A SIMULATION STUDY OF DATA TRANSFER BETWEEN TWO ANYNCHRONOUS DEVICES: AN EXAMPLE OF THE USE OF GPSS IN COMPUTER SYSTEMS DESIGN

by

William T. McClelland, Jr.
Systems Development Division
International Business Machines Corporation
Kingston, New York

and

Kathleen M. Rutter
Systems Development Division
International Business Machines Corporation
Poughkeepsie, New York

SUMMARY

This paper is a conceptual description of a GPSS simulation model employed to investigate the transfer of data from a rotating storage device to a high speed communications channel. The hardware characteristics are studied within the framework of the total system.

INTRODUCTION

The growth of the computer industry over the last two decades has created an ability to solve problems several magnitudes of complexity beyond those even definable prior to the sixties. These technological advancements have at the same time introduced more challenging problem areas, not the least of which is the actual design of computer systems. Justification is found in that with the increase in sophistication has come a corresponding increase in the risk of designing terminal-type periphery independent of major performance considerations. Significant emphasis is currently being placed on the concept of product design for optimum integration into a total systems environment. It is conceivable, for example, that several clusters of expertly designed terminals could actually degrade overall systems performance because of unforseen bottlenecks created elsewhere in the computer network.

Coupled with the need to investigate the performance of complex systems is the fact that detailed evaluations are not easily provided within the constraints of probabalistic or queueing theoretic analytics. Frequently, analysts require a more in-depth description of complicated interrelationships than existing analytical methods can provide. Of fundamental interest is the effect of congestion resulting from the flow of many interacting entities throughout the system. It is becoming increasingly important to understand variations of individual parameters as they relate to critical system response variables.

Simulation languages such as GPSS/360 provide the capability to represent such systems at almost any required level of detail within the constraints of time and cost. These high-level, event-oriented languages permit analysts to define a network in terms of its shared resources, transactions to be serviced, and the queueing disciplines by which the transactions contend for service. Together with the sequential flow of one transaction through the system, pseudo-random number generation allows the analyst to vary critical parameters and study important response variables under stochastic fluctuations. Complex convolutions of dissimilar statistical distributions may be represented easily without the constraints inherent in the underlying assumptions of most analytical techniques. This alone takes the analysis one meaningful step beyond usual worst case approximations.

One major consideration in a simulation effort of this type is identification of those system components demanding the closest scrutiny. In most cases, a detailed description of computer software would add little to a study of device-to-control unit data transfer. Conversly, it would make little sense to implement a detailed representation of peripheral devices in a study of task switching in a multi-programmed operating system. A determination must be made at the outset to pinpoint the potentially crucial areas and to avoid the allocation of expensive resources to detailed investigations of non-related aspects.

Once questions of technique and/or language, level of detail, and critical components are answered, the system must be learned in depth. Clearly, any simulation effort will net results only as reliable as its descriptive input. The analyst should eventually come to know as much or more about total systems performance than any single design engineer. This also leads to increased personnel with a working knowledge of the system and expediency in addressing future design questions.

Several referenced papers[1], [2], [3] discuss the use of some software simulation tools[4], [5] in the investigation of total system performance. Reference [2] concentrates chiefly on line control characteristics and involves only the most necessary representations of system software. Reference [1], on the other hand, deals exten-

sively with both software and hardware aspects of complex teleprocessing systems. These two approaches to the analysis of similar computer systems demonstrate the importance of focus, each maximizing attention on the subject of interest.

Through the use of a particular simulation study and descriptions of a generalized system, we shall attempt to demonstrate the effectiveness of simulation in the field of computer systems design. The investigation of performance implications of data transfer from a shared rotating storage device should serve to indicate the addressable scope of such simulation studies. As examples, read/write time, resource utilizations, and throughput are assessed as functions of channel type and system configuration. Since device-channel data transfer was the key issue of interest, system software merited only minor consideration. Central processing time was therefore represented as a uniform random variate over a specified range and focus was on the rotating storage device itself. Included herein is a brief description of a hypothetical rotating device, a discussion of two GPSS simulation models, and some comments on output analyses.

SYSTEM DESCRIPTION

The basic sturcture of the interactive system under study consisted of a SYSTEM/360 Processing Unit and multiple control units attached via either a selector or multiplexor channel. Clustered subdevices communicated with the processor through a rotating storage device and a 16-byte common storage buffer housed in each control unit. (See Figure 1) Major variables of interest were read/write (response) time, resource utilization, and throughput as a function of channel type, buffer size, and system configuration. Of specific concern was performance degradation as related to synchronization loss during control unit-channel data transfer operations.

The following sections describe the major facets of the computer network: All timings, channel interference factors, and operational sequences pertain to the SYSTEM/360 central processor.

A. The Rotating Storage Device

For brevity of description, consider a rotating storage device (RSD) with several tracks, each having a specific storage area allocated to multiple subdevices. Assume further that one byte of data may be transferred from the RSD to the 16 byte common-storage buffer at a peak rate of 11μs per byte. Since data storage is sequential in nature, message length and location of the data on the RSD are critical to the data transfer operation. It is also conceivable that a single message may be broken into segments and be assigned to several disconnected storage locations. The implications will become obvious later.

B. Data Channels

For this investigation three related simulation models were developed: (1) a multiplexor channel, RSD model, (2) a selector channel, RSD model, and (3) a subset of (2) with a hypothetical "look ahead" feature. Operational characteristics of these systems are described below.

Multiplexor and selector channel operations differ in that the former is shared (with interleaved service) by multiple control units and the latter is dedicated to one control unit for an entire data transfer. In such a system, the selector channel will operate in synchronization with the rotating device and be independent of the intermediate buffer. Data Transfer over the Multiplexor, by its very nature, is highly dependent upon buffer speed, buffer capacity, and message configuration on the RSD.

A complete interactive read/write sequence between a control unit and the central processor consists of an attention interrupt, a read from the control unit to the processor, and a write from the processor following application program processing. A typical non-preemptive priority scheme for locally attached devices is depicted by Figure 2 for the case of eight control units. Note that the attention interrupt and read operations are assigned a priority relative to their control unit number.

A normal selector channel operation allows the data transfer to begin only after the synchronization position (storage location zero) is encountered on the rotating storage device. Given that the RSD was at storage location two when service was requested and that the first desired byte of data was located storage location three, one complete revolution would be necessary prior to the actual transfer of data. To avoid this critical delay period, a hypothetical "look ahead" feature was proposed to initiate the transfer immediately upon encountering the first data position. Hence, selector channel service (exclusive of operating system control sequences) consists of waiting for position zero and/or the first data byte prior to the burst mode transfer of data.

The maximum amount of time a multiplexor channel could devote to any single control unit per service request was set at 32μs. At a rate of 4.4μs per byte, a maximum of eight bytes could be transmitted per request (a byte transfer commencing prior to the time limit would be completed).[1] Figure 3 depicts the sequential flow of multiplexor channel operation. As a result of the shared channel con-

[1] If a higher priority selector channel stole 100ρ percent of the CPU memory cycles, the effective transfer rate would be approximately 4.4/(1-ρ) microseconds per byte. This could significantly decrease the number of bytes transferred per request.

cept, multiplexor service was highley dependent on buffer activity and device rotational speed. For a read operation, data could be removed from the RSD at 11μs per byte and sent over the channel at 4.4μs per byte.

If a buffer filled to capacity before channel service was obtained, at least one revolution of the RSD would be necessary before more data could be transferred. Many simultaneous demands placed upon the multiplexor increase the probability of several such occurrences per read operation. The effect is to seriously degrade systems performance, a phenomenon that is magnified strongly at lower priority control units.

During the CPU write operation, data enters the buffer at 4.4μs per byte and is transferred to the RSD at 11μs per byte. When the buffer fills to capacity, the control unit can no longer accept multiplexor service until space is available. If the buffer empties before service is regained, the result will be loss of synchronization with the RSD. One loss of this type increases the probability of additional losses because of the cyclic nature of service demands placed upon the multiplexor.

C. Assumptions

The following is a brief outline of some representative underlying assumptions to define the system. These, together with the parameters mentioned previously, are combined in the next section to describe message flow through the system.

a) Request arrival rate: An attention interrupt is generated by each subdevice (10±2) seconds after its previous operation. The attention interrupt is a signal from the subdevice requesting that a read/write operation be performed.

b) Read: Nine bytes, beginning in storage location 1057, are transferred to the CPU.

c) Write: Three hundred and twenty bytes are transferred from the CPU to storage locations 1057-1376.

d) CPU Processing: Without loss of generality, application program processing and all internal CPU transfers may be represented by a uniform random variate on a range of (100, 400)ms.

e) Supervisor and Channel Timings:

Attention Interrupt	1000μs
IOS Supervisor Call	1000μs
START I/O	100μs
I/O END INTERRUPT	1000μs

f) Channel Data Rate: While a peak data rate of 91KB per second is assumed, transfer is governed by RSD or buffer speed.

D. Logical Flow Through the System

The four basic segments of a read/write transfer are 1) the attention interrupt, 2) the read to the CPU, 3) CPU processing, and 4) the write from the CPU. The gross representation of CPU processing was necessary to reflect output queueing and is representative of a wide range of application benchmarks. All other operations are defined in depth.

A read/write transaction over a selector channel begins when the control unit receives an electrical pulse from one of its subdevices. Multiple requests are handled on a first-come-first-served basis with the operational priority previously defined. The control unit initiates contact with the CPU by obtaining channel service (at attention priority) to transfer the necessary status and address information. Following this, the channel and control unit are free and a delay of 1000μs is encountered to represent supervisor interrupt handling. A second delay of 1000μs simulates the supervisor call to begin the read operation after which the subdevice again contends for control unit service at read priority.

When all pending attention interrupts and write operations requiring the control unit have been completed, subdevice read operations are served. The channel is obtained according to the control unit priority and a 100μs START I/O operation is executed. The read sequence consists of a synchronization delay (with or without the "look ahead" feature), the message transmission, and a 1000μs I/O END INTERRUPT operation. The channel and control unit are then free and the message queues for CPU processing on a FIFO basis (the CPU was represented by a GPSS storage of capacity four to provide for limited multithreading).

Following the CPU processing, a 1000μs delay is encountered to reflect the supervisor call for the write operation. The message is placed on the output queue where it will be transferred only when the channel and destination control unit are simultaneously free. The transfer sequence consists of a 100μs START I/O, the write operation, and the I/O END INTERRUPT. Following the completion of the write operation, the addressed subdevice will generate another attention interrupt (10±2) seconds later.

With limited exception, operations fundamental to the selector channel apply to the multiplexor channel. Major differences are the shared multi-byte concept of the multiplexor and the impact of selector channel interference. The multiplexor priority structure is also similar to the selector except that START I/O is assigned the lowest priority and I/O END INTERRUPT is assigned the highest priority of any operation. The total duration of these two operations is the same as with the selector channel. However, the multiplexor is held for

only 100μs during the I/O END INTERRUPT and the control unit is forced idle for the entire operation.

After the attention interrupt is processed and START I/O executed, an RSD synchronization period is encountered and the buffer begins to fill. As long as there is one data byte in the buffer, channel service will be requested and obtained on a priority basis for 32μs. Upon the completion of the message transfer, multiplexor service is immediately obtained for the I/O END INTERRUPT. The write operation, similar to the read, entails the execution of a 1000μs supervisor call for write prior to placement of the message on the output queue. According to the priority scheme a free path (Channel and Control Unit) is obtained and a 100μs START I/O is executed. Channel service is requested as long as there is space available in the control unit buffer. The I/O END INTERRUPT is handled identically with that of the read operation.

The above discussion should clarify the complexities encountered in the identification of system performance characteristics. Average read/write time, for example, has significant dependencies on:

a) the particular channel design
b) the number of control units sharing a channel
c) message length and position on the RSD
d) physical position of the control unit on the channel
e) size of the common storage buffer
f) attainable peak data rates for the RSD and channel
g) possible performance improvements such as the "look ahead" feature
h) other system parameters.

Such a system is a suitable representation of the field of computer system design where simulation is at its best as a useful tool. The system poses an extensive list of parameters which will influence performance in some unknown manner, as well as a degree of complexity in the interrelationships between controlling variables that are nearly impossible to assess via alternative techniques. A comprehensive discussion of the innumerable performance dependencies in this system would be impossible to address in a short paper. However, for purposes of illustrating the effectiveness of simulation, we will consider the results of one particular combination of system parameters in the section that follows.

SOME GENERAL RESULTS

Extensive output analyses were conducted for all of the simulation models. The multiplexor model, however, involved considerable detail and computer runtimes were extremely slow. For this reason only those data points of specific necessity were obtained and will not be presented here. Some general data gathered from the model representation of the selector channel with the "look ahead" feature should suffice to describe the type of output of interest.

The basic time unit for the selector model was one millisecond (which compares to a time unit of one microsecond for the multiplexor model). Based on stabilization studies centered around queue lengths and read/write times at specific points across time, it was determined that one minute of simulated time would be necessary to obtain pseudo-steady state conditions. It was further determined that for most cases, a simulated time period of five minutes was adequate to gather meaningful statistics. Four special cases involving a total of 256 subdevices required a six minute stabilization prior to the five minute production period because of the extreme variability involved. In the context of this high contention, multi-priority computer network, the term "steady state transitional probability" is left at loosely defined as possible. That such a system ever attains steady state in the theoretical sense is subject to doubt.

The configurations presented as examples consist of combinations of up to eight control units with a maximum of 256 total subdevices. While some combinations yield unrealistic results, behavior at the extremes of any continuum is often of interest.

Table I describes some representative output from the simulation runs. The columns entitled T_1, T_3, T_5, T_8, and T_{10} are defined below, together with other components making up the total read/write time (T).

T_1 = wait time until attention interrupt is serviced.
T_2 = 1 ms attention handling.
T_3 = wait time to begin a read after T_2.
T_4 = 1.1 ms Supervisor Call, START I/O.
T_5 = read time, including RSD latency.
T_6 = 1 ms, I/O END INTERRUPT.
T_7 = CPU process time (250 ms average).
T_8 = wait time to begin a write after T_7.
T_9 = 1.1 ms Supervisor Call, START I/O.
T_{10} = write time, including RSD latency.
T_{11} = 1 ms, I/O END INTERRUPT.

Note that the final I/O END INTERRUPT occurs after the write has been completed and is not a true component of read/write time. Rough approximations of read/write times for different CPU process times may be obtained by substitution for T_7. In this, however, one must exercise extreme caution since the effect on system queueing characteristics is virtually undefined.

Much insight may be gained from information such as that presented in Table I. For example, the most serious system bottle-

neck is the wait prior to the read operation. In general, the queueing time for this low priority operation increases significantly with the number of subdevices introduced into the system. The fact that subdevices contend for control unit as well as channel service manifests itself in the non-linearity of the queue length increases. This performance tradeoff is reflected as the configuration is varied for a constant number of subdevices. Note the behavior of T_3 for the cases of 128 and 256 subdevices as the number of control units is decreased.

Very few effects are seen in any of the other queueing variables as a result of configuration changes. Delays prior to attention handling vary slightly but are essentially negligable because of the high priority. Delays prior to the write operation vary more significantly, but the overall effects are minimal. Given a constant starting location for all messages, read/write times increase as activity increases because of the corresponding higher probability of being closer to the location where the previous I/O operation terminated (that is, almost one complete rotation away). These effects are also negligible in comparison to the read queueing times.

Control unit and channel utilizations also merit considerations. For the case of one subdevice per control unit, with the assumed arrival rate of I/O requests, an average control unit utilization of 0.3 percent was observed across eight control units. A literal translation is that 99.7 percent of the control unit's maximum capability has been wasted. The concept of clustering multiple subdevices on a control unit is an attempt to put this unproductive time to use. This utilization increases to 43 percent for 128 subdevices on a single control unit, and to 100 percent for 256 subdevices. A general rule of thumb is that beyond the 50 percent point, queues begin to build and read/write time degrades exponentially. Total throughput begins to level off at this point until, at a utilization approaching 100 percent, it achieves its asymptotic maximum. Figure 2 depicts a representative relationship between these two variables as a function of control unit utilization. Other far reaching implications, such as reasonable limits on selector channel utilization will not be discussed.

This simulation effort led to several meaningful conclusions. The first was that the most critical delay was encountered prior to the read operation. This was expected because of the priority structure but it was extremely useful to measure the magnitude of the delay as a function of control unit and channel utilization tradeoffs. An unexpected phenomenon was the behavior of the read delay as a function of system configuration. The ability to achieve an absolute optimum configuration has led to other meaningful studies along this line. Most important, although not quantified here, was the fact that the comparison between the two systems confirmed beyond a reasonable doubt that loss of synchronization caused serious performance degradation when the multiplexor channel was employed. Perhaps equally as dramatic was the effect of message segmentation. Specifically, there are critical dependencies on segment length and the distance between segments for the multiplexor. If handled improperly this alone has the potential to make the entire system unworkable. Having the ability to measure this factor for many different system loads contributed significantly to the entire effort.

CONCLUSION

With a brief description of the system under study and some types of meaningful output statistics, it has been our intent to present one approach to the design and analysis of computer systems. The attempt has been to remain as general as possible and to use one small segment of a particular simulation study to demonstrate the effectiveness of this technique as a design aid. The most important consideration is that with this approach, the analyst has available the means to vary an extremely large number of system parameters and assess their relationship to overall performance. Statistical analyses of the output and queueing theoretic validations associated with the simulation effort would merit extensive presentation in their own right and were not germane to the issue at hand.

While some extremely complex GPSS[2] techniques were necessary to circumvent limitations of this version of GPSS, the selector channel system was represented with relative ease. The transition from a detailed flow chart directly into GPSS code did not present the anticipated problem. There is a loosely defined point of departure, however, after which the use of a general purpose language such as GPSS becomes prohibitive. More difficult modeling techniques were necessary to represent the multiplexor system and still maintain economy of computer time. A computer-oriented simulation language would have had considerable value in this instance. This reflects the trade-off between the level of detail required and the cost of achieving that goal.

Most important of all, however, was the fact that this effort demonstrated the effectiveness of simulation as a valuable design tool--a technique by which the total systems performance of hypothetical devices could be assessed prior to any costly physical design commitment. Peripheral devices do not always perform as expected when integrated into a

[2]A release prior to GPSS Version 2, the IBM Program Product.

systems environment and simulation, within certain contraints, can be extremely useful in a quantification of critical performance issues otherwise undefinable.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to Mr. James D. Russell, System Design, Systems Development Division, IBM, Kingston, N.Y. for his continued support in this and related efforts. We are also indebted to him for many helpful comments regarding the content of the paper itself.

REFERENCES

[1] Fogel, R. M. and McClelland, Jr., W. T., Use of Simulation in the Design of Information Display Systems. Proceedings of the 1970 Summer Computer Simulation Conference, June 10-12, 1970, pp. 78-86, Vol. I. Denver, Colorado.

[2] Catania, Salvatore, C. A GPSS Simulation Of A Communications Network. Proceedings of the 1970 Summer Computer Simulation Conference, June 10-12, 1970, pp. 668-676, Vol. II, Denver, Colorado.

[3] Anderson, H. A. "Simulation of the Time-Varying load on Future Remote-Access Immediate-Response Computer Systems." Proceedings of the Third Conference of Applications of Simulation, 1969, pp. 142-164.

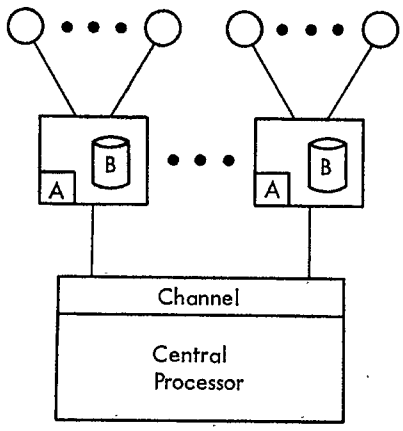
[4] General Purpose Simulation System/360, User's Manual, IBM Corporation, H20-0694.

[5] IBM Systems Journal, Vol. 8, No. 4, IBM Corporation, 1969.

Control Units	Devices Per Control Unit	Total Devices	Read/Write Time Components							Average Percent Utilization	
			T_1	T_3	T_5	T_8	T_{10}	T'	T	Control Unit	Channel
8	1	8	0.1	0.0	13.5	0.6	14.9	29.0	284.2	0.3	2.5
2	16	32	1.0	0.3	12.3	1.2	16.3	31.1	286.3	5.0	10.5
8	8	64	2.2	1.1	13.3	1.5	16.5	34.6	289.8	2.8	21.6
4	16	64	1.8	1.7	12.9	2.7	17.0	36.1	291.3	5.8	22.0
8	16	128	4.8	12.3	15.6	6.7	19.0	58.4	313.6	5.6	48.5
4	32	128	5.1	14.5	14.8	6.7	18.7	58.8	314.0	10.7	48.3
2	64	128	4.9	11.7	14.9	7.1	18.9	57.5	312.7	21.6	48.5
1	128	128	4.9	9.3	15.1	6.9	18.8	55.0	310.2	43.1	48.5
8	32	256	10.5	146.9	16.8	20.3	20.6	215.1	470.3	36.0	96.4
4	64	256	11.6	218.4	20.5	21.9	23.5	295.9	551.1	53.3	96.4
2	128	256	11.7	1815.5	20.7	22.6	23.6	1894.1	2149.3	71.4	96.4
1	256	256	11.6	1574.5	20.7	22.4	23.6	1652.8	1908.0	100.0	96.4

Note: $T' = T_1 + T_3 + T_5 + T_8 + T_{10}$, $T = \sum_{i=1}^{10} T_i$

Table I. Read/Write Time Components (ms.)



Control Unit
A) 16 Byte Buffer
B) Rotating
Storage Device

Fig. 1 Example System Configuration

Operation	CU1	CU2	---	CU8
Attention	17	16	---	10
Write	9	9	---	9
Read	8	7	---	1

Fig. 2 Channel Service Priorities

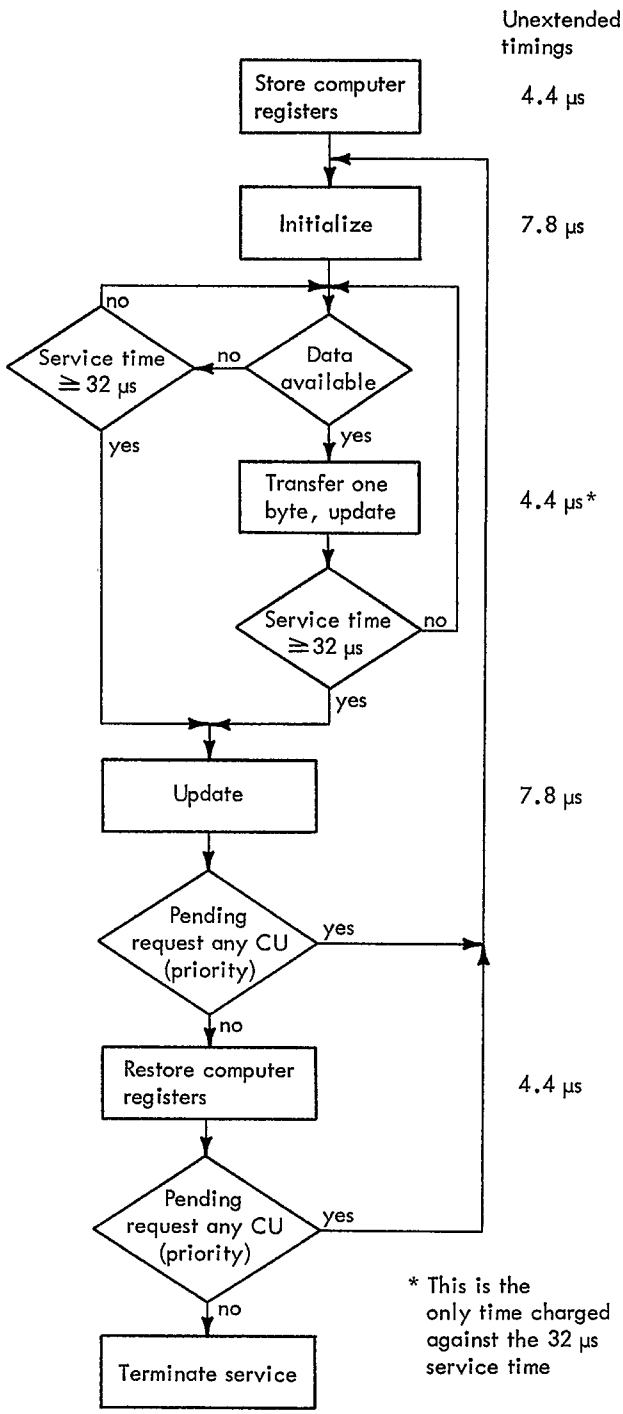


Fig. 3 Multiplexor Operations in Multi-Byte Mode with Control Unit Priorities

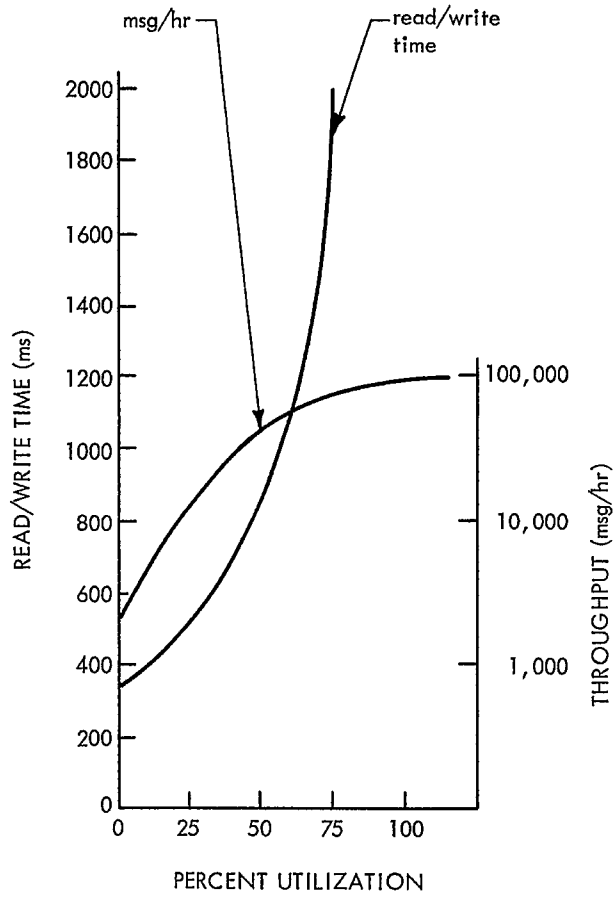


Fig. 4 Read/Write Time and Throughput