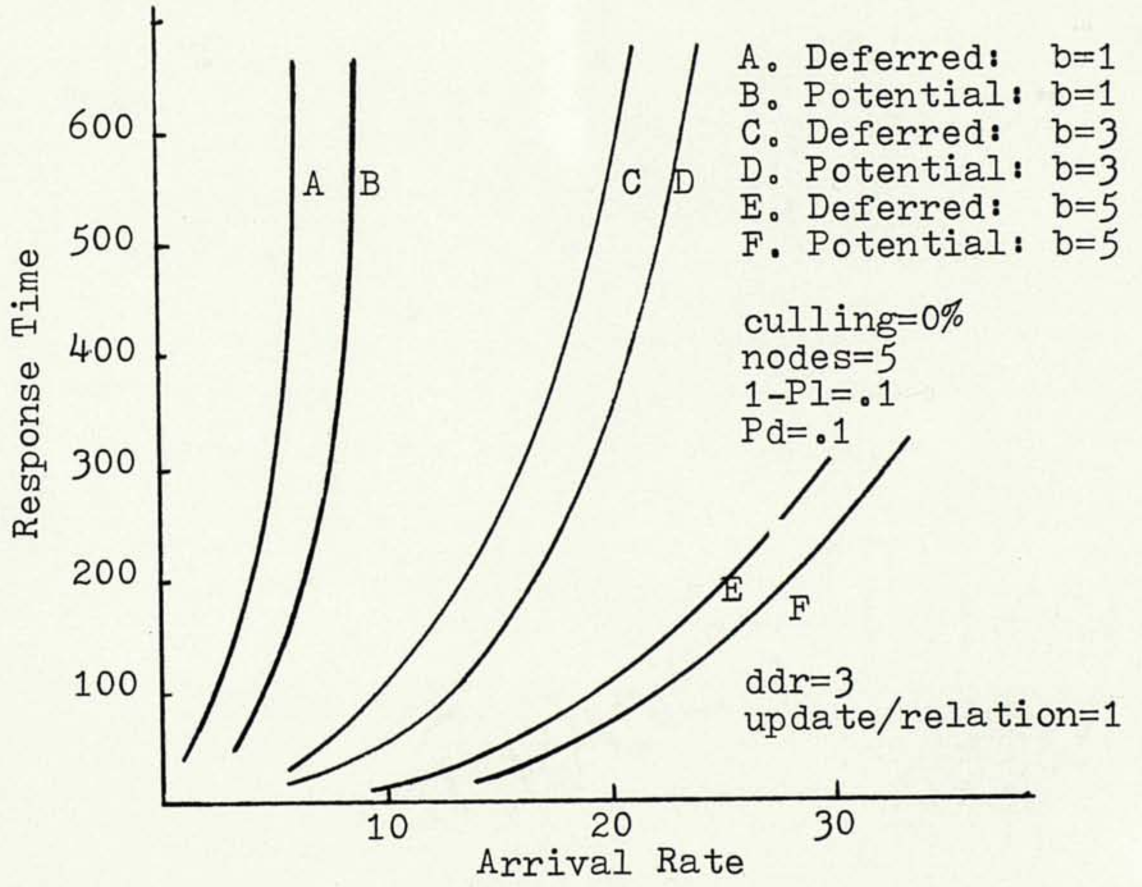
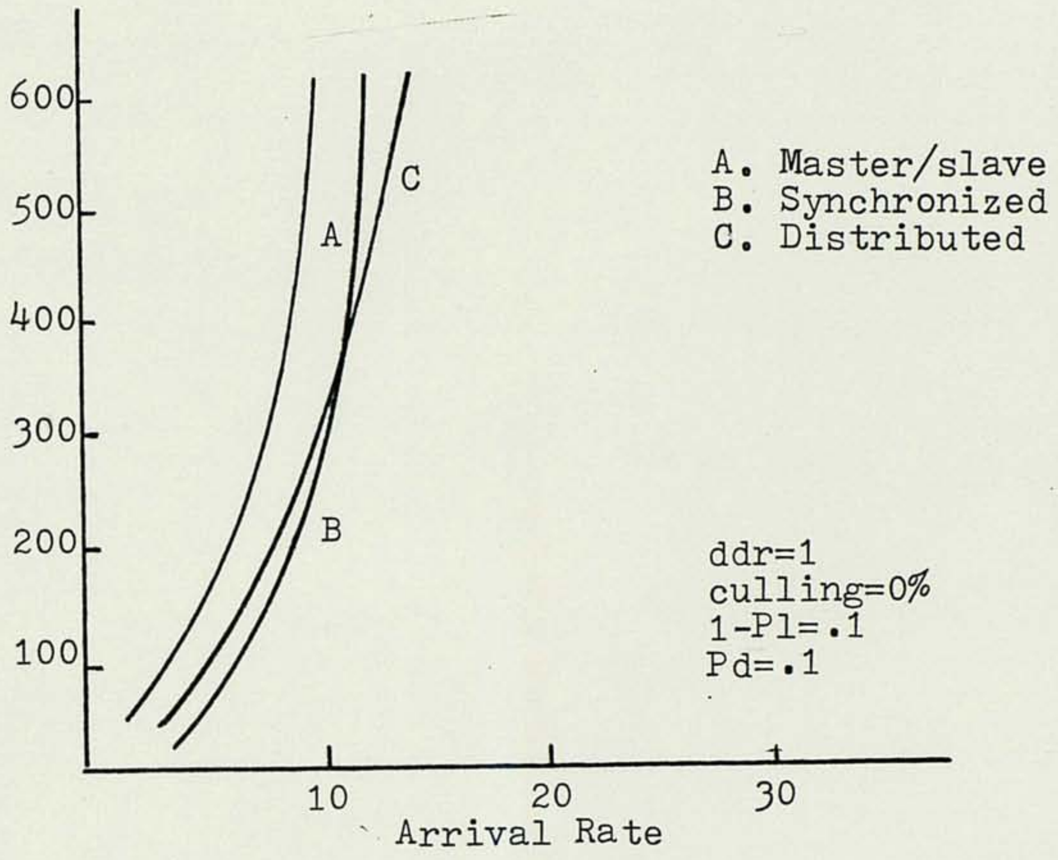


Figure 45. Synchronized: vary the block size (b).



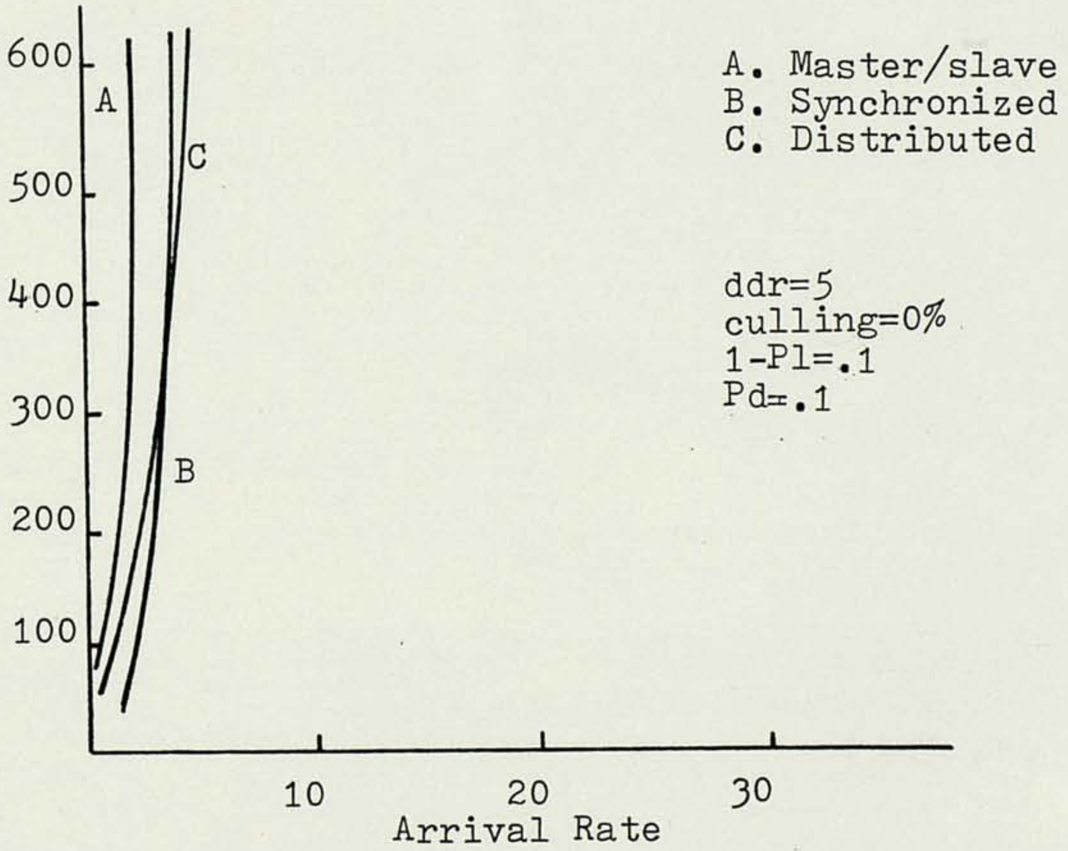


Figure 46. Performance of deferred-actual in three concurrency schemes .

appealing. If it is compared to systems which have no conflict and therefore no rollback, it will not compare favorably. Therefore, in this research, the cost of conflicts was determined, both in increased arrival rate for requesting locks with conflict and in providing an additional service stage for granting the lock. The analysis reflects this extra load on the system.

CHAPTER V

CONCLUSIONS

This chapter gives an overview of what was learned from this research. We will begin by discussing conclusions regarding the ddr support methods and then look at the concurrency performance. The methodology is discussed and possibilities for future work are noted.

A. Discussion of the Three DDR Support Methods

Three methods have been presented for the support of ddrs. The potential method totally reconstructs the ddr each time it is needed. The immediate actual method updates all ddrs defined by the same defining relation whenever one ddr is retrieved. The deferred actual keeps tuples in a differential file, culling whenever the same tuple is used more than once to update a relation.

Because database performance is highly dependent upon patterns of usage, there is no one conclusion that can be gleaned from the analysis performed here. In order to set boundaries, and observe any anomalies that might be present, we first measured the performance of each support method for a set of special cases. These included parameter settings that caused one method to have an extremely poor or good performance. We also chose cases that cause several of the methods to perform in an identical manner. We found that, by limiting the number of ddrs to one, making the ratio of the update to defining relation size and the block size equal to one, the immediate-actual and the potential method, respectively, has the same performance as the deferred method if no overhead and culling are incurred.

Intuitively, these cases do not seem to be the norm. For instance, examining the case of the ratio of derived relation to defining relation size, selections, projections and the set operations of union, intersection and difference would yield a ddr with the same or a few number of tuples/attributes. In the case of join, because we require that a join be done only on the key, the derived data is, again, no larger than the number of tuples in it defining relations. Hence, ddrs are in general smaller than their defining relations. The case that each relation defines at most one ddr is very limiting. For example, suppose one defining relation, STUDENT-ROSTER(SNO,SNAME,CLASS,GPA), exists. Several derived relations might be needed, such as, selection by class, selection by GPA range, and projection on SNO and SNAME.

Although these special cases may not be the norm, they were considered in order to determine the bounds on relative performance. The potential method performs poorly as the update to relation size ratio decreases. Both the deferred-actual and potential methods are sensitive to block size while the actual methods response time increased as the number of ddrs increased. The deferred-actual method was also sensitive to culling and overhead. In general, no one method performed the best in all cases. Different circumstances causes each method to perform differently.

In general, it was found that the deferred-actual method performs very well under what we consider to be normal conditions. Thus, this research has made a case for more study of the actual methods. This result is in contrast to the claims of others in that it supports the use of the actual method in some cases.

B. The Three Concurrency Methods

The ddrs' support scheme performance was measured in three concurrency methods in the distributive database environment. It was found that, as suspected, the master/slave and distributive control performances were sensitive to lock conflicts, while the synchronized method was more sensitive to transmission times between nodes.

In all of the systems, arrival rates were adjusted to consider arrivals generated by the system, as well as from external sources. For instance, the arrival rate at the master node in the master/slave system was adjusted to reflect lock requests from all nodes and requests generated by conflicts. Thus, the model defined here is more realistic that those chosen by other researchers. For this reason, intuitive observations, such as the distributive control should perform better than the master-slave system, were supported. This has not been the case in previous research, but in these efforts, critera such as conflicts and internally generated arrivals were not considered.

C. The Methodology

The major contribution of this dissertation is the development of a methodology through which database algorithms may be analyzed. The specific algorithms looked at here are update schemes for dynamically derived relations.

This methodology has proven to be an important tool in modelling the specific update schemes as well as concurrency control in distributive database environments. It makes use of the M/Hr/1 model which allows parallel stages within the service facility. This is significant in that different classes of customers (in this case, updates and retrievals) can be analyzed. Similar attempts to do this in the M/M/1 model proved very restrictive.

A key factor in the efficacy of the models developed here is their use of the history of how data is employed in the database. Several parameters are included so that the model can be used to analyze actual systems.

The methodology is flexible and uses concepts which are intuitively obvious, while having a sound, theoretical base. This allows the analysis to be done in a straightforward fashion.

D. Future Work

The research presented in this dissertation has indicated several areas for future work.

First, the analysis was done with the relational model. Further research might be warranted to see how appropriate the methods developed here are to the analysis of database problems in other models.

Second, ddrs defined by more than two defining relations and ddrs defining ddrs should be examined. In addition, a combination of two of the methods might be an area of interest. For example, a ddr may exist in actual form until it becomes too large to manage and, at that time, reduces to potential form.

This research examined only partitioned distributed databases. A third area might be to study performance in a partially or totally replicated database. Rollback and noninstantaneous token handling is a fourth area that might be explored.

APPENDICES

APPENDIX 1

USING THE HISTORY OF DATABASE RETRIEVALS AND UPDATES TO DETERMINE THE AVERAGE SERVICE TIME

Chapter III, section B, presented results which illustrated that, when given the frequency of update and retrieval calls, probabilities can be computed which can be used in representing the average retrieval and update times. The details of the calculations are presented here.

Figure 8 is the state transition diagram which represents m times as many update arrivals as ddr retrieval arrivals, no culling and infinite service rates. Computing the probabilities of being in states S1, S2,... we have

$$P(S1) = \frac{m * A * P(So)}{A + m * A} = \frac{m * P(So)}{1 + m}$$

$$P(S2) = \frac{m * P(S1)}{1 + m} = \left(\frac{m}{1 + m}\right)^{**2} * P(So)$$

.
.
.

$$P(Sk) = \frac{m * P(Sk+1)}{1 + m} = \left(\frac{m}{1 + m}\right)^{**k} * P(So)$$

Note that P(So)+P(S1)+...=1. Therefore, solving for P(So), we have

$$P(S_o) + \frac{m}{1+m} P(S_o) + \left(\frac{m}{1+m}\right)^{**2} * P(S_o) + \ldots = 1$$

$$P(S_o)\left(1 + \frac{m}{1+m} + \left(\frac{m}{1+m}\right)^{**2} + \ldots\right) = 1$$

$$P(S_o) = \frac{1}{1 + \frac{m}{1+m} + \left(\frac{m}{1+m}\right)^{**2} + \ldots}$$

$$P(S_o) = \frac{1}{1 + \frac{m}{1+m} + \left(\frac{m}{1+m}\right)^{**2} + \ldots} * \frac{1 - \frac{m}{1+m}}{1 - \frac{m}{1+m}}$$

$$= 1 - \frac{m}{1+m} = \frac{1}{1+m}$$

The expected number of customers, c, is

$$c = \sum_{i=1}^{\infty} i*P(S_i)$$

$$= \frac{1}{1+m} * \left(\frac{m}{1+m} + 2*\left(\frac{m}{1+m}\right)^{**2} + \ldots\right) * \frac{\left(1 - \frac{m}{1+m}\right)^{**2}}{\left(1 - \frac{m}{1+m}\right)^{**2}}$$

$$= \frac{1}{1+m} * \frac{m}{1+m} * \left(\frac{1}{1 - \frac{m}{1+m}}\right)^{2}$$

$$= \frac{m}{(m+1)^{2}} * \frac{(m+1)^{2}}{1}$$

$$= m$$

which is the ratio of the update arrival rate to the retrieval arrival rate. Therefore, by determining the frequency of update and retrieval calls, we can determine how many updates occur between ddr retrievals.

APPENDIX 2

THE ddr SUPPORT METHODS IN A CENTRAL DATABASE

This appendix presents details of the analysis presented in Chapter III which dealt with the central database environment. We will first simplify the M/Hr/1 formulas and then proceed to the derivation of the wait time for each ddr support method.

Chapter III, section C, presented formulas for the M/Hr/1 queueing model. In this section we will simplify the queue length and wait time formulas so that they can be expressed in terms of the average service time, X, the second moment of the probability density function, X2, and the arrival rate, A. We are given that

q = R + (R**2 *((1+C2/(2*(1-R)))

where R=A*X and C2=(X2/X**2)-1

Substituting, we have

q = A*X + (A*X)**2 * ((1+((X2/X**2)-1)/(2*(1-A*X}}

= A*X + (((A**2)*X2)/(2*(1-A*X))

The analysis in this dissertation will use this formula to find the queue length.

2a. The Immediate-Actual ddr Support in a Central Database

Chapter III, section D, presented the wait time for the immediate-actual ddr support method in a central database environment. Details of the calculations are presented here.

The service time

$$X = \frac{P}{1} * \frac{Q * n}{P * U} + \frac{Q}{1} * \frac{1}{U}$$

$$= \frac{Q * (n+1)}{U}$$

The second moment of the pdf

$$X2 = 2*\left(\frac{P}{1} * \frac{(Q*n)**2}{(P*U)**2} + \frac{Q}{1} * \frac{1}{U**2} + \frac{L}{1}\right)$$

$$X2 = 2*\left(\frac{(Q*n)**2}{P*U**2} + \frac{Q}{U**2}\right)$$

The expected queue length

$$q = A*\left(\frac{Q * (n+1)}{U}\right)$$

$$+ \frac{A**2*2 * \left(\frac{(Q*n)**2}{P*U**2} + \frac{Q}{U**2}\right)}{2*\left(1 - A*\left(\frac{Q*(n+1)}{U}\right)\right)}$$

$$= A * \left(\frac{Q * (n+1)}{U} + \frac{A * \left(\frac{(Q * n)^{2}}{P * U^{2}} + \frac{Q}{U^{2}} \right)}{1 - A * \left(\frac{Q * (n+1)}{U} \right)} \right)$$

The expected wait time

$$W = A * \left(\frac{Q * (n+1)}{U} + \frac{A * \left(\frac{(Q * n)^{2}}{P * U^{2}} + \frac{Q}{U^{2}} \right)}{1 - A * \left(\frac{Q * (n+1)}{U} \right)} \right) / A$$

$$= \frac{Q * (n+1)}{U} + \frac{A * \left(\frac{(Q * n)^{2}}{P * U^{2}} + \frac{Q}{U^{2}} \right)}{1 - A * \left(\frac{Q * (n+1)}{U} \right)}$$

$$W = \frac{Q * (n+1)}{U}$$

$$+ \frac{A * \left(\frac{(Q*n)**2}{P*U**2} + \frac{Q}{U**2}\right)}{1 - A * \left(\frac{Q*(n+1)}{U}\right)}$$

2b. The Deferred-Actual ddr Support in a Central Database

Chapter III, section, E presented the wait time for the deferred-actual ddr support method in a central database environment. The details of the calculation is presented here.

The service time:

$$X = \frac{p1}{1} * \frac{k1*Q}{p1*U*b} + \frac{p2}{1} * \frac{k2*Q}{p2*U*b} + \ldots$$

$$+ \frac{pk}{1} * \frac{kn*Q}{pn*U*b} + \frac{Q}{1} * \frac{1}{U}$$

$$= \frac{Q}{U} (k1+k2+\ldots+kn)/b + \frac{Q}{U}$$

$$= \frac{Q * SUM1}{U} + \frac{Q}{U}$$

where $SUM1 = (k1+k2+...+kn)/b$

The second moment of the pdf

$$X2 = 2*\left(\frac{p1}{1} * \frac{(k1*Q)^{2}}{(p1*U*b)^{2}} + \frac{p2}{1} * \frac{(k2*Q)^{2}}{(p2*U*b)^{2}} + \ldots \right.$$

$$\left. + \frac{pn}{1} * \frac{(kn*Q)^{2}}{(pn*U*b)^{2}} + \frac{Q}{1} * \frac{1}{U^{2}} \right)$$

$$= 2*\left(\frac{Q^{2}}{U^{2}} * \left((k1^{2}/p1 + k2^{2}/p2 + ... + kn^{2}/pn)/b^{2} \right) \right.$$

$$\left. + \frac{Q^{2}}{U^{2}} \right)$$

$$= 2*\left(\frac{Q^{2}*SUM2}{U^{2}} + \frac{Q}{U^{2}} \right)$$

$SUM2 = (k1^{2}/p1 + k2^{2}/p2 + ... + kn^{2}/pn)/b^{2}$

The queue length

$$q = A * \left(\frac{Q * SUM1}{U} + \frac{Q}{U} \right)$$

$$+ \frac{A^{2}*2*\left(\frac{Q^{2}*SUM2}{U^{2}} + \frac{Q}{U^{2}} \right)}{2*\left(1 - A * \left(\frac{Q*SUM1}{U} + \frac{Q}{U} \right) \right)}$$

```
           Q  * SUM1         Q
   =A *(------------ + ------
             U               U

                 Q**2*SUM2          Q
        A *(------------- +  --------- )
                  U**2             U**2
  + -------------------------------------------------)
                      Q *SUM1  +    Q
            1-A * (--------      ------  )
                       U             U
```

The expected wait time

```
         Q  * SUM1        Q
W  =A*(------------ + ------ )
            U             U

                Q**2*SUM2          Q
       A *( ------------- +  --------- ))/A
                 U**2           U**2
  + -------------------------------------------------
                    Q *SUM1  +    Q
           1-A * (--------      ------ )
                      U            U


         Q  * SUM1        Q
   =(------------ + ------ )
             U            U

                Q**2*SUM2          Q
       A *( ------------- +  --------- )
                 U**2              U**2
  + -------------------------------------------------
                    Q *SUM1  +    Q
           1-A * (--------      ------ )
                      U           U
```

$$= \left(\frac{Q * SUM1}{U} + \frac{Q}{U}\right) + \frac{A * \left(\frac{Q^{2} * SUM2}{U^{2}} + \frac{Q}{U^{2}}\right)}{1 - A\left(\frac{Q * SUM1}{U} + \frac{Q}{U}\right)}$$

where SUM1 =(k1+k2+...+kn)/b and

SUM2 =(k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2

2c. The Potential ddr Support in a Central Database

Chapter III, section F, presented the wait time for the potential ddr support method in a central environment. The details of the calculations are presented here.

The service time

$$X = \frac{P}{1} * \frac{S}{U*b} + \frac{Q}{1} * \frac{1}{U}$$

$$= \frac{P*S}{U*b} + \frac{Q}{U}$$

The second moment of the pdf

$$X2 = 2*\left(\frac{P}{1} * \frac{S^{2}}{(U*b)^{2}} + \frac{Q}{1} * \frac{1}{U^{2}}\right)$$

$$= 2*\left(\frac{P*S^{2}}{(U*b)^{2}} + \frac{Q}{U^{2}}\right)$$

The queue length

$$q = A*\left(\frac{P*S}{U*b} + \frac{Q}{U}\right)$$

$$+ \frac{A^{2}*2*\left(\frac{P*S^{2}}{(U*b)^{2}} + \frac{Q}{U^{2}}\right)}{2*\left(1 - A*\left(\frac{P*S}{U*b} + \frac{Q}{U}\right)\right)}$$

$$= A*\left(\frac{P*S}{U*b} + \frac{Q}{U}\right.$$

$$\left. + \frac{A*\left(\frac{P*S^{2}}{(U*b)^{2}} + \frac{Q}{U^{2}}\right)}{\left(1 - A*\left(\frac{P*S}{U*b} + \frac{Q}{U}\right)\right)}\right)$$

The wait time

$$W = A*\left(\frac{P*S}{U*b} + \frac{Q}{U} + \frac{A*\left(\frac{P*S**2}{(U*b)**2} + \frac{Q}{U**2}\right)}{\left(1- A*\left(\frac{P*S}{U*b} + \frac{Q}{U}\right)\right)}\right)/A$$

$$W = \frac{P*S}{U*b} + \frac{Q}{U} + \frac{A*\left(\frac{P*S**2}{(U*b)**2} + \frac{Q}{U**2}\right)}{1 - A*\left(\frac{P*S}{U*b} + \frac{Q}{U}\right)}$$

APPENDIX 3

THE MASTER/SLAVE DISTRIBUTIVE DATABASE CASE

This appendix shows details of the calculations for the service time, X, the second moment of the pdf, X2 and the queue length in a master/slave distributive database.

The M/Hr/1 queue requires that the sum of the probabilities of proceeding to each stage be equal to one. We must, therefore, normalize the probabilities with respect to the master node. This is because the master node is receiving all lock requests and, therefore, will be receiving n*Q requests. The normalizing factor is thus

D=Pm+Qm+L

where L, the lock requests is n*Q+Pw*Q and Pw is the probability of a conflict.

3a. The Deferred-Actual Method

Chapter IV, section A presented the wait time for the deferred-actual ddr support method in a master/slave system. The details of that calculation are presented here.

MASTER NODE

The service time:

$$X=\frac{pm1}{D}*\frac{km1*Qm}{pm1*U*b}+\frac{pm2}{D}*\frac{km2*Qm}{pm2*U*b}+\ldots$$

$$+\frac{pmk}{D}*\frac{kmn*Qm}{pmn*U*b}+\frac{Qm}{D}*\frac{1}{U}+\frac{L}{D}*\frac{1}{C}$$

$$=\frac{Qm}{D*U}(km1+km2+\ldots+kmn)/b+\frac{Qm}{D*U}+\frac{L}{D*C}$$

$$=\frac{Qm*SUM1m}{D*U}+\frac{Qm}{D*U}+\frac{L}{D*C}$$

where SUM1m=(km1+km2+...+kmn)/b

The second moment of the pdf

$$X2=2*\left(\frac{pm1}{D}*\frac{(km1*Qm)**2}{(pm1*U*b)**2}+\frac{pm2}{D}*\frac{(km2*Qm)**2}{(pm2*U*b)**2}+\ldots\right.$$

$$\left.+\frac{pmn}{D}*\frac{(kmn*Qm)**2}{(pmn*U*b)**2}+\frac{Qm}{D}*\frac{1}{U**2}+\frac{L}{D}*\frac{1}{C**2}\right)$$

$$=2*\left(\frac{Qm**2}{D*U**2}*((km1**2/pm1+km2**2/pm2+\ldots+kmn**2/pmn)/b**2)\right.$$

$$\left.+\frac{Qm**2}{D*U**2}+\frac{L}{D*C**2}\right)$$

$$= 2\cdot\left(\frac{Qm^{2}\cdot SUM2m}{D\cdot U^{2}} + \frac{Qm}{D\cdot U^{2}} + \frac{L}{D\cdot C^{2}}\right)$$

$SUM2m=(km1^{2}/pm1+km2^{2}/pm2+\ldots+kmn^{2}/pmn)/b^{2}$

The queue length

$$q = Am\cdot\left(\frac{Qm \cdot SUM1m}{D\cdot U} + \frac{Qm}{D\cdot U} + \frac{L}{D\cdot C}\right)$$

$$+ \frac{Am^{2}\cdot 2\cdot 2\cdot\left(\frac{Qm^{2}\cdot SUM2m}{D\cdot U^{2}} + \frac{Qm}{D\cdot U^{2}} + \frac{L}{D\cdot C^{2}}\right)}{2\cdot\left(1-Am\cdot\left(\frac{Qm\cdot SUM1m}{D\cdot U} + \frac{Qm}{D\cdot U} + \frac{L}{D\cdot C}\right)\right)}$$

$$= Am\cdot\left(\frac{Qm \cdot SUM1m}{D\cdot U} + \frac{Qm}{D\cdot U} + \frac{L}{D\cdot C}\right.$$

$$\left. + \frac{Am\cdot\left(\frac{Qm^{2}\cdot SUM2m}{D\cdot U^{2}} + \frac{Qm}{D\cdot U^{2}} + \frac{L}{D\cdot C^{2}}\right)}{1-Am\cdot\left(\frac{Qm\cdot SUM1m}{D\cdot U} + \frac{Qm}{D\cdot U} + \frac{L}{D\cdot C}\right)}\right)$$

The expected wait time

$$Wm = A*\left(\frac{Qm * SUM1m}{D*U} + \frac{Qm}{D*U} + \frac{L}{D*C}\right.$$

$$\left. + \frac{Am*\left(\frac{Qm^{2}*SUM2m}{D*U^{2}} + \frac{Qm}{D*U^{2}} + \frac{L}{D*C^{2}}\right)\Big)/A}{1-Am*\left(\frac{Qm*SUM1m}{D*U} + \frac{Qm}{D*U} + \frac{L}{D*C}\right)}\right.$$

$$= \left(\frac{Qm * SUM1m}{D*U} + \frac{Qm}{D*U} + \frac{L}{D*C}\right)$$

$$+ \frac{Am*\left(\frac{Qm^{2}*SUM2m}{D*U^{2}} + \frac{Qm}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{1-Am*\left(\frac{Qm*SUM1m}{D*U} + \frac{Qm}{D*U} + \frac{L}{D*C}\right)}$$

where $SUM1m=(km1+km2+\ldots+kmn)/b$ and

$SUM2m=(km1^{2}/pm1+km2^{2}/pm2+\ldots+kmn^{2}/pmn)/b^{2}$

SLAVE NODE

The service time:

$$X= \frac{ps1}{1} * \frac{ks1*Qs}{ps1*U*b} + \frac{ps2}{1} * \frac{ks2*Qs}{ps2*U*b} + \ldots$$

$$+ \frac{psk}{1} * \frac{ksn*Qs}{psn*U*b} + \frac{Qs}{1} * \frac{1}{U}$$

$$= \frac{Qs}{U} (ks1+ks2+...+ksn)/b + \frac{Qs}{U}$$

$$= \frac{Qs * SUM1s}{U} + \frac{Qs}{U}$$

where SUM1s=(ks1+ks2+...+ksn)/b

The second moment of the pdf

$$X2=2*(\frac{ps1}{1} * \frac{(ks1*Qs)**2}{(ps1*U*b)**2} + \frac{ps2}{1} * \frac{(ks2*Qs)**2}{(ps2*U*b)**2} + ...$$

$$+ \frac{psn}{1} * \frac{(ksn*Qs)**2}{(psn*U*b)**2} + \frac{Qs}{1} * \frac{1}{U**2})$$

$$=2*(\frac{Qs**2}{U**2}*((ks1**2/ps1+ks2**2/ps2+...+ksn**2/psn)/b**2)$$

$$+ \frac{Qs**2}{U**2})$$

$$= 2*(\frac{Qs**2*SUM2s}{U**2} + \frac{Qs}{U**2})$$

where SUM2s=(ks1**2/ps1+ks2**2/ps2+...+ksn**2/psn)/b**2

The queue length

$$q =As*(\frac{Qs * SUM1s}{U} + \frac{Qs}{U})$$

```
             Qs**2*SUM2s          Qs
 As**2*2*(------------- +  ---------)
                U**2          U**2
+ --------------------------------------------------
                Qs*SUM1s +    Qs
  2* (  1-As* (--------      ------ ))
                  U            U


          Qs  * SUM1s       Qs
  =As*(------------ + ------
           U              U

             Qs**2*SUM2s          Qs
      As*(------------- +  --------- )
              U**2           U**2
+ --------------------------------------------------)
                Qs*SUM1s +    Qs
        1-As* (--------      ------  )
                  U            U
```

The expected wait time

```
        Qs  * SUM1s       Qs
Ws  =A*(------------ + ------ )
           U              U

             Qs**2*SUM2s          Qs
      As*( ------------- +  --------- ))/A
              U**2           U**2
+ --------------------------------------------------
                Qs*SUM1s +    Qs
        1-As* (--------      ------ )
                  U            U
```

$$= \left(\frac{Qs * SUM1s}{U} + \frac{Qs}{U} \right) + \frac{As * \left(\frac{Qs^{2} * SUM2s}{U^{2}} + \frac{Qs}{U^{2}} \right)}{1 - As * \left(\frac{Qs * SUM1s}{U} + \frac{Qs}{U} \right)}$$

where SUM1s=(ks1+ks2+...+ksn)/b and

SUM2s=(ks1**2/ps1+ks2**2/ps2+...+ksn**2/psn)/b**2

$$Ws = \left(\frac{Qs * Sum1s}{U} + \frac{Qs}{U} \right) + \frac{As * \left(\frac{Qs^{2} * SUM2s}{U^{2}} + \frac{Qs}{U^{2}} \right)}{1 - As \left(\frac{Qs * SUM1s}{U} + \frac{Qs}{U} \right)}$$

where SUM1s=(ks1+ks2+...+ksn)/b and

SUM2s=(ks1**2/ps1+ks2**2/ps2+...+ksn**2/psn)/b**2

3b. The Immediate-Actual Method

Chapter IV, section A, presented the wait time for the immediate-actual ddr support method in a master/slave distributive database system. The details of that calculation follow.

MASTER NODE

The service time

$$X = \frac{Pm}{D} * \frac{Qm * n}{Pm * U} + \frac{Qm}{D} * \frac{1}{U} + \frac{L}{D} * \frac{1}{C}$$

$$= \frac{Qm * (n+1)}{D*U} + \frac{L}{D*C}$$

The second moment of the pdf

$$X2 = 2*(\frac{Pm}{D} * \frac{(Qm*n)**2}{(Pm*U)**2} + \frac{Qm}{D} * \frac{1}{U**2} + \frac{L}{D} * \frac{1}{C**2})$$

$$X2 = 2* (\frac{(Qm*n)**2}{D*Pm*U**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2})$$

The expected queue length

$$q = A*(\frac{Qm * (n+1)}{D*U} + \frac{L}{D*C})$$

$$+ \frac{Am**2*2 * (\frac{(Qm*n)**2}{D*Pm*U**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2})}{2*(1 - Am*(\frac{Qm*(n+1)}{D*U} + \frac{L}{D*C}))}$$

$$q=A*\left(\frac{Qm*(n+1)}{D*U}+\frac{L}{D*C}+\frac{Am*\left(\frac{(Qm*n)^{2}}{D*Pm*U^{2}}+\frac{Qm}{D*U^{2}}+\frac{L}{D*C^{2}}\right)}{1-Am*\left(\frac{Qm*(n+1)}{D*U}+\frac{L}{D*C}\right)}\right)$$

The expected wait time

$$Wm=Am*\left(\frac{Qm*(n+1)}{D*U}+\frac{L}{D*C}+\frac{Am*\left(\frac{(Qm*n)^{2}}{D*Pm*U^{2}}+\frac{Qm}{D*U^{2}}+\frac{L}{D*C^{2}}\right)}{1-Am*\left(\frac{Qm*(n+1)}{D*U}+\frac{L}{D*C}\right)}\right)/Am$$

$$=\frac{Qm*(n+1)}{D*U}+\frac{L}{D*C}+\frac{Am*\left(\frac{(Qm*n)^{2}}{D*Pm*U^{2}}+\frac{Qm}{D*U^{2}}+\frac{L}{D*C^{2}}\right)}{1-Am*\left(\frac{Qm*(n+1)}{D*U}+\frac{L}{D*C}\right)}$$

SLAVE NODE

The service time

$$X = \frac{Ps}{1} * \frac{Qs * n}{Ps * U} + \frac{Qs}{1} * \frac{1}{U}$$

$$= \frac{Qs * (n+1)}{U}$$

The second moment of the pdf

$$X2 = 2*\left(\frac{Ps}{1} * \frac{(Qs*n)**2}{(Ps*U)**2} + \frac{Qs}{1} * \frac{1}{U**2} + \frac{L}{1}\right)$$

$$X2 = 2*\left(\frac{(Qs*n)**2}{Ps*U**2} + \frac{Qs}{U**2}\right)$$

The expected queue length

$$q = A*\left(\frac{Qs * (n+1)}{U}\right)$$

$$+ \frac{As**2*2 * \left(\frac{(Qs*n)**2}{Ps*U**2} + \frac{Qs}{U**2}\right)}{2*\left(1 - As*\left(\frac{Qs*(n+1)}{U}\right)\right)}$$

```
      Qs * (n+1)
 =A*(----------
        U


              (Qs*n)**2       Qs
     As   * (--------- +   ------   )
               Ps*U**2       U**2
 +     ---------------------------------------)
                   Qs*(n+1)
       1 - As*(----------)
                    U
```

The expected wait time

```
        Qs * (n+1)
 Ws=As*(----------
           U


              (Qs*n)**2       Qs
       As * (--------- +   ------   )
               Ps*U**2       U**2
 +     ---------------------------------------)/As
                   Qs*(n+1)
       1 - As*(----------)
                    U



        Qs * (n+1)
 =     ----------
           U


              (Qs*n)**2       Qs
       As * (--------- +   ------ )
               Ps*U**2       U**2
 +     ---------------------------------------
                   Qs*(n+1)
       1 - As*(----------)
                    U
```

$$= \frac{Qs * (n+1)}{U}$$

$$+ \frac{As * \left(\frac{(Qs*n)**2}{Ps*U**2} + \frac{Qs}{U**2}\right)}{1 - As*\left(\frac{Qs*(n+1)}{U}\right)}$$

<u>3c. The Potential Method</u>

Chapter IV, section A, presented the wait time for the potential ddr support method in a master/slave distributive database environment. The details of that calculation follow.

MASTER NODE

The service time

$$X= \frac{Pm}{D} * \frac{S}{U*b} + \frac{Qm}{D} * \frac{1}{U} + \frac{L}{D} * \frac{1}{C}$$

$$= \frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}$$

The second moment of the pdf

$$X2= 2*(\frac{Pm}{D} * \frac{S**2}{(U*b)**2} + \frac{Qm}{D} * \frac{1}{U**2} + \frac{L}{D} * \frac{1}{C**2})$$

$$= 2 *(\frac{Pm*S**2}{D*(U*b)**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2})$$

The queue length

$$q= Am*(\frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C})$$

$$+ \frac{Am**2*2*(\frac{Pm*S**2}{D*(U*b)**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2})}{2*(1- \frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C})}$$

$$= Am*(\frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}$$

$$+ \frac{Am*(\frac{Pm*S**2}{D*(U*b)**2} + \frac{Qm}{D*U**2} + \frac{L}{D*C**2})}{(1- \frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C})})$$

The wait time

$$Wm = Am*\left(\frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}\right.$$

$$\left. + \frac{Am*\left(\frac{Pm*S^{2}}{D*(U*b)^{2}} + \frac{Qm}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{\left(1- \frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}\right)}\right)/Am$$

$$Wm = \frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}$$

$$+ \frac{Am*\left(\frac{Pm*S^{2}}{D*(U*b)^{2}} + \frac{Qm}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{1 - Am*\left(\frac{Pm*S}{D*U*b} + \frac{Qm}{D*U} + \frac{L}{D*C}\right)}$$

SLAVE NODE

The service time

$$X = \frac{Ps}{1} * \frac{S}{U*b} + \frac{Qs}{1} * \frac{1}{U}$$

$$= \frac{Ps*S}{U*b} + \frac{Qs}{U}$$

The second moment of the pdf

$$X2 = 2*\left(\frac{Ps}{1} * \frac{S^{2}}{(U*b)^{2}} + \frac{Qs}{1} * \frac{1}{U^{2}}\right)$$

$$= 2*\left(\frac{Ps*S^{2}}{(U*b)^{2}} + \frac{Qs}{U^{2}}\right)$$

The queue length

$$q = As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right)$$

$$+ \frac{As^{2}*2*\left(\frac{Ps*S^{2}}{(U*b)^{2}} + \frac{Qs}{U^{2}}\right)}{2*\left(1- As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right)\right)}$$

$$= As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right.$$

$$\left. + \frac{As*\left(\frac{Ps*S^{2}}{(U*b)^{2}} + \frac{Qs}{U^{2}}\right)}{\left(1- As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right)\right)}\right)$$

The wait time

$$Ws = As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U} + \frac{As*\left(\frac{Ps*S**2}{(U*b)**2} + \frac{Qs}{U**2}\right)}{\left(1 - As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right)\right)}\right)/As$$

$$Ws = \frac{Ps*S}{U*b} + \frac{Qs}{U} + \frac{As*\left(\frac{Ps*S**2}{(U*b)**2} + \frac{Qs}{U**2}\right)}{1 - As*\left(\frac{Ps*S}{U*b} + \frac{Qs}{U}\right)}$$

APPENDIX 4

THE DISTRIBUTED CONCURRENCY CONTROL CASE

This appendix shows details of the calculations for the service time, second moment of the pdf and queue length for the distributed concurrency control database.

The normalizing factor for the distributed control case involves locks for only those files local to the node, since the distributed control handles locking at each node, and the conflicting update requests. Therefore, the normalizing factor is

$$D= P+Q+L$$

where L is Q+Pw*Q.

4a. The Deferred-Actual Method

Chapter IV, section B presented the wait time for the deferred-actual ddr support method in a distributed concurrency control system. The details of that calculation is presented here.

The service time:

```
    p1        k1*Q          p2          k2*Q
X= ---- * --------- + ------- * --------- + ...
    D       p1*U*b        D          p2*U*b


     pk        kn*Q         Q        1        L        1
  + ------- * -------  + ----- * ----- + ----- * -----
     D        pn*U*b        D        U        D        C

      Q                                  Q          L
  = -------- (k1+k2+...+kn)/b + -------- + --------
     D*U                                D*U        D*C

       Q * SUM1          Q          L
  =   ----------- + ------ + ------
         D*U            D*U        D*C

  where SUM1=(k1+k2+...+kn)/b
```

The second moment of the pdf

```
          p1     (k1*Q)**2        p2            (k2*Q)**2
X2=2*( ---- * ----------- + ------- * ----------- + ...
          D     (p1*U*b)**2        D          (p2*U*b)**2

      pn      (kn*Q)**2       Q        1         L        1
  + ----- * ----------  + ----- * ----- + ----- * -----)
      D     (pn*U*b)**2       D       U**2       D      C**2


        Q**2
 =2*( ------*((k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2)
      D*U**2

          Q**2        L
      +  ------- + ------)
         D*U**2    D*C**2

         Q**2*SUM2            Q              L
  = 2*( ------------- + --------- + --------)
           D*U**2            D*U**2        D*C**2
```

where $SUM2 = (k1^{2}/p1 + k2^{2}/p2 + \ldots + kn^{2}/pn)/b^{2}$

The queue length

$$q = A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right)$$

$$+ \frac{A^{2} * 2 * \left(\frac{Q^{2} * SUM2}{D*U^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}} \right)}{2 * \left(1 - A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right) \right)}$$

$$= A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right.$$

$$\left. + \frac{A * \left(\frac{Q^{2} * SUM2}{D*U^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}} \right)}{1 - A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right)} \right)$$

The expected wait time

$$W = A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right.$$

$$\left. + \frac{A * \left(\frac{Q^{2} * SUM2}{D*U^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}} \right)}{1 - A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right)} \right) / A$$

$$= \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right)$$

$$+ \frac{A * \left(\frac{Q^{2} * SUM2}{D*U^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}} \right)}{1 - A * \left(\frac{Q * SUM1}{D*U} + \frac{Q}{D*U} + \frac{L}{D*C} \right)}$$

where $SUM1 = (k1+k2+...+kn)/b$ and

$SUM2 = (k1^{2}/p1+k2^{2}/p2+...+kn^{2}/pn)/b^{2}$

4b. The Immediate-Actual Method

Chapter IV, section B, presented the wait time for the immediate-actual ddr support method in a distributive concurrency database system. The details of that calculation follow.

The service time

$$X = \frac{P}{D} * \frac{Q * n}{P * U} + \frac{Q}{D} * \frac{1}{U} + \frac{L}{D} * \frac{1}{C}$$

$$= \frac{Q * (n+1)}{D*U} + \frac{L}{D*C}$$

The second moment of the pdf

$$X2=2*(\frac{P}{D} * \frac{(Q *n)**2}{(P *U)**2} + \frac{Q}{D} * \frac{1}{U**2} + \frac{L}{D} * \frac{1}{C**2})$$

$$X2=2* (\frac{(Q *n)**2}{D*P*U**2} + \frac{Q}{D*U**2} + \frac{L}{D*C**2})$$

The expected queue length

$$q=A*(\frac{Q * (n+1)}{D*U} + \frac{L}{D*C})$$

$$+ \frac{A**2*2 * (\frac{(Q*n)**2}{D*P*U**2} + \frac{Q}{D*U**2} + \frac{L}{D*C**2})}{2*(1 - A *(\frac{Q *(n+1)}{D*U} + \frac{L}{D*C}))}$$

$$q=A*(\frac{Q * (n+1)}{D*U} + \frac{L}{D*C}$$

$$+ \frac{A * (\frac{(Q*n)**2}{D*P*U**2} + \frac{Q}{D*U**2} + \frac{L}{D*C**2})}{1 - A *(\frac{Q*(n+1)}{D*U} + \frac{L}{D*C})})$$

The expected wait time

$$W = A * \left(\frac{Q * (n+1)}{D*U} + \frac{L}{D*C} + \frac{A * \left(\frac{(Q*n)^{2}}{D*P*U^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}} \right)}{1 - A * \left(\frac{Q*(n+1)}{D*U} + \frac{L}{D*C} \right)} \right) / A$$

$$= \frac{Q * (n+1)}{D*U} + \frac{L}{D*C} + \frac{A * \left(\frac{(Q*n)^{2}}{D*P*U^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}} \right)}{1 - A * \left(\frac{Q*(n+1)}{D*U} + \frac{L}{D*C} \right)}$$

4c. The Potential Method

Chapter IV, section A, presented the wait time for the potential ddr support method in a distributive concurrency database environment. The details of that calculation follow.

The service time

$$X= \frac{P}{D} * \frac{S}{U*b} + \frac{Q}{D} * \frac{1}{U} + \frac{L}{D} * \frac{1}{C}$$

$$= \frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}$$

The second moment of the pdf

$$X2= 2*\left(\frac{P}{D} * \frac{S^{2}}{(U*b)^{2}} + \frac{Q}{D} * \frac{1}{U^{2}} + \frac{L}{D} * \frac{1}{C^{2}}\right)$$

$$= 2*\left(\frac{P*S^{2}}{D*(U*b)^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}}\right)$$

The queue length

$$q= A*\left(\frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right)$$

$$+ \frac{A^{2}*2*\left(\frac{P*S^{2}}{D*(U*b)^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{2*\left(1- \frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right)}$$

$$= A*\left(\frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right.$$

$$\left. + \frac{A*\left(\frac{P*S^{2}}{D*(U*b)^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{\left(1- \frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right)}\right)$$

The wait time

$$W = A*\left(\frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right.$$

$$\left. + \frac{A*\left(\frac{P*S^{2}}{D*(U*b)^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{\left(1- \frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right)}\right)/A$$

$$= \frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}$$

$$+ \frac{A*\left(\frac{P*S^{2}}{D*(U*b)^{2}} + \frac{Q}{D*U^{2}} + \frac{L}{D*C^{2}}\right)}{1 - A*\left(\frac{P*S}{D*U*b} + \frac{Q}{D*U} + \frac{L}{D*C}\right)}$$

APPENDIX 5

THE SYNCHRONIZED CONCURRENCY CONTROL ENVIRONMENT

This appendix will present details of the analysis presented in Chapter IV, section C, regarding the synchronized concurrency control method in a distributive database.

5a. The Deferred-Actual Method

Chapter IV, section C presented the wait time for the deferred-actual ddr support method in a synchronized concurrency control system. The details of that calculation are presented here.

The service time

$$X = \frac{2}{D*N*T} * R + \frac{p1*A}{D} * \frac{Q*k1}{p1*U*b} + \frac{p2*A}{D} * \frac{Q*k2}{p2*U*b} + \ldots$$

$$+ \frac{pn*A}{D} * \frac{Q*kn}{pn*U*b}$$

$$= \frac{2*R}{D*N*T} + \frac{A*Q}{D*U} * (k1+k2+\ldots+kn)/b$$

$$= \frac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U}$$

where $SUM1=(k1+k2+...+kn)/b$

The second moment of the pdf

$$X2=2*(\frac{2}{D*N*T}*R**2 + \frac{p1*A}{D}*(\frac{Q*k1}{p1*U*b})**2 + \frac{p2*A}{D}*(\frac{Q*k2}{p2*U*b})**2$$

$$...\quad + \frac{pn*A}{D}*(\frac{Q*kn}{pn*U*b})**2)$$

$$=2*(\frac{2*R**2}{D*N*T} + \frac{A*Q**2}{D*U**2}*(k1**2/p1+k2**2/p2+...+kn**2/p2)/b**2))$$

$$= \frac{2*(2*(R*U)**2 + N*T*A*SUM2*Q**2)}{D*N*T*U**2}$$

where $SUM2=(k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2$.

The queue length

$$q = A* \frac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U}$$

$$+ \frac{A**2*\frac{2*(2*(R*U)**2 + N*T*A*SUM2*Q**2)}{D*N*T*U**2}}{2*(1 - A * (\frac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U}))}$$

```
        2*R*U + N*T*A*Q*SUM1
  =A*( --------------------
            D*N*T*U

         A*(2*(R*U)**2 + N*T*A*SUM2*Q**2)
         --------------------------------
              D*N*T*U**2
    +    ---------------------------------------)
                  2*R*U + N*T*A*Q*SUM1
      1 - A * (   --------------------  )
                       D*N*T*U
```

The expected wait time

```
        2*R*U + N*T*A*Q*SUM1
 W =A*( --------------------
            D*N*T*U

         A*(2*(R*U)**2 + N*T*A*SUM2*Q**2)
         --------------------------------
              D*N*T*U**2
   +    ---------------------------------------)/A
                  2*R*U + N*T*A*Q*SUM1
      1 - A * (   --------------------  )
                       D*N*T*U

        2*R*U + N*T*A*Q*SUM1
  =     --------------------
            D*N*T*U

         A*(2*(R*U)**2 + N*T*A*SUM2*Q**2)
         --------------------------------
              D*N*T*U**2
   +    ---------------------------------------
                  2*R*U + N*T*A*Q*SUM1
      1 - A * (   --------------------  )
                       D*N*T*U
```

where SUM1=(k1+k2+...+kn)/b and

SUM2=(k1**2/p1+k2**2/p2+...+kn**2/pn)/b**2.

5b. The Immediate-Actual ddr Support Method

Chapter 4, section C, presented the expected wait time for the immediate actual ddr support method in a distributed concurrency control system. The details of that calculation are presented here.

The service rate

$$X = \frac{2}{D*N*T} * R + \frac{P*A}{D} * \frac{n*Q}{P*U}$$

$$= \frac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U}$$

The second moment of the pdf

$$X2 = 2*\left(\frac{2}{D*N*T} * R**2 + \frac{P*A}{D} * \left(\frac{n*Q}{P*U}\right)**2\right)$$

$$= \frac{2*(2*P*(R*U)**2 + N*T*A*(Q*n)**2)}{D*N*T*P*U**2}$$

The queue length

$$q = A* \frac{2*R*U + N*T*A*Q*SUM1}{D*N*T*U}$$

$$+ \frac{A**2*2*\dfrac{(2*P*(R*U)**2 + N*T*A*(Q*n)**2)}{D*N*T*P*U**2}}{2*\left(1 - A * \left(\dfrac{2*R*U + N*T*A*Q*n}{D*N*T*U}\right)\right)}$$

```
        2*R*U + N*T*A*Q*SUM1
  =A* (--------------------
              D*N*T*U

            A*(2*P*(R*U)**2 + N*T*A*(Q*n)**2)
            ---------------------------------
                  D*N*T*P*U**2
   +    ----------------------------------------- )
                   2*R*U + N*T*A*Q*n
     ( 1 - A * (  ---------------------  ))
                       D*N*T*U
```

The wait time

```
        2*R*U + N*T*A*Q*SUM1
 W=A* (--------------------
              D*N*T*U

            A*(2*P*(R*U)**2 + N*T*A*(Q*n)**2)
            ---------------------------------
                  D*N*T*P*U**2
   +    ----------------------------------------- )/A
                   2*R*U + N*T*A*Q*n
     ( 1 - A * (  ---------------------  ))
                       D*N*T*U


        2*R*U + N*T*A*Q*SUM1
  =    --------------------
              D*N*T*U

            A*(2*P*(R*U)**2 + N*T*A*(Q*n)**2)
            ---------------------------------
                  D*N*T*P*U**2
   +    ----------------------------------------- )
                   2*R*U + N*T*A*Q*n
     ( 1 - A * (  ---------------------  ))
                       D*N*T*U
```

5c. The Potential Method

Chapter IV, section C, presented the wait time for the potential ddr support method in a synchronized concurrency control system. The details of that calculation are presented here.

The service time

$$X = \frac{2}{D*N*T}*R + \frac{P*A}{D} * \frac{S}{U*b}$$

$$= \frac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}$$

The second moment of the pdf

$$X2 = 2*\left(\frac{2}{D*N*T} * R**2 + \frac{P*A}{D} * \left(\frac{S}{U*b}\right)**2 \right)$$

$$= 2*\left(\frac{A*(2*(R*U*b)**2 + N*T*A*P*S**2)}{D*N*T*(U*b)**2}\right)$$

The queue length

$$q = A* \frac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}$$

$$+ \frac{\dfrac{A**2*2*(2*(R*U*b)**2 + N*T*A*P*S**2)}{D*N*T*(U*b)**2}}{2*(\ 1 - A * (\ \dfrac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}\ ))}$$

$$= A*(\frac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}$$

$$+ \frac{\dfrac{A*(2*(R*U*b)**2 + N*T*A*P*S**2)}{D*N*T*(U*b)**2}}{(\ 1 - A * (\ \dfrac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}\ ))}\ )$$

The wait time

$$= A*(\frac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}$$

$$+ \frac{\dfrac{A*(2*(R*U*b)**2 + N*T*A*P*S**2)}{D*N*T*(U*b)**2}}{(\ 1 - A * (\ \dfrac{2*R*U*b + N*T*A*P*S}{D*N*T*U*b}\ ))}\ )/A$$

```
          2*R*U*b + N*T*A*P*S
 =      ----------------------
             D*N*T*U*b

             A*(2*(R*U*b)**2 + N*T*A*P*S**2)
             ---------------------------------
                   D*N*T*(U*b)**2
  +     ----------------------------------------
                      2*R*U*b + N*T*A*P*S
        1 - A * (    ----------------------  )
                          D*N*T*U*b
```

LIST OF REFERENCES

Astrahan, M. M.; Blasgen, W. W.; Chamberlin, D. D.; Eswaran, K.P.; Gray, J. N.; Griffiths, P.; King, W. F.; Lorie, R. J.; McJones, P. R.; Melh, J. W.; Putzola, G. R.; Traiger, L.; Wade, B. W.; and Watson, V. "System R: Relational Approach to Database Management." ACM Transactions on Database Systems 1 (June 1976): 97-137.

Bernstein, P.; Shipman, D.; and Wong, W. "Formal Aspects of Serializability in Database Concurrency Control." IEEE Transactions on Software Engineering 5 (May 1979): 203-215.

Bernstein, P.; Shipman, D.; and Rothnie, J. "Concurrency Control in a System for Distributive Databases (SDD-1)." ACM Transactions on Database Systems 5 (March 1980): 18-51.

Blasgen, M.W., and Eswaran, K. P. "Storage and Access in Relational Data Bases." IBM Systems Journal 6 (July 1977): 203-215.

Cardenas, A.F. "Evaluation and Selection of File Organization -- A Model and System." Communications of the ACM 16 (September 1973): 540-548.

Chamberlin, D.D.; Gray; J. N.; and Traiger, I. L. "Views, Authorization and Locking in a Relational Database System." In AFIPS Proceedings-- National Computer Conference, 1975. Vol. 44, pp. 425-430.

Chen, P.S. "Optimal File Allocation in Multi-level Storage Systems." In AFIPS Proceedings-- National Computer Conference, 1973. Vol. 42, pp. 277-282.

Chu, W.W. "Performance of File Directory Systems for Data Bases in Star and Distributed Networks." AFIPS Conference Proceedings--National Computer Conference, 1976, Vol. 45, pp. 577-587.

CODASYL, Data Base Task Group Report, April 1971. New York: ACM, 1975.

Codd, E.F. "A Relational Model of Data for Large Shared Data Banks." Communications of the ACM 13 (June 1970): 377-387.

Codd, E.F. "Extending the Relational Database Model to Capture More Meaning." ACM Transactions on Database Systems 4 (December 1979):397-434.

Coffman, E.G.; Gelenbe, E.; and Plateau; B. "Optimization of the Number of Copies in a Distributed Data Base." IEEE Transactions on Software Engineering, 7 (January 1981): 74-84.

Collmeyer, A.J., and Shemer, J. E. "Analysis of Retrieval Performance for Selected File Organization Techniques." AFIPS Conference Proceedings-- National Computer Conference, 1970. Vol 37, pp. 207-210.

Date, C.J. An Introduction to Database Systems. 3rd ed. Reading, Mass.: Addison-Wesley, 1979.

Denoia, L.A. Performance and Timeliness in a Database (Ph.D. dissertation, Stanford University, 1979). NTIS AD-A075268. Springfield, VA: National Technical Information Service, 1979.

Dutton, B. A., and Kinsley, K. C. "RAQUEL II-- A Relational Query Language." In Proceedings of the 16th Annual Southeast Regional ACM Conference, Atlanta, November 1977, pp. 327-333. New York: ACM, 1977.

Ellis, C. "Consistency and Correctness of Duplicate Data Base Systems." In Proceedings of the Sixth Symposium on Operating Systems Principles, Lafayette, IN, November 1977 pp. 67-84. New York: ACM, 1977.

Enslow, P. "What is a Distributed Data Processing System." Computer (January 1978): 13-21.

Eswaran, K.P., and Chamberlin, D. D. "Functional Specification of a Subsystem for Database Integrity." In Very Large Data Bases (First International Conference, September 1975), pp. 624-633. Edited by Douglas S. Kerr. New York: ACM, 1975.

Garcia-Molina, H. Performance of Update Algorithms for Replicated Data in a Distributive Database (Ph. D. dissertation, Stanford University, 1979). NUSC Technical Report No. 6099. New London, CT: Naval Underwater Systems Center, 1979.

Gardarin, G. and Chu, W. W."A Reliable Distributed Control Algorithm for Updating Replicated Databases." In Sixth Data Communications Symposium, Pacific Grove, CA, Nov. 27-19, 1979, pp. 42-51. New York: IEEE Press, 1979.

Gelenbe, E., and Sevcik, K. "Analysis of Update Synchronization for Multiple Copy Data-bases." In Proceedings of the Third Berkeley Workshop on Distributed Data Management and Computer Networks, University of California, August 1978, pp. 69-90. NTIS LBL-7953 (also published in IEEE Transactions on Computing 28 (October 1979): 737-746).

Ghosh, S.P., and Tuel, W. G. "A Design of an Experiment to Model Data Base System Performance." IEEE Transactions on Software Engineering 2 (June 1976): 97-106.

Gotlieb, L.R. "Computing Joins of Relations." In ACM-SIGMOD International Conference on Management of Data, San Jose, CA, May 1975, pp. 100-110. New York: ACM, 1975.

Hall, P. "Optimization of a Single Relational Expression in a Relatioal Database System." IBM Journal of Research and Development 20 {June 1975):244-257.

Hammer, M., and Shipman, D. "Reliability Mechanisms for SDD-1." ACM Transactions on Database Systems 5 (December 1980): 431-466.

Hitchcock, P. "User Extensions to the Peterlee Relational Test Vehicle" In Systems for Large Data Bases (2nd International Conference on Very Large Data Bases, September, 1976), pp. 169-180. Edited by P. C. Lockemann and E. J. Neuhold. New York: North-Holland Publishing, 1977.

Kim, W. "Relational Database Systems." ACM Computing Surveys 11 (September 1979):185-210.

Kinsley, K., and Driscoll, J. "Dynamic Derived Relations Within the RAQUEL II DBMS" In Proceedings of the ACM Annual Conference, Detroit, October 1979, pp. 69-80. New York: ACM, 1979.

Kinsley, K., and Driscoll, J. "Efficiently Maintaining Dynamic Derived Relations in Actual Form." Technical Report CS-TR-48, Department of Computer Science, University of Central Florida, Orlando, Florida, October 1980.

Kleinrock, L. Queueing Systems, Volume 1: Theory. New York: John Wiley and Sons, 1975.

Koenig, S. and Paige, R. "A Transformational Framework for the Automatic Control of Derived Data." In Very Large Data Bases (Seventh International Conference, September 1981), pp. 306-318. New York: IEEE, 1981.

Lavenberg, S.S., and Shedler, G.S. "Stochastic Modeling of Processor Scheduling with Application to Data Base Management Systems." IBM Journal of Research and Development (September 1976): 437-448.

LeLann, G. "Distrubutive Systems -- Towards a Formal Approach." In Information Processing '77 (Proceedings of IFIP Congress '77, Toronto, Canada, August 1977) pp. 259-171. Edited by Bruce Gilchrist. New York: Elsevier North-Holland Publishing, 1977.

Lowe, T.C. "The Influence of Data Base Characteristics and Usage on Direct Access File Organization." Journal of ACM 15 (October 1978): 535-548.

Mahmoud, S., and Riordon, J. S. "Optimal Allocation of Resources in Distributed Information Networks" ACM Transactions on Database Systems 1 (March 1976): 66-78.

MacDonald, N. and Stonebraker, M. "CUPID: The Friendly Query Language." In Proceedings of the ACM Pacific Regional Conference, San Francisco, April 1975, pp. 127-131. New York: ACM, 1975.

Menasce, D., and Muntz, R. "Locking and Deadlock Detection in Distributed Data Bases." IEEE Transactions on Software Engineering 5 (May 1979): 195-202.

Menon, M.J., and Hsiao, D.K. "Design and AnalySis of a Relational Join Operation for VLSI." In Very Large Data Bases (Seventh International Conference, September 1981), pp. 203-211. New York: IEEE Press, 1981.

Rosenkrantz, D.; Stearns, R. ;and Lewis, P. "System Level Concurrency Control for Distributive Database Systems." ACM Transactions on Database Systems 3 (June 1978): 178-198.

Rosenkrantz, P., and Hunt, H. "Processing Conjunctive Predicates and Queries." Technical Report, Department of Computer Science, State University of New York at Albany, June 1980.

Rothnie, J. B. "Evaluating Inter-entry Retrieval Expressions in a Relational Data Base Management System." In AFIPS Conference Proceedings- National Computer Conference, 1975. Vol. 44, pp. 417-424.

Severance, D.G. "Differential Files: Their Application to the Maintenance of Large Databases." ACM Transactions on Database Systems 1 (September 1976):256-267.

Siler, K.F. "A Stochastic Evaluation Model for Database Organizations in Data Retrieval Systems."Communications of the ACM 19 (February 1976): 84-95.

Stearns, R., and Rosenkrantz, D. "Distributed Database Concurrency Controls Using Before-Values." Technical Report, State University of New York at Albany, February 1981.

Stonebraker, M. "Implementation of Integrity Constraints and Views-- Query Modification." In ACM-SIGMOD International Conference on Management of Data, San Jose, CA, May 1975, pp. 65-78. New York: ACM, 1975.

Stonebraker, M.; Wong, E.; Kreps, P.; and Held, G. "The Design and Implementation of INGRES." ACM Transactions on Database Systems 1 (September 1976):189-122.

Stonebraker, M. "Concurrency Control and Consistency of Multiple Copies of Data in Distributive INGRES." IEEE Transactions on Software Engineering, 5 (May 1979): 203-215.

Teory, T. J., and Oberlander, L. B. "Network Database Evaluation Using Analytical Modeling." Technical Report 78-7, College of Engineering, University of Michigan, Ann Arbor, Michigan, 1978.

Thomas, R. "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases." _ACM Transactions on Database Systems_ 3 (June 1979): 180-209.

Todd, S. "The Peterlee Test Vehicle - A System Overview." _IBM Systems Journal_ 15 (May 1976):285-308.

Wiederhold, G. _Database Design_. New York: McGraw-Hill, 1977.

Yao, S. B. "Approximating Block Accesses in Database Organizations." _Communications of the ACM_ 20 (April 1977): 260-261.

Yao, S. B. "An Attribute Based Model for Database Access Cost Analysis." _ACM Transactions on Database Systems_ 2 (March 1977): 45-67.