A NOTE ON PERFORMANCE OF VM/370 IN THE INTEGRATION OF MODELS AND DATABASES

John J. Donovan

Energy Laboratory in Association with
the Alfred P. Sloan School of Management

Working Paper No. MIT-EL-76-017WP

June 1976

OUTLINE

Abstract

1. Introduction

2. Overview of System Architecture

3. Experimental Analysis of Overhead Costs

4. Analysis of Response Time Degradation Due to Locking

5. Implication of Theoretical Results

6. Conclusion

Acknowledgment

References

ABSTRACT

As the proliferation of programming systems and database systems continues and, correspondingly, as the need for integrating these systems for certain applications increases, VM/370 offers a mechanism for such integration. This paper analyzes the performance of a configuration of virtual machines using VM/370 that allows for the sharing of a database system among several incompatible programs in an interactive environment.

Specifically, two aspects of performance are addressed--an experimental study of the overhead cost incurred in the interface mechanisms employed, and a theoretical study of the degradation of response time due to the locking mechanisms employed. The conclusion of the experimental observations is that for sophisticated, complex accesses to the database system, the overhead costs are relatively small. The result of the theoretical study is the quantification of that degradation as a function of speeds of the database machine and the rate with which queries are made. The discussion of the practical implications of this theoretical study presents ways to improve this degradation. The observed conclusion of this work is our feeling that, for certain application areas, the benefits resulting from increased effectiveness of users outweigh the costs incurred.

1. INTRODUCTION

This paper discusses two aspects of performance of a configuration of virtual machines using VM/370 [IBM, 1972]. Such a configuration as will be analyzed facilitates the sharing of data between several seemingly incompatible programs in an interactive system. By "seemingly incompatible" we mean that each of these programs, or databases, may be running simultaneously under different IBM/360 or 370 operating systems.

The two aspects of performance that are analyzed are (1) an experimental study, which makes explicit the overhead incurred in interfacing and communicating between virtual machines; and (2) a theoretical study of the degradation in response time due to locking strategy, used to permit multiple users access to the same database system.

It has also been suggested by others [Bagley et al., 1976] that virtual machines (in particular, VM/370) can be interconnected. We have extended this concept to the development of several operational decision support systems [Donovan and Keating, 1975; M.I.T., 1975]. These systems allow users interactive access to standard analytical facilities (e.g., PL/I, APL, etc.), modeling facilities (e.g., TROLL [NBER, 1974], TSP [Hall, 1975], EPLAN [Schober, 1975], etc.), and database facilities (e.g., SEQUEL [Chamberlain, 1975], IMS, etc.). Some of these facilities were formerly thought to be incompatible, in that they may have required a special operating system or single machine.

VM/370 provides a mechanism for all these systems to be integrated. Essentially, VM/370 accomplishes this by simulating several 370 computers on one machine and hence allowing each of these facilities to run in its own simulated environment.

The VM/370 concept, however, was based on "isolation," that is, each virtual machine was to be unaware of other virtual machines. Our applications demand communication among these machines. Several mechanisms for facilitating the transfer of data between independent virtual machines are reported in the literature [Hsieh, 1974; Parmelee et al., 1972; Donovan and Jacoby, 1975].

2. OVERVIEW OF SYSTEM ARCHITECTURE

M.I.T.'s Center for Information Systems Research, the M.I.T. Energy Laboratory, and the IBM Cambridge Scientific Center have developed a system of interconnected virtual machines, called GMIS (Generalized Management Information System) [Donovan and Jacoby, 1975]. It is not the purpose of this paper to describe GMIS; rather, we use GMIS here as an operational illustrative example of these concepts and as an environment to study performance issues.

GMIS is implemented on an IBM System/370 computer using VM/370. It uses the virtual machine (VM) concept extensively [Parmelee, 1972; Buzen et al., 1973; Goldberg, 1974]. A virtual machine may be defined as a replica of a real computer system simulated by a combination of a virtual machine monitor (VMM) software program and appropriate hardware support. For example, the VM/370 system enables a single IBM System/370 to appear functionally as though it were multiple independent System/370's (i.e., multiple "virtual machines"). Thus, a VMM can make one computer system function as though it were multiple, physically isolated, systems.

A configuration of virtual machines used in GMIS is depicted in Figure 1, where each box denotes a separate virtual machine. Those virtual

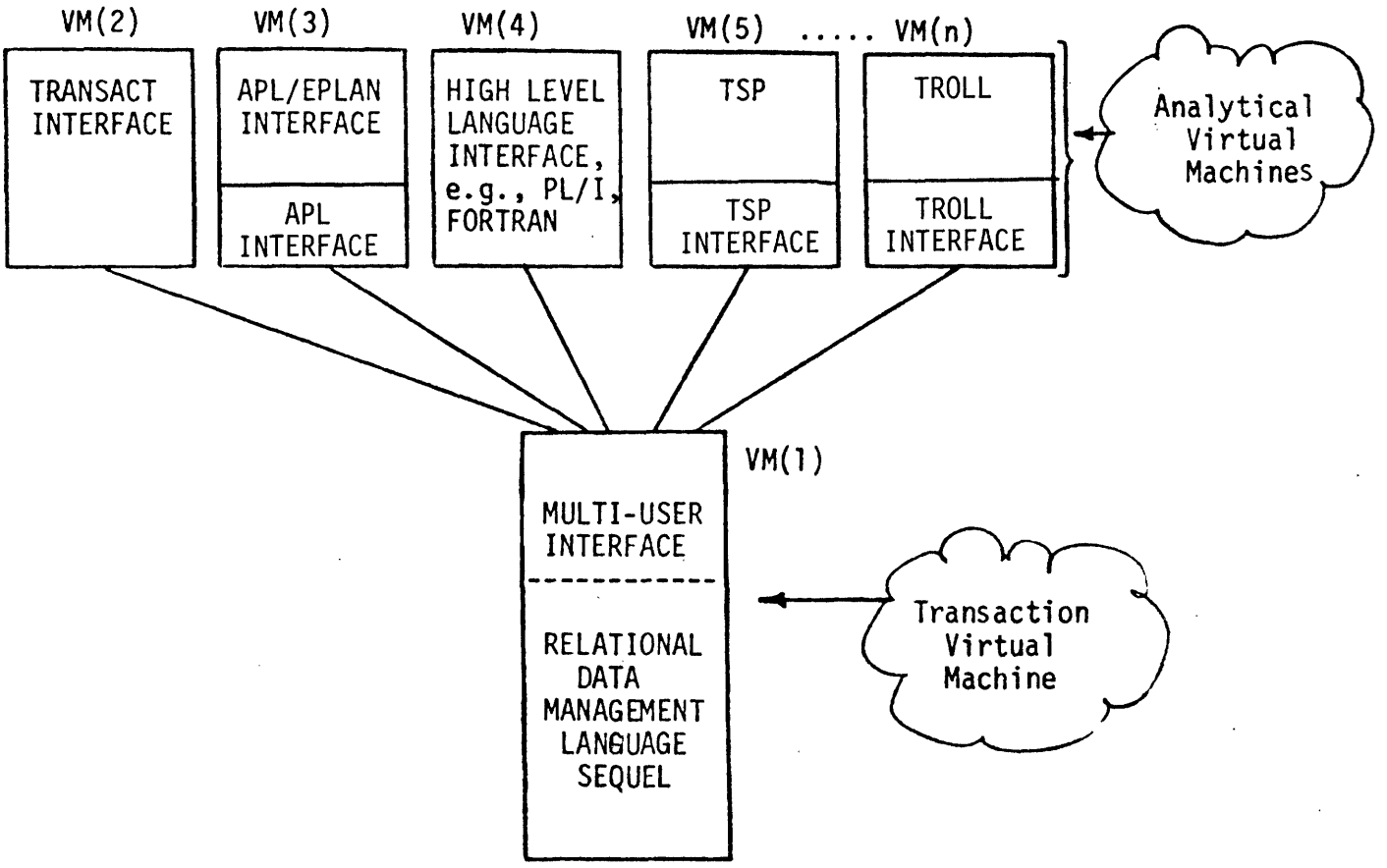

Figure 1: Overview of the Software Architecture of GMIS

machines across the top of the figure are executing programs that interact with the user, whether they are analytical facilities, existing models, or database systems. All these programs can access data managed by the general data management facility running on the virtual machine depicted in the center of Figure 1. A sample use of this architecture might proceed as follows: A user activates a model, say in the APL/EPLAN machine. That model requests data from the general database machine (called the Transaction Virtual Machine, or TVM), which responds by passing back the requested data. That data can then be manipulated in the APL machine.

Figure 2 depicts such a user console session, in which the user is logged into an APL machine. The statement 'QUERY' is an APL function within the APL interface that passes the SEQUEL statement 'SELECT PRICE, CONSUMPTION FROM ENERGY WHERE WINTER IN(73,74,75);' to the SEQUEL database machine. That SEQUEL machine assembles the specified data and passes it back to the APL machine as the vectors PRICE and CONSUMPTION. The remaining statements are APL and EPLAN statements that perform certain analytical functions on the data. Specifically, the PLOT function is an EPLAN function which produced the plot. The statement ∇ ELASTICITY is an APL statement that allows the user to define a function ELASTICITY. The following seven statements are user-inputted APL statements. The statement ELASTICITY causes the execution of the user function. (As an aside, this session was used to compute the price elasticity of residential heating fuel oil in New England.)

Note that all the analytical facilities and database facilities may be incompatible with each other, in that they may run under different operating systems. Hence users are not required to learn a new analytical capability in order to gain access to the data. Each computer may run

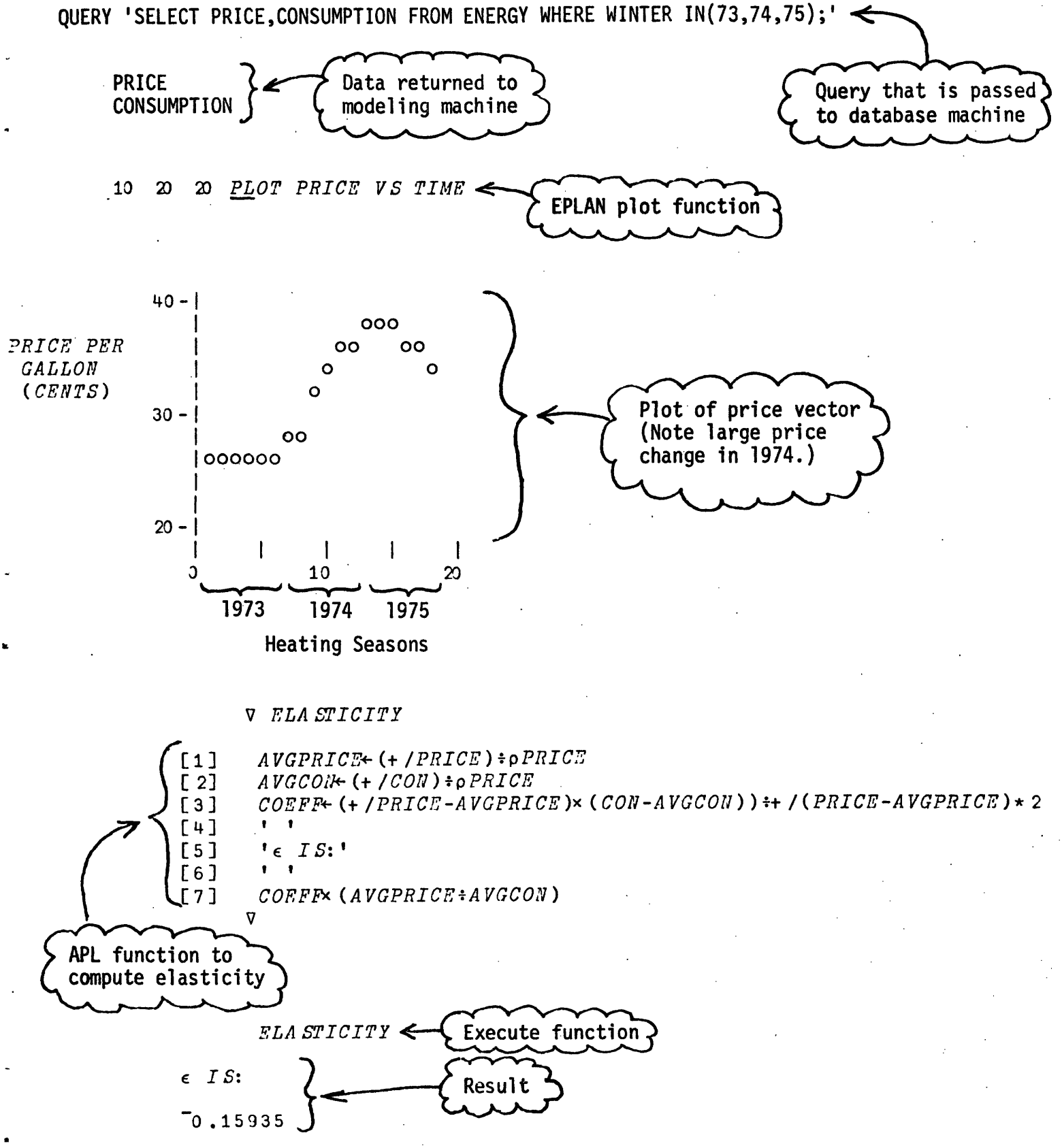

Figure 2: Sample Session of APL/EPLAN Machine Connected to SEQUEL Machine

any existing model or program with no transfer costs. Such a configuration also eliminates the need to devote resources to transporting application languages and programs between operating systems and permits interaction between application languages and programs not originally envisioned by their developers. For example, an analytical package has its data management capabilities greatly enhanced. Further, all analytical facilities may access a common database.

The communications facility between virtual machines is incorporated in the program's Multi-User Interface. The implementation of this communications facility is described more fully in [Gutentag, 1975; and Donovan and Jacoby, 1975]. That basic problem is to allow communication between VM's. Essentially what is needed is a means of passing commands and data to the database machine, returning data, and a locking and queueing mechanism.

The mechanism implemented in GMIS is as follows (note that this mechanism may be invisible to a modeler): Each user virtual machine (UVM), which is accessed by logging on to a separate account ID under VM/370, sends transactions to the Transaction Virtual Machine through a communications facility shared files and virtual card punchers and readers. The Multi-User Interface (MUI) stacks these transaction requests and processes them one at a time. The results of each transaction are passed back to the virtual machine that made the request through the same communications facility. Replies to the transactions may be processed with any software interface that is required for the application.

While more uses of VM's in an interconnected environment are being found, even more efficient intercommunications facilities are being developed, e.g., virtual machine to virtual machine core transfers [Hsieh, 1974]. But with the software available, with the compatibility and protection problems, and in light of the fact that many of the modeling systems do no use a standard file system, such as CMS, we chose the above mechanisms for the prototype GMIS.

3. EXPERIMENTAL ANALYSIS OF OVERHEAD COSTS

For the experimental study reported in this section, the configuration used is an APL modeling machine and a SEQUEL database machine, as depicted in Figure 3. (SEQUEL [Chamberlain, 1974] is an experimental relational data management system.) The study analyzes the overhead incurred in sending a request for data to the SEQUEL machine from the user APL machine and in receiving the requested data back. That overhead consists of two components, A and B, where

A = time spent in APL interface (passing commands down and converting returned data's format into host's system); and

B = time spent in Multi-User Interface (linking to user VM disk, security, and sending back data).

The data is processed in the SEQUEL machine in time C, where

C = time spent in SEQUEL (processing request for information).

Therefore we may write:

$$\text{Percent Overhead} = \frac{A + B}{A + B + C}$$

That is, if C is very large, then the percent overhead is small. Or, said in another way, if C is very large, a larger portion of time was spent formulating the data in the data management machine than in the interface routines.

The overhead amount may be viewed as a function of the type of query made and of the amount of data requested. To perform this experiment, we use four classes of queries of varying degrees of "complexity," yet all capable of

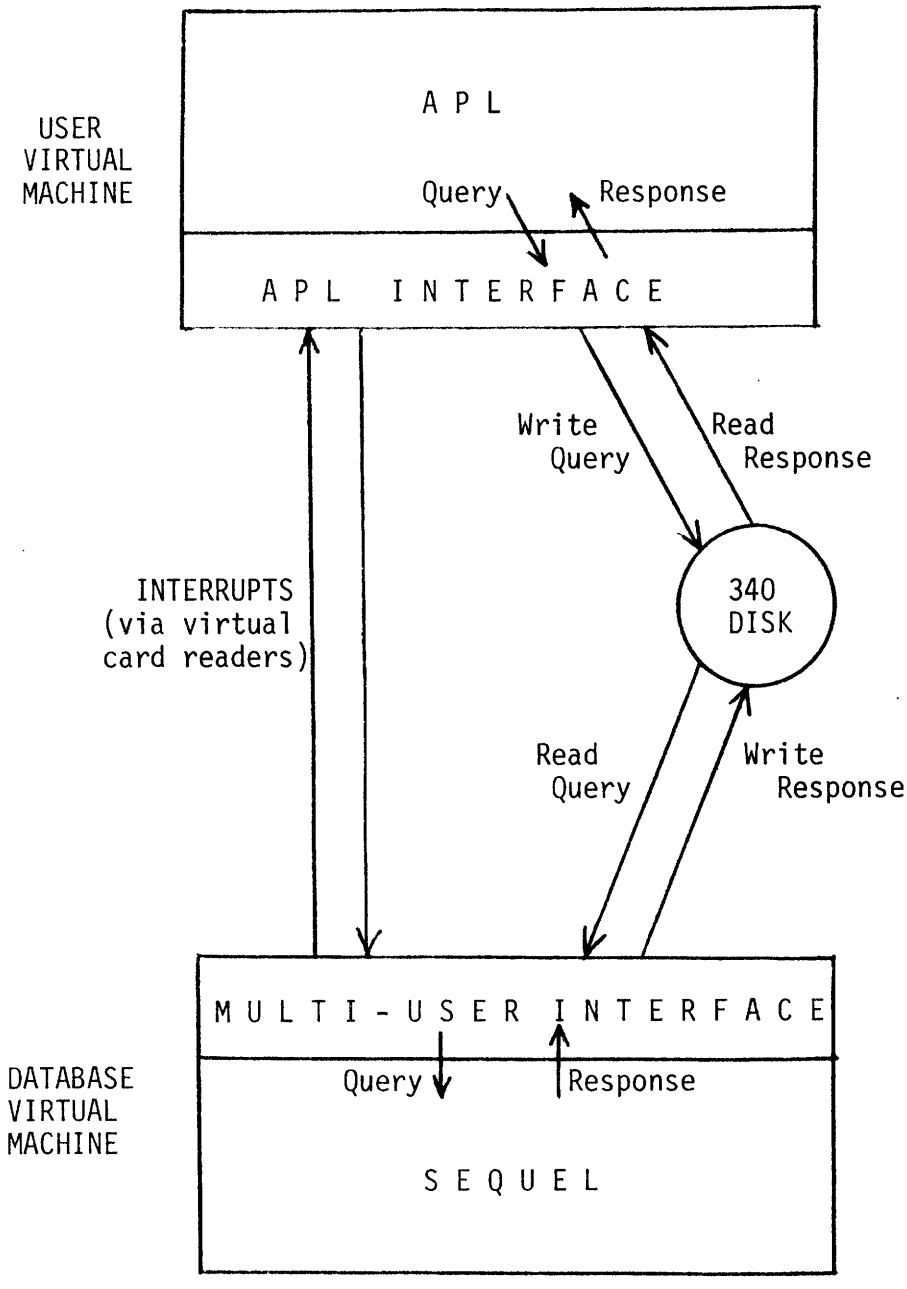

Figure 3: Experimental Configuration

retrieving the same amount data.

Those queries are SEQUEL queries, where complexity numbers 1, 2, 3, and 4 correspond to the following types of queries. Note that all queries were selected so that they actually resulted in retrieving all the data in the table.

1 -- a simple retrieval of all data in a table

```
(SELECT * FROM table;)
```

2 -- a retrieval of all data meeting one condition

```
(SELECT * FROM table WHERE STATE IN (CT, VT, ME, NH, RI, MA);)
```

3 -- a retrieval of all data meeting two conditions

```
(SELECT * FROM table WHERE STATE IN (CT, VT, ME, NH, RI, MA) OR STATE
   IN SELECT STATE FROM INCOME WHERE TWOTOTHREEK > 0 OR STATE = MA;;)
```

4 -- a retrieval of all data meeting three conditions

```
(SELECT * FROM table WHERE STATE IN (CT, VT, ME, NH, RI, MA) OR STATE
   IN SELECT STATE FROM INCOME WHERE COUNTY IN SELECT COUNTY FROM
   TERMINAL WHERE PLACEMENT >= 0;;;)
```

To vary the amount of data, four tables of linearly increasing size were used:

Table Name	Number of Rows	Number of Columns
R1	457	2
R2	457	4
R3	457	6
R4	457	8

Figure 4 depicts the observed results for constant amounts of data retrieved. Note that a high percentage of overhead is incurred when using the interface

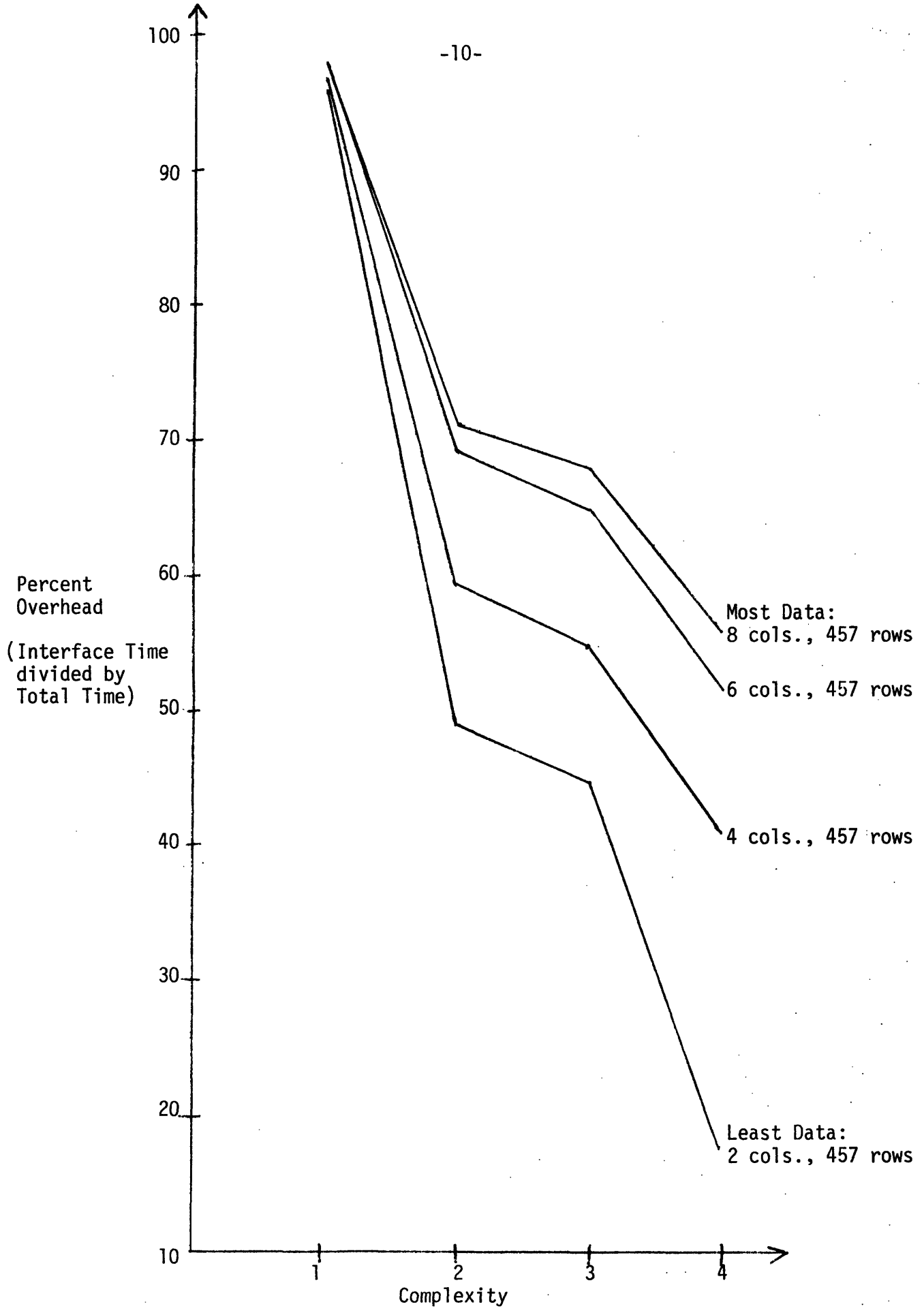

Figure 4: Overhead as a Function of Complexity

mechanism for simple queries, regardless of the amount of data retrieved. The system is most efficient for complex queries on small tables.

Figure 5 depicts the time, in milliseconds, observed in each of the interfaces, as a function of the amount of data requested and using a constant complexity of query. Note the linear nature of this figure.

4. ANALYSIS OF RESPONSE TIME DEGRADATION DUE TO LOCKING

The construction of a system of communicating VM's brings the previously mentioned advantages, but these come at the expense of some sacrifice in performance. Various performance studies of VM's are available in the literature [Hatfield, 1972; Goldberg, 1974]. We report here on a theoretical analysis and in the next section on the practical implications of the degradation of response time as a function of the number of modeling machines. The direction of this work can be seen by considering a configuration as in Figure 1, where several modeling facilities, each running on a separate virtual machine, are accessing and updating a database that is managed by a database management system running on its separate virtual machine. What is the degradation of performance with each additional user? What determines the length of time the database machine takes to process a request? What is the best locking strategy?

An access or update to the database machine may be initiated either by a user query, which would be passed on by the modeling machine, or by a model executing on the modeling machine. In either case, the database machine, while processing a request, locks out (queues) all other requests. The analysis is further complicated by the fact that as some VM's become

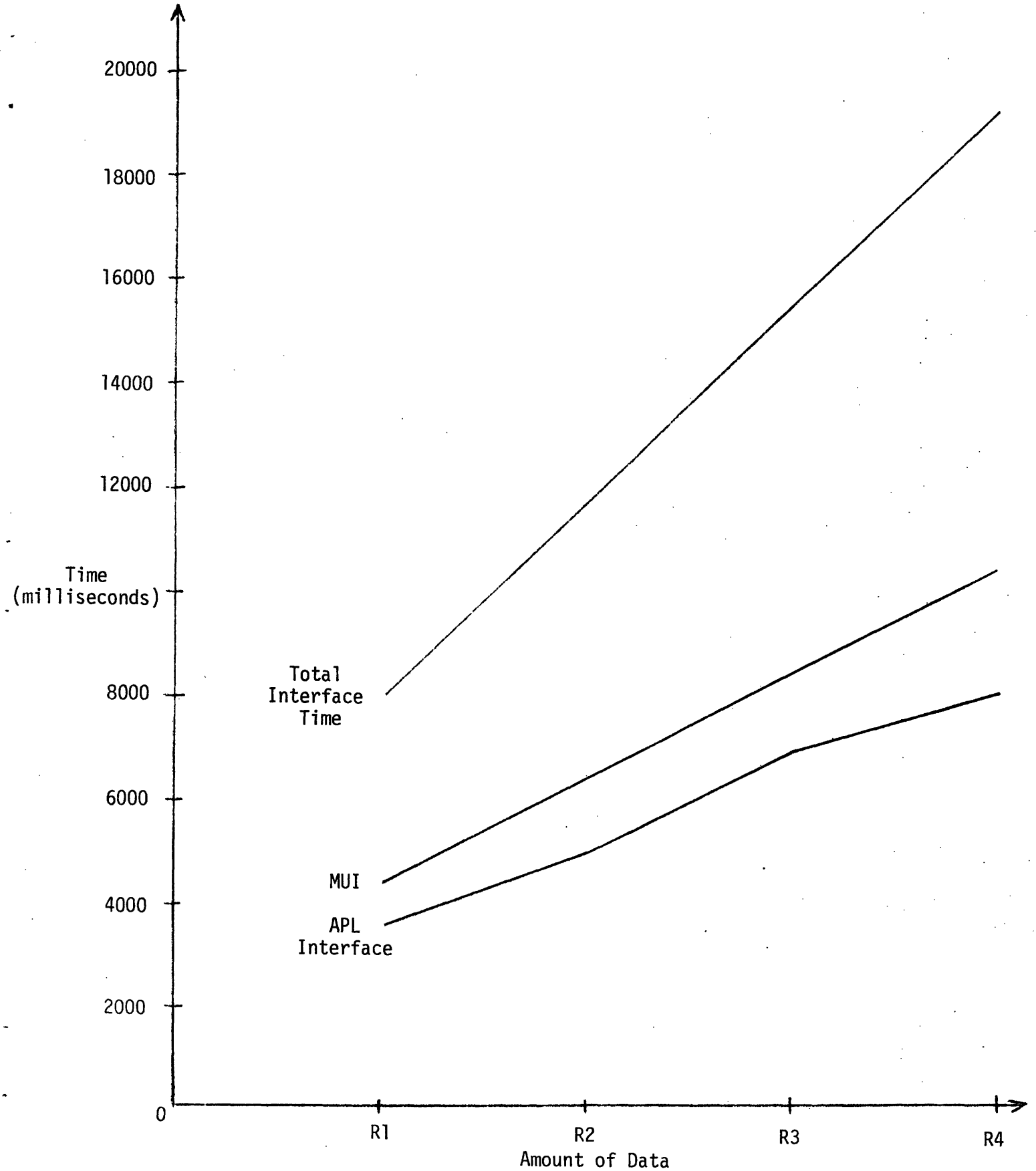

Figure 5: Time Spent in Interfaces

locked, then others get more of the real CPU's time and therefore generate requests faster. However, the database VM gets more of the CPU's time and thereby processes requests faster. For example, if there are ten virtual machines, each one receives one-tenth of the real CPU. If seven of the ten are in a locked state, however, then the remaining three receive one-third of the CPU. Thus these three run (in real time) faster than they did when ten were running.

To try to analyze this circumstance for the uses outlined in this paper, we have assumed that the virtual speeds of VM's are constant and equal. When some VM's (including the database VM) are allocated a larger share of CPU processing power, however, they become faster in real time. We assume that each unblocked VM receives the same amount of CPU processing power and that at the initial state m machines are running (i.e., the database machine is stopped if no modeling machines are making requests). 'λ' is the request rate of each modeling VM when there are m VM'm running. 'μ' is the service rate at which the database virtual machine is running when there are m-1 modeling VM's and one database VM running. Thus we may write the following relations:

$$\mu_i = \frac{m}{m-i+1}\,\mu \qquad (i = 1, 2, \ldots, m)$$

$$\lambda_0 = \lambda$$

$$\lambda_i = \frac{m}{m-i+1}\,\lambda \qquad (i = 1, 2, \ldots, m)$$

where i is the number of modeling VM's being blocked. Using a birth/death process model [Drake, 1967], and using a queueing analysis [Little, 1961],

we get the following for the response time of the model, where P_i is the steady state probability that there are i modeling machines waiting, and 'N' is the number of modeling machines:

$$T'_{model} = N * \left(\frac{\sum_{i=0}^{m-1} P_i \left(\frac{m-i}{m}\right)}{\sum_{i=0}^{m-1} P_i \left(\frac{m-i}{m}\right) \lambda_i} \right)$$

$$T'_{overhead} = \text{constant}$$

$$T'_{wait\text{-}for\text{-}data} = N * \frac{\sum_{i=1}^{m} iP_i}{\sum_{i=1}^{m} \mu_i P_i}$$

$$T'_{total} = T'_{overhead} + T'_{model} + T'_{wait\text{-}for\text{-}data}$$

5. IMPLICATION OF THEORETICAL RESULTS

Figure 6 illustrates the total time to execute three different models as a function of the number of modeling VM's. Let us consider some of the implications of the above analysis.

First, for $\lambda/\mu = .1$, a model executing in a configuration of one modeling machine takes 110 units of time to execute. When the same model, run in an environment of 10 modeling machines all executing similar models, takes approximately 135 units of time to execute, the degradation of performance is slightly more than 15 percent. Intuitively, λ denotes the speed of the modeling machine, and μ is the speed of the database machine.

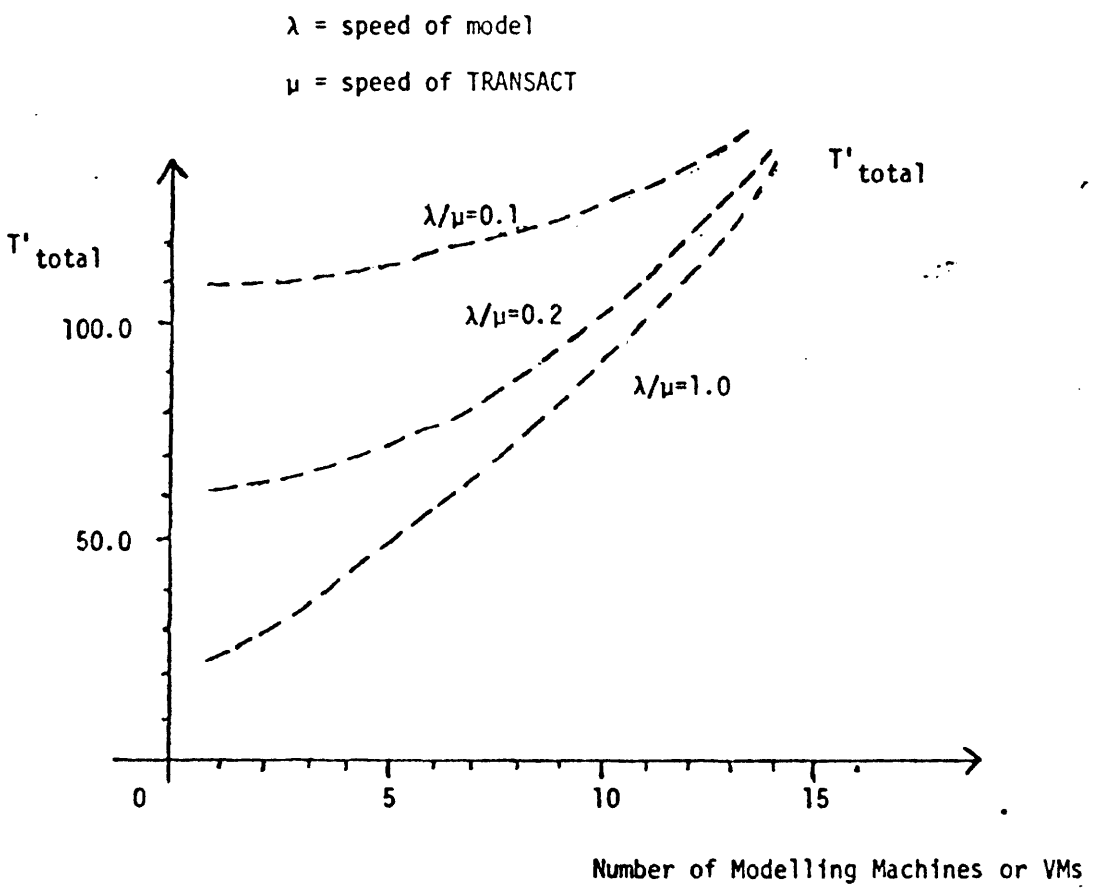

Figure 6: Total Elapsed Times for a VM Configuration

Thus a situation where $\lambda/\mu = .1$ indicates that the database machine is ten time faster than the modeling machine. From the same figure, with the ratio $\lambda/\mu = 1$, a model executing with a configuration of one modeling machine takes 20 units of time; whereas with ten machines the same model takes approximately 90 units of time--over four times longer.

If such a degradation of performance is not tolerable, there are several ways to improve performance. The theoretical study would indicate that increasing μ for a given configuration helps performance. Practically, this could be done by changing the processor scheduling algorithm of VM so that the real processor is assigned to the database management VM more often, thus speeding it up and increasing μ.

Observing the equation for T'_{total} above, another way of reducing T'_{total} is to reduce $T'_{wait\_for\_data}$. One way to reduce $T'_{wait\_for\_data}$ is to extend the VM architecture of Figure 1 to allow for multiple database machines. In this configuration $T'_{wait\_for\_data}$ could be reduced by locking out all database machines only when one modeling machine is doing a write. For all read requests the multiple database machines would operate without locking. Shared locks between machines would have to be created, as well as a mechanism for keeping a write request pending until all database machines can be locked.

A way of improving performance further would be to extend the single locking mechanism used in the above multi-database machine configuration to handle multiple locks. Locks would be associated with groupings of data, e.g., a table. The locking policy would be to have all machines locked out of a portion of the data only when one machine is writing into that portion. Thus requests could be processed simultaneously for reads into tables not being written in and for reads to different tables. Thus

adding another real processor to the multiple-lock VM configuration could greatly improve performance. There is a tradeoff with the multi-locking scheme between increases in overhead time in maintaining multiple locks versus increases in wait time for locked databases. We have not yet extended the theoretical analysis to quantify this tradeoff.

Hence this study indicates that there may be a degradation in performance with multiple users, but that there are mechanisms for ameliorating the effects of this degradation.

Other theoretical extensions and analyses of this synchronization model would include extending the model to cover a more common VM operating circumstance, namely, that where the GMIS system (multiple modeling machines and one database machine) would have to share the physical machine with other users also executing under VM, e.g., a payroll program under VS2 under VM, multiple CMS users, etc.

6. CONCLUSION

Our experience with configurations of virtual machines has to date been useful in the application areas of decision support systems. The conclusion of the study reported here is twofold: (1) The overhead incurred in the interface is small for complex queries on small amounts of data; and (2) There exist methods for reducing degradation of response time due to locking mechanisms. They include changing the VM scheduling algorithm, adding locks, and incorporating additional virtual machines. We feel that further studies on cost benefits analysis and on increased effectiveness of users of this sort of system will quantitatively confirm our observation of the benefits of this approach.

ACKNOWLEDGMENT

Acknowledgment is given to Professor Stuart Madnick of the Sloan School for helping in formulating the performance equations and the experiment; to Professor Peter Chen, also of the Sloan School, for helping with the analytical construction and solutions to the equations; and to Peter DiGiammarino for his efforts in gathering the measurements for the experimental analysis of overhead.

We thank Drs. Stuart Greenberg and Ray Fessel of the IBM Cambridge Scientific Center for their assistance in implementing GMIS; the IBM San Jose Research Center for the use of their relational system SEQUEL and for their assistance in adapting their system to the GMIS applications; Louis Gutentag and the M.I.T. students who worked with him for their efforts in making the system operational; Jeff Buzen of Harvard for his review of the theoretical work; and Professor Henry Jacoby of the Sloan School for his assistance in articulating some of the ideas expressed herein.

REFERENCES

Bagley, J. D., E. R. Floto, S. C. Hsieh, and V. Watson. "Sharing Data and Services in a Virtual Machine System," ACM SIGOPS Conference Proceedings, 1976.

Buzen, J. P., P. Chen, and R. P. Goldberg. "Virtual Machine Techniques for Improving System Reliability," Proceedings of the ACM Workshop on Virtual Computer Systems, March 26-27, 1973.

Chamberlain, D. D. and R. F. Boyce. "SEQUEL: A Structured English Query Language," Proceedings of 1974 ACM/SIGFIDET Workshop, 1974.

Codd, E. F. "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, vol. 13, no. 6, June 1970, pp. 377-387.

Donovan, John J. and Henry D. Jacoby. "GMIS: An Experimental System for Data Management and Analysis," Working Paper No. MIT-EL-75-011WP, September 1975.

Donovan, John J. and W. Robert Keating. "Text of Governors Presentation," Report CISR-18, December 1975.

Drake, A. W. Fundamentals of Applied Probability, McGraw-Hill, New York, 1967.

Goldberg, R. P. "Survey of Virtual Machine Research," Computer, vol. 7, no. 6, June 1974, pp. 34-35.

Gutentag, L. M. "GMIS: Generalized Management Information System--an Implementation Description," M.S. Thesis, M.I.T. Sloan School of Management, June 1975.

Hatfield, D. J. "Experiments on Page Size Program Access Patterns and Virtual Memory Performance," IBM Journal of Research and Development, vol. 16, no. 1, pp. 58-66, January 1972.

Hsieh, S. C. "Inter-Virtual Machine Communication under VM/370," RC 5147, IBM Research, Yorktown Heights, NY, November 1974.

IBM Corporation. IBM Virtual Machine Facility/370: Introduction, System Reference Library, Form GC20-1800, White Plains, NY, July 1972.

Little, J. D. C. "A Proof of the Queueing Formula: $L = \lambda\omega$," Operations Research, vol. 9, 1961, pp. 383-387.

Madnick, S. E. "Time-Sharing Systems: Virtual Machine Concept vs. Conventional Approach," Modern Data, vol. 2, no. 3, March 1969, pp. 34-36.

Madnick, S. E. and J. J. Donovan. Operating Systems, McGraw-Hill, New York, 1974.

Morrison, J. E. "User Program Performance in Virtual Storage Systems," IBM Systems Journal, vo. 12, no. 3, 1973, pp. 34-36.

Parmelee, R. P., T. I. Peterson, C. C. Tillman, and D. J. Hatfield. "Virtual Storage and Virtual Machine Concepts," IBM Systems Journal, vol. 11, no. 2, 1972, pp. 99-130.