USING STORAGE FACTORS TO BALANCE

STORAGE SUBSYSTEM LOADS

By

ROGER R. HILL

Bachelor of Science

Brigham Young University

Provo, Utah

1979

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1992

USING STORAGE FACTORS TO BALANCE

STORAGE SUBSYSTEM LOADS

Thesis Approved:

G. E. Hedrick

Thesis Adviser

J Chandler

M. Samadzadeh-H.

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGEMENTS

As with most significant tasks, many assisted in completing this work. First, I would like to thank my father for his suggestion of the topic and his listening ear when difficulties arose in developing this thesis. I would also like to sincerely thank my thesis advisor, Dr. G. E. Hedrick, for his thorough reviews and suggestions for improvements, as well as his continued patience over the considerable time necessary to complete this work. Thanks also to Drs. M. Samadzadeh and J. Chandler for their willingness to act as members of my thesis committee.

I would also like to thank my employer, Conoco Inc., for the use of their computer facilities in the preparation of this thesis. Thanks also to the many relatives and friends who continued to help me keep sight of the goal.

Of course, without the support and encouragement of my wife and children, none of this would have been possible. I deeply appreciate their patience with my absence during the many evenings and weekends necessary to complete this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Storage subsystems provide two capabilities that nearly every computer system requires [15,18]:

- permanent data storage
- timely data retrieval

Some effort usually is required to configure a storage subsystem that cost effectively provides these capabilities. For small, single user systems, configuring a subsystem is typically a matter of obtaining the fastest devices that satisfy the storage space requirements without exceeding the budget. For large, multi-user systems, however, configuring a storage subsystem is much more difficult. With these systems, capacity planners must expend considerable effort to configure a storage subsystem that matches the anticipated loads with the capacities of the storage devices while maintaining reasonable costs. Numerous models have been developed to analyze the performance of proposed configurations in order to determine whether they will perform adequately. Using these models, a variety of configurations are evaluated until a configuration is found that meets the space, performance, and cost criteria specified for the storage subsystem.

Obtaining and maintaining the performance predicted by a model, however, is a very difficult task. This is primarily due to the continuously changing demands experienced by storage subsystems. The changing demands result from the ever changing business environment. As business conditions change, data requirements also change. Thus, data is added to the system; data is removed from the system; data is accessed more frequently; data is accessed less frequently. These changing demands can result in conditions that do not permit the storage subsystem to function efficiently, affecting both I/O performance and storage space. These conditions are discussed in the following sections.

Literature Review

I/O Performance and DASD Skew

One of the most important conditions created by changing I/O demand that affects I/O performance for direct access storage devices (DASD) is known as "DASD I/O rate skew" [20] or simply DASD skew. DASD skew is the disproportionate distribution of access demand among the storage devices within a storage subsystem. This means that within a group of similar storage devices, only a few of the devices handle a large percentage of the access load. The remainder of the devices handle a much smaller percentage of the load. As a result, the heavily utilized devices become bottlenecks to system performance because the devices cannot handle the access demands adequately. The usual indications of this condition include long I/O request queue times, increased path and channel contention, increased rotational position sensing (RPS) misses, and increased seek times due to concurrent activity on more than one data set, all of which result in longer I/O service times.

Because of its prevalence and potentially serious impact on storage subsystem performance, DASD skew is a major problem that must be managed in all large storage subsystems. The existence of DASD skew and its impact on DASD performance has been observed by many individuals who have studied DASD performance problems. Suggestions for dealing with the problem are also numerous. These suggestions usually involve investigation of current activity rates on the devices in the subsystem, then locating the most active devices and data sets, and finally moving the highly active data sets to less active devices. This more evenly distributes the load, allowing each device to be utilized better.

Following is a summary of the information provided by others relating to DASD skew. Some authors simply discuss the nature of DASD skew and its impact on storage subsystems in general, frequently as part of a discussion of other topics which are also affected by DASD skew. This information is described first. Others discuss the topic more directly, usually providing suggestions about how to handle the problem as part of a discussion of tuning strategies.

The Nature and Impact of DASD Skew. McNutt [20] probably provides the most thorough analysis of the DASD skew. The intent of his research, however, is not to reduce DASD skew but to predict its behavior. He defines a technique for generating skew profiles for use in capacity planning so that I/O loads for individual devices can be determined given an overall I/O load. His premise is that DASD skew exists in nearly all storage subsystems and must be considered when modeling systems because of its significant impact on performance. He identifies three major categories of data with distinctively different skew characteristics due to the methods used to manage the skew. These are 1) dynamic services such as paging, spooling, and scratch space, 2) fixed services, or "key system data", such as resident system volumes, catalog data, etc., and 3) user data for applications such as TSO, databases, etc. The dynamic services category consists of temporary data sets created and managed by the operating system as needed. The operating system dynamically determines file placement and skew problems with dynamic data sets are not generally a problem because of the highly dynamic nature of these data sets. Fixed services data sets are generally monitored by the support staff, and the data sets are moved to different volumes when problems occur. User data usually receives little skew control because of the volume and volatility of usage patterns.

In a later study, McNutt [21] also addresses the question of performance when loading data from lower capacity storage devices to higher capacity storage devices. He concludes that standard M/M/1 queueing models do not reflect the actual patterns of I/O demand for application data stored on DASD. Based on his earlier studies of skewed data discussed previously, he concludes that there will only be a marginal increase in the I/O rate on the busiest volume, which will cause only a small decrease in performance. This does not mean that skewed loads will not occur on higher capacity devices. In fact, McNutt points out that it is possible for two highly active data sets to end up on the same volume, a condition that would require the attention of a DASD tuning specialist.

Mungal [23] discusses the importance of skewed loads on the performance of I/O subsystems and the need to give careful consideration to the impact of skewed loads when configuring a storage subsystem. He mentions the use of access density as an "interesting way to view I/O actuator loading," which can be used to determine storage "pools" within the storage subsystem.

Friedman [11], in looking at DASD access patterns while studying caching opportunities, noted the presence of highly skewed access patterns in at least two of the subsystems on which he performed cache simulations. The first subsystem had two-thirds of the I/O activity performed by one-third of the devices, while the second subsystem had three-quarters of the I/O activity performed by one-third of the devices.

Duhl [10] analyzed the effects of the introduction of DASD devices with twice the storage capacity of the original DASD devices. Simulating the disk performance for 10 gigabytes of storage, with a skew of 70 percent of the I/O requests being handled by half the devices and 30 percent of the I/O requests being handled by the other half of the devices, and varying the I/O load from 50 to 200 I/O requests per second, both the single capacity and the double capacity devices showed increased response times with skewed loading versus an evenly balanced load. The higher loads showed a more significant difference with the double capacity devices being more severely impacted. This indicates that alleviating skew can improve response time performance, especially at higher I/O rates.

Brandwajn [7] makes the following statement regarding the importance of considering DASD skew in his development of a model of DASD Dynamic Reconnection: "Note also that measurements of actual DASD subsystems show that there is often a considerable imbalance in the load, i.e., rates of

I/O's and, possibly, other I/O characteristics, among strings of disks, as well as among devices of a single string. This can be the case when a small subset of drives accounts for much of the string activity ("dominant devices"), and could also be expected in strings mixing single and multiple capacity DASD's. Therefore, it is important to be able to accurately represent multipath DASD path reconnection configurations with imbalanced load."

Wilmot [34] evaluates the skewness of the access rates for all of the data sets in the overall storage subsystem, noting that "extreme skewness of file usage appears to be ubiquitous in the file systems we have so far examined." However, he does not look at the skew experienced among the different devices of the system. It is actually the skewness of data set access rates that leads to device skew problems. If all of the data were about the same size and had the same access rates, then access skew would not be a problem. However, because of these differences it is easy for one device to have a large number of active files while another device has none.

DASD Skew Reduction. The following provide a variety of suggestions for reducing DASD skew in a storage subsystem.

Piepmeier [29] describes a mathematical method for determining the proper distribution of I/O loads over multiple devices. Minimizing the "total response time" resulted in evenly distributed access loads for devices with like access capabilities. The total load is distributed proportionately across the sets of devices with different capacities. The proportion of the load managed by the higher capacity devices, however, is not equivalent to the ratio of the access speeds of the different type devices. Higher capacity devices are able to handle a load somewhat greater than the load determined from a simple ratio of the access speeds. Generalized examples illustrated that better performance is obtainable with access loads evenly distributed across all devices in accordance with their capacities.

Beretvas [6], in a general discussion of performance tuning problems and techniques within the IBM OS/VS2 MVS operating system, indicates that when I/O performance problems occur, the I/O load may be "incorrectly distributed. ... Bad data set placement in any operating system makes it impossible for the system to achieve its full potential. ... and may therefore require load balancing, which begins with the time honored task of data set placement. Data set placement is distribution of data sets such that no channels, control units, and I/O devices are excessively used." He continues with a description of how to determine where bottlenecks exist and then, among other things, discusses the possible need to spread "TSO user catalogues, spool data sets, user data bases, and scratch space... across volumes, control units, and channels to minimize contention and enhance availability."

Schardt [30], in a description of his approach to IBM MVS operating system tuning, indicates that when a device is dominating a control unit, it may be necessary to move some data sets to different devices with less active control units; that is, it may be necessary to distribute the load. This may also be necessary when excessive seek times are being caused by multiple active data sets residing on the same volume.

Surveying I/O optimization procedures and problems at the time, Smith [33] reiterates the recommendations of Berevtas [6] and Piepmeier [29] -- data sets that are used concurrently should be located on different volumes "to reduce congestion and improve access time."

Barkatski [3] states in his report of performance problems on a Sperry system that several system files with high access demands were stored on only two of the drives. It was proposed that the files be evenly distributed among the available disk units. The implementation of this recommendation plus others did improve both the mean response time and the response time variance, however, the improvements were not able to eliminate user complaints. The reasons are not clear and the author indicates that further studies are needed.

Singh et. Al. [32], discussing the performance management of MVS systems, indicate that the I/O system requires tuning and optimization to achieve its full potential. One of the important aspects of the tuning effort is to assure that the I/O load is reasonably balanced across all the devices within the system.

Beretvas [5] concludes as the result of modeling studies of both MVS S/370 and MVS X/A that it is "important to balance the load among actuators." These studies showed skewed loading has a very negative impact on channel utilization, with path utilizations dropping from forty percent to fifteen percent in the particular modeling conditions used.

Wong and Chanson [35] describe a general purpose software package called OPTIMAL that is designed to determine the optimal upgrade equipment that would improve the performance of a computer system. A first step in the upgrade optimization process is the balancing of loads across the I/O devices. By reducing the load on highly utilized devices, system bottlenecks may be eliminated and the need for upgrades may be eliminated.

Papy [26] provides a case study in which improvements to the load balance among the various storage devices of the system were essential in to resolve the performance problems being experienced. The actions taken included moving a highly active device to a different string, moving some high activity data sets to different devices, and breaking up some data sets into multiple data sets and then redistributing the new data sets onto different devices. All of these efforts were directed toward balancing the loads of the storage subsystem more evenly across all of the devices.

Papy [27] also addresses some of the general principles involved in resolving performance problems in the storage subsystem. He suggests that the principal techniques for tuning a storage subsystem are data set placement within a device, data set placement between devices, and volume placement between strings. He also indicates that determination of which volumes and data sets are problems is not the only critical element to the resolution of the problem. Another important requirement is to determine where a volume or data set is to be moved. All of these are directly related to the elimination of DASD skew.

Baker [2], in a review of the fundamentals of DASD tuning, includes in his recommended tuning procedures, the balancing of I/O loads between over-utilized and under-utilized devices. He also suggests this as means of resolving path contention in the subsystem.

Buzen and Shum [8], in their review of various trade-offs involved in I/O performance tuning, point out that DASD skew reduction is the most important tuning strategy.

Griffith [14], though not discussing the problems of skewed access loads directly, recognizes the problem in his discussion of combining loads from two or more smaller capacity devices to a single higher capacity device. He suggests combining high activity volumes with low activity volumes in order to "spread the activity more evenly" to prevent lengthy queues.

Space Utilization

Space utilization is another important consideration in the management of storage subsystems. The principal problem with space utilization is large amounts of unutilized space. [9,12,16] This does not create problems for the user, but it does indicate resources have been purchased that may not be necessary. For most people this is an unacceptable condition and should be avoided. Therefore, the storage subsystem manager must work toward efficient utilization of storage space.

When maximizing the utilization of space, however, there are some restrictions that must be considered. Of course, it is physically impossible to exceed the capacity of a storage device. The physical capacity of the device is an absolute limit. This is in contrast to the I/O capacity which can be exceeded because of the capability of storage subsystems to queue access requests. However, it is not feasible to fully utilize all of the available space. A certain amount of free space is required for temporary files and for future growth. Capacity planners determine the amount of free space required from their modeling of the expected loads that the storage subsystem must be able to handle. Levy describes a method for determining the amount of free space that should be available [17]. Tuning personnel must resolve the problem of how this free space should be allocated among the different storage devices.

Initially, the particular device on which the free space is located may not appear to be important. However, because the utilization of free space also has an associated access requirement, the access loads on the devices will be affected by the allocation of free space among the devices. As has already been discussed, system performance is improved by distributing the access load across the available devices. Therefore, to provide better access performance, distribution of free space is also important.

Another consideration affecting the distribution of free space in the subsystem is that some devices may have different access capacities. Therefore, depending on the nature of the typical access requirements for temporary data, providing free space on all of the different device types could improve performance by providing the access capability that best fits the requirement of the temporary data.

Storage Subsystem Tuning

It is apparent from the recommendations of others who have studied the problems of DASD skew and space utilization that steps must be taken to assure that the system continues to provide the space and access capabilities required by the users of the system. This necessitates some type of periodic tuning of the system to make adjustments for problems that have been detected in capacity, performance, or cost efficiency.

The challenge of tuning a storage subsystem is to provide sufficient storage space and access capacity as cost effectively as possible without creating unacceptable performance delays. This requires optimization of both the access and space capabilities of the subsystem. That is, both the system's storage capacity must be utilized fully to reduce the cost per megabyte of storage, and the system's access capacity must be utilized fully to reduce the cost per access without creating excessively long service times.

Optimizing these capabilities is difficult because optimization of one capability can oppose optimization of the other. For example, if a device has available storage space, placing more data on the device can increase the access demand on the device and possibly increase the I/O response time. Alternatively, if a device is experiencing excessive delays in handling I/O requests, moving data from the device to reduce the I/O demand also reduces the storage space utilization. This results in wasted storage space and an increased cost per megabyte to store the remaining data.

Tuning for I/O performance also has additional conflicting objectives. There are two principal DASD access performance measures that must be considered: 1)throughput, or the total number of accesses performed during a specified period, and 2)service time, the time required to perform an individual access request. Throughput optimization maximizes utilization of a device by reducing the device's idle time and by removing system inefficiencies that increase the time necessary to complete an I/O request. These include long seek times and RPS misses. Idle time is reduced by assuring that an access request is always available to a device. However, because the arrival of I/O requests is not constant, maintaining access demand requires the development of long queues. Long queues, however, cause unacceptably long I/O service times, leaving many users of the system dissatisfied even though a large volume of I/O is performed.

Optimization of service times attempts to reduce the time required to process an I/O request to a minimum. When waiting is eliminated, an I/O request is performed in the minimum amount of time. Optimization techniques to reduce I/O service times include elimination of I/O request queues, reduction of seek times within DASD devices, and reduction of RPS delays while waiting for available data paths. These assure that excessive I/O request queues do not develop, and they minimize wait times for the I/O service to be performed. Reducing service times, however, can result in less I/O being accomplished per unit of hardware and thus increase the amount of hardware required for the system. Therefore, because of these opposing objectives, the tuning process must attempt to reduce the response time for each I/O while at the same time maximizing the total quantity of I/O performed by each unit of hardware.

Olcott [25] divides current tuning methods into three categories: rules of thumb, application performance standards, and modeling.

The first category uses "Rules of Thumb" which specify maximum loads for various components of the system. The loads on each of these components are monitored and adjustments are made periodically when the loads on a component exceed these limits. It is assumed that these adjustments will provide acceptable performance for the system's users.

The second category sets standards for acceptable performance from the application or user perspective. If a particular application is experiencing unacceptable delays, a study is initiated to find both the source of the problem and the actions that should be taken to resolve the problem.

The third category seeks to determine a system configuration and proper loading levels for each device in the system based on a model of the system. The model provides system specific load limits that are then monitored. If a problem develops, the model is altered to reflect the actual operating conditions. Subsequently, alternatives for resolving the problem are evaluated. When a solution is decided upon, the actual system is reconfigured and possibly new rules are implemented.

Problem Statement

One of the major problems with these tuning techniques is that they rely on manual methods. In larger multi-user systems with numerous storage devices (sometimes numbering in the hundreds), several people may manage the storage subsystems. Gelb [12] describes some of the problems as follows: "The determination of which data sets are to be placed on which devices is often a difficult and time-consuming manual process. In addition, the placement is frequently performed after-the-fact. That is, data are moved because a problem has already occurred, which often creates new performance and contention problems. It is virtually impossible for manually driven or application-driven procedures to optimally place data in a timely manner." Major [19] also notes that "tuning tends to be costly in skilled people resources as well as in hardware. Tuning is an ongoing effort, and the results are often unstable. Human resources are growing more costly, whereas hardware is decreasing in cost. Therefore, tuning should not be an objective; rather, it should merely be an unavoidable temporary measure."

Because of the time and expense of manual tuning methods, only the most serious problems are given attention. And usually these are the problems dealing with performance. Other problems, such as under-utilization of resources, get far less attention because they do not generate immediate complaints. Thus, current methods frequently seek to optimize access capabilities at the expense of space utilization resulting in wasted storage space and extra costs.

Ideally, a computer system should, as Merrill [22] suggests, "dynamically manage" its own storage subsystems, freeing system managers from this time consuming and expensive task. In 1983, the GUIDE IBM user group suggested, among other things, that "productivity of support personnel must exceed storage growth rate," and "that the subsystem must be self-adjusting to a changing environment." [12] Of course, implicit in these suggestions is the need for automation of the storage subsystem tuning process.

Turning the tuning function over to the computer system is not a simple task, but efforts are being made to do it. IBM has, over the last few years, introduced System Managed Storage in their large computer systems in an effort to simplify the tasks of allocating storage space and dealing with inactive data. At the time a data set is created, the expected access requirement is specified by the user. Then the data set is located automatically on the device that has the capacity to handle the specified access requirement. Additionally, as data sets become inactive, they migrate to off-line storage with an automatic recall capability over an extended period of time. Of course, migrating inactive data to off-line storage has a tendency to increase the access load on a device which increases the potential for performance problems on the device. Also, System Managed Storage does not tune the system once a data set has been allocated. Adjustments are not made if the user did not define the access or space requirements properly or if the requirements change over time. [28]

The objective of this study is to define a method for automating the tuning of storage subsystems that minimizes DASD skew and, in turn, provides efficient utilization of the space and access capabilities of the subsystem. The method described significantly reduces the requirement for manual tuning of storage subsystems by maintaining the subsystem in a balanced condition; the I/O loads and free space are equally distributed across all devices according to their capacity. As subsystem loads change, the system automatically adjusts to the changes. DASD skew is controlled so that no manual intervention is required to eliminate the problems introduced by skewed I/O and space utilization within the system.

The tuning method proposed is based on the use of vector representations of storage subsystem capacities and loads to balance the loads across all of the devices in the storage subsystem. These vectors are referred to as storage factors. Storage factors are the vector representation of the concept of access density which was first defined by Hill in the IBM technical report *Access Density--A Data Storage Figure of Merit* [15]. Storage factors provide an effective means to conceptualize and correlate mathematically the space and access capacities of a set of storage devices with the space and access requirements of the data being stored. Theoretically, it is possible to balance a storage subsystem by equalizing the storage factor representing the load for each device with the storage factor representing the capacity of each device in the subsystem. By moving data sets to different devices, the difference between the load and capacity storage factors of each device is reduced as much as possible.

Although a statistical analysis of the effectiveness of using storage factors would have been preferable, a legitimate statistical analysis was not feasible for this study because data were not available from a statistically representative sample of large storage subsystems. Additionally, the volume of data necessary to complete a statistical analysis with the measurement tools currently available would have resulted in processing costs that were unacceptably high. Alternatively, a simulated tuning effort using actual data from a large storage subsystem is presented to demonstrate the use of the method and to explore any potential problems.

A detailed description of the proposed method follows. In chapter 2, storage factors are defined in detail. In chapter 3, the proposed tuning method is described in a step-by-step procedure. Next, a case study is presented in chapter 4, followed in chapter 5 by an analysis of the results obtained in the case study. Conclusions and suggestions for future study are discussed in chapter 6.

The terminology and examples refer primarily to IBM storage subsystem architecture and devices. This is because of IBM's dominance among large computer systems and the prevalence of information about these systems. However, the problems discussed here and solutions proposed apply to any storage subsystem with a large number of storage devices that experience variable access requirements among a large group of data storage devices.

CHAPTER II

STORAGE FACTORS

Access density space is described by Hill [15] as "the Cartesian coordinate space described by *volume of capacity* as the ordinate and *accesses per second* . . . as the abscissa," where *volume of capacity* is generally measured in megabytes of storage space and *accesses per second* is the number of times per second that a read or write operation is performed on a block of data. Within this coordinate system, the number of accesses per second relative to the amount of data stored is depicted by plotting the storage space along the abscissa and the accesses per unit of time along the ordinate. The ratio of the access rate to the space, which is the slope of the plot, is defined by Hill [15] as access density as shown in Equation 1.

$$\text{ACCESS DENSITY} = \frac{\text{ACCESSES PER SECOND}}{\text{MEGABYTES OF STORAGE}} \tag{1}$$

Hill later observed that a plot of this type is similar to a vector and that storage requirements could be represented using vectors. Based on this observation, he suggested that vector analysis may be a useful tool for automating the balancing of a storage subsystem, which is the thesis of this paper.

Meaningful vector analysis requires that the dimensional units be consistent. Since storage space and access requirements are measured in different units, it is not possible to vectorize the storage characteristics in terms of their actual units. However, by scaling the actual units it is possible to convert to common units that allow the storage requirements to be represented as vectors. The access scale factor must be proportional to the accesses per second, and the space scale factor must be proportional to the storage space. For example, an access scale factor may be 2.00 inches/access/second, and a space scale factor may be 0.002 inches/megabyte.

Once the storage characteristics have been scaled, a space vector with a magnitude equal to the scaled megabytes of storage can be plotted along the abscissa, and an access vector with a magnitude equal to the scaled accesses per second can be plotted along the ordinate. The sum of the space vector and the access vector defines a third vector known as the storage factor vector. This is shown in Figure 1.

The direction of the storage factor vector is related to the access density as shown in Equation 2. Therefore, access density is also used to refer to the direction of the storage factor vector.

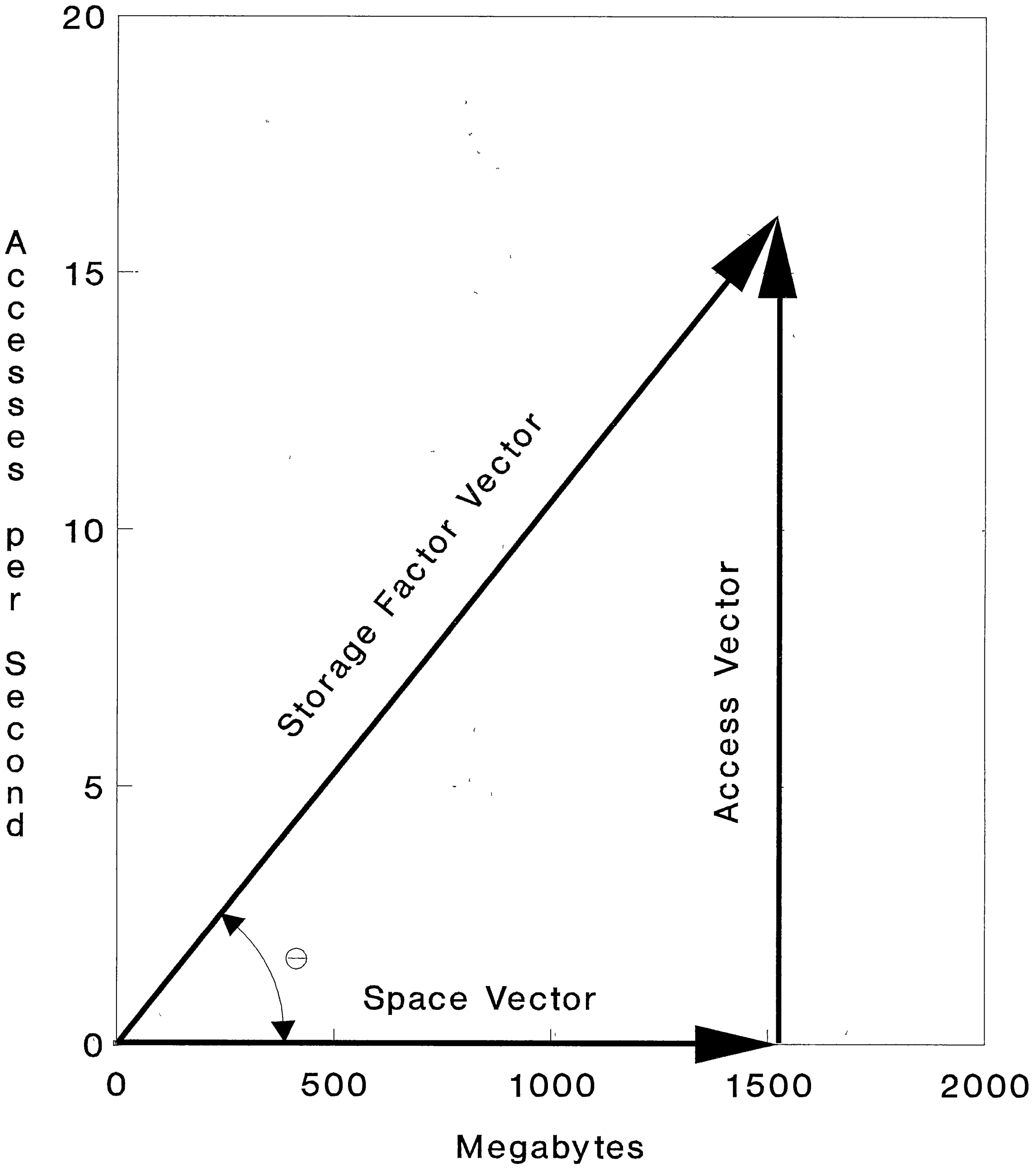


Figure 1. Storage Factor Vector

$$\text{Tan}\,\Theta = \text{ACCESS DENSITY} * \frac{SF_{\text{accesses}}}{SF_{\text{space}}} = \frac{\text{ACC/SEC} * SF_{\text{accesses}}}{\text{MB} * SF_{\text{space}}} \quad (2)$$

The magnitude of the storage factor vector is defined as the storage load. Using vector algebra, it is calculated as shown in Equation 3 [31].

$$\text{STORAGE LOAD} = \sqrt{(\text{MB} * SF_{\text{space}})^2 + (\text{ACC/SEC} * SF_{\text{accesses}})^2} \quad (3)$$

Since the magnitude of a storage factor vector is dependent on the values of the space scale factor and the access scale factor, the selection of storage factors can make a significant difference when comparing storage loads. Selection of scaling factors is discussed later under Determining Scaled Factors for Balancing in Chapter III.

Storage factors quantitatively express the relationship between the space and access requirements of data storage. They allow access and space problems to be analyzed and resolved simultaneously while giving each problem equal consideration. Since storage factors are vectors, vector mathematics can be used to analyze a storage subsystem's condition and verify the suitability of possible solutions to loading and capacity problems. Also, vectors are well suited to graphical presentation; therefore, storage factors, when plotted, can provide an easily understood depiction of a storage subsystem's condition.

Hardware vs. Data Storage Factors

Storage factors can be used to characterize both storage hardware and stored data. Hardware includes physical storage devices such as disk drives (floppy disks, hard disks, optical disks, etc.), tape drives, and solid state devices (caches, RAM, ROM, etc.). Stored data includes any collection of related pieces of information identified as a single entity. These data collections are referred to as a file or data set. In this paper, the term data set is used. However, a stored data entity also can be a collection of all the data sets stored on a device or in a storage subsystem. Or, if a data set is stored on more than one device, the stored data entity could be only that portion of the data set stored on the device being analyzed.

The storage factor is useful for tuning storage subsystems because a vector can simultaneously represent both the access and the space requirements of the storage problem, both for stored data as well as for storage hardware. By correlating the storage factors of the hardware and the data, utilization of the storage and access capabilities can be improved while maintaining a reasonable assurance that access bottlenecks will not occur. Determining the storage factor for data sets and for storage devices is described in the next two sections.

Data Set Storage Factors

The storage factor of a data set is defined by the amount of data stored in the data set and the access rate that must be provided to the data set.

The amount of data stored in a data set is the storage space allocated for the data set. The unit of measure is usually megabytes, but gigabytes can be used also.

The calculation of a data set's access rate is shown in Equation 4.

$$\text{ACCESSES PER SEC} = \frac{\text{READ/WRITE REQUESTS}}{\text{TIME PERIOD}} \tag{4}$$

The READ/WRITE REQUESTS is the predicted number of times that a block of data will be read from or written to the data set during a specified TIME PERIOD. This prediction is made by monitoring the data set activity and recording the I/O requests during the TIME PERIOD. On large IBM systems, the SMF monitoring system is available for this purpose. However, this type of monitoring generates very large volumes of data that must be condensed and analyzed to be useful. It would be preferable to incorporate in the operating system a method of continuously capturing, analyzing, and condensing this data and storing it in the Volume Table of Contents (VTOC) with the other data set information.

A conservative algorithm for determining the characteristic access rate of a data set, which could be incorporated into the operating system, is shown in Figure 2. An EXCP refers to the IBM MVS command issued to initiate an I/O request.

By defining a time period for evaluating access skew, such as weekly, monthly, bimonthly, or quarterly, it is possible to determine a data set with a declining access rate. This is necessary to prevent the data from becoming skewed to the high side over time. Thus, at some regular interval of time designated for monitoring access skew, the characteristic EXCP count is updated with the maximum EXCP count since the last time period.

By storing the characteristic access rate for a data set, it would always be readily available. This would allow the system to rebalance the storage loads regularly, as time and resources are available, possibly on a daily or weekly basis. Of course, this approach assumes that activity in the future will be similar to the activity in the past. This assumption, however, may not be valid. Therefore, if conditions are known that will modify the access requirements of the data set, adjustments should be made to the predicted access rates.

The TIME PERIOD is the length of time used to define the data set access characteristics. Access requests vary not only from device to device but also in time. [20] Some periods of the day have much higher activity rates than

```
lastskew = last skew check period
currskew = current skew check period
counthr = hour for which EXCP's were counted
currhr = current hour
excpct = EXCP count
maxexcp(counthr) = maximum EXCP count for
                   the data set for the
                   count hour during the
                   latest skew period
characct(counthr) = characteristic EXCP
                    count for the count hour
if currhr > counthr then
     if excpct > maxexcp(counthr) then
        maxexcp(counthr) = excpct
        if excpct > characct(counthr) then
          characct(counthr) = excpct
        endif
     endif
     if currskew > lastskew then
        characct(counthr) = max-
excp(counthr)
        maxexcp(counthr) = 0
        lastskew = currskew
     endif
     excpct = 0
     counthr = currhr
endif
excpct = excpct + 1
```

Figure 2. An Algorithm to Calculate the Characteristic Access Rate

other periods. [2] The busiest times of the day are usually when problems resulting from DASD skew are encountered. Therefore, since the intent of tuning the subsystem is to eliminate these problems and prevent future problems, tuning should be performed for the high access periods. Tuning for other time periods will have little or no benefit if the accesses fall well below the device's capabilities. Typically, the high access periods occur sometime during the morning and then again in the afternoon on regular business days [2,26].

For this study, one hour periods are used. Over several days or weeks, analyzing data accesses within one hour periods allows meaningful access patterns to be determined. These access patterns are then compared and tuning is performed for those periods with the highest I/O activity since they will have the most significant impact on system performance.

When comparing data set storage factors, larger magnitude storage factors indicate larger amounts of data are being stored, more accesses per second are required, or both. A data set with a large access density indicates that the data being stored is accessed at a high rate relative to the amount of data stored. A small access density indicates that a large amount of data is stored in the data set relative to the number of accesses to the data.

Table 1 lists some example data sets and their characteristics. These are plotted in Figure 3.

Device Storage Factors

The storage factor of a storage device (hardware)

TABLE 1

DATA SET CHARACTERISTICS

Data Set Name	Read/Write Requests	Time Period (Seconds)	Average Accesses /Second	Megabytes Stored
DATA SET 1	151.00	3600.00	0.042	7.12
DATA SET 2	80.80	3600.00	0.022	3.75
DATA SET 3	226.00	3600.00	0.063	9.49
DATA SET 4	191.00	3600.00	0.053	3.75
DATA SET 5	266.00	3600.00	0.074	3.75
DATA SET 6	403.00	3600.00	0.112	3.84
DATA SET 7	207.00	3600.00	0.058	1.80

relates a device's ability to store data and its ability to access data. A device's storage factor is defined by the amount of data it can store (usually specified in megabytes) and the number of blocks of data it can read from or write to the device in one second--the sustainable access rate.

The maximum access rate that can be sustained by a device is calculated in Equation 5.

$$\text{ACCESSES PER SEC} = \frac{1}{\text{SERVICE TIME}} * \text{QUEUEING FACTOR} \qquad (5)$$

There are many factors that determine the access rate

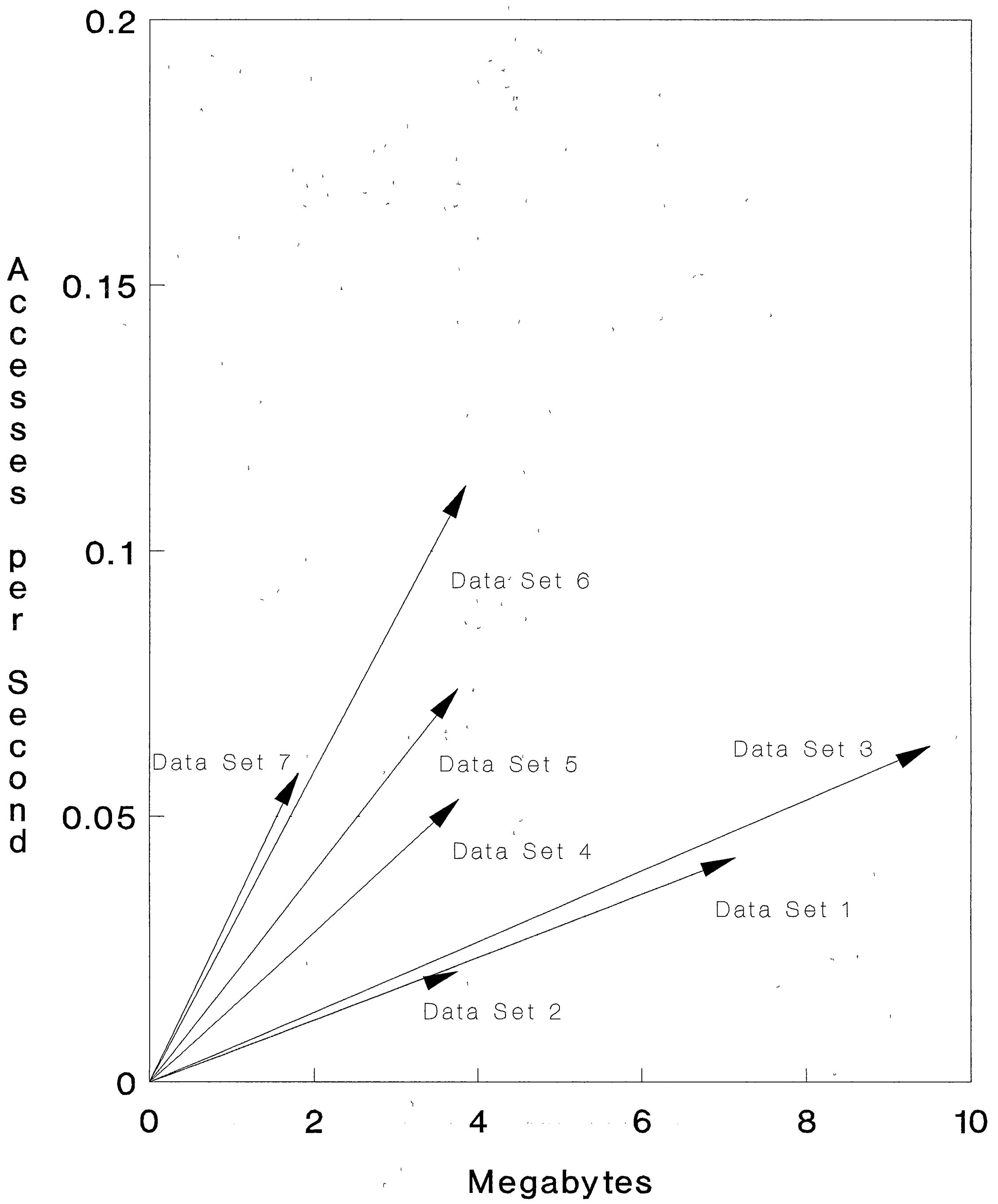


Figure 3. Example Data Set Storage Factors

that can be sustained by a rotating disk device. These vary from subsystem to subsystem and are dependent on the exact configuration of the subsystem. Beretvas [4] and Gray [13] provide a thorough discussion of these factors.

SERVICE TIME is the time required for a storage device to process a request to either read or write a block of data. A simplified calculation of the SERVICE TIME for a rotating disk device is shown in Equation 6.

$$\text{SERVICE TIME} = \text{AVG SEEK TIME} + \text{AVG LATENCY TIME} \\ + \text{AVG DATA TRANSFER TIME} \tag{6}$$

where

$$\text{AVG DATA TRANSFER TIME} = \frac{1}{\text{TRANSFER RATE}} * \text{AVG BLOCK SIZE}$$

The AVG SEEK TIME is the average number of milliseconds it takes to move the read/write head to the track or cylinder where the data to be retrieved is stored. The AVG LATENCY TIME is the average number of milliseconds required for the proper block of data to rotate under the read/write head. The AVG DATA TRANSFER TIME is the number of milliseconds required to move an average size block of data to or from the device once the read/write head is positioned properly. A device's average block size is the average of the block sizes of all the data sets stored on the device. A larger block size will increase the transfer time per access because more data is transferred during each access. Although the average block size for a device may vary somewhat, the difference it causes in the AVG DATA TRANSFER TIME is usually insignificant when compared to the seek time or latency time. If desired, the average block size can be easily adjusted since it is readily available on the VTOC.

Of course, the average seek time also may vary depending on the locality of reference and the cylinder placement of more active data sets. This can be alleviated to some extent by placing higher access density data sets on those cylinders that minimize read/write head movements.

The QUEUEING FACTOR is required because input/output requests are not received by the device at a constant rate. This is because the CPU can process data at a much faster rate than an I/O device. Therefore, an I/O device often receives access requests much faster than its capacity to handle them. When this occurs, a queue of I/O requests is generated. At other times, the I/O device receives requests at a rate well below its capacity. During this lower activity period, the requests in the queue can be processed until the queue is empty.

The queueing theory steady state equation for the wait time in a queue for an M/M/1 queueing system [1] shows the relationship between the arrival rate and the service rate. If it is assumed that an I/O device is an M/M/1 queueing system (arrival rate and service rate follow a Poisson distribution) and that an acceptable wait time for service is 50 percent of the service time, then by the wait time equation just mentioned, the average service rate of a device should be approximately 30 percent of its maximum service rate. Otherwise, excessive delays will occur because of the time required for the server to service a request.

McNutt [21] has suggested that the M/M/1 model may not be appropriate for storage devices that are not being used by a large number of users with relatively high access demands because the requests for service probably do not follow a Poisson arrival process. He found that for TSO and large databases, no degradation in service occurred for service request rates up to sixty percent of the maximum and, in some instances, up to eighty or ninety percent of the maximum service rates. Thus, if it is found that the access requests are not arriving according to the predictions of the Poisson arrival process, it would be feasible to adjust the queueing factor appropriately.

Example Device Calculations

The calculations to determine an allowable access rate for the 3380 Model AK4/BK4 are illustrated in the following equations. First the Service Time is calculated in Equation 7. Technical specifications were obtained from

IBM literature [36]. Then the Accesses per Second is calculated in Equation 8.

$$\text{SERVICE TIME} = \text{AVG SEEK TIME} + \text{AVG LATENCY TIME} + \text{DATA TRANSFER TIME} \tag{7}$$

$$\begin{aligned}\text{SERVICE TIME} &= 16\text{x}10^{-3}\text{secs/access} + 8.3\text{x}10^{-3}\text{secs/access} \\ &\quad +(1\text{sec}/3\text{x}10^{6}*6500\text{bytes/access}) \\ &= 26.47\text{x}10^{-3}\text{secs/access}\end{aligned}$$

$$\begin{aligned}\text{ACCESSES/SEC} &= \frac{1}{\text{SERVICE TIME}} * \text{QUEUEING FACTOR} \\ &= \frac{1}{26.47\text{x}10^{-3}\text{secs/access}} * .3 \\ &= 37.77\text{accesses/sec} * .3 \\ &= 11.33\text{accesses/sec}\end{aligned} \tag{8}$$

The MEGABYTES OF STORAGE for a device is the maximum number of megabytes to be stored on the device. This will not be the maximum physically possible to store on the device because some free space is required to allow for data set growth and temporary data sets. Typically, the desirable maximum allowable storage size is eighty-five to ninety percent of the physical capacity of the device. Of course, it is possible to characterize temporary storage demands just as permanent data sets are characterized.

When comparing two devices, the device with the larger storage load has more capability for accessing data, storing data, or both. The device with the larger access density has a greater capacity to retrieve data relative to the amount of data stored than the device with the smaller access density.

As an example, Table 2 provides a comparison of the characteristics of various models of IBM Direct Access Storage Devices. Figure 4 portrays the storage factor for each device [36].

TABLE 2

EXAMPLE STORAGE DEVICE CHARACTERISTICS

Device	Model Number	Sustainable Access Rate (Accesses/sec)	Storage MB
3380	AK4/BK4	11.34	1606.500
3380	AJ4/BJ4	13.35	535.500
3380	A04/AA4/B04	11.34	535.500
3380	AD4/BD4	11.78	535.500
3380	AE4/BE4	10.92	1071.000
3350		8.19	269.875

After comparing the access densities of these devices, it is clear that the 3380 Model AK4/BK4 has the least access capacity relative to the amount of data stored,

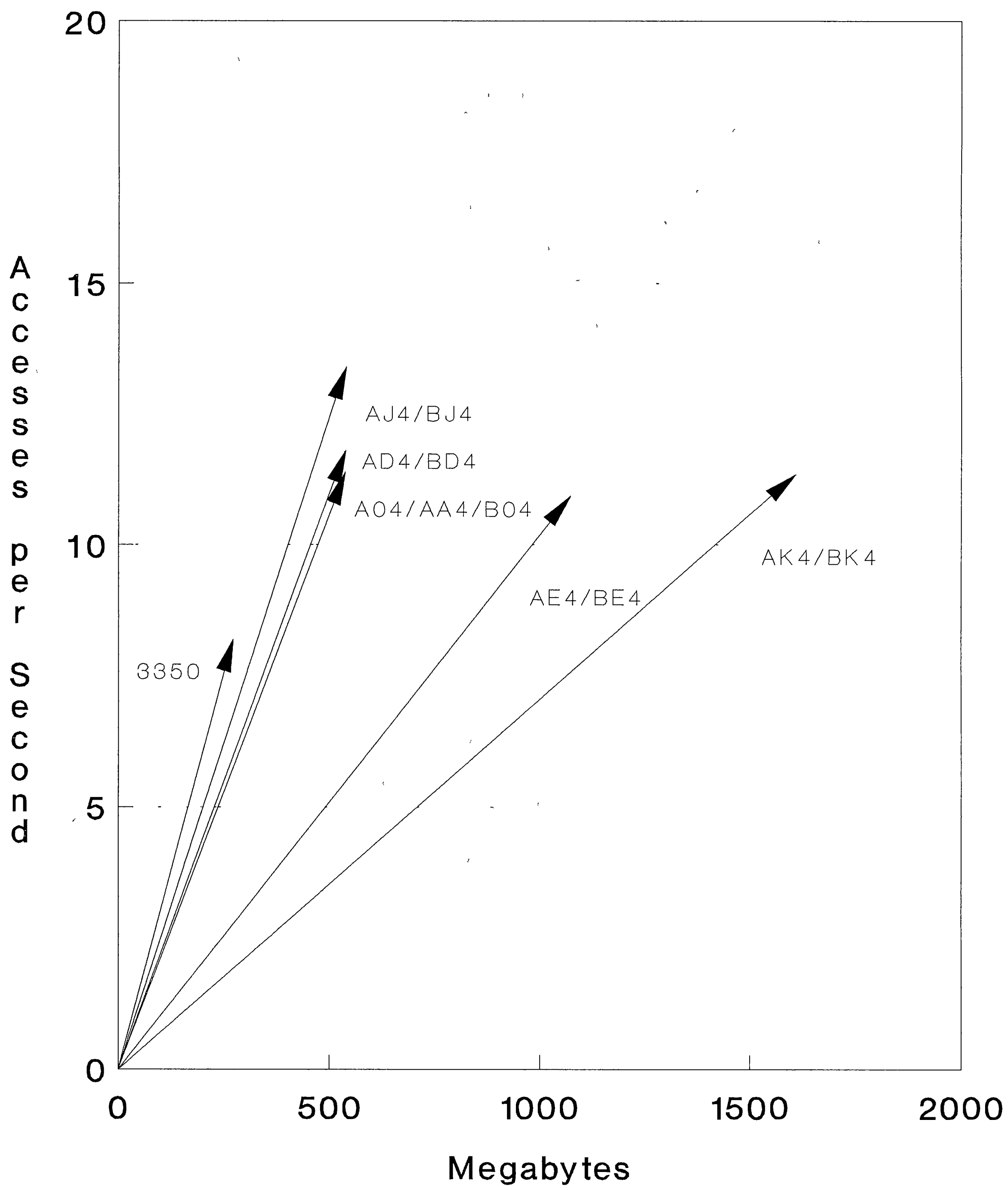


Figure 4. Example Device Storage Factors

while the 3350 Model has the largest access capacity relative to the amount of data stored. This means that a 3350 device is suited to storing higher access density data sets better than a 3380 device because the access density of a 3350 device is higher than the access density of a 3380 device. On the other hand, a 3380 device would be suited to storing a large amount of data with low access requirements better than would a 3350 device.

Caching Effects

It is beyond the scope of this paper to discuss I/O caches in detail. However, since caching is prevalent in large storage subsystems today, their impact on the storage devices that they service is discussed briefly. An I/O cache is in reality a storage device with different storage characteristics than the storage devices they service. It generally has a very high access capacity and a much smaller storage space. That is, its access density is much higher. It also does not provide permanent storage.

Cache is non-permanent storage. Therefore, it must be considered separately from permanent storage since any cached data also must be provided storage space and access capacity on a permanent storage device. However, since data is moved from permanent storage to cache when the data set is accessed, the cache can reduce the amount of I/O that the permanent storage device must support. This reduces the access density of the data set with respect to the permanent storage device. Since the access density affects how the system manages a data set, a cache can affect the placement of the data set on the permanent storage devices. In a non-cached system the same data set with the same access demand can have a completely different access requirement at the permanent storage device level. Therefore, it is important that access demand be measured at the queueing level of the device that is being analyzed.

Aggregate Storage Factors

An aggregate storage factor is the characteristic of a set of data sets or devices when viewed as a whole. Thus, for data, the data set divisions are ignored and all the data is viewed as one entity. Usually the data sets include all the data sets on a device or all the data sets in a subsystem. For hardware, the physical device separations are ignored and all device capabilities are considered as one device. Using vector algebra, the calculation of the aggregate storage factor is shown in Equations 9 and 10 where n equals the number devices or data sets.

$$\text{AGGREGATE ACCESS DENSITY} = \frac{\sum_{i=1}^{n} \text{ACC/SEC}_i * \text{SF}_{\text{accesses}}}{\sum_{i=1}^{n} \text{MB}_i * \text{SF}_{\text{space}}} \tag{9}$$

$$\text{AGGREGATE STORAGE LOAD} = \sqrt{\left(\sum_{\iota=1}^{n} \text{ACC/SEC}_{\iota} * \text{SF}_{\text{accesses}}\right)^2 + \left(\sum_{\iota=1}^{n} \text{MB}_{\iota} * \text{SF}_{\text{space}}\right)^2} \tag{10}$$

A comparison of the aggregate device access density and the aggregate data access density will show any inefficiencies in the utilization of the storage subsystem. If the aggregate data access density is significantly different from the aggregate device access density of the devices, then the devices are not well suited to storing the data. If the aggregate data access density is considerably larger than the aggregate device access density, there must be a large amount of unutilized storage space if the access capabilities of the devices are not exceeded. An example is shown in Figure 5.

If the device storage capacity is being utilized fully, then the device access capabilities are being exceeded, and a bottleneck is created. This is illustrated in Figure 6. Of course, other conditions can exist, but under no circumstances can the storage subsystem be operated without inefficiencies in either the access rates, the quantity of stored data, or both.

Residual Storage Factors

The residual storage factor is the difference between the aggregate device storage factor and the aggregate data storage factor as shown in Equation 11.

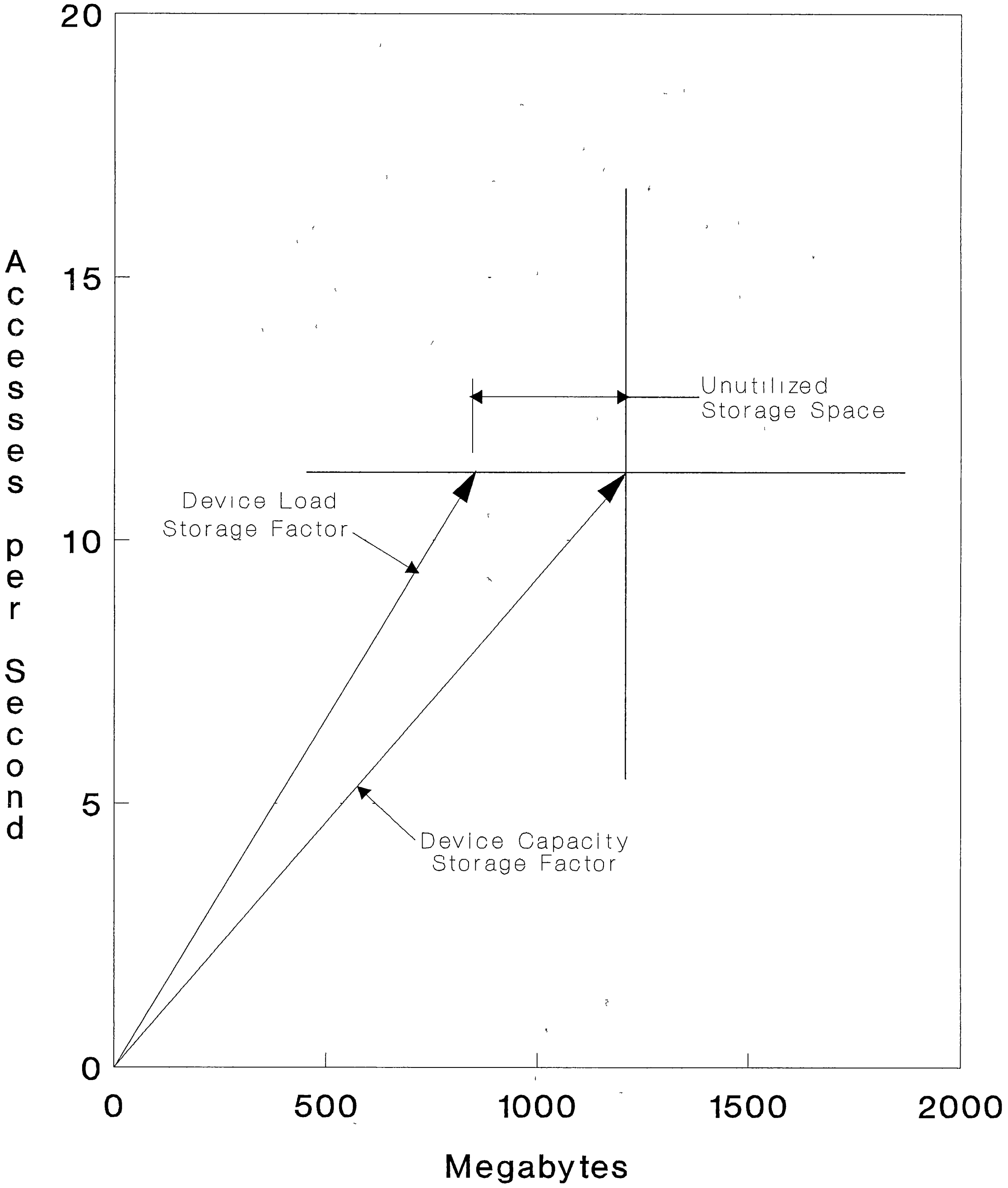


Figure 5. Unutilized Storage Space

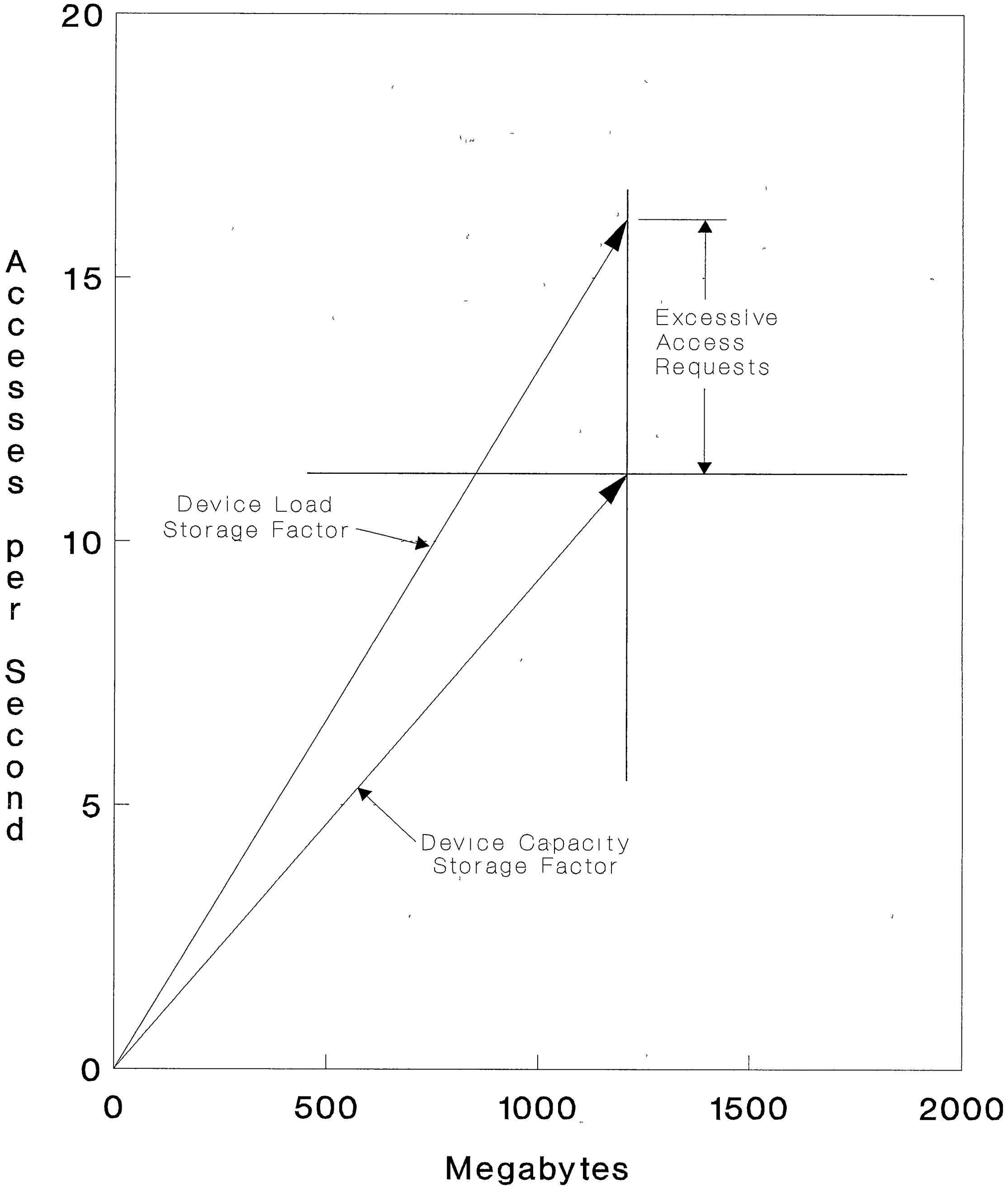


Figure 6. Excessive Accesses

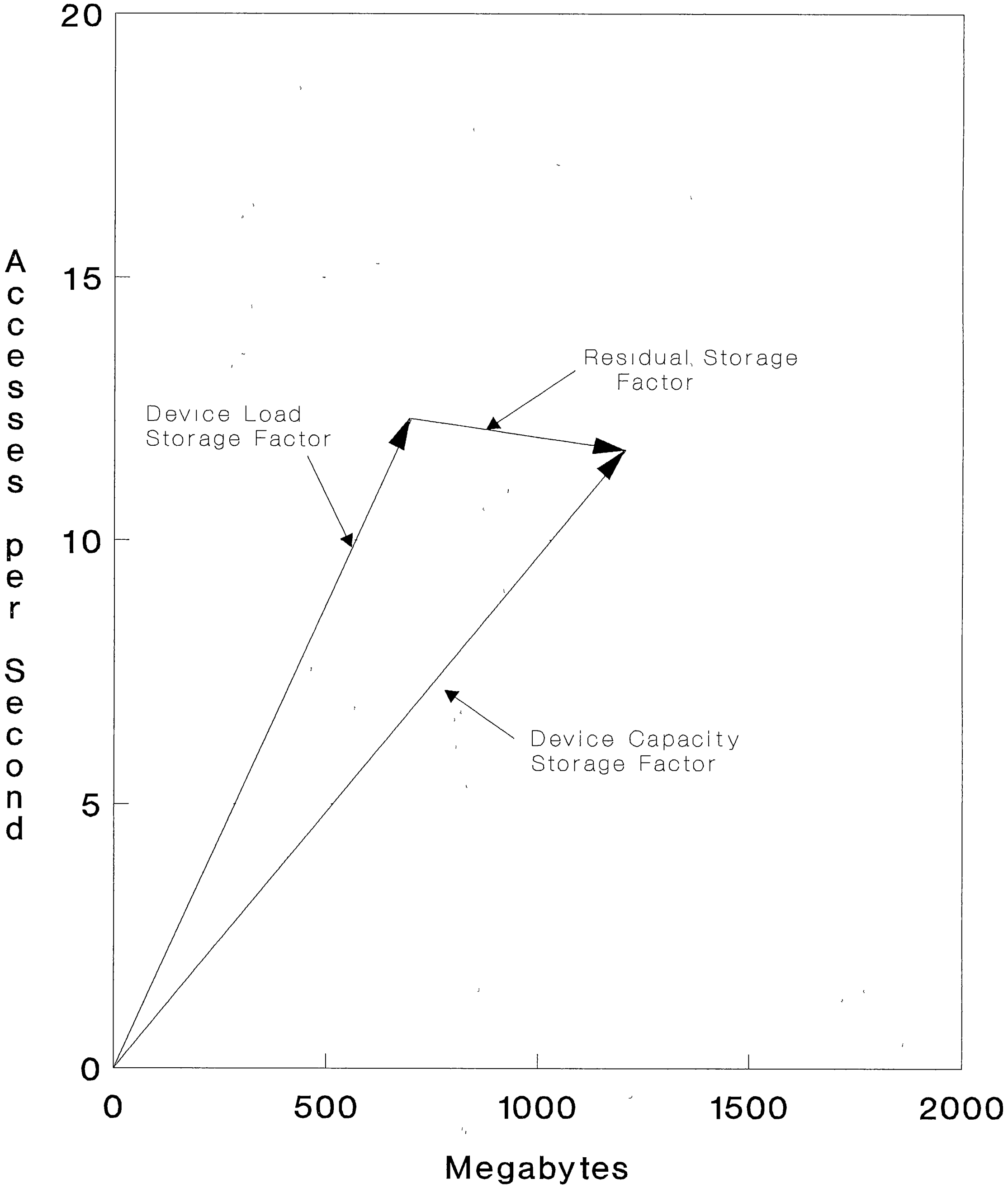


Figure 7. Residual Storage Factor

$$\text{RESIDUAL STORAGE FACTOR} = \text{DEVICE STORAGE FACTOR} - \text{DATA STORAGE FACTOR} \tag{11}$$

In reality, the residual storage factor defines a virtual device consisting of the unutilized storage capabilities of the storage device. Figure 7 shows a residual storage factor.

The residual storage factor shows the condition of the storage subsystem being analyzed. The larger the magnitude of the residual storage factor, the more out of balance is the system. A zero magnitude residual storage factor indicates a perfectly balanced system; the data storage requirements equal the data storage capabilities. This is the most economic condition, and therefore, it defines the desired result of most tuning efforts.

CHAPTER III

TUNING PROCEDURES

As discussed previously, in the ideal storage subsystem the storage and access capacities of the subsystem are utilized fully without degrading its performance. Because conditions frequently change within a subsystem, periodic tuning is necessary to maintain balanced utilization of the subsystem's resources and thus prevent performance problems resulting from an out-of-balance condition, or DASD skew.

Tuning a storage subsystem first requires verification that the storage subsystem has the capacities necessary to handle the storage requirements of the system. Next, the space and the access loads on each of the storage devices within the subsystem must be balanced with the device's capacities. Balancing the loads is accomplished by moving an appropriate group of data sets from those storage devices that are overloaded to those devices that are under-utilized. The tuning procedures developed during this study are summarized in the following outline of the balancing algorithm. Each step is then explained in detail in the following sections.

Outline of the Balancing Algorithm

Step 1: Calculation of storage factors

The following storage factors are determined:

Data set storage factors by time period
Device storage factors
Device load storage factors by time period
Subsystem capacity storage factor
Subsystem load storage factor by time period

Step 2: Selection of the critical time period

This is the time period with the greatest magnitude Subsystem Load Storage Factor.

Step 3: Evaluation of the hardware compatibility

If the subsystem capacity storage factor and the subsystem load storage factor are significantly different in magnitude or direction, a change in hardware configuration may be required.

Step 4: Determination of the virtual device storage factors

These are the desirable balanced loads for each device based on a weighted average of the total loads.

Step 5: Determine the scale factors for balancing, and scale the appropriate storage factors

Select a scale which equalizes the magnitudes of the components of the virtual device storage factor. Scale the virtual device storage factor, the device load storage factors, and the data set storage factors.

Step 6: Calculate the residual storage factor for each device.

Step 7: Balancing the loads (moving data sets)

Step 7A: Determination of a data set to move

Determine the device with the largest magnitude residual storage factor.

```
Condition 1

If the access load < virtual access capacity
and the space load < virtual space capacity
then
   No data sets are moved--access and space
   capacity remains on the device

Else Condition 2

If the access load < virtual access capacity
and the space load > virtual space capacity
then
   The data set with the lowest access density
   is selected

Else Condition 3

If the access load > virtual access capacity
and the space load > virtual space capacity
then
   The data set with an access density closest
   to the residual access density is selected

Else Condition 4

If the access load > virtual access capacity
and the space load < virtual space capacity
then
   The data set with the largest access density
   is selected
```

Step 7B: A check is made that removal of the data set will reduce the residual storage load.

If removing the data set will increase the residual storage load, the move is canceled and the next data set is selected.

Step 7C: Determination of the device to receive the moved data set.

The device with the residual access density closest to the data set access density is determined.

If adding the data set increases the magnitude of the residual storage factor, the device with the next closest residual access density is tried. This is repeated until a device is found to receive the data set. If no device is found, the move is cancelled, and the next data set is tried.

Step 7D: Move the data set.

Step 7E: Goto Step 7A and repeat until all devices are balanced or no more data sets can be moved.

Calculation of the Storage Factors

The first step in tuning a storage subsystem is to calculate the following storage factors: the data set storage factors, the device storage factors, the device load storage factors, the subsystem capacity storage factor, and the subsystem load storage factor. They are described briefly below.

The data set storage factors are defined by the characteristic access requirement by time period and the allocated storage space of each data set. This information should be contained in the VTOC. If it is not, it must be calculated from monitoring data. A data set storage factor must be determined for each data set in the subsystem for each time period to be analyzed.

The device storage factors are defined by the maximum sustainable access capacity and the storage space capacity available on each device, less any required space for future growth and temporary data sets. A device storage factor must be determined for each device in the subsystem.

The device load storage factor is the aggregate storage factor of all the data sets stored on a single device. A device load storage factor must be determined for each device in the subsystem for each time period.

The subsystem device storage factor is the aggregate storage factor of all the devices in the subsystem.

The subsystem load storage factor is the aggregate storage factor of all the data sets stored in the subsystem. A subsystem load storage factor must be determined for each time period.

Selection of the Critical Time Period

The second step in the tuning procedure is to determine the time period that has the greatest demand for resources. The critical time period is the hour with the largest system data storage factor access density, or the hour with the largest accesses per second requirement. The space requirement is not a factor in determining the critical time period because the quantity of data stored remains constant through all the time periods.

Evaluation of the Hardware Requirements

After determining the critical time period, the subsystem is evaluated as a whole; that is, total capacities are compared to total demands for the critical time period. This is accomplished by plotting the system data storage factor versus the system device storage factor for the critical time period. By comparing these two storage factors, it can be determined whether modifications to the number or type of hardware devices are necessary in order to provide the required storage capabilities economically.

If there is unutilized storage and access capacity equal to the storage and access capacities of one or more of the devices in the subsystem, then it is possible to remove one or more devices from the subsystem without affecting its performance. If either the storage capacity or the access capacity, but not both, is under-utilized, the existing storage devices may need to be replaced with devices that have a storage factor similar to the storage factor of the data. This should allow better utilization of the devices.

Acquiring other equipment also may be necessary if access or space demands exceed the capacities of the present devices. If the total access demand exceeds the total access capacity of all the devices, there probably are performance problems in the system. If there is not enough free space for growth and temporary data sets, then jobs requiring additional storage space will fail because of out-of-space errors. If either of these conditions exists, new storage capacity must be acquired. Again, only devices with storage factors as close to the device load storage factors as possible should be obtained. This assures that unnecessary capabilities are not purchased.

Determining the Virtual Device Storage Factors

Once it has been determined that the existing equipment meets the current and projected future data storage requirements, the next step is to define the storage loads that each device should handle. Of course, since the object of tuning the storage subsystem is to assure that the access and space requirements of the data are matched as closely as possible to the capacities of the device on which the data is stored, it may seem logical to use the device storage factors as the tuning objective. Nevertheless, because the total storage loads rarely exactly match the total subsystem capacities, it is preferable to use a weighted distribution of the total storage load as the desired balanced load rather than the actual device capacities. The weights are based on the capacities of the devices. In this way, all the devices carry a load proportional to their capacities.

For example, if the total demand is for 1200 megabytes of data and 30 access per second, Table 3 illustrates the calculated distribution of the load for balancing the subsystem. Thus, devices 1 and 2 would be expected to store 67 percent more data and provide 25 percent more accesses than devices 3 and 4 because devices 1 and 2 have 67 percent more space capacity and 25 percent more access capacity than devices 3 and 4.

Distributing the loads in this manner can be viewed as defining a virtual device, then attempting to maximize the loads on each virtual device during the balancing process. The storage factor associated with each of these virtual devices is referred to as the virtual device storage factor.

TABLE 3

EXAMPLE -- DESIRABLE DEVICE LOADING

Device	Megabytes Capacity	Accesses /Sec. Capacity	Weighted Load Megabytes	Weighted Average Acc./Sec.
Device 1	500	10	375	8.33
Device 2	500	10	375	8.33
Device 3	300	8	225	6.67
Device 4	300	8	225	6.67
Total	1600	36	1200	30.00

Determining Scale Factors for Balancing

The balancing procedure should give highest priority to the condition that is most out of balance. Thus, when balancing a device with an access requirement that exceeds the desirable by 25 percent and a space requirement that exceeds the desirable by 10 percent, a higher priority should be given to reducing the access requirement over reducing the space utilization.

To assure that this prioritization occurs, the storage factors must be scaled appropriately. For example, a device may have a desirable load of 1500 megabytes of storage space and 15 accesses per second. If 1700 megabytes of data are actually stored on the device and 18 accesses per second are required, a vector analysis of the difference between the desirable and the actual loads, using scale factors of one, results in a vector with 200 units for the space component and 3 units for the accesses per second component. The magnitude of this vector is almost completely the result of the space component. Consequently, the balancing effort would first try to remove the excess storage load, even though the access requirement is 20 percent out-of-balance, and the storage requirement is only 13 percent out-of-balance.

To overcome this problem, a scale that plots the subsystem device storage factor at 45 degrees is necessary. That is, the access rate and space components of the subsystem storage capacity vector should have equal magnitudes. This will give equal consideration to the space and access characteristics during the balancing effort. Thus, using the previous example, if the megabytes are scaled by a factor of 0.01, then the 1700 megabytes load becomes 17 and the accesses per second remains at 18. Now, when the vector representing the difference between the actual and desirable loads is calculated, the magnitude of the access component is 3 and the magnitude of the space component is only 2. In this case the access component is more critical, as it should be since the access component is more out-of-balance.

Once a scale is selected, the virtual device storage factor, the device load storage factors, and each of the data set storage factors are rescaled accordingly.

Balancing the Loads (Moving Data Sets)

The next step is to balance the loads on the virtual storage devices. This is achieved by seeking to equalize the device load storage factor and the virtual device storage factor for each of the devices in the subsystem. When both storage factors are equal for a device, the demand on the device is balanced with the device's capabilities. This assures that the device's capabilities are fully utilized without creating performance delays. If the data and device storage factors are not equal, then the storage device is being under-utilized, or a potential access bottleneck exists for the device.

To determine whether the loads on a set of storage devices require adjustment, the device load storage factor for each device is calculated and compared to the related virtual device storage factor. For devices that are not properly balanced, data must be moved from the overloaded devices to the under-utilized devices. Improperly loaded devices are recognizable immediately when the device load storage factors are plotted as shown in Figure 8. Devices

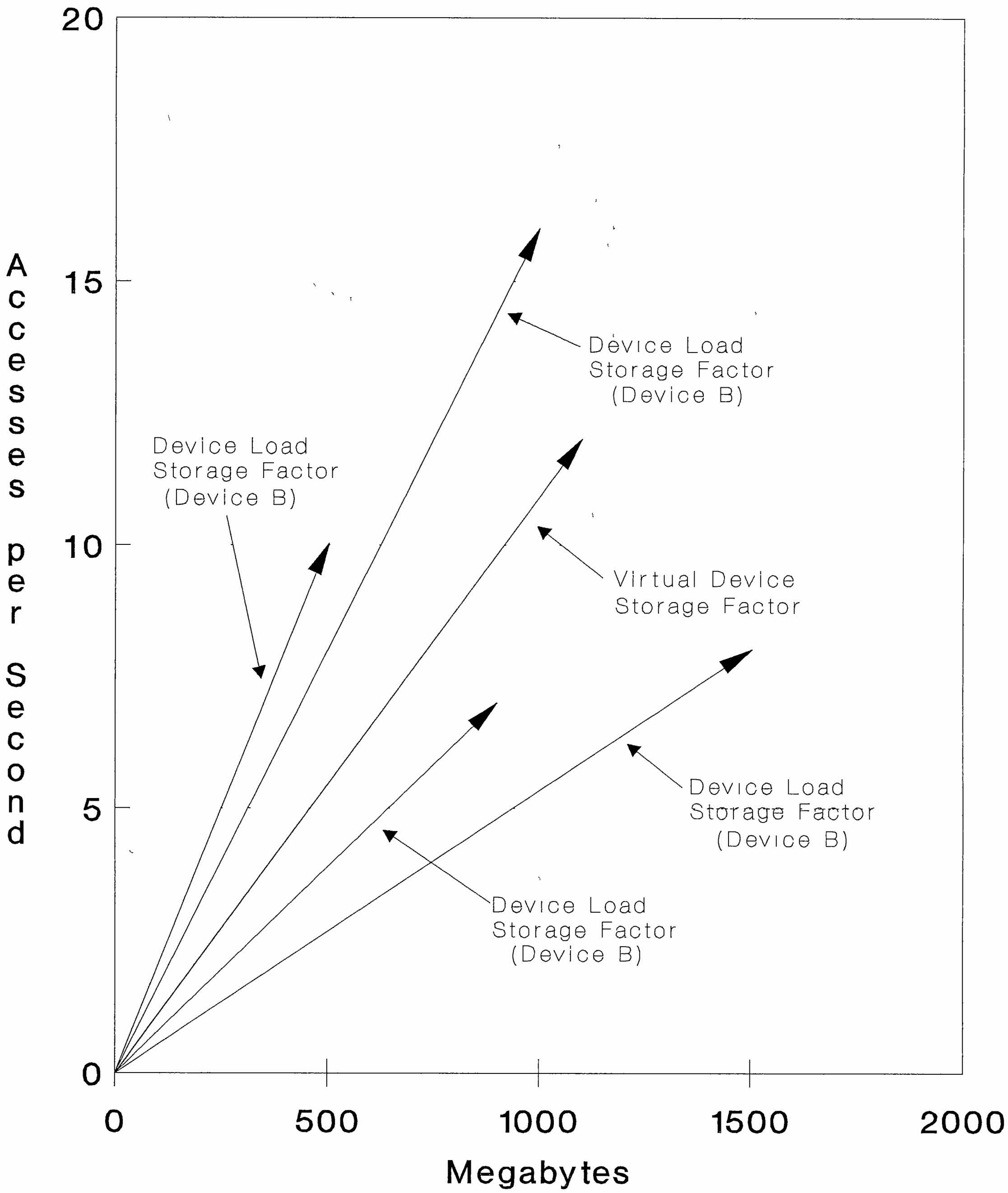


Figure 8. Device Loads vs. Virtual Device Storage Factor

A and B have excess capacity and devices C & D are overloaded. Data must be taken from devices C and D and placed on A and B until all the loads are balanced.

To determine which data sets should be moved, the residual storage factor is calculated for each device by taking the difference between the device load storage factor and the virtual device storage factor. A larger residual storage load indicates a more significant out-of-balance condition. Therefore, a data set is moved from the device with the largest residual storage load. It is then added to the under-utilized device that has a residual access density that most nearly matches the access density of the data set being moved.

Balancing a set of storage devices requires that the residual storage load for each device be reduced to as near zero as possible. This is accomplished by continuing to remove data sets from overloaded devices and then adding them to under-utilized devices while assuring that the residual storage load for each of the devices involved in the move is reduced. When no data sets can be moved so that the residual storage loads are reduced, the subsystem balance has been improved as much as possible.

The specific data set to move is dependent on the nature of the out-of-balance condition as determined by comparing the data loads to the virtual storage device capacities. Four different conditions are possible as shown in Table 4.

TABLE 4

OUT-OF-BALANCE CONDITIONS
RELATIVE TO THE VIRTUAL STORAGE DEVICE CAPACITIES

Condition	Space	Accesses/Second
Condition 1	Data Load < Device Capacity	Data Load < Device Capacity
Condition 2	Data Load > Device Capacity	Data Load < Device Capacity
Condition 3	Data Load > Device Capacity	Data Load > Device Capacity
Condition 3	Data Load < Device Capacity	Data Load > Device Capacity

These conditions are represented graphically by dividing the space surrounding the endpoint of the virtual device storage factor into four areas as shown in Figure 9. The area where a storage factor terminates identifies the out-of-balance condition for that device. The out-of-balance condition determines the criteria used to select a data set for movement. Following is a description of each condition and the criteria used to select a data set for movement.

Condition one exists when the data access requirements and the data space requirements are less than the capacity of the virtual storage device. In other words, additional access capacity and storage capacity exist on the device.

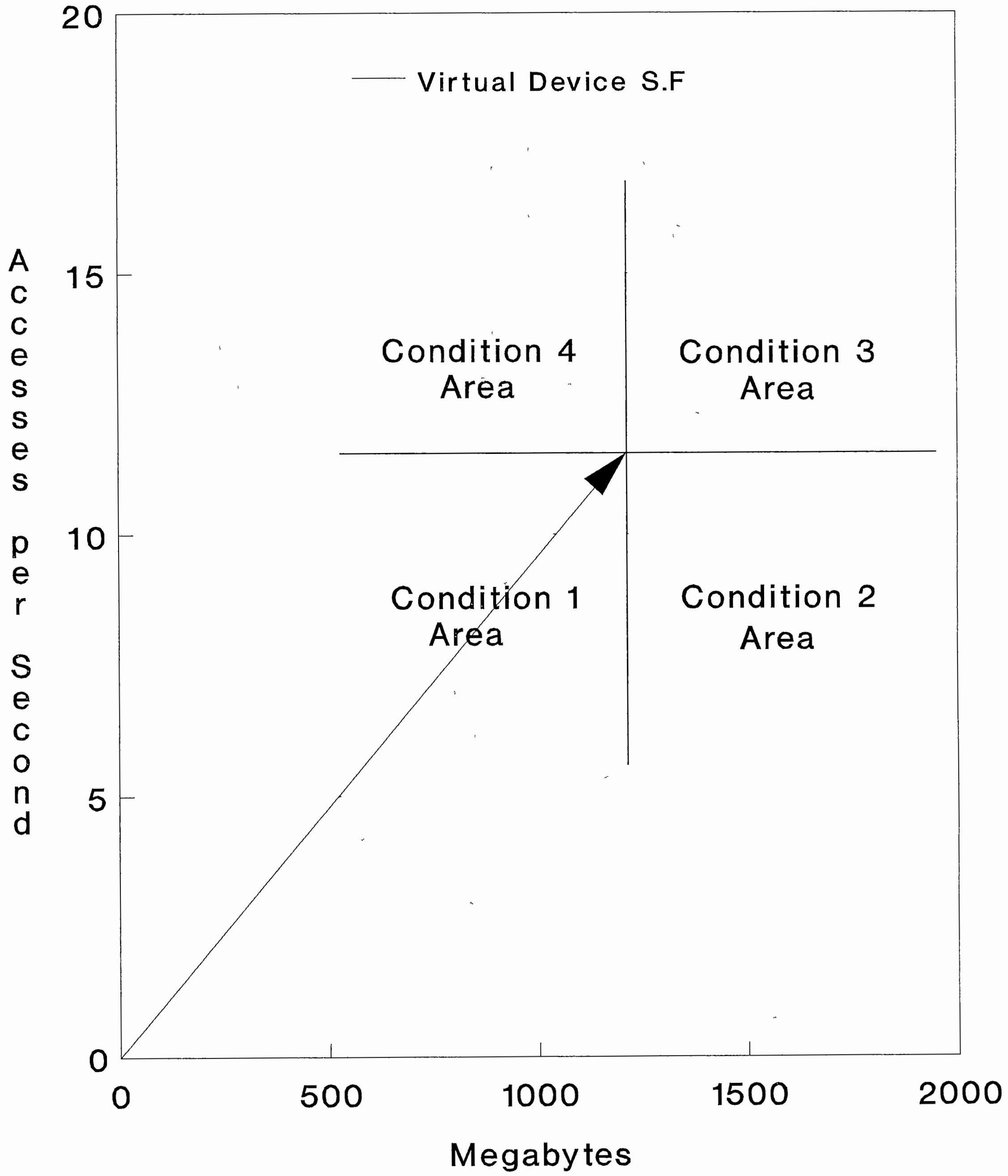


Figure 9. Out-of-Balance Conditions

Reduction of the residual storage load for these devices requires the addition of other data sets to the device. This occurs as data sets are moved from other devices. Since moving data sets off a device with this out-of-balance condition would create a larger residual storage load, moving data sets from these devices is not permitted.

Condition two exists when the amount of data stored on a device exceeds the capacity of the virtual storage device and the data access requirements are less than the capacity of the virtual storage device. In other words, too much data is stored on the device, while additional capacity remains for accessing data. In this case, data sets with the smallest access densities should be removed from the device and added to another device since their removal will reduce the amount of data stored as much as possible while minimizing the reduction in the access load.

Condition three exists when the amount of data stored and the data access requirements for a device exceed the capacity of the virtual storage device. In other words, too much data is stored on the device, and too many accesses per second are required. The data sets with access densities closest to the residual access density should be removed from the device since their removal will reduce both the access and space requirements as much as possible.

Condition four exists when the data access requirements for a device exceed the capacity of the virtual storage device and the data space requirements are less than the capacity of the virtual storage device. In other words, too many accesses per second are required, but additional storage capacity remains on the device. Data sets with the largest access densities should be removed since their removal will reduce the access requirements as much as possible while minimizing the reduction in the amount of data stored.

The device from which a data set should be removed is the device with the largest residual storage load and with an out-of-balance condition of two, three, or four. A data set being moved should then be added to the device with a residual storage factor access density that is closest to the access density of the data set being moved. The receiving device will always be a device with an out-of-balance condition of condition one, condition two, or condition four.

Before moving a data set, however, two requirements must be met. The first requirement is that the residual storage load of the device from which the data set is removed must be reduced. This is illustrated for condition two in Figure 10, condition three in Figure 11, and Condition 4 in Figure 12. If removal of the selected data set can be plotted within the cross hatched area for the applicable condition, then the data set is eligible for

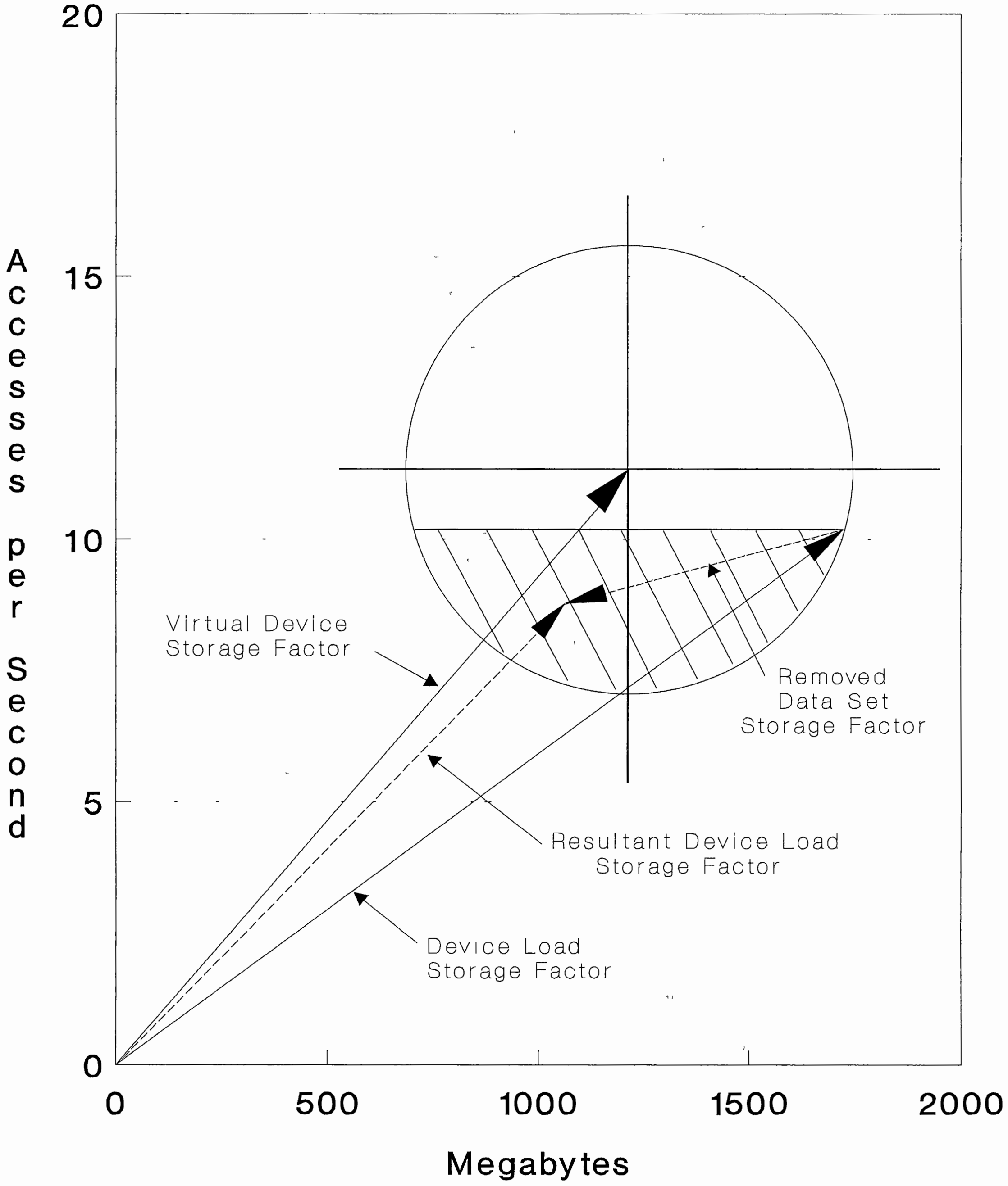


Figure 10. Valid Data Set Removal -- Condition 2

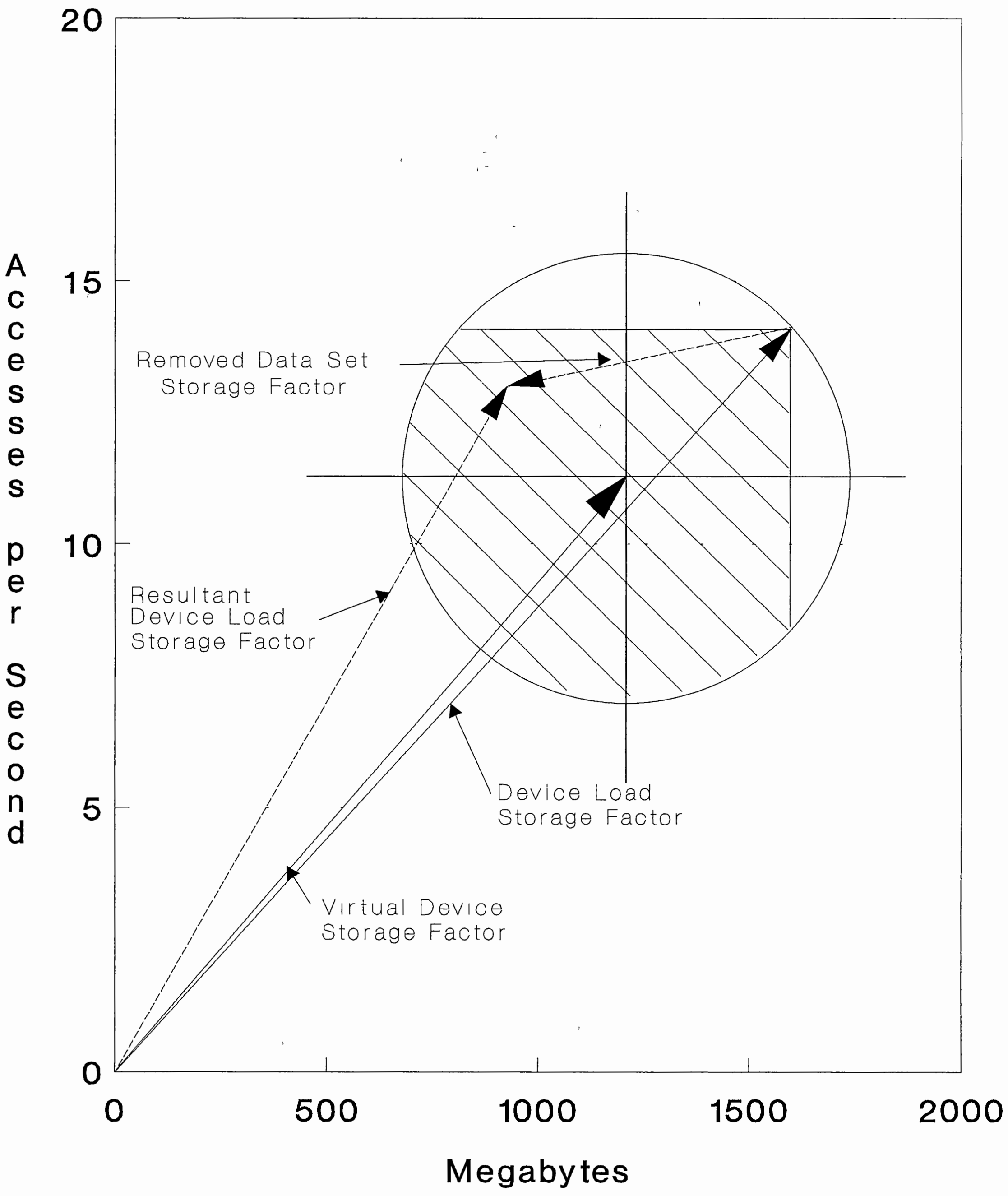


Figure 11. Valid Data Set Removal -- Condition 3

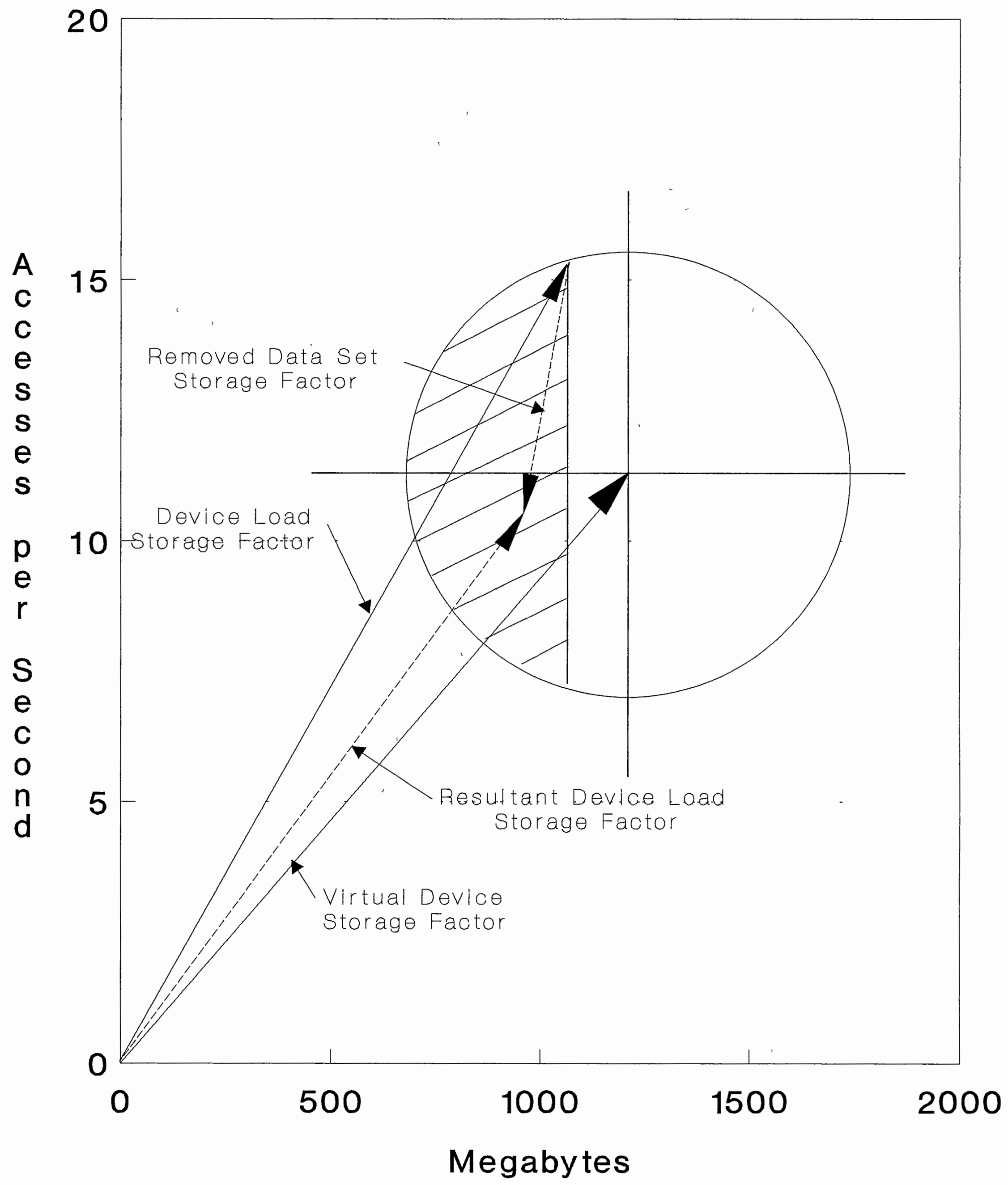


Figure 12. Valid Data Set Removal -- Condition 4

removal. If not, another data set must be selected. Of course, the size of the cross hatched area is a function of the termination point of the device load storage factor.

The second requirement that must be satisfied is that the residual storage load of the device receiving the data set must be reduced. This is illustrated in Figure 13.

The receiving device is determined by finding the device with a residual access density that most nearly equals the access density of the data set being moved. The residual storage load that would result if the data set was actually moved to this device is then calculated. If placement of the data set on the device would result in a larger residual storage load, the device with the next closest residual access density is tried. This is repeated until a device is found that can accept the data set or until all devices are eliminated as possible destinations.

These conditions are necessary to assure termination of the balancing process. Otherwise, it is possible that a loop in which a set of data sets is moved back and forth between the same set of devices could occur and the balancing process would continue indefinitely.

The movement of data sets is continued until the magnitude of all of the residual storage factors is reduced to zero or until no more data sets can be moved.

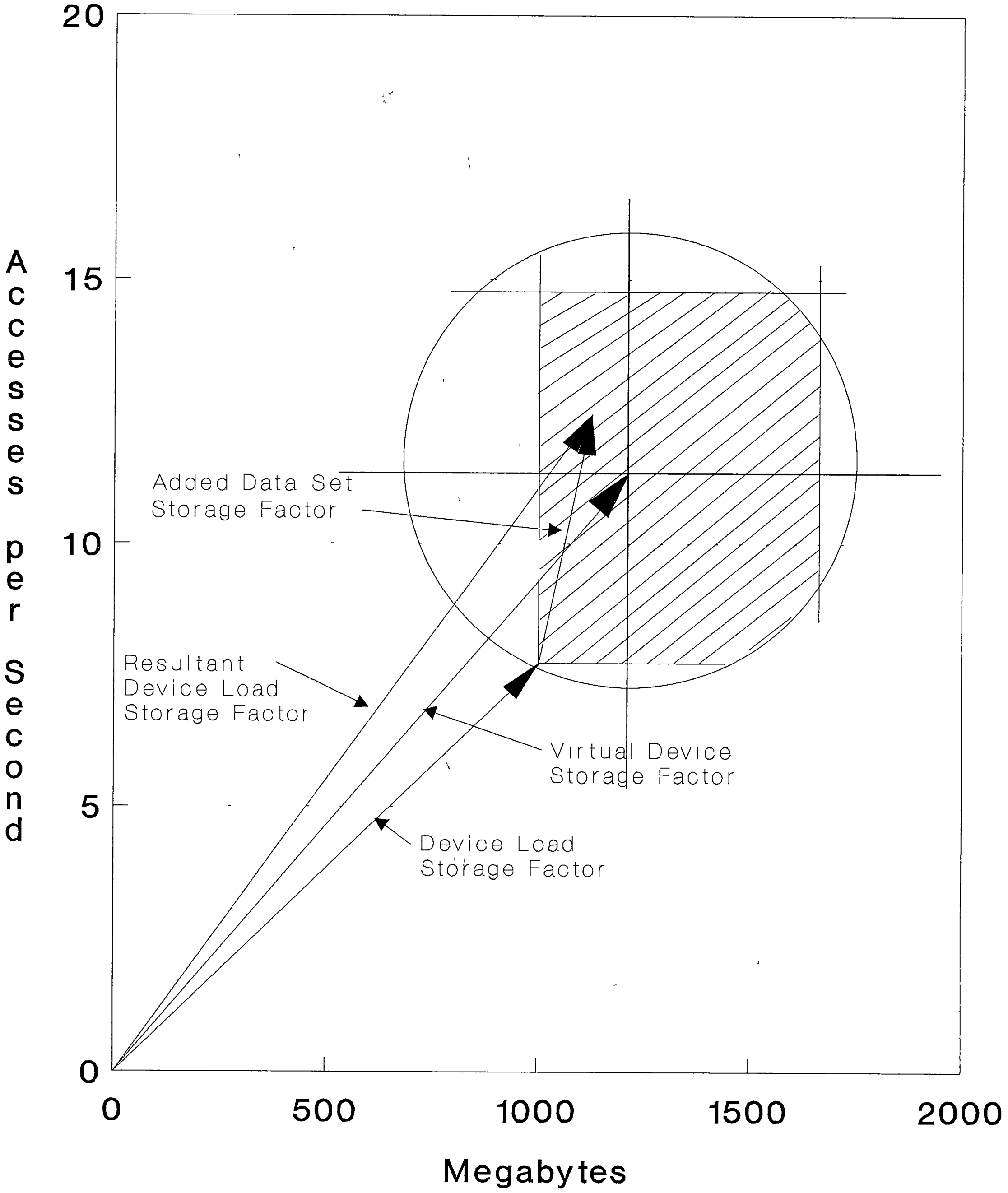


Figure 13. Valid Data Set Addition

Proof

The following is a proof of the validity of using residual storage factors to balance a storage subsystem. It must be considered as theoretical because it assumes no granularity of the data. In practical applications, the ability to move data sets is limited by the requirement to maintain the integrity of the data sets.

A general depiction of DASD skew using storage factors is shown in Figure 14.

Given that:

$n =$ the total number of devices

$y =$ the magnitude of the accesses per second

$x =$ the magnitude of the space requirements

$y_r =$ the residual accesses per second

$x_r =$ the residual space requirments

$y_a =$ the average accesses per second across all the devices

$x_a =$ the average space requirement across all the devices

The average accesses per second is calculated as shown in Equation 12.

$$y_a = \frac{y_1 + y_2 + y_3 + \ldots + y_n}{n} \tag{12}$$

The residual accesses per second for each device is calculated as shown in Equation 13.

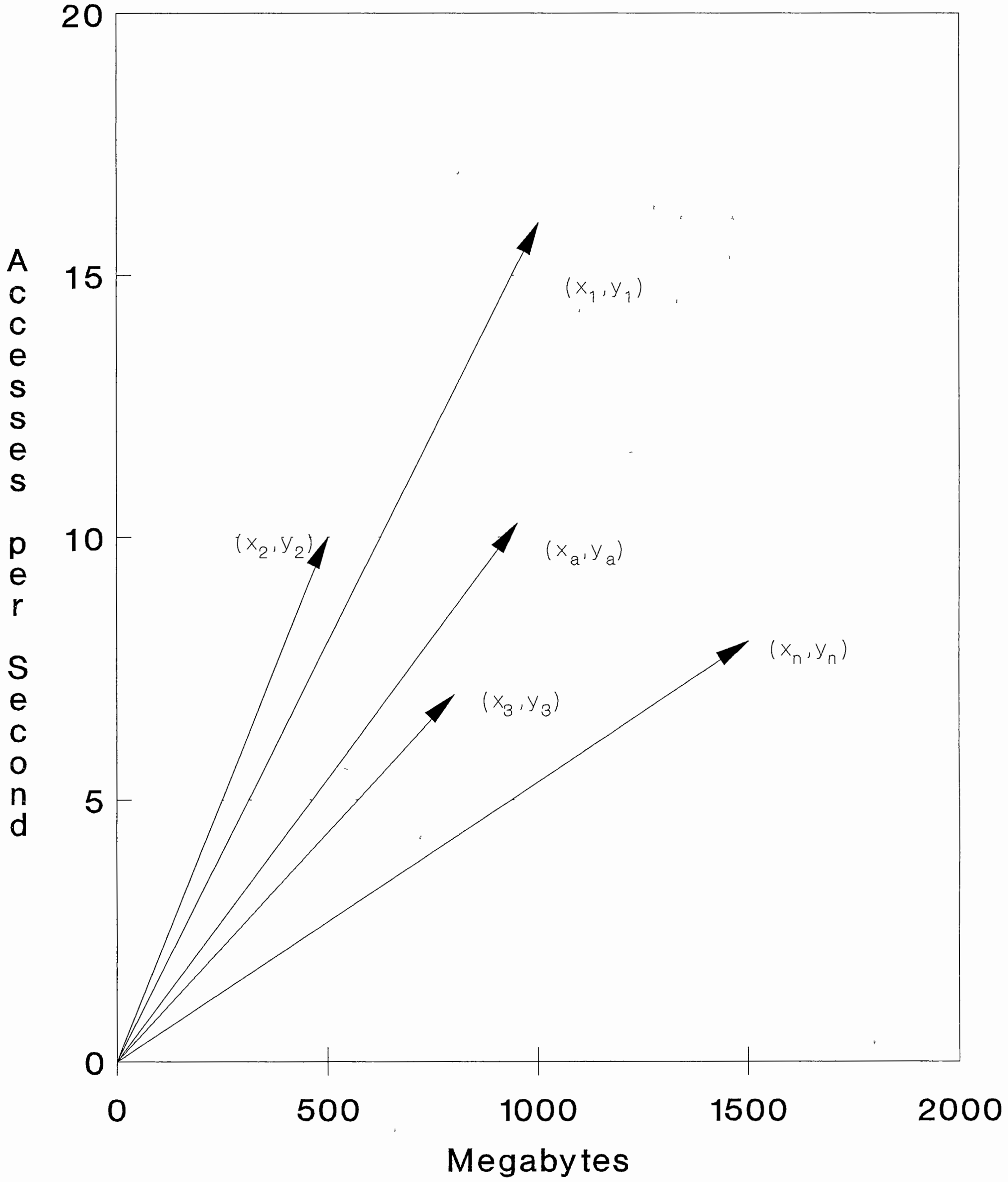


Figure 14. General Storage Factor Depiction of DASD Skew

$$
\begin{aligned}
y_{r1} &= y_1 - y_a \\
y_{r2} &= y_2 - y_a \\
y_{r3} &= y_3 - y_a \qquad (13) \\
&. \\
&. \\
&. \\
y_{rn} &= y_n - y_a
\end{aligned}
$$

As Equation 14 shows, the sum of the residual accesses per second is equal to zero.

$$
\begin{aligned}
\sum_{i=1}^{n} y_{ri} &= y_{r1} + y_{r2} + y_{r3} + \ldots + y_{rn} \\
&= (y_1 - y_a) + (y_2 - y_a) + (y_3 - y_a) + \ldots + (y_n - y_a) \\
&= \left(y_1 - \frac{y_1}{n} - \frac{y_2}{n} - \frac{y_3}{n} - \ldots - \frac{y_n}{n} \right) + \ldots + \left(y_n - \frac{y_1}{n} - \frac{y_2}{n} - \frac{y_3}{n} - \ldots - \frac{y_n}{n} \right) \qquad (14) \\
&= y_1 - n\left(\frac{y_1}{n}\right) + y_2 - n\left(\frac{y_2}{n}\right) + \ldots + y_n - n\left(\frac{y_n}{n}\right) \\
&= (y_1 - y_1) + (y_2 - y_2) + \ldots + (y_n - y_n) \\
&= 0
\end{aligned}
$$

If all the devices with a residual access load greater than or equal to zero are labeled from 1 to k, then, as shown in Equation 15, the sum of the residual access loads for the remaining devices from $k+1$ to n must equal the negative of the sum of the residual access loads for the devices form 1 to k. This follows from the fact that the sum of all the residual access loads must equal zero.

$$0 = \sum_{i=1}^{k} y_{ri} + \sum_{i=k+1}^{n} y_{ri}$$

$$\sum_{i=k+1}^{n} y_{ri} = -\sum_{i=1}^{k} y_{ri} \qquad (15)$$

It is also apparent that when some access load y_m is removed from one device and added to a different device, the sum of all of the residual storage factors remains zero.

Since it has been shown that the residual access load consists of two equal but opposite components and that an access load can be removed from one device and added back to a different device without affecting the sum of the residual access loads, then it follows that if the residual access loads greater than zero are removed from their respective devices and are added to the devices with a residual access load less than zero, then the residual access loads on all devices will equal zero.

This same logic can be applied to the storage space of the devices. In the previous equations, x and x_r can be substituted for y and y_r. It is then easily shown that moving the positive residual space loads to the devices with negative residual space loads adjusts the residual storage space load to zero on each device.

When the residual loads are adjusted to zero, the system is balanced. Therefore, given the ability to move the loads as desired, it is possible to balance a storage subsystem using the procedures outlined previously.

Balancing Constraints

The physical requirement of maintaining data sets as a single entity stored on a single device constrains the movement of both the access and space loads from one device to another device. A data set cannot be divided arbitrarily according to the residual conditions. At the time of balancing, the size and access demand of the data are fixed. Therefore, as data sets are moved during the tuning process and as a device approaches its calculated desirable loads, it may not be possible to find a data set with the exact characteristics necessary to match the device characteristics perfectly. Usually, a compromise must be made because an exact match cannot be found. This situation is aggravated by larger data sets and data sets with larger access demands.

Thus, if a device is loaded to within 0.1 megabytes of the desirable total megabytes and 0.08 accesses per second of the desirable total accesses per second, there may not be a data set available for movement that matches this condition. And, of course, the larger the data sets to be moved, both in terms of access demand and the data set size, the more difficult it will be to find a device that can accommodate its requirements. Thus, depending on the size of the demands, the ideal condition will most likely not be achievable. Therefore, termination of the balancing effort will very likely be the result of an inability to move more data sets rather than the achievement of a perfectly balanced subsystem.

CHAPTER IV

CASE STUDY

Data Set Information

To demonstrate the use of storage factors for tuning storage subsystems, access and storage space information was collected from a large corporate information center using the IBM System Management Facilities (SMF). The center uses IBM 3090 mainframes with an extensive array of 3380 model AK4/BK4 disk storage devices. The device storage factors section contains a description of these devices. Because of the large quantity of data stored in this subsystem, the data gathered was restricted to a single string of sixteen devices over one 24 hour period. The device string contains sixteen 3380 model AK4/BK4 storage devices or volumes. Because SMF provides inaccurate access counts (EXCP's) for Virtual Storage Access Methods (VSAM) [22,34], a device string on which 14 of the 16 devices were designated for TSO storage and the other two were assigned to specific user groups for special storage requirements was used. This should have eliminated the problems with the VSAM access methods because

TSO data sets usually do not use this access method. Over 41,000 data sets were stored on these 16 volumes.

The SMF counts EXCP's without regard to any I/O caches that may exist in the system. This is because the SMF is not designed to differentiate between cached and non-cached devices. In the system from which the data for this study was obtained, each channel was cached for reads only. Therefore, many of the EXCP's would have been handled by the cache. For this investigation, however, cacheing was ignored and it was assumed that the disk storage devices were required to handle the full I/O demand. These assumptions should not adversely affect the results of this study since the purpose of the data was to provide a realistic demand scenario for a large storage subsystem.

The SMF writes I/O information each time a data set is closed within a step. These records are referred to by the SMF as Type 14 or Type 15 records, depending on whether the data set was opened for input, output, or update, etc. These Type 14/15 records include the following pertinent information:

- The data set name.
- The volume that the data set is stored on.
- The time that the data set was opened.
- The time that the data set was closed.
- The number of EXCP requests initiated for the data set while the data set was opened.

A large amount of additional information is also logged on the Type 14 and Type 15 records but is not relevant to this analysis and was ignored. Merrill and IBM [22,24] provide a complete description of the Type 14 and Type 15 records from the SMF system.

Since the purpose of this analysis was to evaluate the use of storage factors to balance storage subsystem loads, it was assumed that the data gathered represented a characteristic access requirement for each data set. In reality, a monitoring system such as the one described in the data set storage factors section would be required to determine a true characteristic access requirement. However, this data provides a sampling of a large number of data sets from an actual working environment, and therefore, should provide a realistic simulation of the tuning problem.

Data Reduction

For the 24 hours during which data was gathered on the 16 volumes, over 129,000 Type 14/15 records were produced. However, not all these records contained useful information. As explained by Merrill [22], this is because the SMF writes a record each time the data set is closed; but, the true EXCP count is only written when the last close is executed during a job step. In examining the SMF data after it was sorted by the volume label, the data set name, and the time the data set was opened, it appeared that this condition occurred (multiple open/closes within the same step) when there was a record with open and close times that included the open and close times for several of the records that followed. Therefore, records that had open and close times between the open and close times of a previous record for the same data set were excluded from the EXCP count. Table 5 provides a typical example of the data produced by the SMF when this condition occurred. Only the first line of data contains reliable information. All the other information was discarded because the open and close times fall within the open and close times of the first line of data.

TABLE 5

SAMPLE SMF DATA--MULTIPLE RECORDS
WITH UNRELIABLE EXCP COUNTS

Data Set Name	Open Time	Close Time	EXCP Count
USER.XYZ.DATA	12:22:40.77	12:28:49.99	16
USER.XYZ.DATA	12:28:41.88	12:28:42.11	11
USER.XYZ.DATA	12:28:42.11	12:28:42.26	13
USER.XYZ.DATA	12:28:42.31	12:28:42.42	6
USER.XYZ.DATA	12:28:42.48	12:28:42.88	10
USER.XYZ.DATA	12:28:47.80	12:28:47.95	19
USER.XYZ.DATA	12:28:47.96	12:28:48.09	21
USER.XYZ.DATA	12:28:48.14	12:28:48.32	14
USER.XYZ.DATA	12:28:48.32	12:28:48.45	16

With this fact in mind, all of the Type 14/15 records were analyzed to provide an EXCP count for each data set opened during the time period analyzed. Each record was read and checked to assure that it contained reliable data as explained above. The EXCP count was then added to the total for the data set. When a data set was opened in one time period and closed in the next, the EXCP count was pro-rated to the time period being analyzed based on the ratio of the amount of time the data set was opened within a time period relative to the total time the data set was open. Usually, a data set used several times during the day had different access requirements depending on the time of day. Thus, a data set used between 9:00 and 10:00 a.m., 10:00 and 11:00 a.m., and 12:00 and 1:00 p.m. would probably have a different access requirement for each time period.

The size of the data set was determined from the space allocated as specified in the VTOC of the device where the data set was stored.

Once the access requirement and size of a data set were determined, the data set's storage factor was also defined. The data set storage factors were the key element in analyzing and balancing the storage subsystem.

Selecting the Analysis Time Period

Using the storage factor for each data set, a subsystem load storage factor was calculated for each one hour time period during the day. The critical time periods are those with the maximum access demands, or, using storage factors, those time periods with the largest subsystem load access densities. The storage space does not have an impact because the storage space required for all the data sets stored in the subsystem is considered constant when the subsystem is analyzed. Table 6 lists the data for each time period, and Figure 15 displays the subsystem load storage factors. 15:00 - 16:00 p.m. was selected as the time period to be analyzed for this study.

TABLE 6

SUBSYSTEM LOAD STORAGE FACTORS BY TIME PERIOD

Time Period	Accesses per Second	Megabytes of Storage
00:00 to 01:00	8.423	22,272
01:00 to 02:00	7.368	22,272
02:00 to 03:00	1.340	22,272
03:00 to 04:00	5.112	22,272
04:00 to 05:00	3.724	22,272
05:00 to 06:00	1.214	22,272
06:00 to 07:00	4.024	22,272
07:00 to 08:00	19.439	22,272
08:00 to 09:00	56.055	22,272
09:00 to 10:00	46.487	22,272
10:00 to 11:00	78.008	22,272
11:00 to 12:00	37.086	22,272
12:00 to 13:00	29.418	22,272
13:00 to 14:00	50.824	22,272
14:00 to 15:00	49.570	22,272
15:00 to 16:00	83.952	22,272
16:00 to 17:00	46.827	22,272
17:00 to 18:00	25.839	22,272
18:00 to 19:00	8.175	22,272
19:00 to 20:00	10.345	22,272
20:00 to 21:00	7.121	22,272
21:00 to 22:00	0.752	22,272
22:00 to 23:00	1.002	22,272
23:00 to 24:00	2.689	22,272

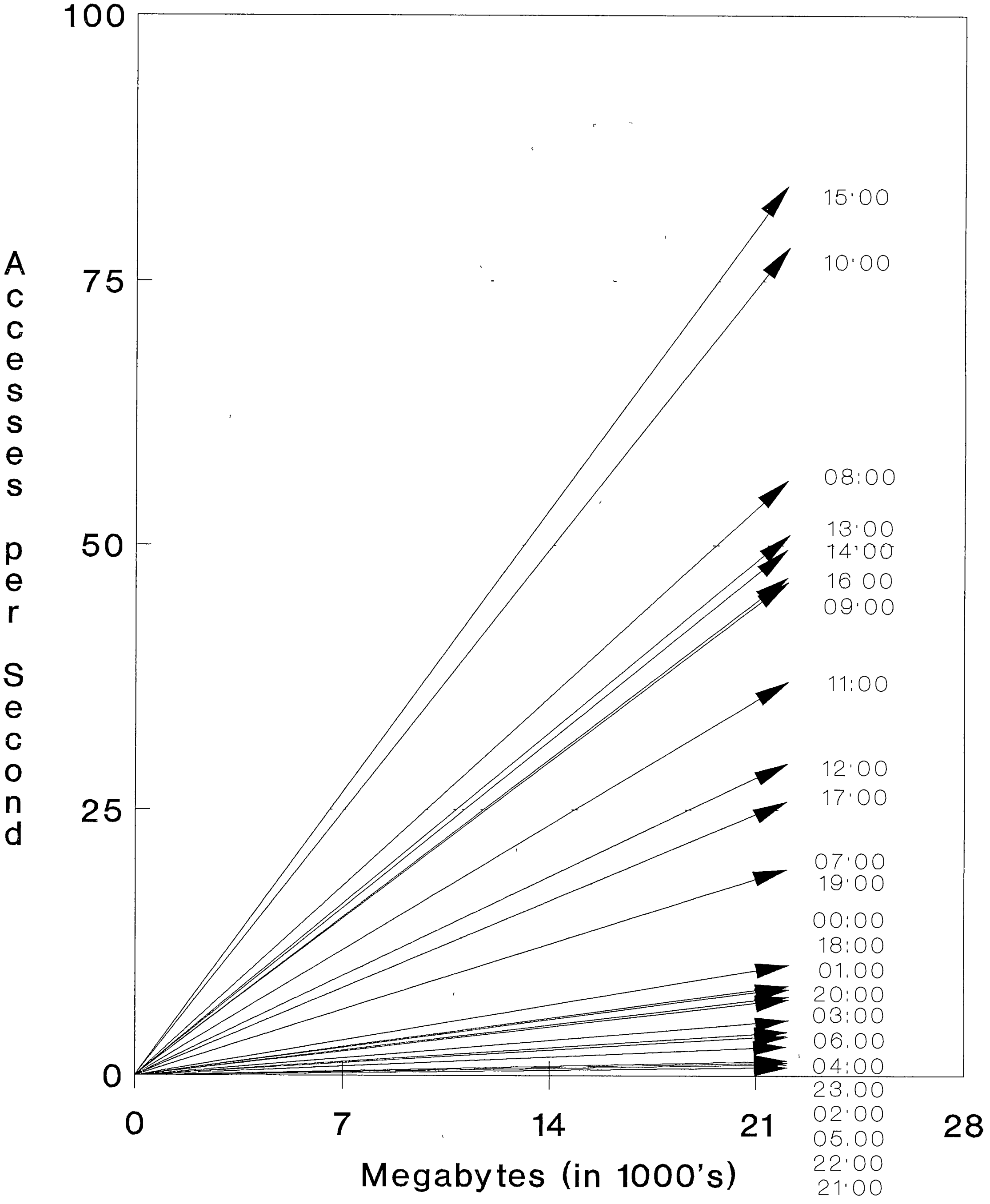


Figure 15. Subsystem Load Storage Factors by Time Period

Hardware Analysis

After determining the time period to be analyzed, the subsystem's hardware was evaluated by comparing the total hardware capacities to the total data storage requirements. The subsystem load storage factor was calculated previously when determining the time period to be analyzed. The subsystem capacity storage factor was determined by summing the device storage factors for each device. This provided a picture of the total capabilities of the system. Table 7 and Figure 16 summarize the subsystem being analyzed.

TABLE 7

SUBSYSTEM STORAGE FACTORS COMPARISON
FOR 15:00 - 16:00 P.M.

Storage Factor Type	Accesses per Second	Megabytes of Storage
Subsystem Capacity	181	25,696
Subsystem Load	84	22,272
Subsystem Residual	97	3,424

Comparing these storage factors indicates that the number of devices can be reduced and that the type of devices can be modified to handle the data requirements more economically. By subtracting two device storage factors

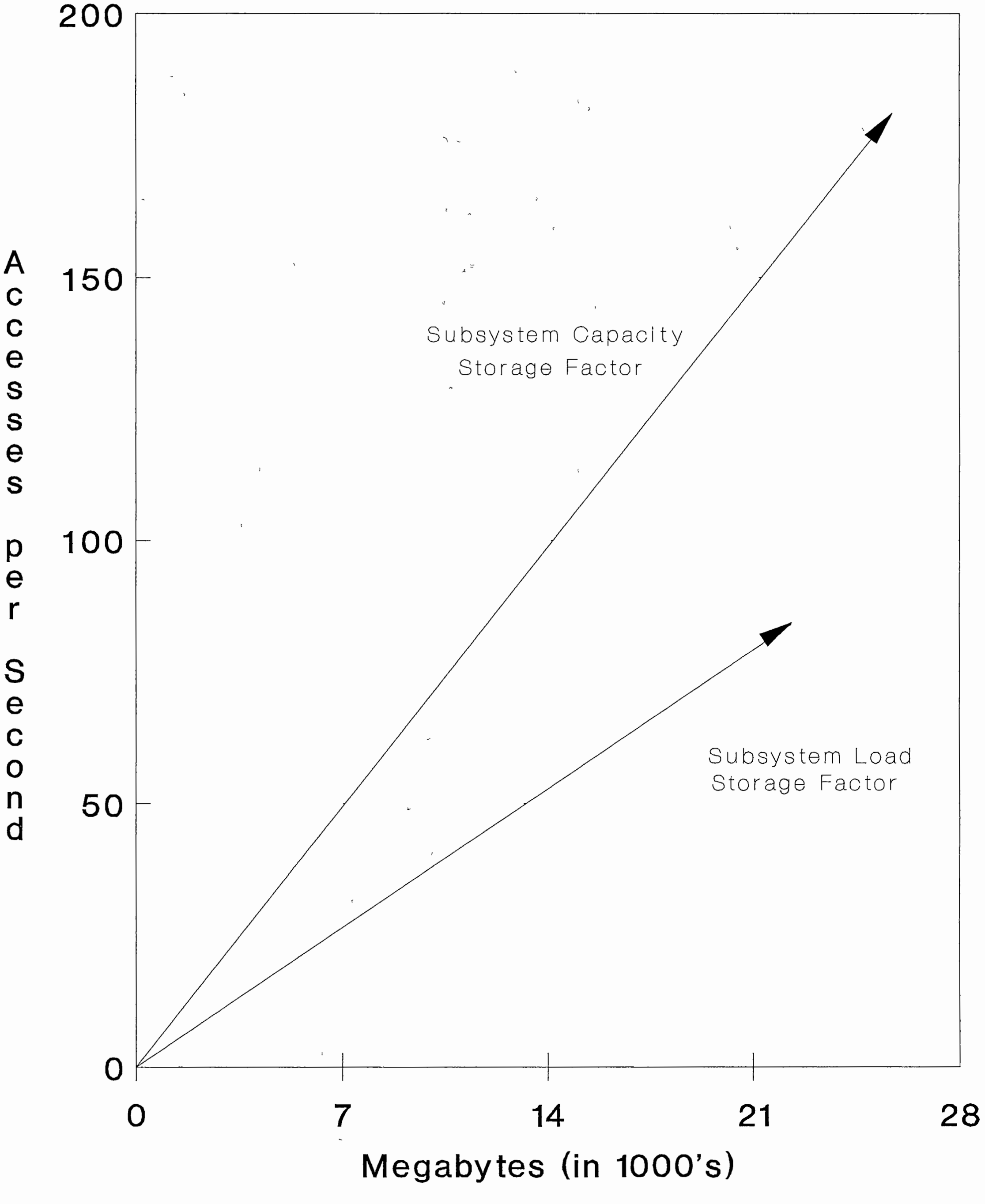


Figure 16. Overall Subsystem Condition--15:00-16:00 P.M.

from the subsystem capacity storage factor, it is apparent that there is adequate capacity to handle the system data storage factor. Thus, it is feasible to remove two devices from the system and have the system function adequately. This is illustrated in Figure 17.

Also, other adjustments may be justified economically. There is still a significant amount of additional capacity for accesses per second provided by the existing hardware, even after removal of two devices. This suggests that other hardware devices with lower access density specifications may be more economical since they would provide less access capacity relative to the amount of data stored, a theoretically less costly device. For this analysis, however, no adjustments were made to the hardware configuration.

Determining Virtual Device Storage Factors

After evaluating the subsystem as a whole and making any hardware adjustments necessary, data loads must then be adjusted on each device so that the loads are more evenly distributed across all the devices in the subsystem. This is necessary to eliminate any loads that may exceed the capacity of a device. Also, it reduces the probability that any of the devices in the subsystem will become overloaded in the future.

To balance the storage loads, the device load storage factor is first calculated for each device in the

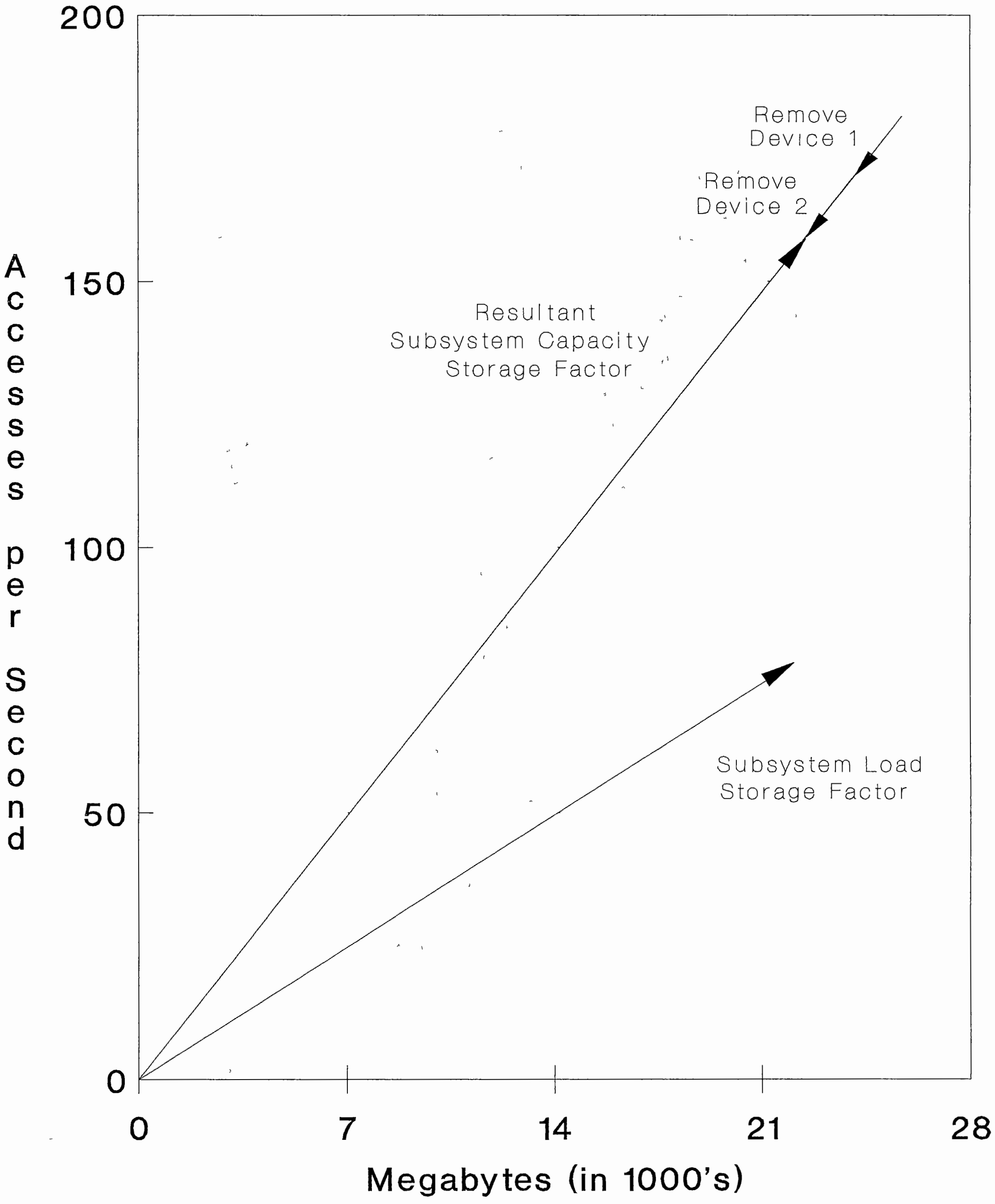


Figure 17. Removal of Two Devices From the Subsystem

subsystem. This is the sum of the data set storage factors for all of the data sets stored on a device for the time period being analyzed. Table 8 summarizes this data for the subsystem being analyzed. Figure 18 also displays the device load storage factor for each device, clearly showing that there is a wide diversity in the direction and magnitude of the device load storage factors. Since all of the devices in the subsystem are the same, this indicates that during the 15:00-16:00 p.m. time period, considerable DASD skew exists in the subsystem and that balancing of the subsystem is required.

TABLE 8

DEVICE LOAD STORAGE FACTORS BEFORE BALANCING
15:00 - 16:00 P.M. TIME PERIOD

Volume	Accesses per Second	Megabytes of Storage
RES018	16.626	1230.50
TSH000	4.259	1618.88
TSH001	4.272	1141.89
TSH002	2.765	1361.78
TSH003	11.195	1506.37
TSH004	8.243	1401.85
TSH005	6.441	1290.69
TSH006	2.853	1568.64
TSH007	4.434	1423.16
TSH008	2.663	1391.16
TSH009	1.703	1503.97
TSH010	2.709	1387.71
TSH011	2.880	1587.67
TSH013	3.572	1238.44
TSH014	2.288	1276.59
USDD01	7.048	1343.14

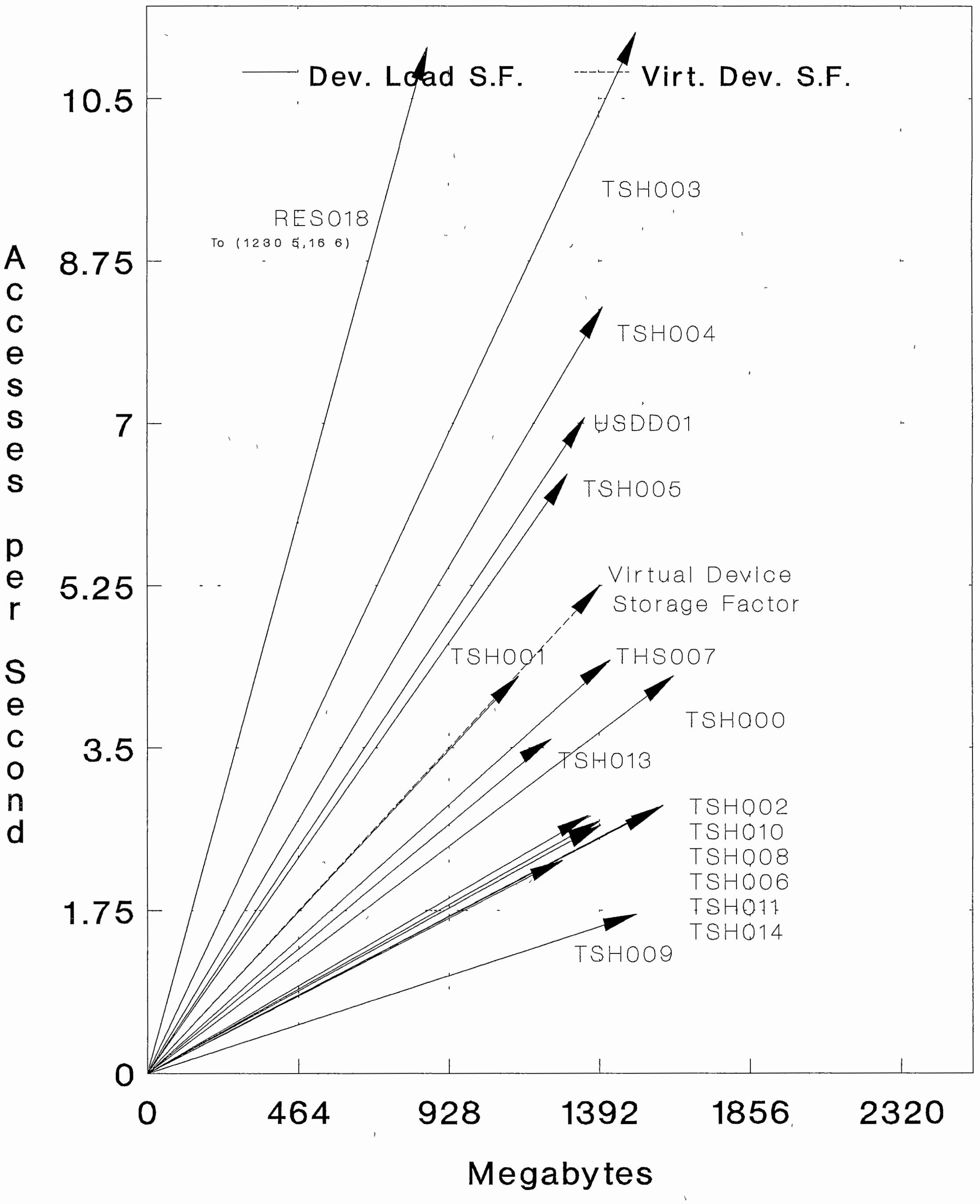


Figure 18. Device Load Storage Factors Before Balancing 15:00-16:00 P.M.

To balance the subsystem, the target load, or virtual device storage factor, is calculated for each device using the subsystem capacity load storage factor and the device storage factors. In this study, because all the devices are the same, it is easily calculated by dividing the subsystem device load storage factor by 16--the number of devices in the system. The result was an access rate of 5.25 accesses per second and 1392 megabytes of storage space. These are the target loads defining the virtual device storage factor for the 15:00 to 16:00 p.m. time period.

Determining the Scale Factors

The virtual device storage factor has an access component of 5.25 accesses per second and a space component of 1392 megabytes. These were scaled to plot three inches in length. Therefore, the scale factor for accesses per second is 0.57142 and the scale factor for megabytes of storage is 0.002155.

Balancing the Loads

As previously described, balancing the loads on a string of storage devices requires the magnitude of the residual storage factor for each device to be reduced to as near zero as possible. This is accomplished by moving data sets from those devices that are overloaded to those devices that are under-utilizing their capacities. To balance the system analyzed in this study, a computer program was written that determines which data sets should be moved and on which device they should be placed. The program automates the procedures described in the previous chapter for moving data sets. The process is briefly described below.

The program calculates the residual storage factor for each device in the system. A data set is then selected for movement from the device having the largest residual storage load that is also overloaded. The data set selected is determined by the rules that apply to the specific out-of-balance condition. If movement of the data set will reduce the residual storage load of the device where the data set currently resides, a device is selected for placement of the data set. If removal of the data set will increase the residual storage load, then the data set with the next closest access density characteristic for the out-of-balance condition of the device is analyzed to see if it can be moved. This is repeated until a data set that can be moved is found, or, if none is found, the device with the next largest device load access density is checked for a data set to move.

The device on which to place a data set is determined by finding the device with a residual access density that is closest to the data set access density. If movement of the data set to the selected device reduces the residual storage load of that device, then the data set is moved to the selected device. If the placement of the data set on the selected device would increase the magnitude of the residual storage load of the device, the placement is tried on the device with the next closest residual access density. This is repeated until a device is found for placing the data set. If no device can be found on which to place the data set, then the data set must remain on its current device.

Each time a data set is moved, the device load storage factors for the "from" and "to" devices are recalculated, as are the residual storage factors. Data sets are moved until no more data sets whose movement will reduce the residual storage load of both the "from" device and the "to" device can be found. Termination is guaranteed because a data set cannot be moved unless the residual storage loads of the affected devices are reduced.

CHAPTER V

RESULTS OF THE CASE STUDY

The data used for this study was taken from a large corporate mainframe in its normal production environment. Because of the expense and impracticality of experimenting with a production system, it was not possible to balance the subsystem and monitor the results in the actual production system. Therefore, to evaluate the effect of balancing the subsystem using the procedures described previously, analytical and simulated results are used.

The results of the subsystem balancing are summarized in Table 9 and the accompanying Figure 19. Appendix B contains a plot of the storage factors for each device showing the before and after balancing condition. When compared to the pre-balanced condition as shown in Figure 18, it is apparent from the greater clustering of the device load storage factors that the amount of DASD skew has been reduced. Table 10 summarizes the changes in the magnitude of the residual storage factor for each device. The changes ranged from 0 percent to 98 percent with an average change of 40 percent.

TABLE 9

DEVICE LOAD STORAGE FACTORS AFTER BALANCING -- 15:00-16:00 P.M.

Volume	Accesses per Second	Megabytes of Storage
RES018	16.111	1222.32
TSH000	5.089	1577.61
TSH001	5.262	1224.51
TSH002	5.235	1378.82
TSH003	5.244	1424.10
TSH004	7.091	1385.35
TSH005	5.258	1290.54
TSH006	2.851	1563.75
TSH007	4.434	1422.59
TSH008	2.732	1392.16
TSH009	1.703	1500.65
TSH010	4.164	1392.13
TSH011	2.875	1579.69
TSH013	5.259	1278.81
TSH014	5.256	1311.28
USDD01	5.388	1328.13

TABLE 10

RESIDUAL STORAGE LOADS BEFORE AND AFTER BALANCING 15:00-16:00 P.M.

Volume	Residual Storage Load Before Balancing	Residual Storage Load After Balancing	Residual Storage Load Difference	% Residual Storage Load Difference
TSH002	1.420	0.029	1.390	97.9%
TSH003	3.408	0.069	3.338	98.0%
USDD01	1.035	0.138	0.897	86.7%
TSH014	1.709	0.139	1.571	91.9%
TSH005	0.716	0.219	0.498	69.5%
TSH013	1.013	0.244	0.768	75.9%
TSH001	0.775	0.361	0.414	53.4%
TSH000	0.747	0.410	0.337	45.1%
TSH007	0.469	0.469	0.000	0.0%
TSH010	1.450	0.619	0.831	57.3%
TSH004	1.712	1.054	0.658	38.5%
TSH011	1.417	1.414	0.002	0.2%
TSH006	1.420	1.418	0.001	0.1%
TSH008	1.477	1.437	0.040	2.7%
TSH009	2.040	2.039	0.001	0.0%
RES018	6.512	6.219	0.293	4.5%
Average	1.707	1.017	0.690	40.4%

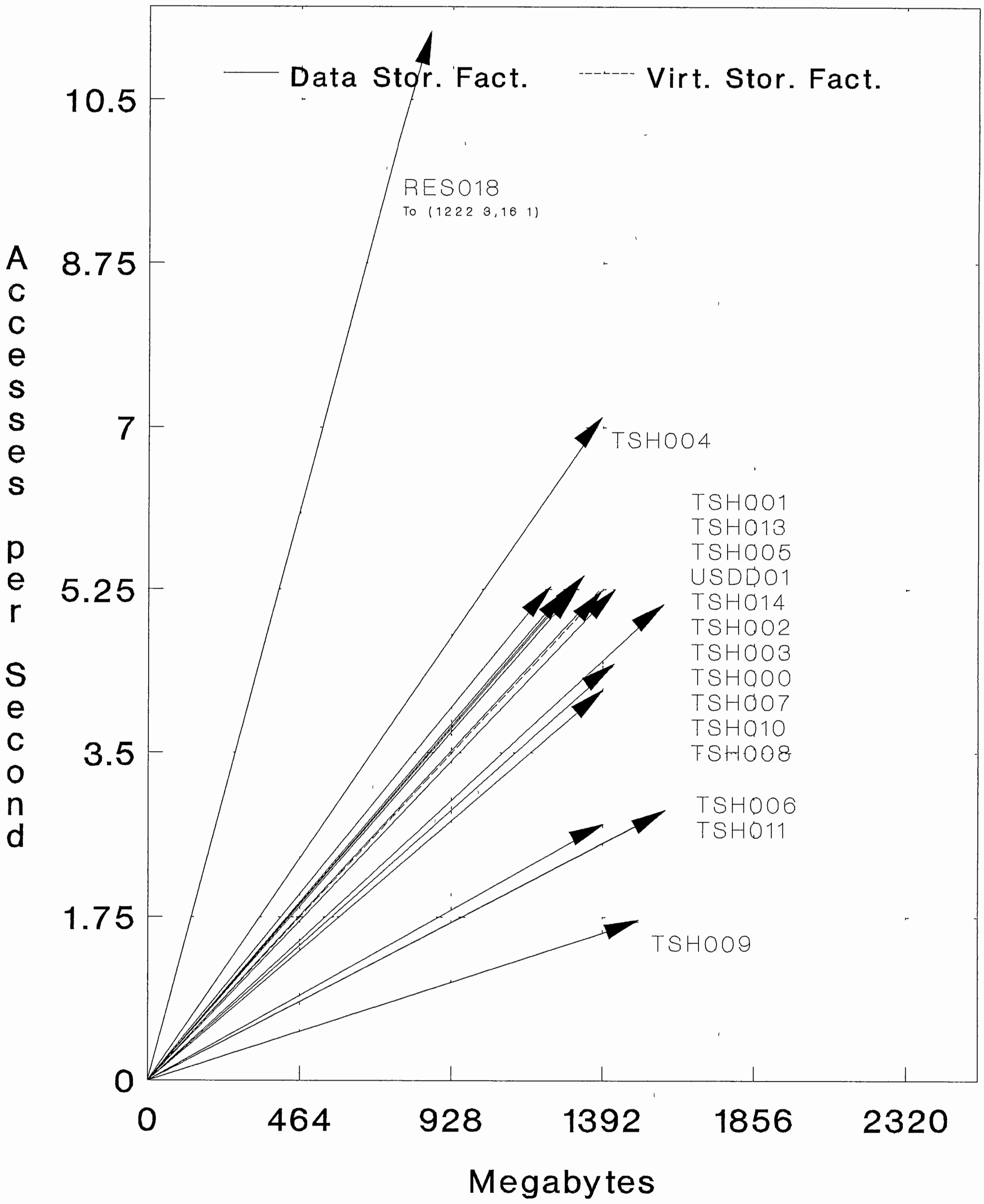


Figure 19. Balanced Device Load Storage Factors

The average 40 percent reduction in the residual storage factor was significantly smaller than anticipated. However, the reason for this relatively small reduction in the DASD skew became apparent when the data set storage factors were investigated for those devices that were most out of balance.

From Figure 18, RES018 had the greatest imbalance. The balancing procedure resulted in one data set stored on the RES018 device that was accessed during the 15:00 to 16:00 p.m. time period. This data set had an access rate of 16.11 accesses per second. However, due to the loads already existing on the other devices, it was impossible to add the data set to any other device and reduce the DASD skew. This left a residual access rate exceeding the virtual storage device by 11.36 accesses per second. Therefore, because the sum of all residual capacities must equal zero, a total of at least 11.36 access per second of unutilized residual accesses per second remains on the other devices. This at least partially accounts for the low utilization of the access capacity of the devices labeled TSH009, TSH001, TSH006, TSH010, and TSH007. It is not possible to add any additional access load to these devices because the access load must come from another device, and the access load cannot be moved from the overloaded devices.

Nearly the same condition exists on the device labeled TSH004. Two data sets with I/O activity occurring during the 15:00 to 16:00 p.m. time period remain on this device. One of these data sets had a very high access rate of 7.051 accesses per second. In this case, however, removal of the data set would have increased the magnitude of the residual storage factor of the device on which it was loaded. Therefore, it was impossible to remove the data set, which left a residual access rate of 1.804 accesses per second on the device. This would have added to the unutilized access capacity for the devices mentioned previously.

Although much less severe in this particular case, imbalances in the storage space utilization also contributed to the remaining DASD skew. The devices labeled TSH009, TSH011, TSH006, and TSH000 showed the largest excess utilization of storage space relative to the virtual storage device. To correct this problem, removal of the data sets with low access densities is required since removal of higher access density data sets has a greater tendency to reduce the access utilization of the device. Of course, this is undesirable in this case since the access capacity of the devices is already under-utilized. After reviewing the data sets assigned to these devices, it was apparent that all of the lowest access density data sets were removed from the devices. The higher access density data sets would each cause an increased residual storage factor if they were removed from their respective devices.

It should be noted, however, that numerous unutilized data sets were stored on these devices. For this study, they were combined to form one pseudo data set of extremely large size in order to reduce the amount of data to be handled. For example, the TSH009 device had a total of 1493 megabytes of data that was not accessed during the time period studied. If these data sets had been left separate, the storage space imbalance would very likely have been reduced further since some of the smaller inactive data sets could have been moved from a device that was over-utilized to one that was under-utilized.

From these results, it is apparent that the devices were not completely balanced because of the granularity of the data sets. Granularity refers to the fixed size and access requirement of each data set. This would not be a problem if data sets could be divided arbitrarily into separate smaller data sets and moved according to the imbalances indicated by the residual storage factors of the subsystem. However, it is not generally desirable to divide a data set into separate smaller data sets. This is because dividing a data set does not necessarily allocate the access load among the different divisions in the desired manner since one portion of a data set may be more frequently accessed than another portion.

Therefore, since division of data sets is not a viable option, data set granularity requires that the movement of loads must be made in increments according to the data set characteristics, rather than by simply moving the loads indicated by the residual storage factor. Consequently, there are some loading conditions for which a data set with a small enough space or access characteristic cannot be found to allow movement of the loads from an overloaded device to an under-utilized device.

As illustrated by the problem data sets on RES018 and TSH004, one of these conditions occurs when a data set has an access rate that exceeds the access rate of the balanced condition. In these situations it is obviously impossible to reduce the access load to the balanced level because the data set must be maintained as a single entity. Therefore, no matter which device the data set is placed on, the device will be out of balance by the amount of the data set's excessive access load. When this occurs, it is an indication that the data set is placed on a device with incompatible characteristics. A device with larger access capacity is required to handle this type of data set effectively.

Two other related loading conditions restrict data set movement. These occur when the storage space is slightly over-utilized and the excess access capacity is relatively high, or when the access capacity is slightly over-utilized and the excess storage space is relatively high. These conditions are illustrated in Figure 20 and Figure 21. The cross-hatched area represents the region where the removal of a data set must plot in order to

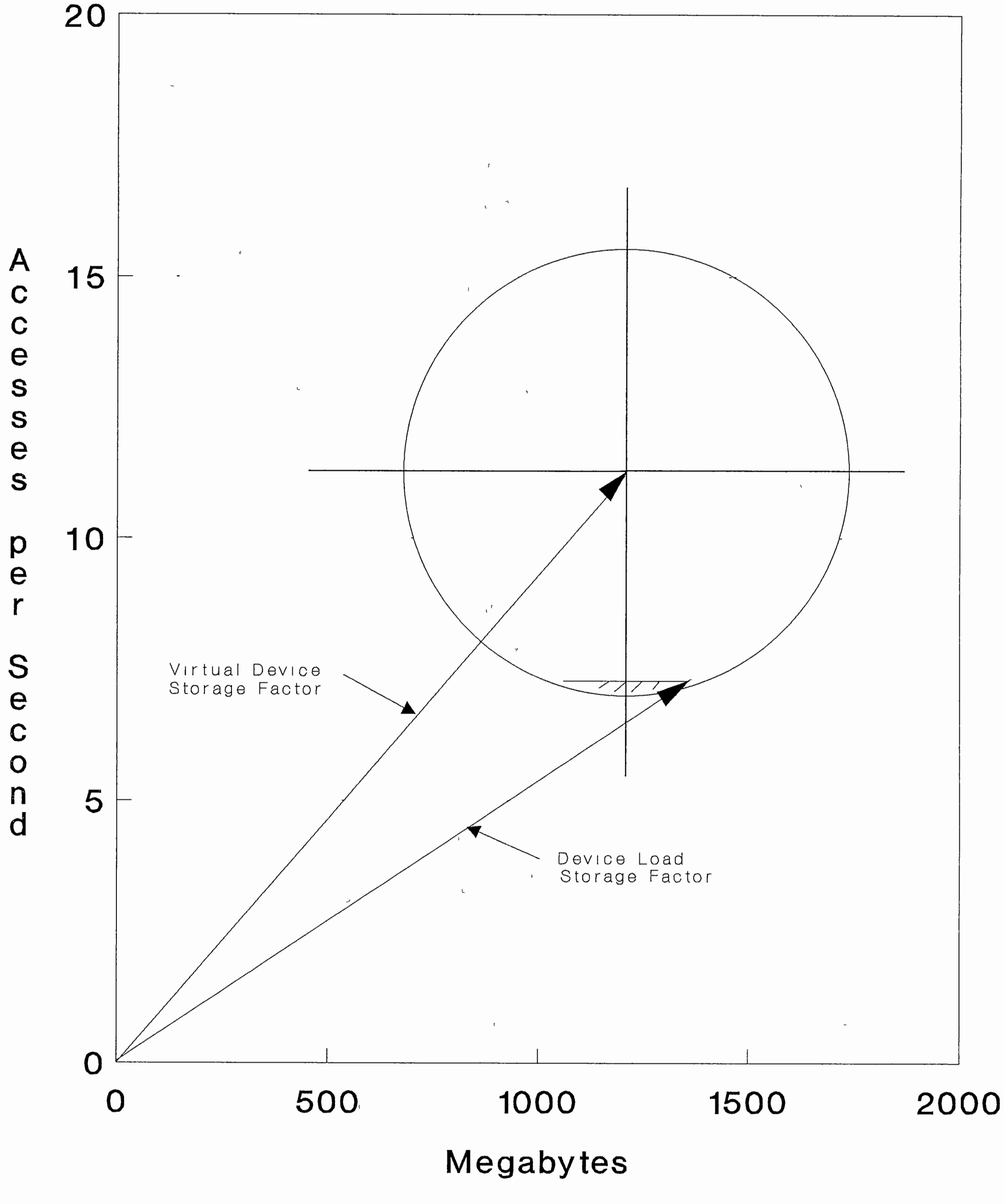


Figure 20. Restricted Data Set Movement Possibilities
Condition 2

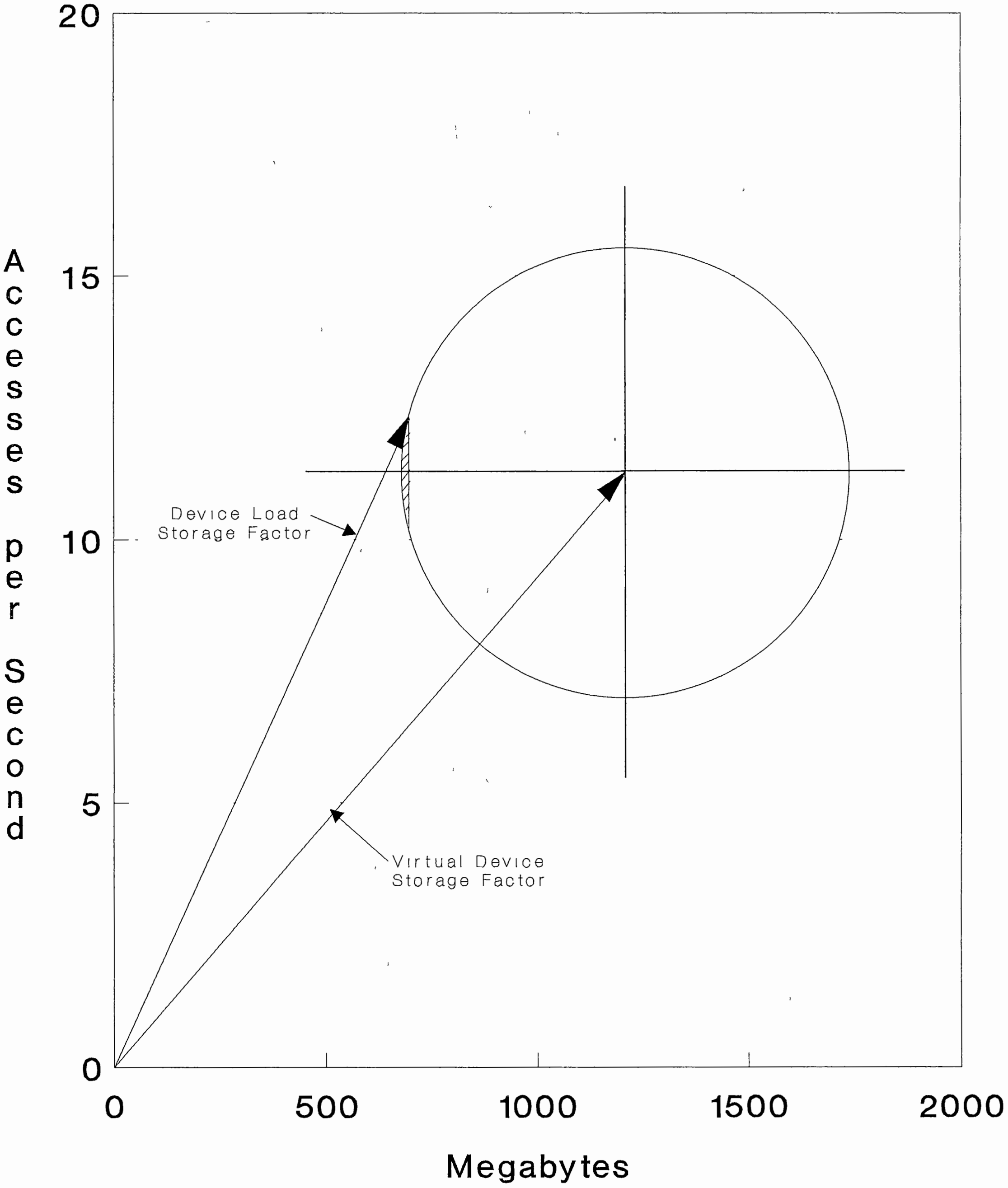


Figure 21. Restricted Data Set Movement Possibilities
Condition 4

reduce the magnitude of the residual storage factor. As the over-utilized characteristic is brought closer and closer to the balanced condition by removing loads from the device, the size of this area approaches zero. The size of the area is calculated as shown in Equation 16 [31].

$$Area = \frac{1}{2} R^2 (\theta - \sin\theta) \tag{16}$$

where θ is double the residual storage angle and R is the magnitude of the residual storage factor. This can also be expressed in terms of the residual access rate as shown in Equation 17 [31].

$$Area = R^2 \mathrm{Cos}^{-1} \frac{y_r}{R} - y_r \sqrt{R^2 - y_r^2} \tag{17}$$

where R is the magnitude of the residual access rate and $y_r$is the magnitude of the residual access rate. If R is held constant then the area becomes a function of θ. A plot of the function $\theta - \sin\theta$ is plotted in Figure 22. As theta approaches zero --that is, either the access rate or the space characteristic becomes balanced--the size of the area asymptotically approaches zero. Thus the ability to remove a data set becomes increasingly difficult as one of the loads approaches the balanced condition. Then, the only means for improving the balance of the other load is to add a data set to the device.

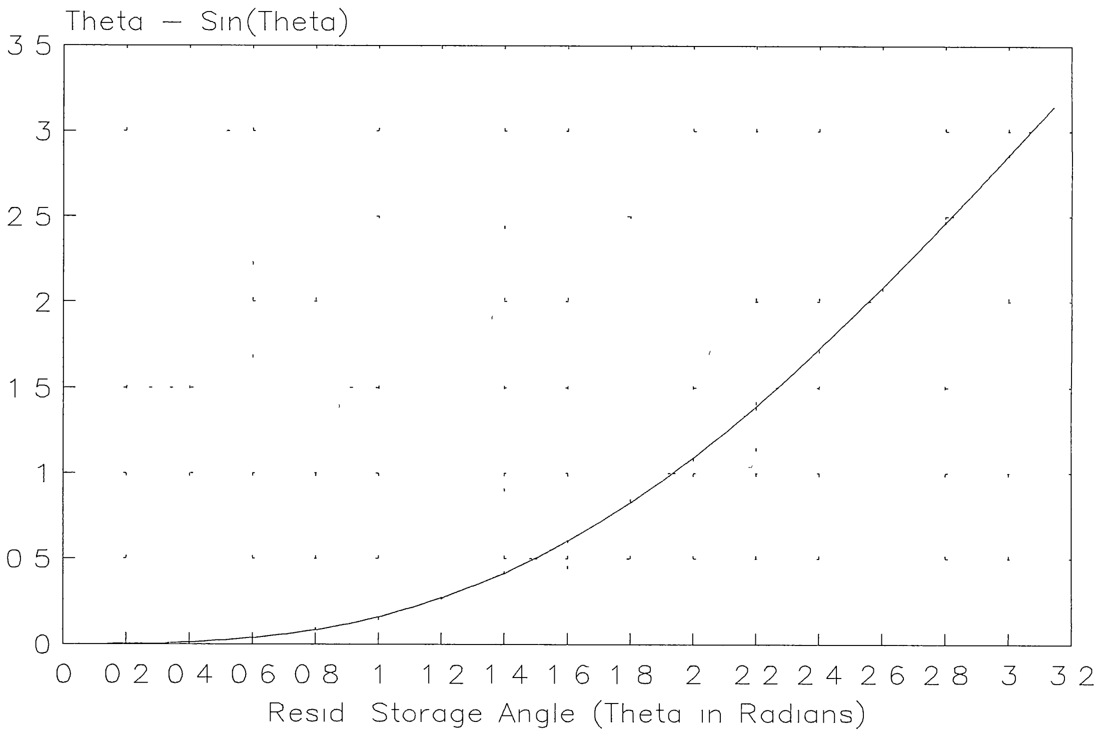


Figure 22. As Residual Storage Angle (Theta) Approaches Zero, the Potential for Data Set Removal, a Function of Theta - Sin(Theta), Asymptotically Approaches Zero.

The device labeled TSH011 is an example of a device with a slightly over-utilized space characteristic and a relatively high available access capacity. The 1580 megabytes stored on the device after balancing is approximately 188 megabytes more than the ideal 1392 megabtyes, or 13.5 percent more than the balanced condition. The 2.88 accesses per second are 2.37 accesses per second less than the ideal 5.247 accesses per second, or 45 percent less than the balanced condition. The balanced condition could not be improved because none of the remaining data sets on this device could be moved.

To illustrate the effect of the granularity of the data sets on the balancing, the data set on the RES018 device with an extremely high access rate was divided artificially into twenty separate data sets, each with equal space and load requirements. The system was then balanced, and the results are shown in Table 11 and 12 and Figure 23. Reducing the granularity of this data set to a level well below the amount of the virtual storage device improved the balancing significantly. The average reduction in the magnitude of the residual storage factor more than doubled to almost 83 percent. As expected, the data set on the TSH004 device continued to cause a problem. However, all other access loads appear to be very nearly balanced.

To aid in determining the changes to the actual access patterns that would have occurred if the data sets were actually moved, the accesses to each data set were simulated as though the data sets were stored on the devices assigned by the balancing procedure. Histogram plots of the access rates produced for each 15 second interval during the 15:00 to 16:00 p.m. time period were produced for each volume for both the prebalanced storage loads and the after-balancing storage loads. These are contained in Appendix C.

TABLE 11

DEVICE LOAD STORAGE FACTORS WITH SPLIT DATA SET AFTER BALANCING 15:00-16:00 P.M. TIME PERIOD

Volume	Accesses per Second	Megabytes of Storage
RES018	5.275	1242.24
TSH000	5.235	1565.79
TSH001	5.265	1279.99
TSH002	5.253	1368.57
TSH003	5.247	1424.19
TSH004	7.091	1385.35
TSH005	5.260	1200.19
TSH006	5.235	1492.57
TSH007	5.195	1403.89
TSH008	3.468	1392.19
TSH009	5.050	1512.39
TSH010	5.247	1391.84
TSH011	5.232	1542.96
TSH013	5.258	1313.27
TSH014	5.254	1328.83
USDD01	5.388	1328.13

TABLE 12

RESIDUAL STORAGE LOADS WITH SPLIT DATA SET BEFORE AND AFTER BALANCING 15:00-16:00 P.M.

Volume	Residual Storage Load Before Balancing	Residual Storage Load After Balancing	Residual Storage Load Difference	% Residual Storage Load Difference
TSH010	1.450	0.000	1.450	100.0%
TSH007	0.469	0.039	0.430	91.6%
TSH002	1.420	0.051	1.369	96.4%
TSH003	3.408	0.069	3.338	98.0%
TSH014	1.709	0.136	1.573	92.0%
USDD01	1.035	0.160	0.875	84.6%
TSH013	1.013	0.170	0.843	83.2%
TSH005	0.716	0.198	0.518	72.4%
TSH006	1.420	0.217	1.203	84.7%
TSH001	0.775	0.242	0.534	68.8%
TSH009	2.040	0.238	1.757	86.1%
RES018	6.512	0.323	6.188	95.0%
TSH011	1.417	0.325	1.091	77.0%
TSH000	0.747	0.375	0.372	49.9%
TSH008	1.477	1.017	0.460	31.2%
TSH004	1.712	1.054	0.658	38.5%
Average	1.707	0.291	1.416	82.9%

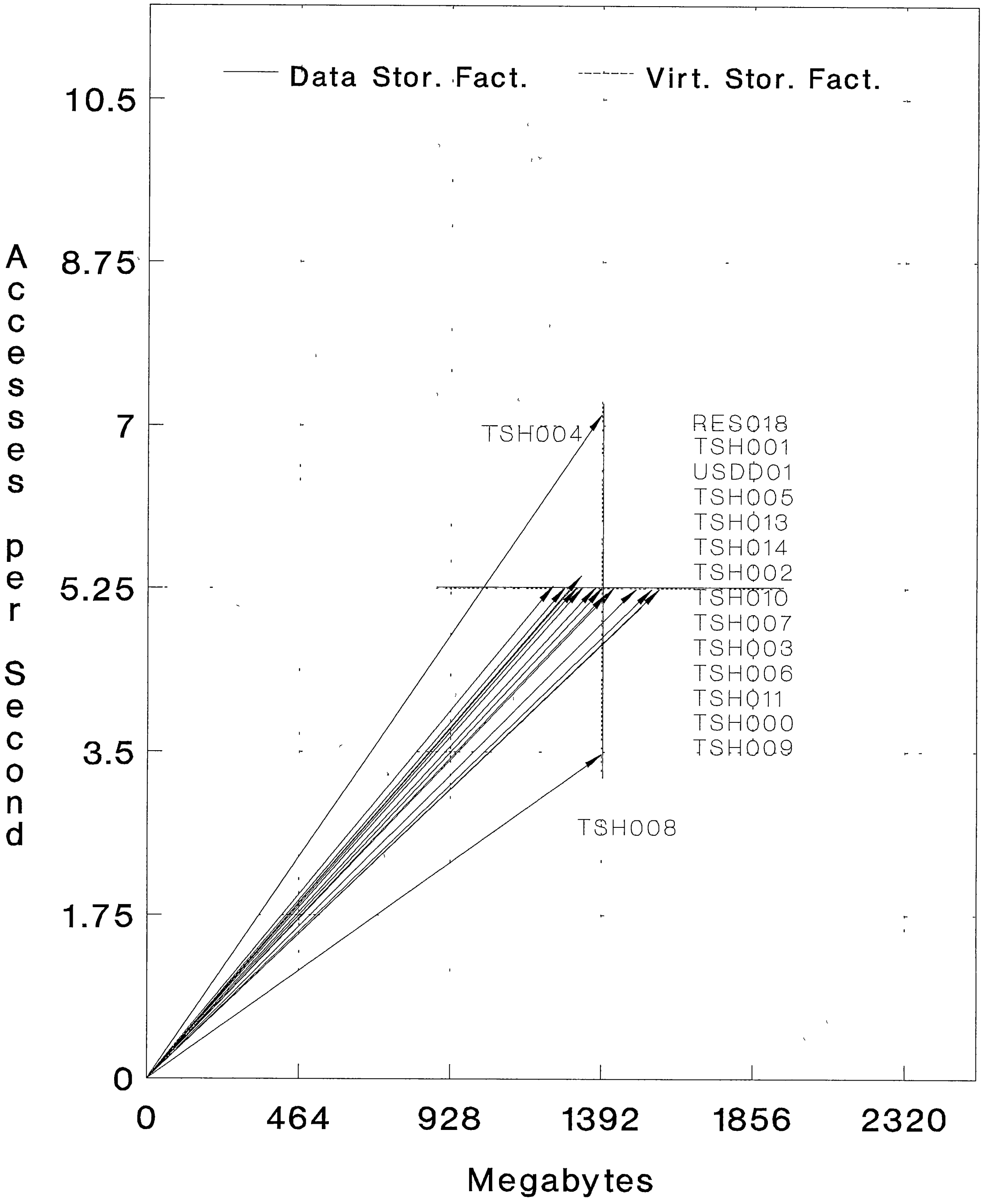


Figure 23. Balanced Storage Factors with Split Data Set

When the plots for the unbalanced loads are compared to the plots for the balanced loads, it appears that changes in the histograms were, as expected, dependent on the degree and direction of the out of balance condition and on the granularity of the data. The device labeled TSH014, for example, was lightly loaded prior to balancing. To increase its load, several data sets with very high access densities were moved to this volume. At the same time, all of the data sets with lower access densities have been removed from volume TSH003 to bring its overall access rate in line with the average. This left only a few high access density data sets on volume TSH003. Of course, storing only a few high access density data sets would assist in assuring reasonable response times since the full access capacity for the device would be dedicated to servicing the data sets that require most, if not all, of the access capabilities of the device.

Thus, the data set movements did not eliminate the irregularity of the access demand across the time period. But, as was its intent, the total demand was balanced with the capabilities of the device. Where large access requirements were necessary for a small number of data sets, these needs were accommodated by eliminating smaller access requirements and allowing some longer periods when the device was almost idle. This is somewhat disturbing at first, but when evaluated, is reasonable because it is not possible to predict the exact time when the high access capabilities will be required; therefore, the capacity must remain available if delays are to be avoided.

CHAPTER VI

CONCLUSIONS AND SUGGESTIONS FOR FUTURE STUDY

The method proposed in this thesis for balancing storage subsystems has the theoretical capability of eliminating DASD skew within a storage subsystem. This was shown in the proof in chapter 3. The method, however, is constrained by the granularity of the data sets stored in the subsystem. The larger the space and access requirements of a data set, the more difficult it can be to eliminate imbalances in the system. In fact, if the granularity is large enough, it may be impossible to balance the system totally. However, the methods proposed here will deal with these constraints as effectively as possible. This was demonstrated in the case study by the handling of the TSH004 and RES018 devices. Both devices had very high access loads, and though these loads were created by single data sets on each device, the balancing procedure removed nearly every other data set from these devices in an effort to bring the access load to a balanced condition.

Also of significance is that the tuning method is automated. It can be integrated into an operating system, which, during idle periods, can then analyze the storage subsystem, and then, using these methods, improve the balance of the loads across all of the devices in the storage subsystem. Automating these procedures should significantly reduce the need for manual tuning efforts and thus reduce costs. It should also increase the performance of the system by allowing the system to maintain itself in a more optimally configured state. As the system loads and the system configuration change, the system then can adjust itself to these changing conditions automatically.

There are, however, some difficulties that need further study. One of the most difficult problems is characterizing the access requirements of data sets. For large production systems such as banking systems, reservations systems, tracking systems, etc., or high usage data sets such as indexes, etc., usage patterns are somewhat consistent. Other data sets, however, may be more difficult to analyze. On TSO, for example, usage patterns for data sets often fluctuate dramatically. Thus, a data set may have had 5 accesses per second most of the time but on a couple of occasions may have had 25 accesses/second. Also, some data sets may only be used once a month, others several times a month, and a few may be used every day. This variation in usage must be reduced statistically to a single access characteristic.

Though a simple algorithm has been suggested for determining a data set's access characteristic, additional work must be done to determine whether there are better methods for characterizing the access requirements of a data set. Several storage subsystems must be analyzed over a period of several weeks or months to determine whether the data exhibits statistically predictable usage patterns. Using statistical predictions with defined confidence limits would allow system managers to designate the level of performance that must be assured. However, high confidence levels would tend to increase the required access capacity for the data and, thus, increase the capacity requirements of the subsystem. Of course, more reliable predictions of data set usage will produce more reliable results from the balancing work.

Additional work also must be done with balancing the system over more than one time period. This study only looked at the 15:00 to 16:00 p.m. time period. 10:00-11:00 a.m. was also a high use period. The algorithms developed here must be refined to include the effects of balancing on other time periods. For example, moving a data set to a different device may not only affect performance for the period being analyzed, but it also may have an impact on a different time period since the data set may have been accessed during that time period also. This could complicate the balancing effort considerably. Perhaps one approach could be to balance a time period until it no longer has the largest residual storage factor then switch to the time period that has become the most out-of-balance and balance it until it's out-of-balance condition is reduced below another device. This could be repeated until no further balancing is possible.

Another area that needs further study is the balancing algorithm itself. The suggested algorithm is a "greedy" type algorithm. The data set with an access density closest to that specified for the out-of-balance condition is selected for movement. It is not clear whether this results in the best solution in terms of minimizing the out-of-balance condition within the constraints of the granularity of the data. In the case study, the solution appears to be optimal since the balancing result was constrained by two data sets that exceeded the capacity of the virtual device. But, this may not be the case under all circumstances.

One possible alternative algorithm that may result in a better solution is to consider the movement of several data sets as though they are one data set. Multiple data sets may have an aggregate characteristic completely different than the characteristics of each individual data set. An algorithm using this technique may be able to provide movement options which otherwise would be invalidated because movement of one of the individual data sets would increase the out-of-balance condition.

It is also unclear whether the suggested algorithm uses the least possible number of data set moves to achieve a balanced subsystem. Most desirable would be an algorithm that determined which data set moves would require the minimum number of moves to produce the maximum reduction in the residual storage factor. This approach would assure that data sets did not move more than once and that only the minimum number of data sets were moved.

With this objective in mind, an alternative algorithm was investigated briefly. In this algorithm, the data set that resulted in the greatest reduction in the residual storage factor was moved. However, this approach produced significantly longer processing times due to the exhaustive search required to determine the data set that should be moved. Consequently, the effort was abandoned.

Despite these difficulties, storage factors provide a significant opportunity to improve the tuning of large storage subsystems. Effective balancing of both the space requirements and the access requirements with the capacities of the storage devices is achievable using the techniques proposed here. Additionally, the balancing can be managed by the computer system itself, allowing the system to adjust automatically to changing conditions, and thus, improve performance and reduce costs. With balanced loads, problems created by DASD skew are eliminated producing a highly efficient and cost effective storage subsystem.

REFERENCES

1. Allen, Arnold O., "Queueing Models of Computer Systems," IEEE Computer, Vol. 13, No. 4, April 1980, pp. 13-24.

2. Baker, Michael G., "DASD Tuning - Understanding the Basics", Computer Measurement Group Conference Proceedings, 1989, pp. 1251-1257.

3. Barkataki, Shan, "A CPE Project in a Fast Transaction Processing Environment", Computer Measurement Group Conference Proceedings, 1984, pp. 9-13.

4. Beretvas, Thomas, "DASD and Cache Performance Analysis Using Modeling", Computer Measurement Group Conference Proceedings, 1988, pp. 1017-1035.

5. Beretvas, Thomas, "DASD Performance Analysis Using Modeling", Computer Measurement Group Transactions, Vol. 54, Fall 1986, pp. 33-43.

6. Beretvas, T. "Performance Tuning in OS/VS2 MVS", IBM Systems Journal, Vol. 17, No. 3, 1978, pp. 290-313.

7. Brandwajn, Alexandre, "Load Imbalance in DASD Dynamic Reconnection", Computer Measurement Group Transactions, Winter 1988, pp. 61-69.

8. Buzen, J. P., and Shum, A. W., "I/O Performance Trade-Offs and MVS/ESA Considerations", Computer Measurement Group Conference Proceedings, 1990, pp. 695-702.

9. Cheung, Duncan, "Storage Strategy", Computer Measurement Group Conference Proceedings, 1986, pp. 333-336.

10. Duhl, B., "Notes on Performance Considerations for Single and Dual Capacity DASD", Computer Measurement Group Conference Proceedings, 1986, pp. 710-715.

11. Friedman, Mark B., "DASD Access Patterns", Computer Measurement Group Conference Proceedings, 1983, pp. 51-61.

12. Gelb, J. P., "System Managed Storage", IBM Systems Journal, Vol. 28, No. 1, 1989, pp. 77-103.

13. Gray, W. T., "The Use of Analytic Modeling To Compare Various I/O Configurations", Computer Measurement Group Transactions, Vol. 62, Fall 1988, pp. 71-85.

14. Griffith, David. P., "Combining DASD Volumes Without Losing Performance", Enterprise Systems Journal, Vol. 8, No. 10, 1991, pp. 84-90.

15. Hill, Reed A., "Access Density -- A Data Storage Figure of Merit", IBM Technical Report, January 1980, 16 pages.

16. Jones, A. L., "A Strategy for Improving I/O Subsystem Design Using Cost Per I/O Analysis," Computer Measurement Group Conference Proceedings, 1987, pp. 672-684.

17. Levy, Kenneth R., "A Simulation Model for Determining Optimal Freespace Levels on MVS Disk Storage Volumes", Computer Measurement Group Conference Proceedings, 1987, pp. 217-221.

18. Liu, Cathy, and Papy, Wayne, "The DASD Tuning & Planning Challenge," Computer Measurement Group Conference Proceedings, 1986, pp. 492-496.

19. Major, J. B., "Processor, I/O Path, and DASD Configuration Capacity", IBM Systems Journal, Vol. 20, No. 1, 1981, pp. 63-85.

20. McNutt, Bruce, "An Empirical Study of Variations in DASD Volume Activity", Computer Measurement Group Conference Proceedings, 1986, pp. 274-283.

21. McNutt, Bruce, "Large Capacity DASD: Is Performance Something to Be Afraid Of?," Computer Measurement Group Conference Proceedings, 1987, pp. 399-404.

22. Merrill, H. W., Merrill's Expanded Guide to Computer Performance Using the SAS System, SAS Institute Inc., Cary, NC, 1984, pp. 482-498.

23. Mungal, Anthony G., "Availability and Performance Considerations of the I/O Subsystem", Computer Measurement Group Conference Proceedings, 1991, pp. 665-674.

24. MVS/Extended Architecture Systems Programming Library: System Management Facilities(SMF), GC28-1153-7, IBM Corp., 8th Edition, Sept. 1989., pp. 7-36 to 7-42.

25. Olcott, Rich, "I/O Tuning in the MVS/XA Environment," Computer Measurement Group Conference Proceedings, 1987, pp. 889-900.

26. Papy, Wayne, "DASD I/O Performance Tuning: A Case Study of Techniques and Results," Computer Measurement Group Conference Proceedings, 1988, pp. 665-671.

27. Papy, Wayne, "Performance Implications of System Dataset Placement", Computer Measurement Group Conference Proceedings, 1989, pp. 1197-1202.

28. Papy, Wayne, "DASD Storage, I/O Performance and SMS", Computer Measurement Group Conference Proceedings, 1990, pp. 998-1003.

29. Piepmeier, William F., "Optimal Balancing of I/O Requests to Disks", Communications of the ACM, Vol. 18, No. 9, September 1975, pp. 524-527.

30. Schardt, R. M., "An MVS Tuning Approach", IBM Systems Journal, Vol. 19, No. 1, 1980, pp. 102-119.

31. Selby, Samuel M. (Editor), Standard Mathematical Tables, The Chemical Rubber Co., Cleveland, Ohio, 19th Edition, 1971, pp. 12, 538-539.

32. Singh, Y., King, G. M., and Anderson, J. W., "IBM 3090 Performance: A Balanced System Approach," IBM Systems Journal, Jan. 1986, Vol. 25, No. 1, pp. 20-35.

33. Smith, A. J., "Input/Output optimization and disk architecture", Performance Evaluation, Vol. 1 No. 2 (May 1981), pp. 104-117.

34. Wilmot, R. B., "File Usage Patterns from SMF Data: Highly Skewed Usage", Computer Measurement Group Conference Proceedings, 1989, pp. 668-677.

35. Wong, Angela S. O., and Chanson, Samuel T., "A Fully Automatic Analytic Approach to Budget-Constrained System Upgrade", Computer Measurement Group Conference Proceedings, 1987, pp. 620-629.

36. IBM Storage Subsystem Library: 3380 Direct Access Storage Introduction, GC26-4491-1, IBM Corp., 2nd Edition, 1989, pp. 9-11.

APPENDICES

APPENDIX A

GLOSSARY

GLOSSARY

Access Density -- The relationship of the accesses per second to the storage capacity of a device or a data set expressed as Accesses/Sec/MB.

Aggregate Access Density -- The access density of a group of devices or data sets expresses the slope of the storage factor vector representing a group of storage devices or a group of data sets.

Aggregate Storage Load -- The aggregate storage load is the magnitude of the storage factor vector representing a group of storage devices or a group of data sets. The storage factor vector is determined by summing the storage factor vectors for each member of the group.

DASD -- Direct Access Storage Device. Usually refers to disk storage devices but can refer to solid state storage devices as well.

Device Storage Factor -- The vector representing the maximum sustainable access capacity and the storage space capacity available on each device, less any required space for future growth and temporary data sets.

Device Load Storage Factor -- The aggregate storage factor of all the data sets stored on a single device.

EXCP -- Execute Channel Program. In large IBM systems, command issued by the CPU that initiates an I/O request.

Granularity of Data Sets -- Differences in data set access and space characteristics occur in incremental steps rather than according to functions based on smooth curves.

IBM -- International Business Machines Corporation.

I/O -- Input/Output. Usually refers to the reading and writing of data from a peripheral storage device.

Latency Time -- Time required for the proper data sector to rotate under the read/write head during an I/O operation. This is a function of the rotational speed of the disk.

M/M/1 -- A symbolic representation of a queueing system. This represents a queue with an exponential interarrival time distribution, an exponential service time distribution, and one server.

MVS -- Multiple Virtual Storage. An operating system for large systems related to the OS/VS2 operating system.

MVS S/370 -- Multiple Virtual Storage, System 370. An MVS operating system used on the IBM System 370.

MVS X/A -- Multiple Virtual Storage, Extended Architecture. A more advanced version of the MVS operating system from IBM.

OS/VS2 -- Operating System/Virtual Storage 2. A virtual storage operating system from IBM used on their large systems.

Queueing Factor -- Based on queueing theory, a factor that reduces a storage device's sustainable access rate from the maximum capability of the device. This reflects the distributed arrival rate of I/O requests and the need to maintain acceptable service times.

Queueing Models -- Mathematical descriptions of a server. Includes expected arrival rate distribution function, queue lengths, service times, etc.

Residual Storage Factor -- The difference between the aggregate device storage factor and the aggregate data storage factor.

RPS -- Rotational Position Sensing. The process of releasing the path during head positioning on a rotating disk device and then reconnecting to the path as proper positioning approaches.

RPS Miss - The path is busy during the process of attempting to reconnect to the path as part of the Rotational Position Sensing. The disk must then rotate one full revolution before an attempt to reconnect can be made again.

Scratch Space -- Space available on a storage device for use as temporary storage space during a process.

Seek Time -- The time required to move a read/write head to the proper track on a rotating disk storage device, usually expressed in milliseconds.

Skew -- Describes the wide variation of utilization of the storage devices in the storage subsystem.

SMF -- System Management Facilities. An IBM system providing performance information for evaluating large IBM computer systems.

SMS -- System Managed Storage. Refers to a group of IBM software products that are designed to perform certain management functions within the storage subsystems of its large mainframe systems.

Storage Device -- The storage media, access mechanisms, and controlling electronics that, when combined appropriately, can store and retrieve data. In this paper, a storage device usually refers to a disk storage device. Frequently, these are referred to as actuators. Users usually refer to these as volumes. IBM often includes multiple volumes or actuators in a single box.

Storage Factor -- The vector representation of the storage characteristics of a storage device or a data set. The components of the vector are the amount of storage space and the access rate.

Subsystem Load Storage Factor -- The aggregate storage factor of all the data sets stored in the subsystem.

Subsystem Capacity Storage Factor -- The aggregate storage factor for each device in the storage subsystem.

TSO -- Time Sharing Option. This is the interactive multi-user option on large IBM systems.

Storage Subsystem Tuning -- The process of modifying system configuration and data set placement to improve storage subsystem performance and reduce costs.

Virtual Device Storage Factor -- The storage factor based on the weighted distribution of the total storage load which defines the desired balanced load rather than the actual device capacities.

Volume -- A term synonymous with storage device.

VSAM -- Virtual Storage Access Method. An access method used by IBM on direct access storage devices which allows organization of the data set by key field, sequentially, or by relative record number.

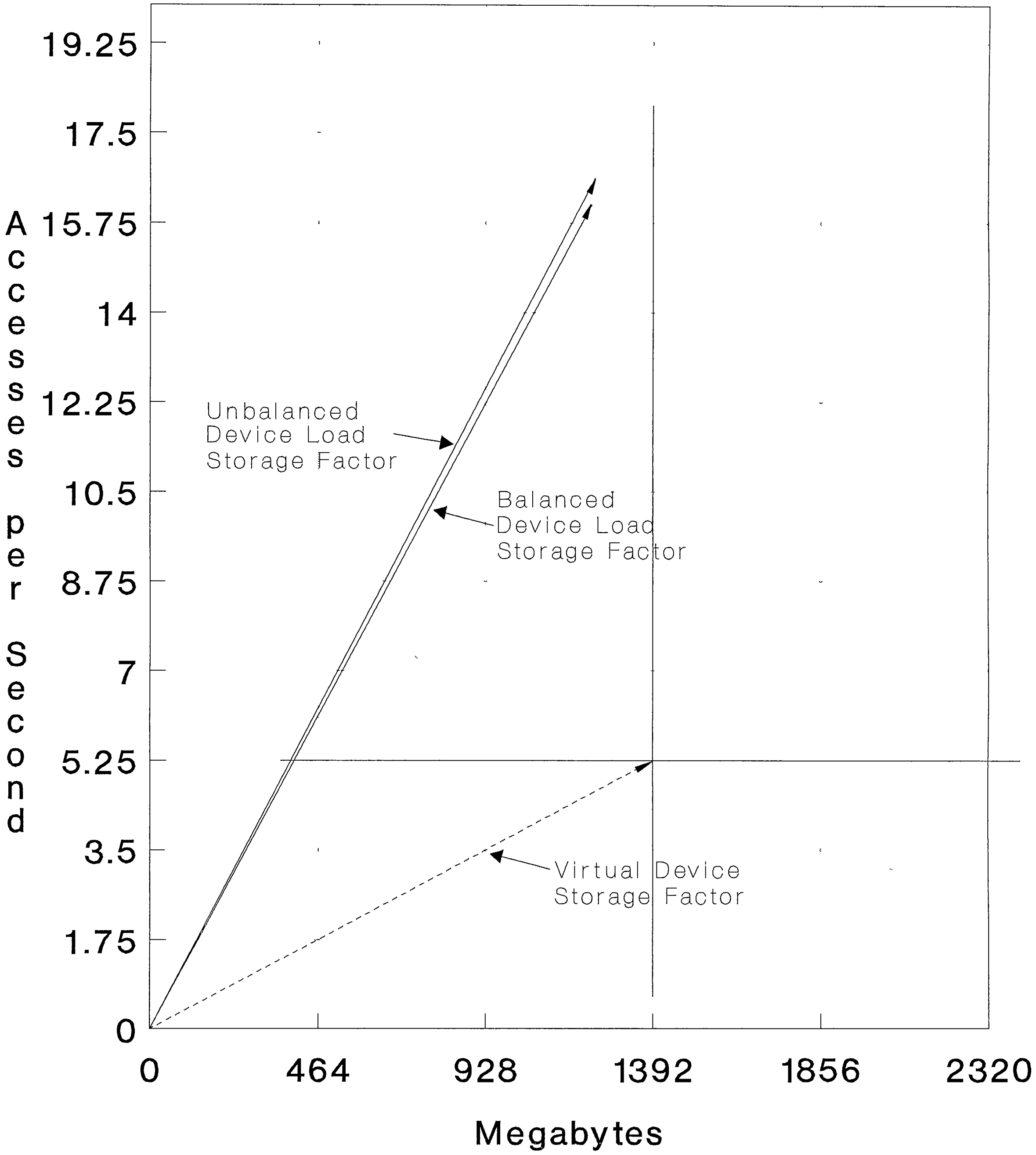
VTOC -- Volume Table of Contents. A table on a storage device that stores information about each data set on the device.

APPENDIX B

STORAGE FACTORS BEFORE AND AFTER BALANCING BY DEVICE

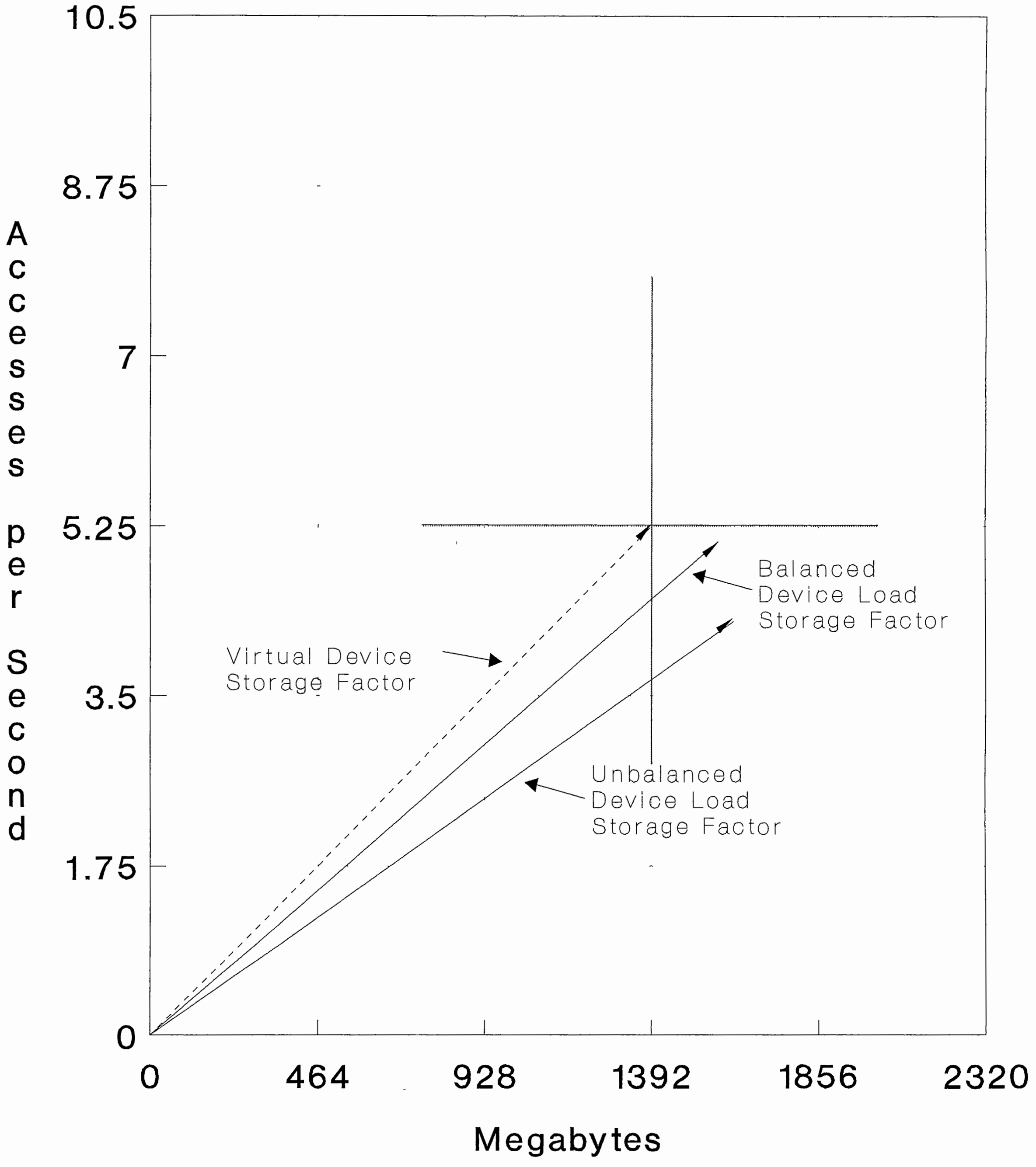
15:00-16:00 P.M.

DEVICE LOAD STOR. FACT.--RES018
15:00-16:00 p.m.



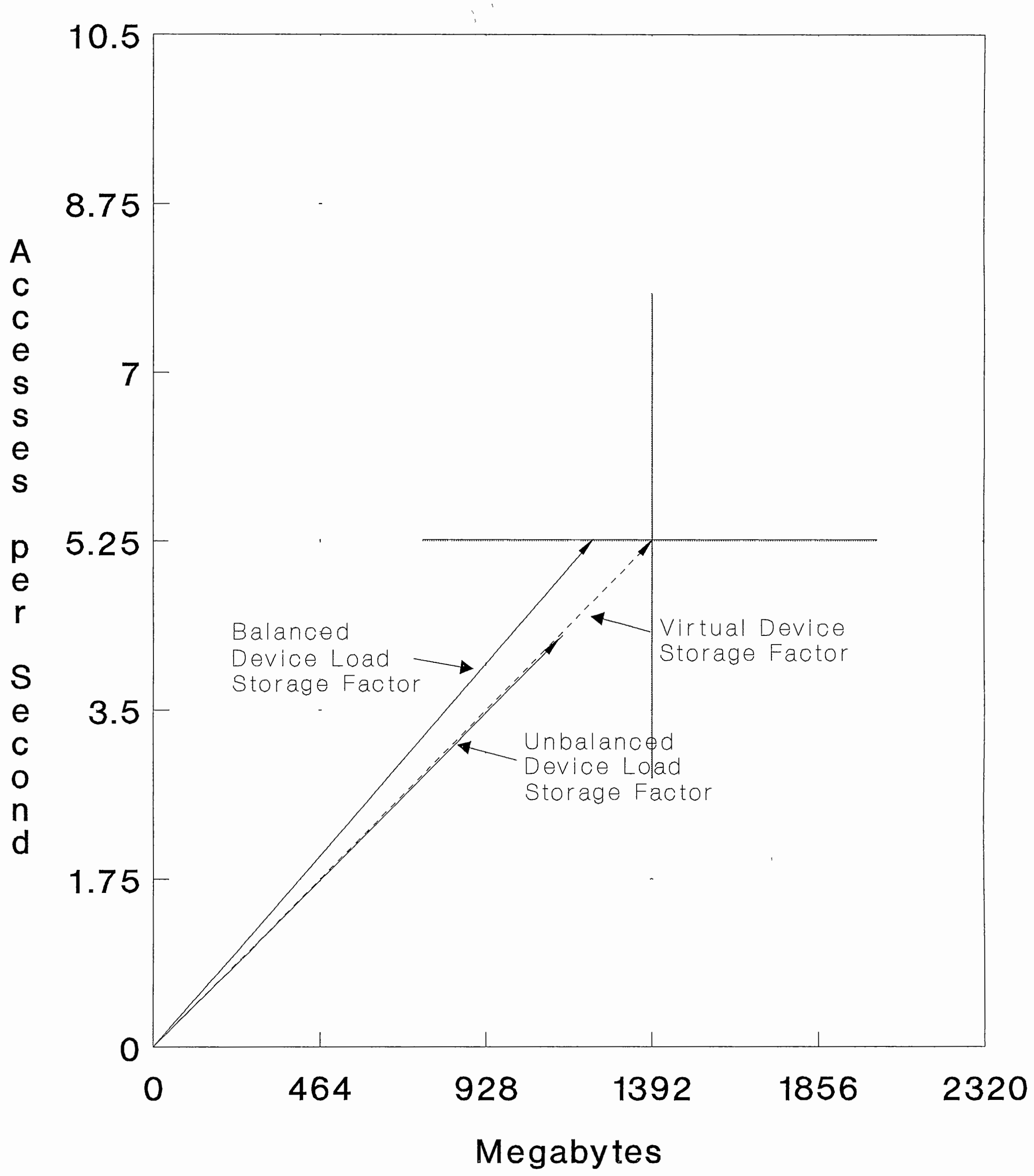
DEVICE LOAD STOR. FACT.--TSH000

15:00-16:00 p.m.

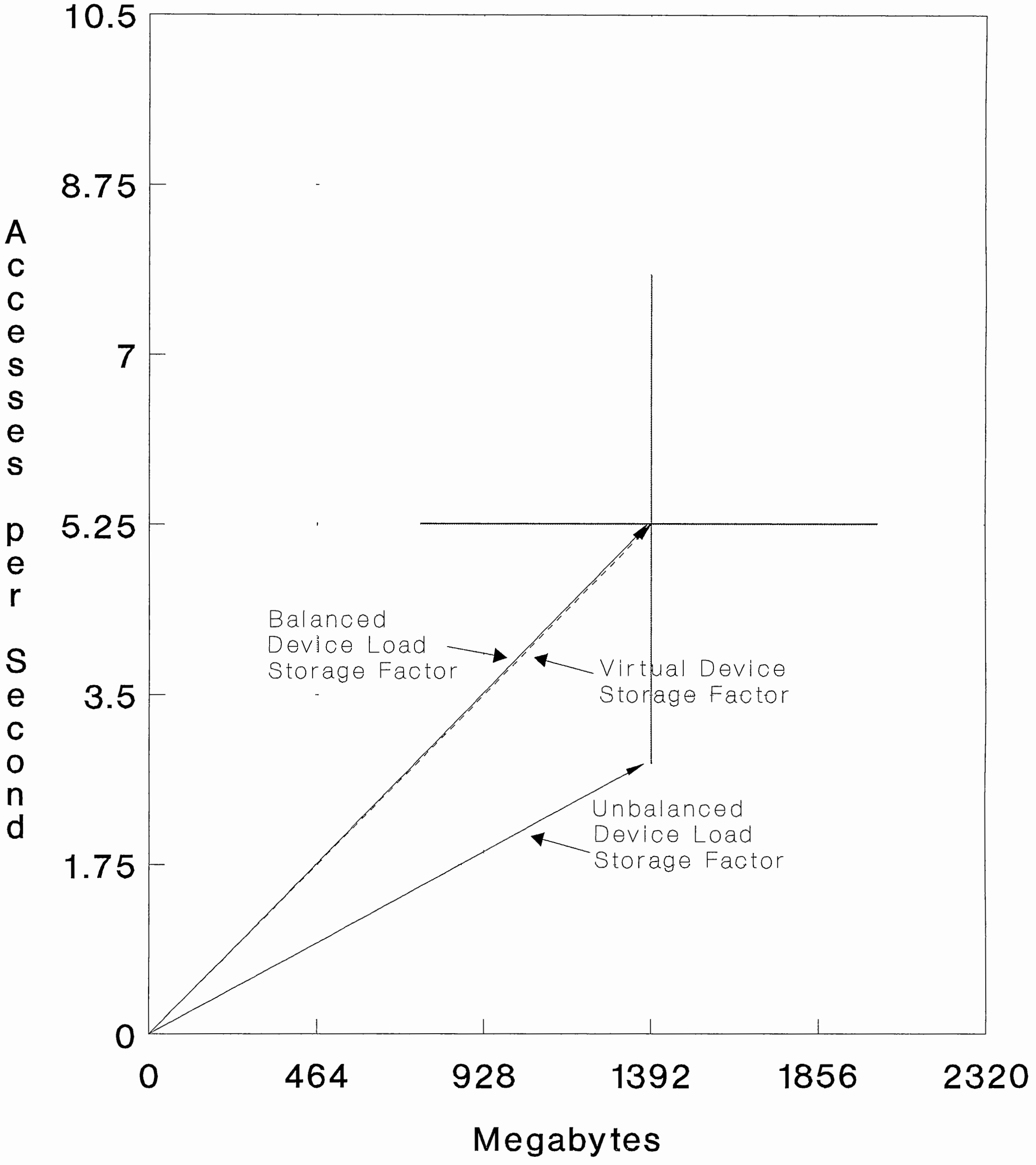


DEVICE LOAD STOR. FACT.--TSH001

15:00-16:00 p.m.

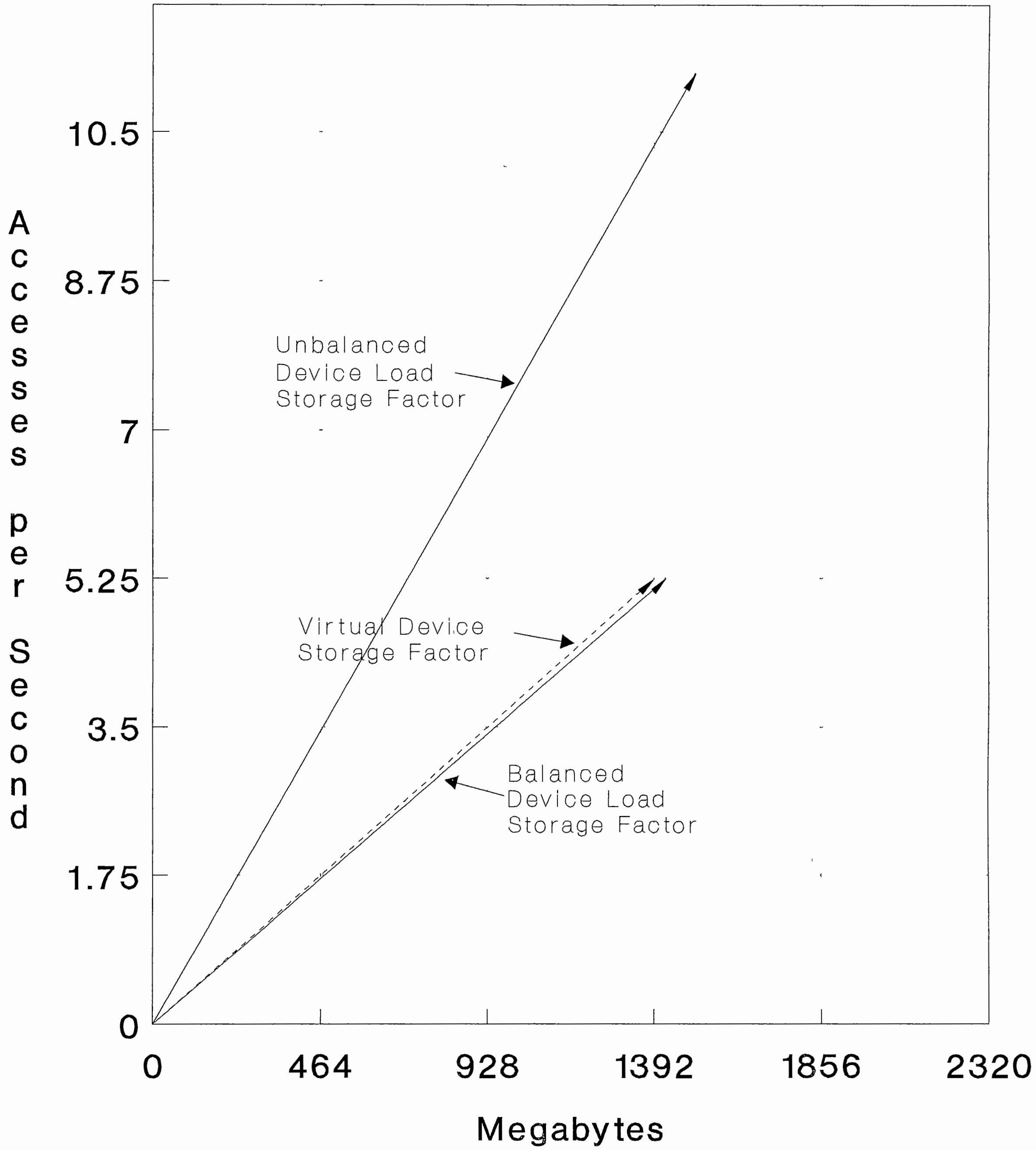


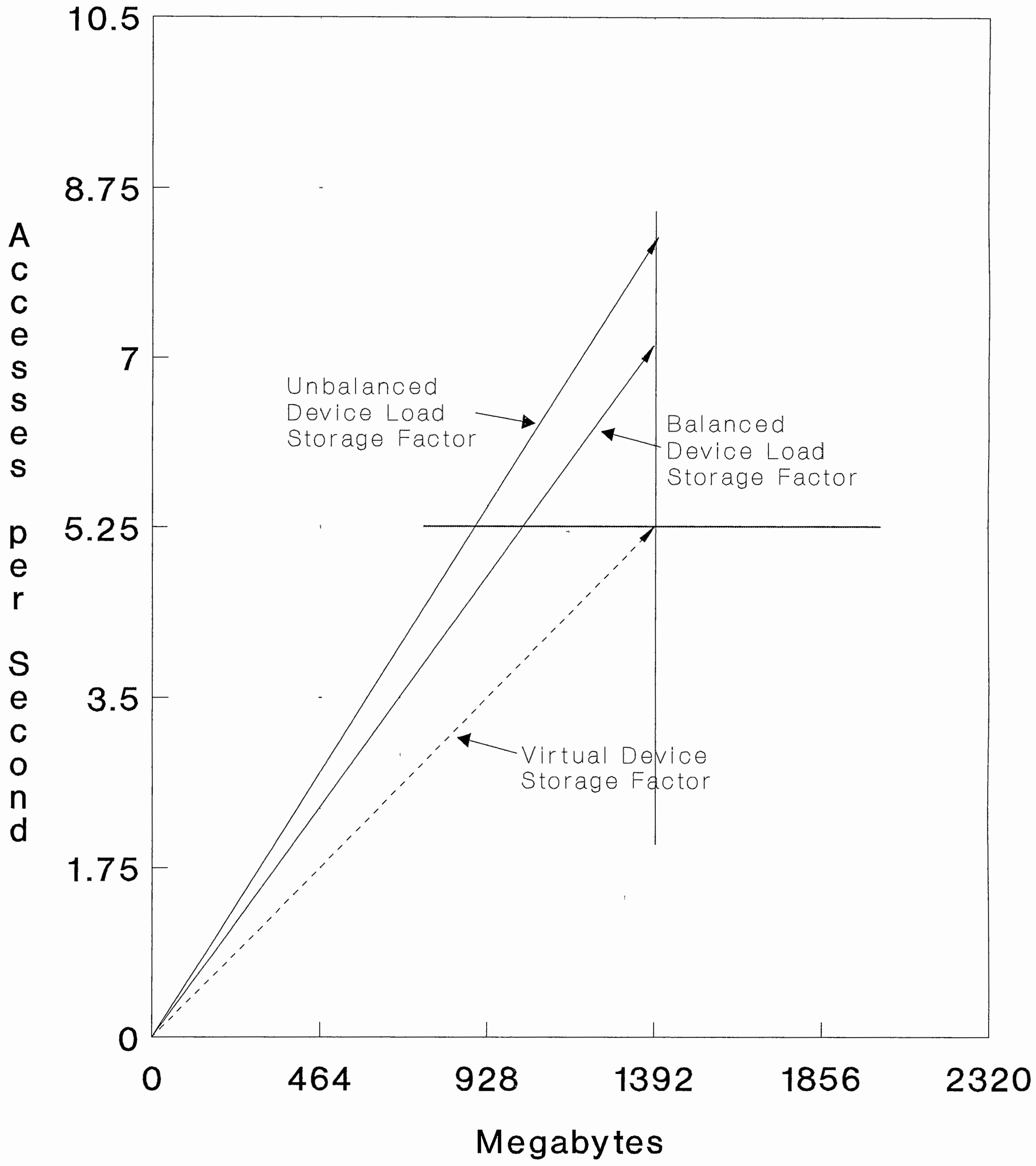
DEVICE LOAD STOR. FACT.--TSH002
15:00-16:00 p.m.



DEVICE LOAD STOR. FACT.--TSH003

15:00-16:00 p.m.



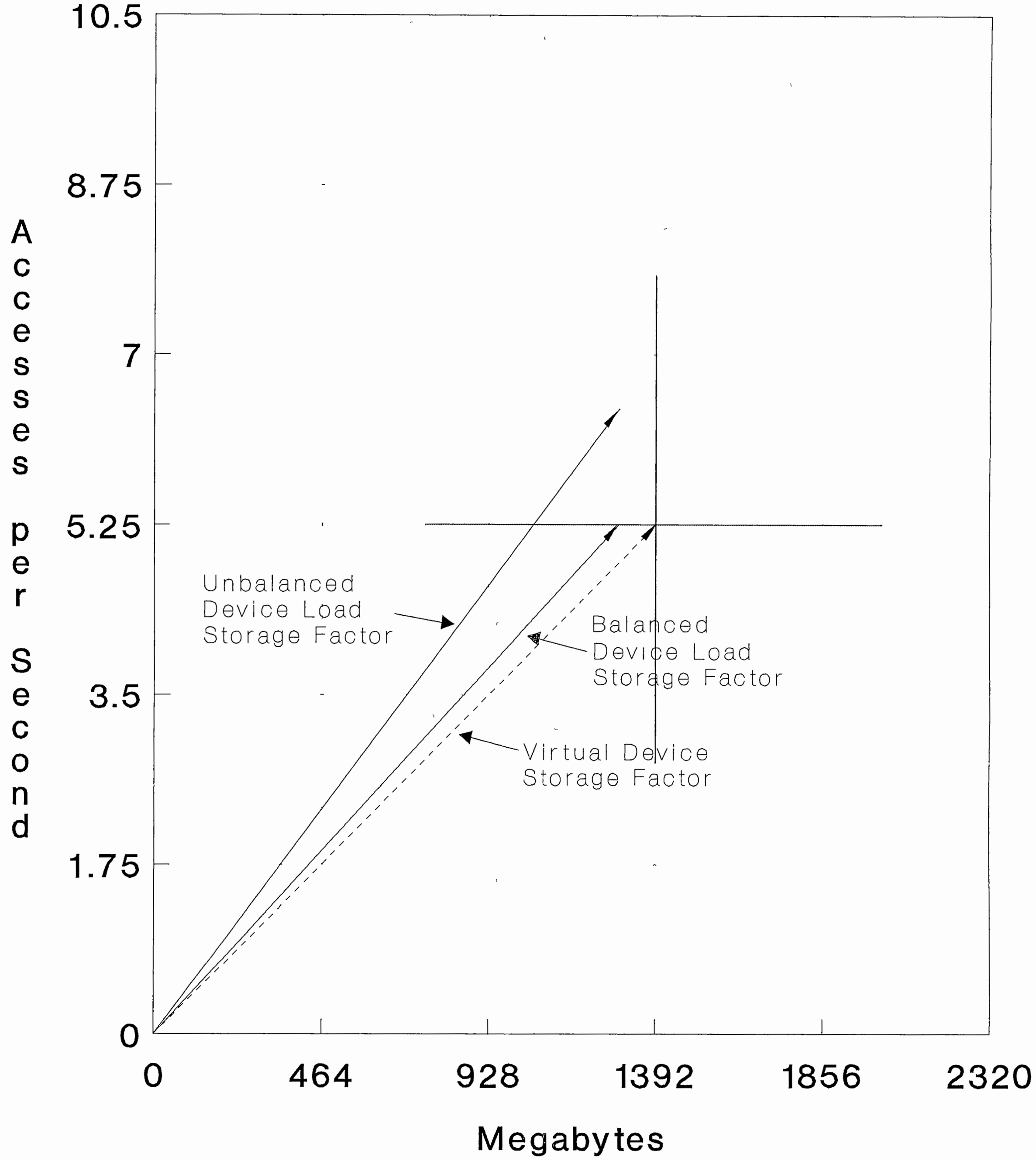


DEVICE LOAD STOR. FACT.--TSH004

15:00-16:00 p.m.

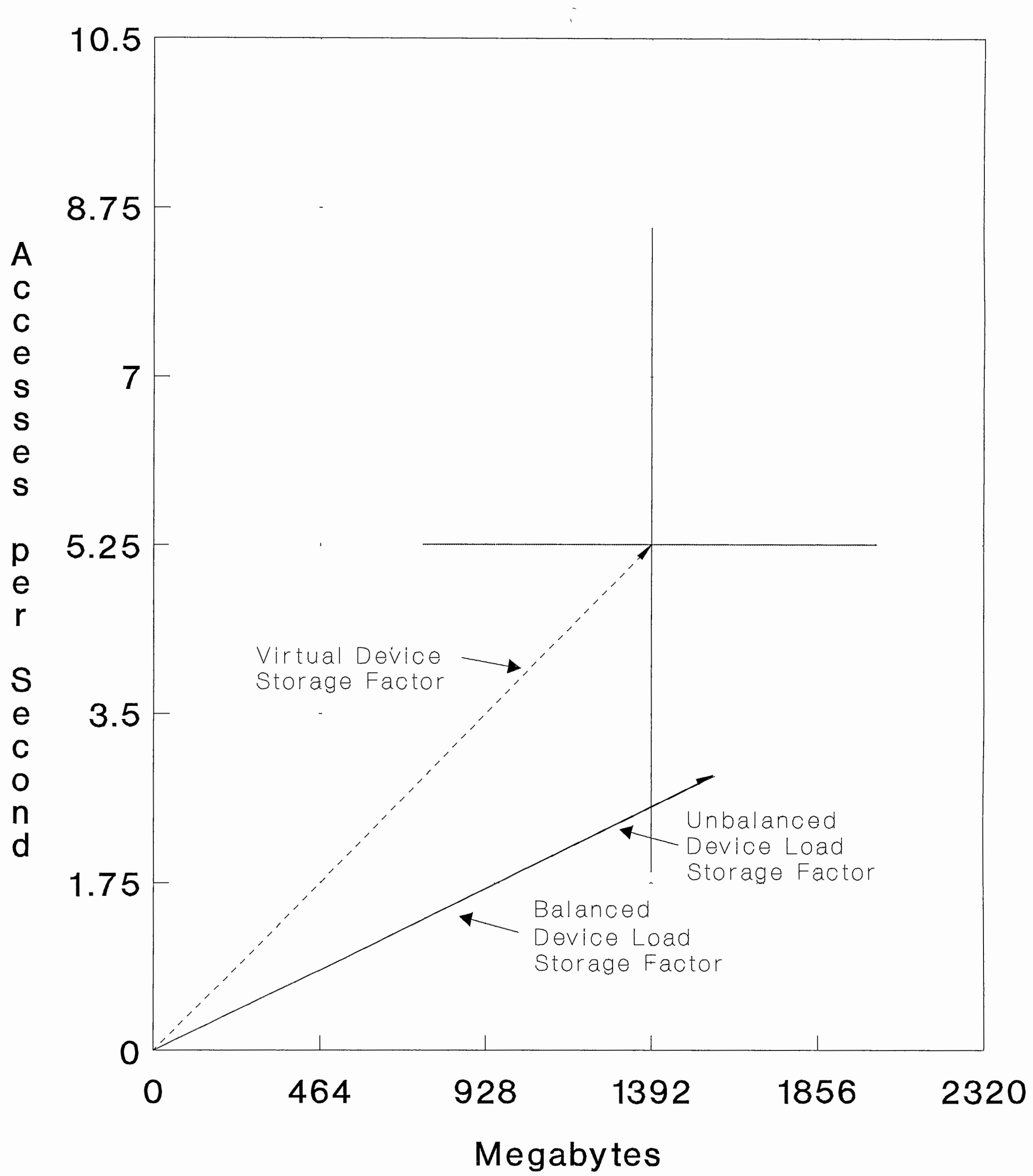
DEVICE LOAD STOR. FACT.--TSH005

15:00-16:00 p.m.



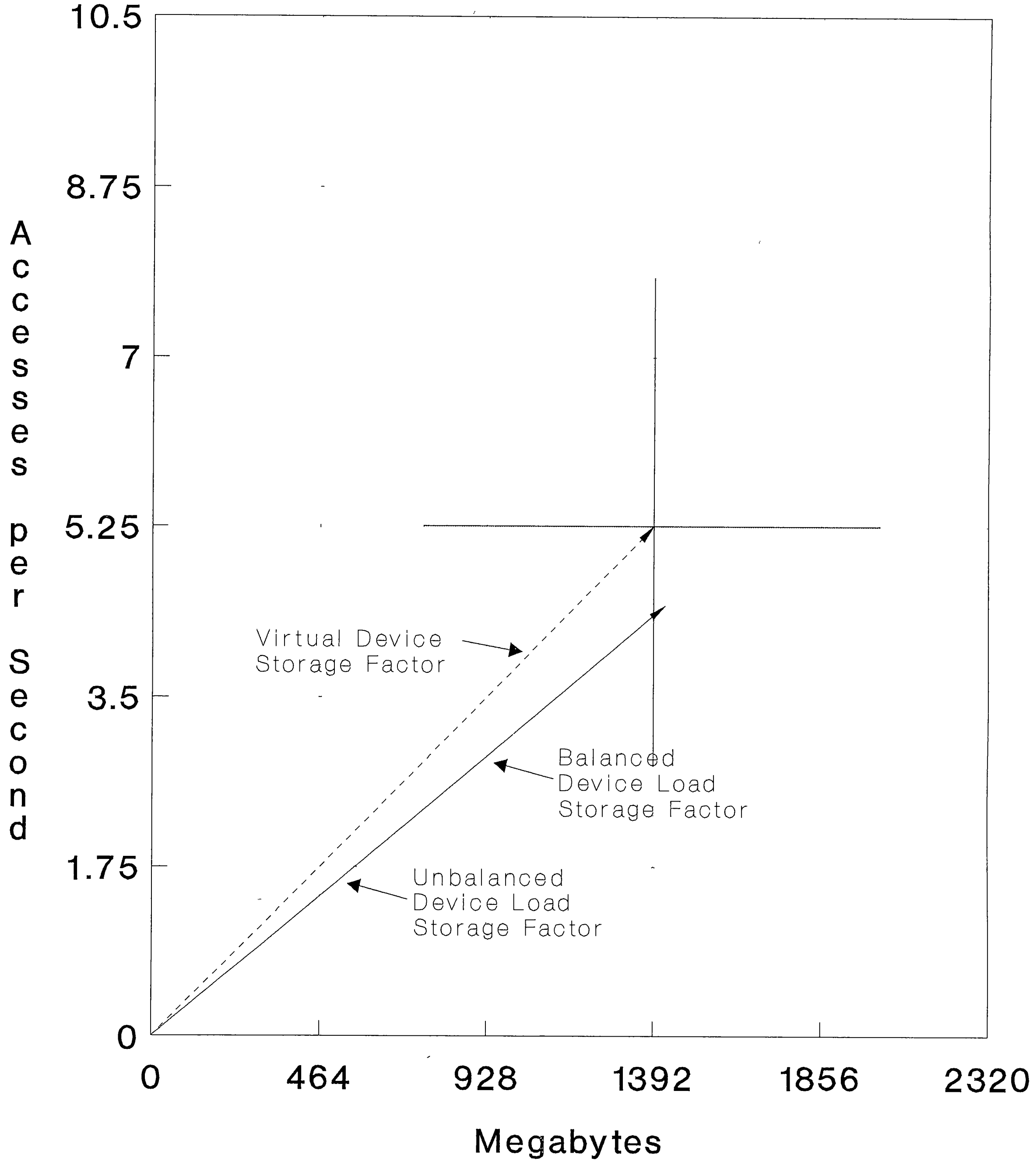
DEVICE LOAD STOR. FACT.--TSH006

15:00-16:00 p.m.



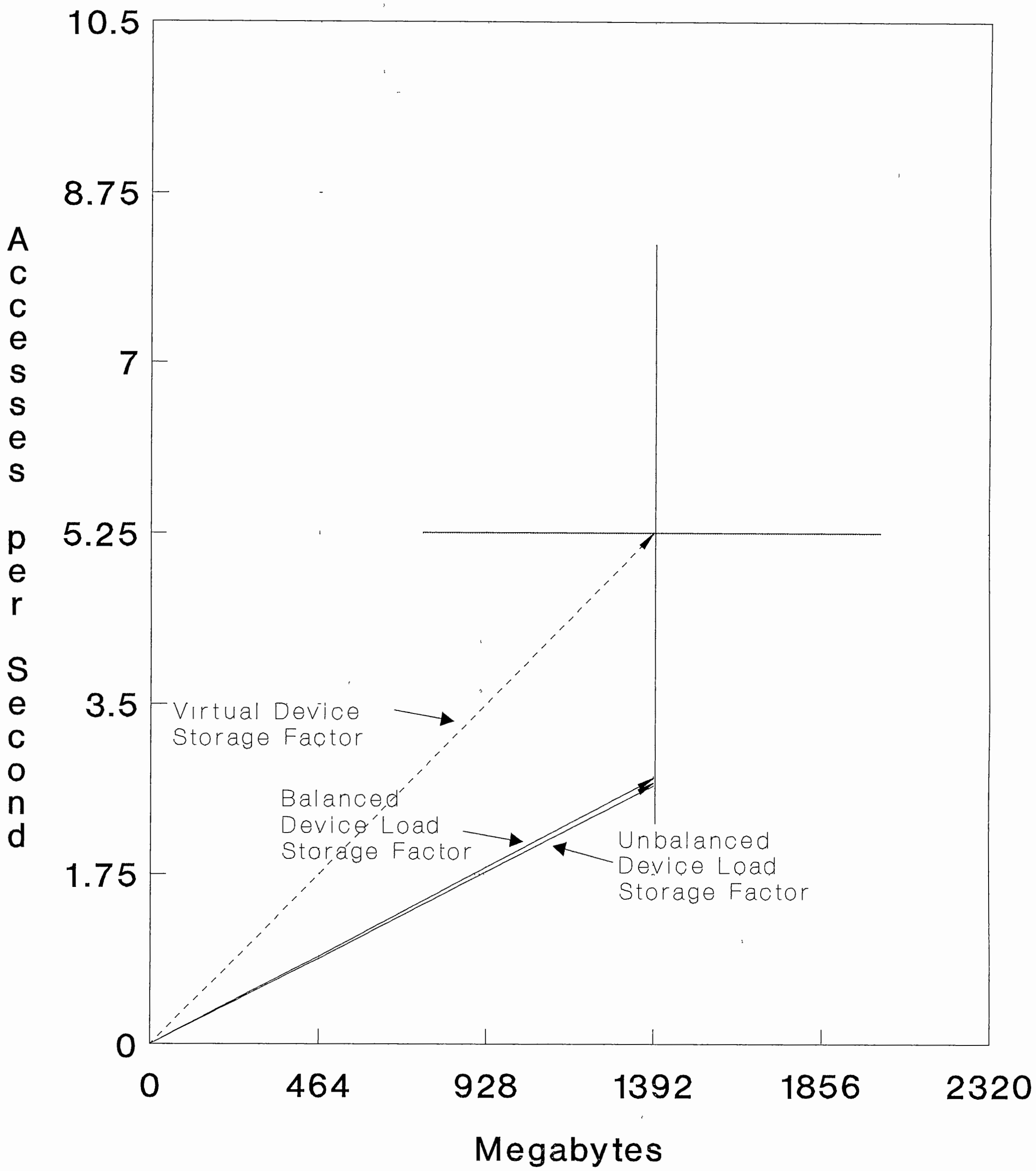
DEVICE LOAD STOR. FACT.--TSH007

15:00-16:00 p.m.



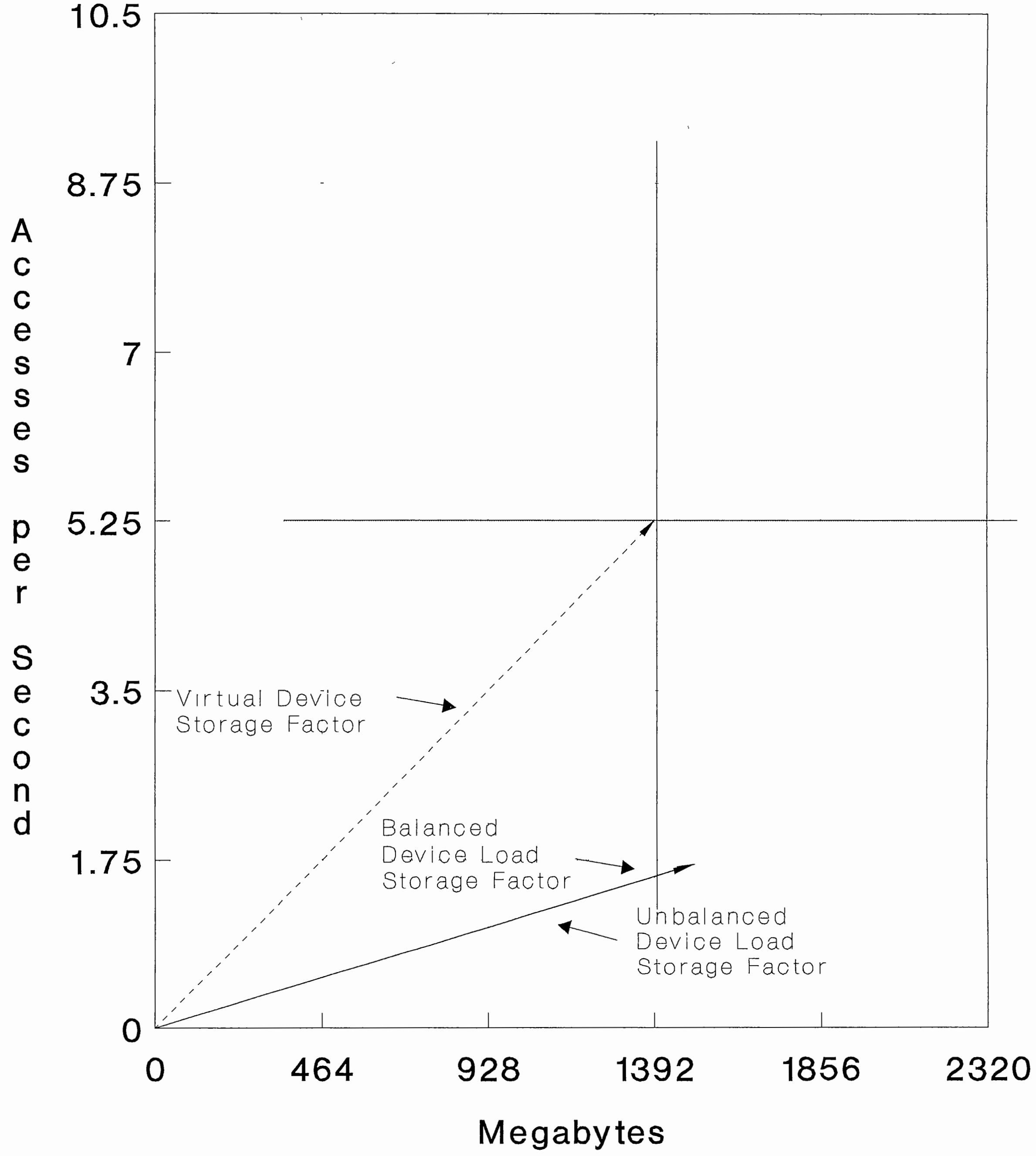
DEVICE LOAD STOR. FACT.--TSH008

15:00-16:00 p.m.



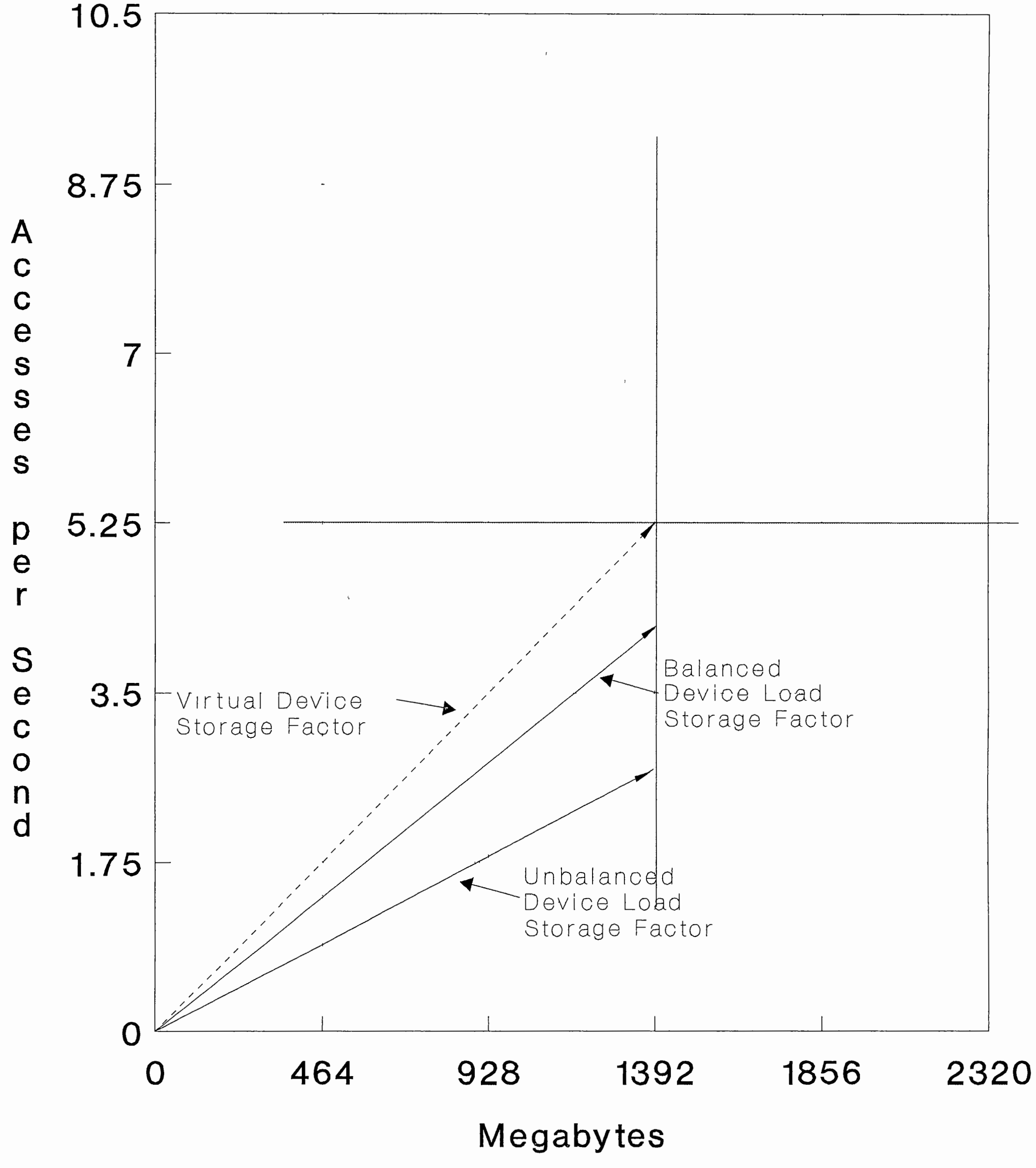
DEVICE LOAD STOR. FACT.--TSH009

15:00-16:00 p.m.



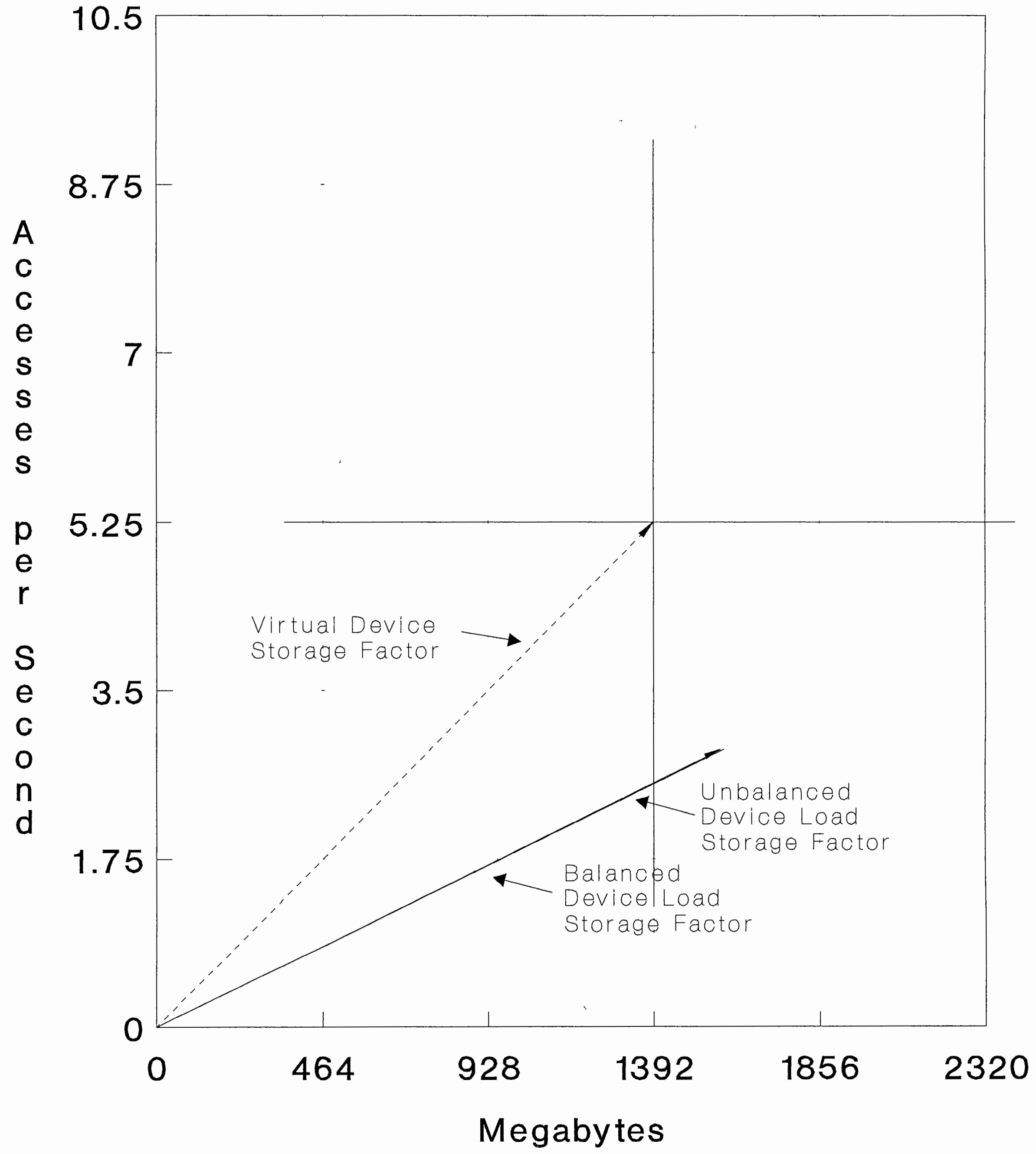
DEVICE LOAD STOR. FACT.--TSH010

15:00-16:00 p.m.



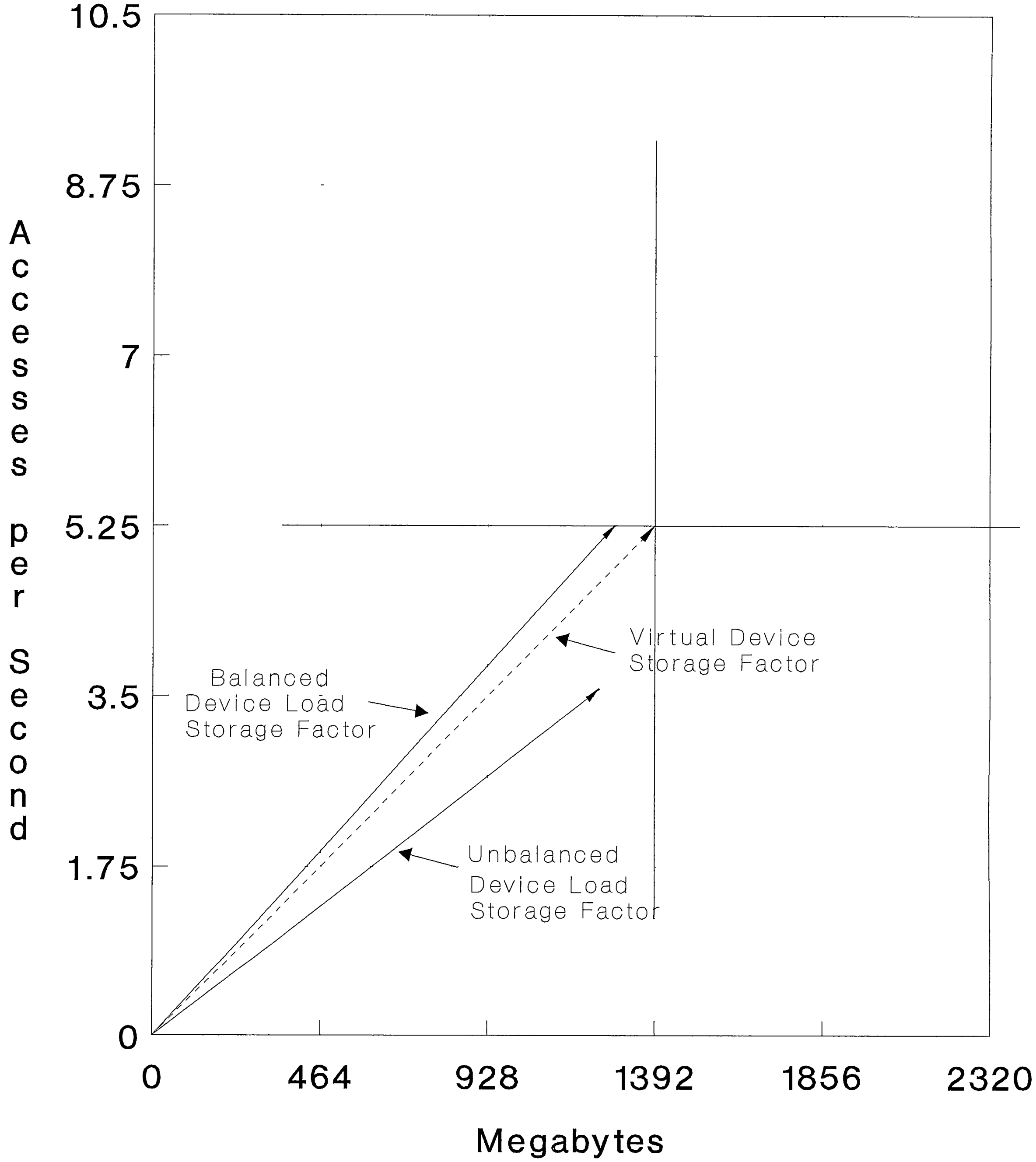
DEVICE LOAD STOR. FACT.--TSH011

15:00-16:00 p.m.



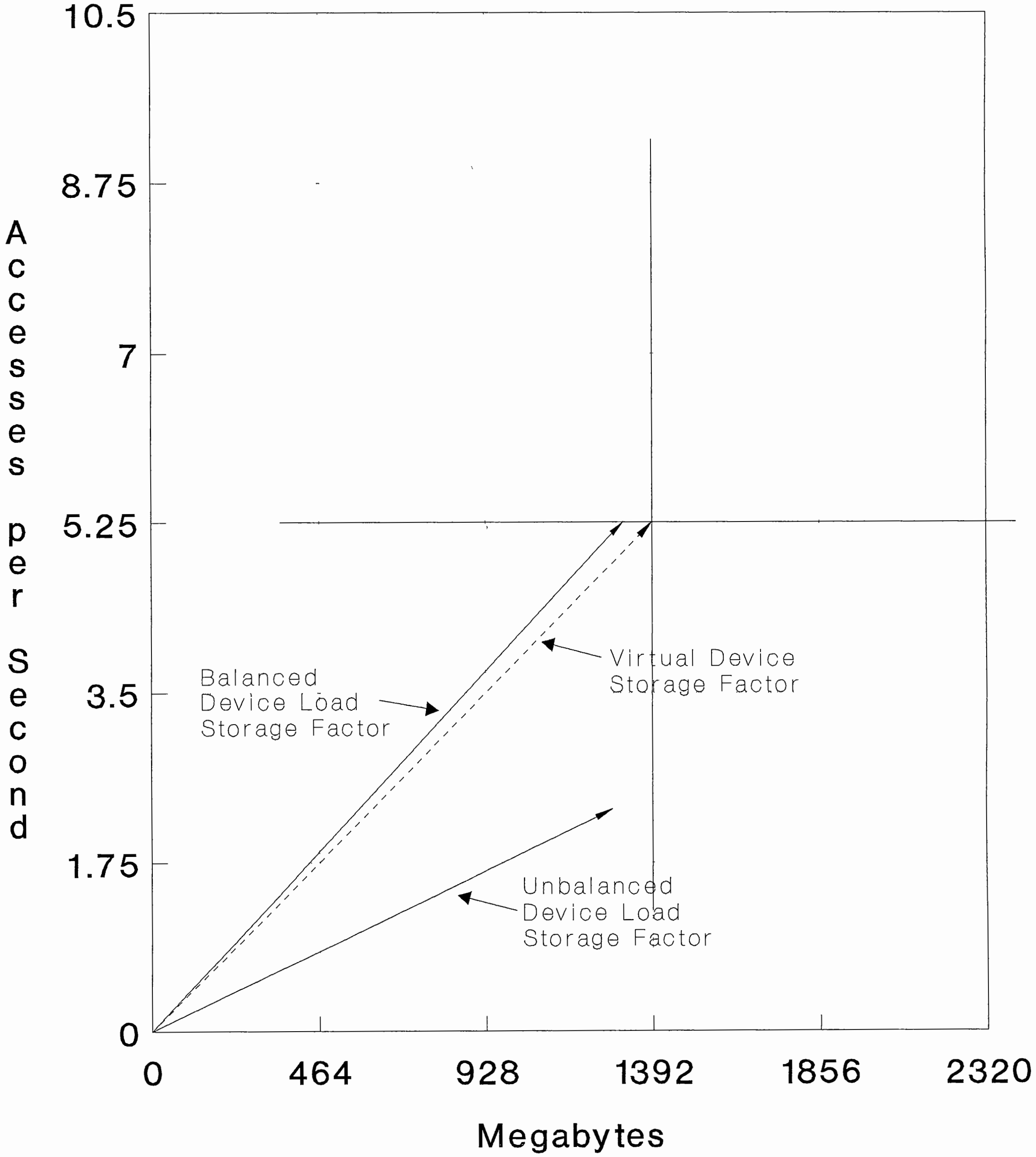
DEVICE LOAD STOR. FACT.--TSH013

15:00-16:00 p.m.



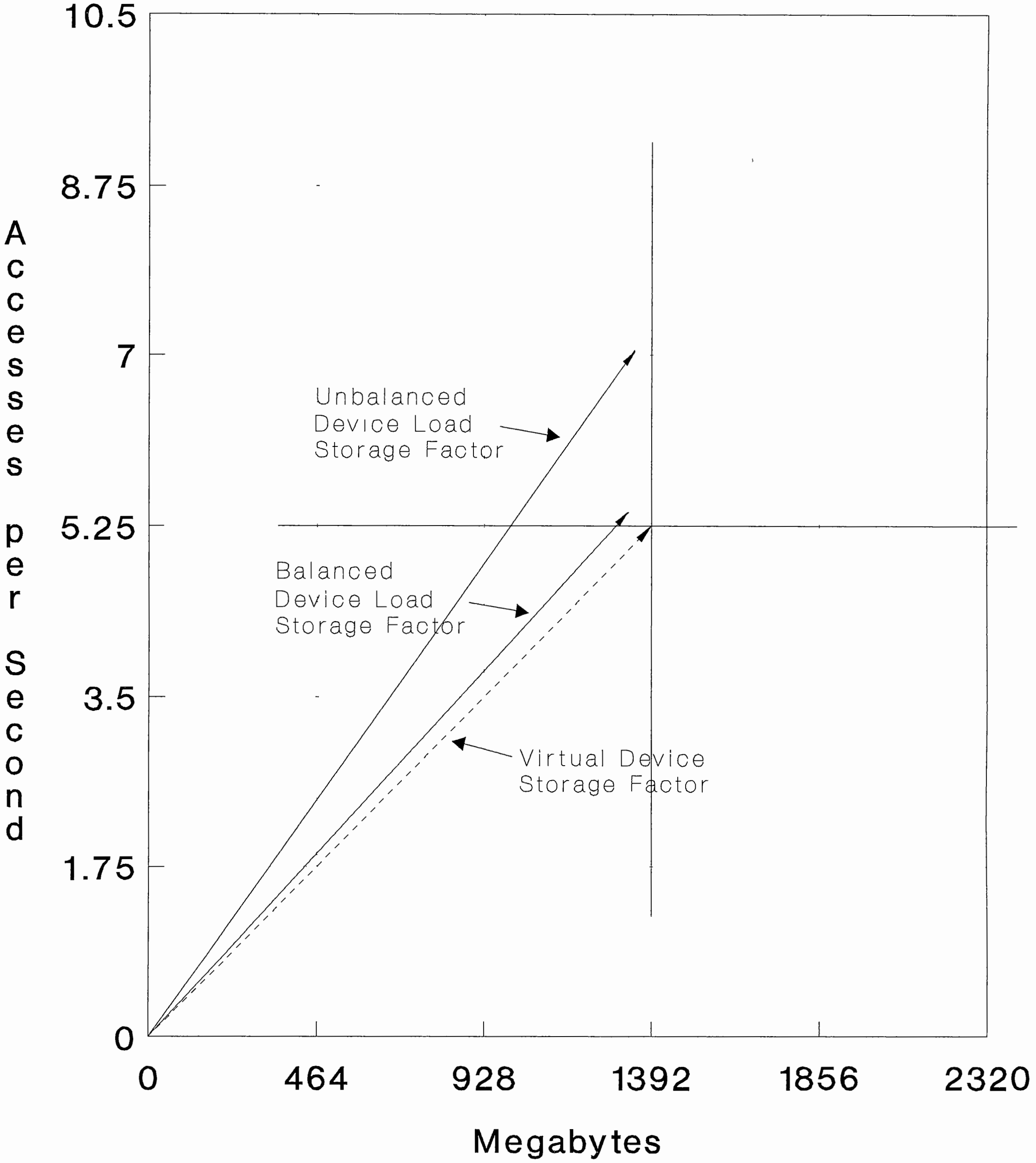
DEVICE LOAD STOR. FACT.--TSH014

15:00-16:00 p.m.



DEVICE LOAD STOR. FACT.--USDD01

15:00-16:00 p.m.



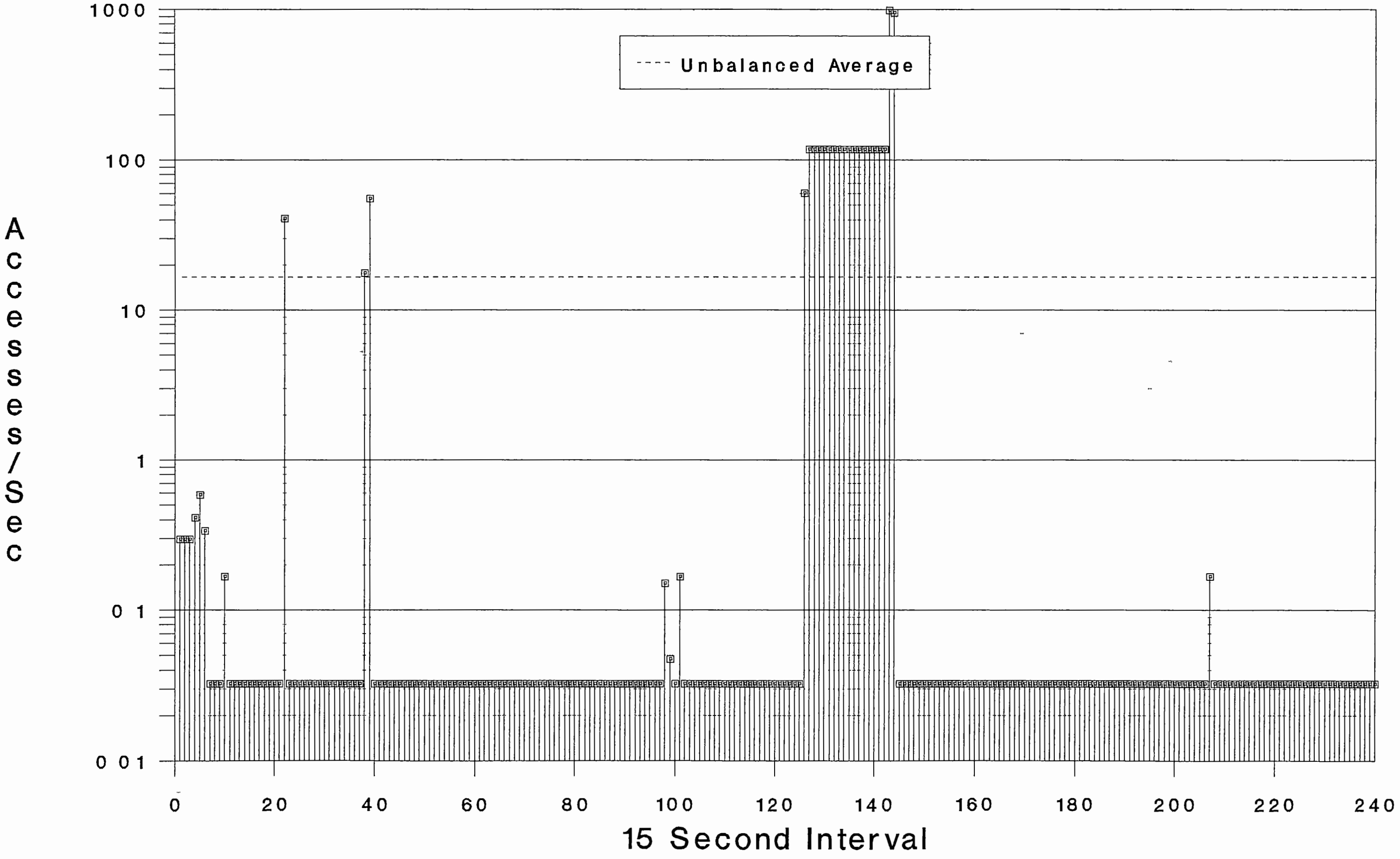
APPENDIX C

HISTOGRAMS OF ACCESS RATES BY DEVICE

15:00-16:00 P.M.

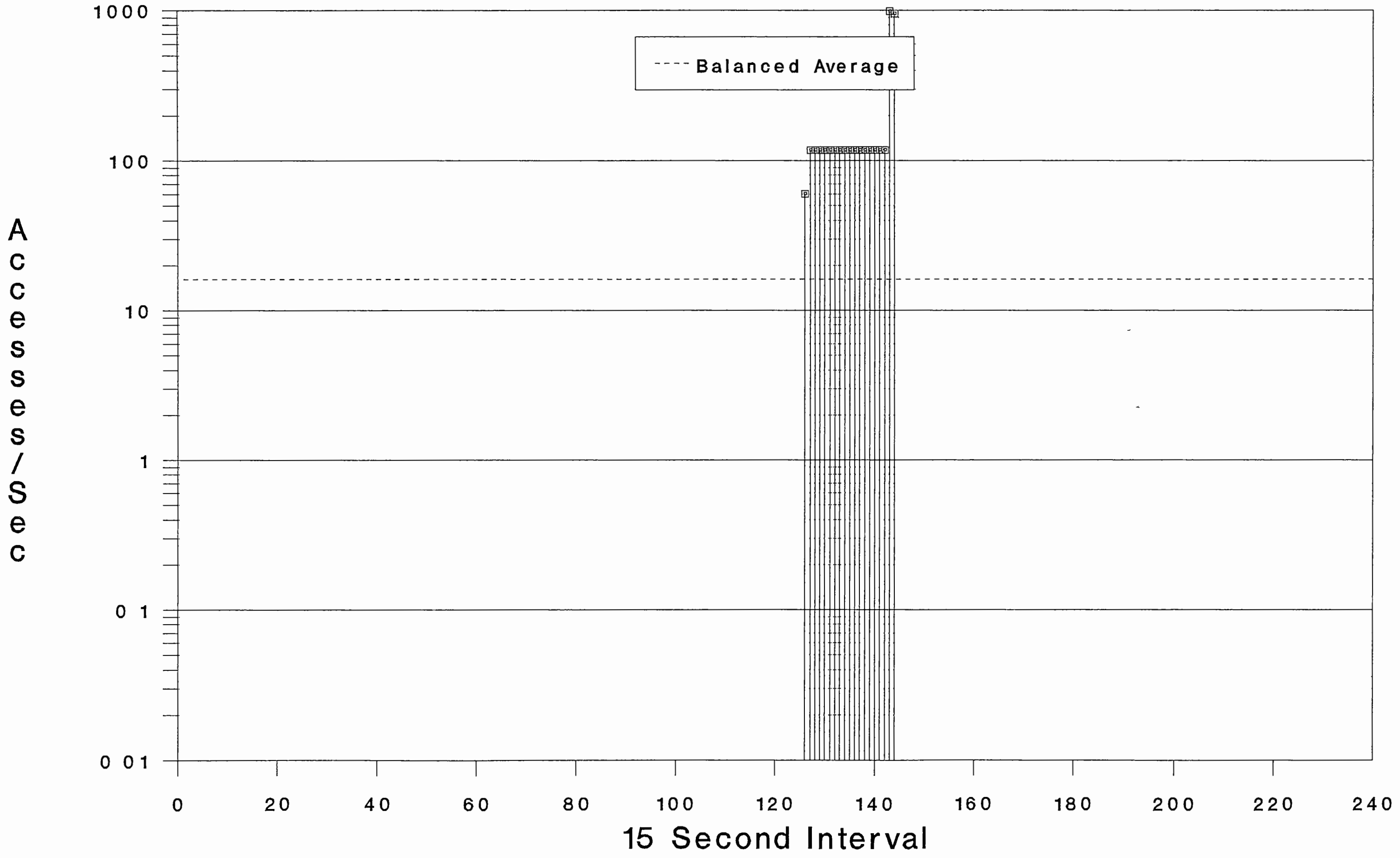
ACCESSES/SECOND - 15:00 TO 16:00 PM

RES018 -- UNBALANCED



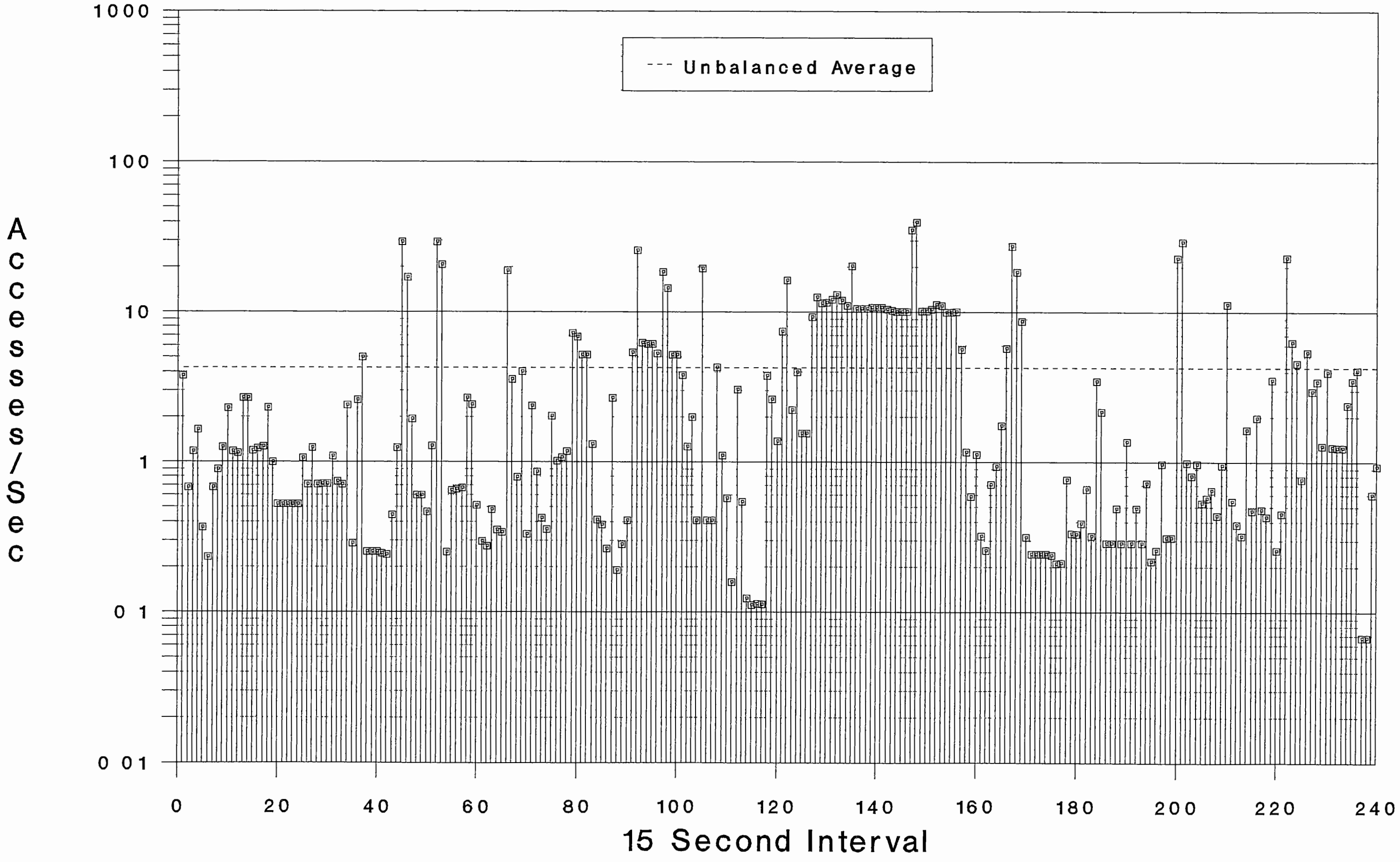
ACCESSES/SECOND - 15:00 TO 16:00 PM

RES018 -- BALANCED



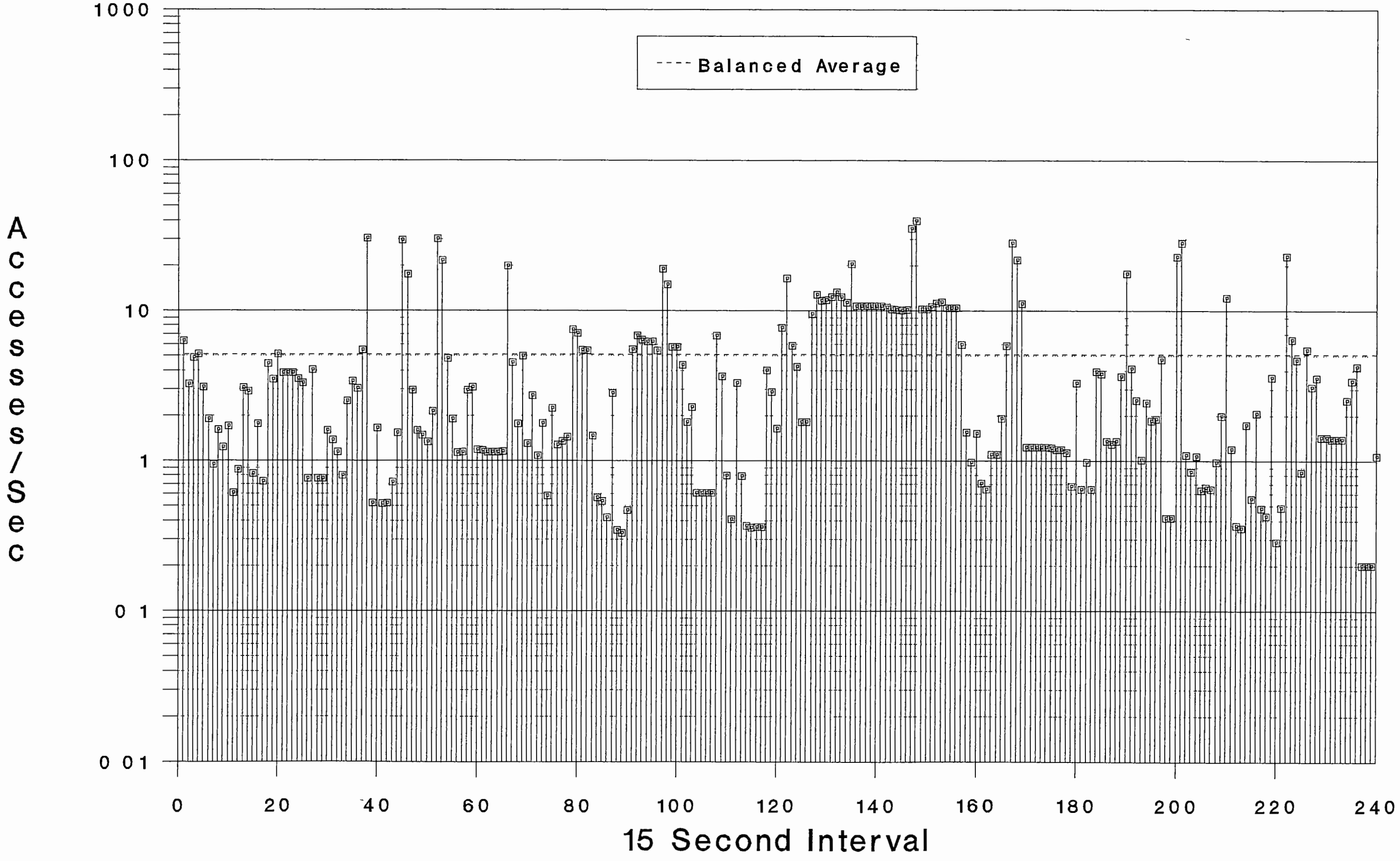
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH000 -- UNBALANCED



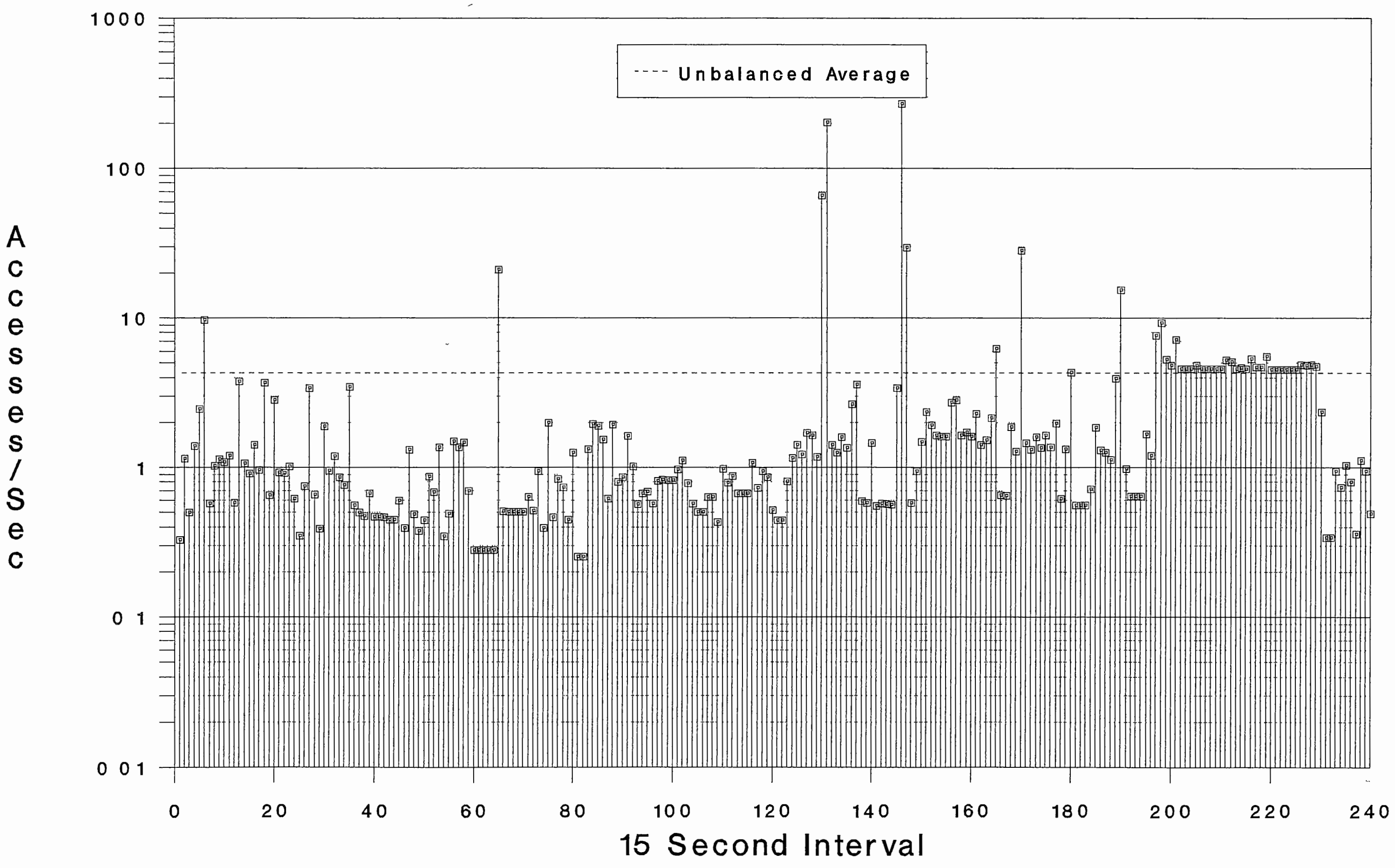
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH000 -- BALANCED



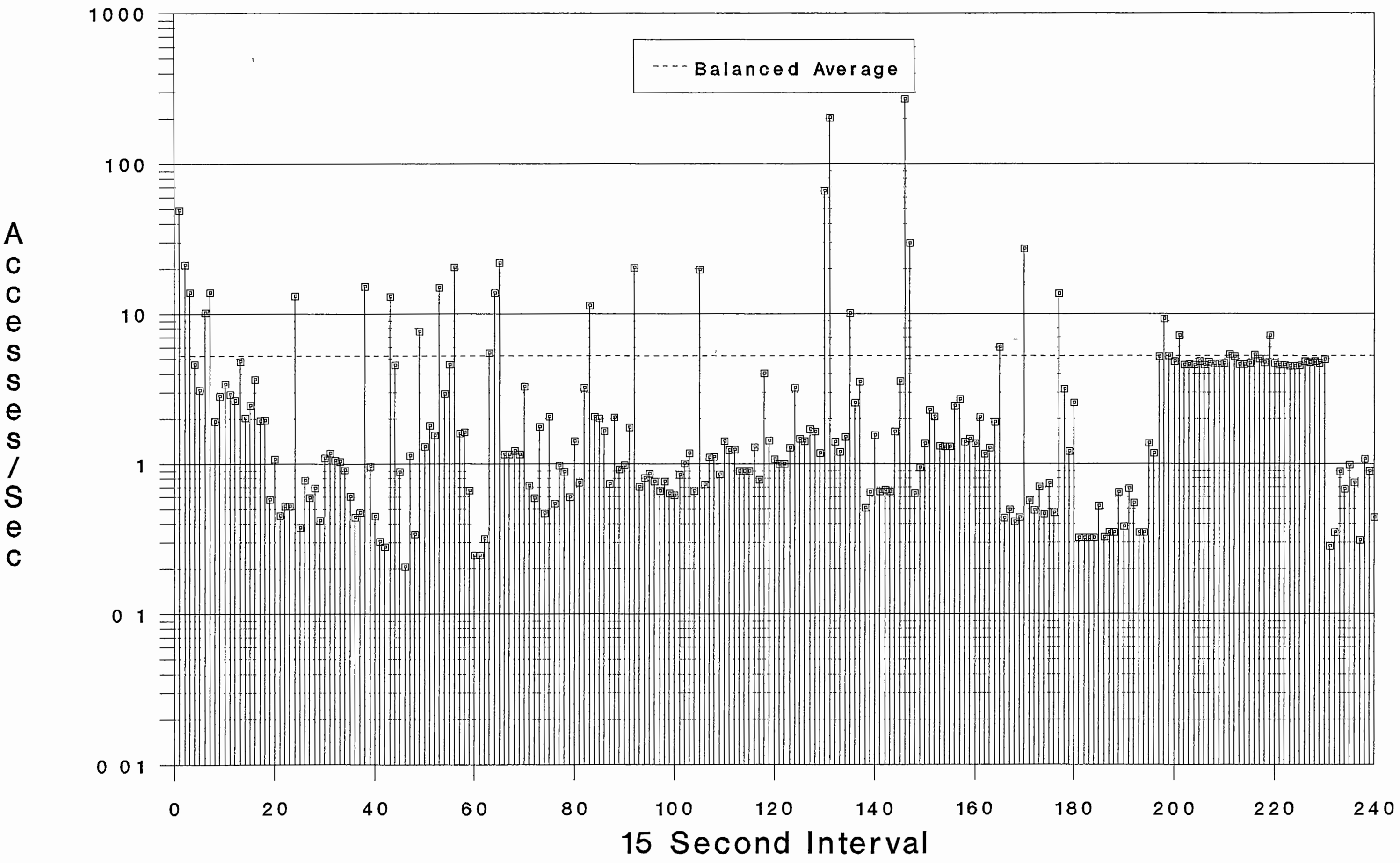
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH001 -- UNBALANCED



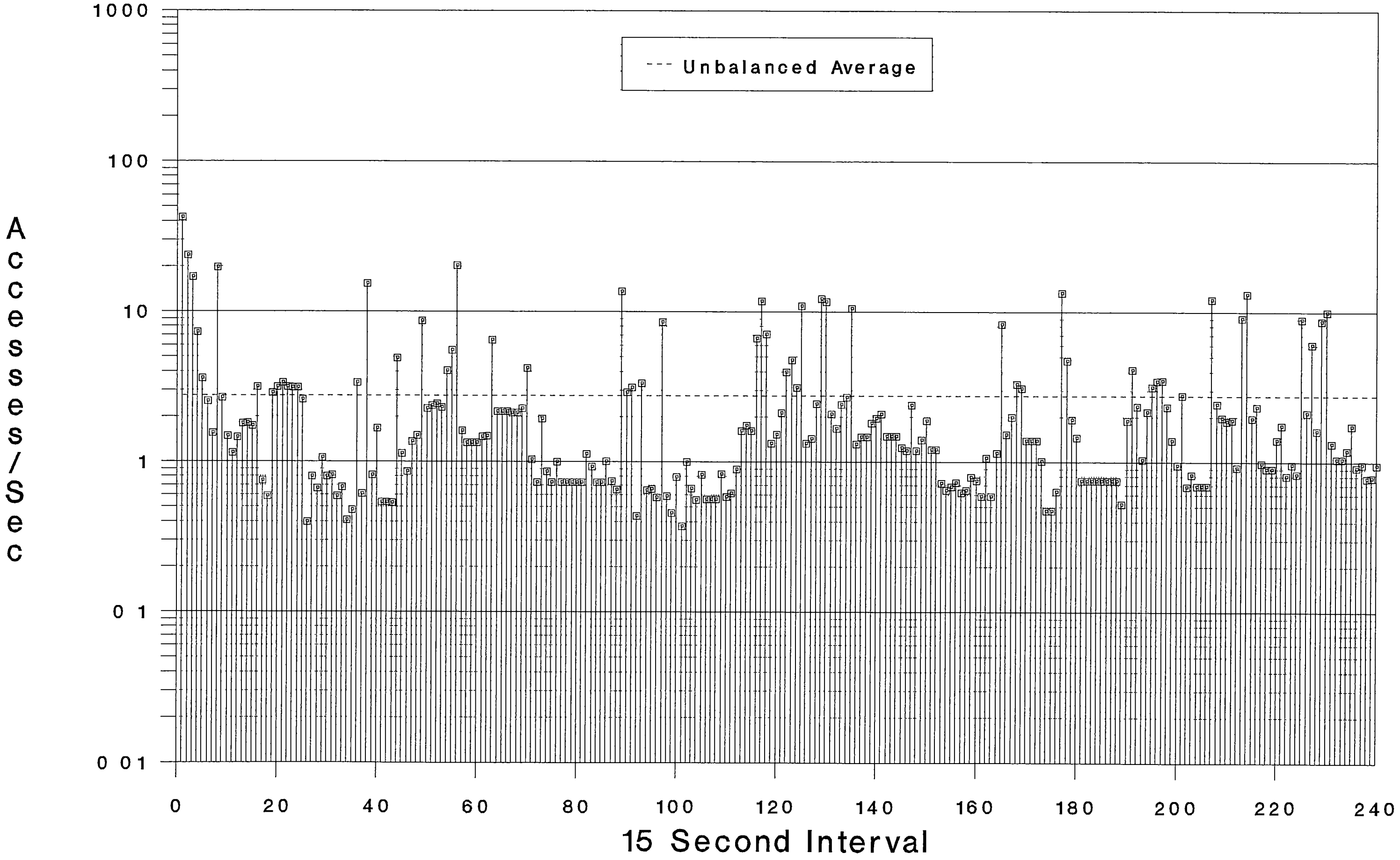
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH001 -- BALANCED



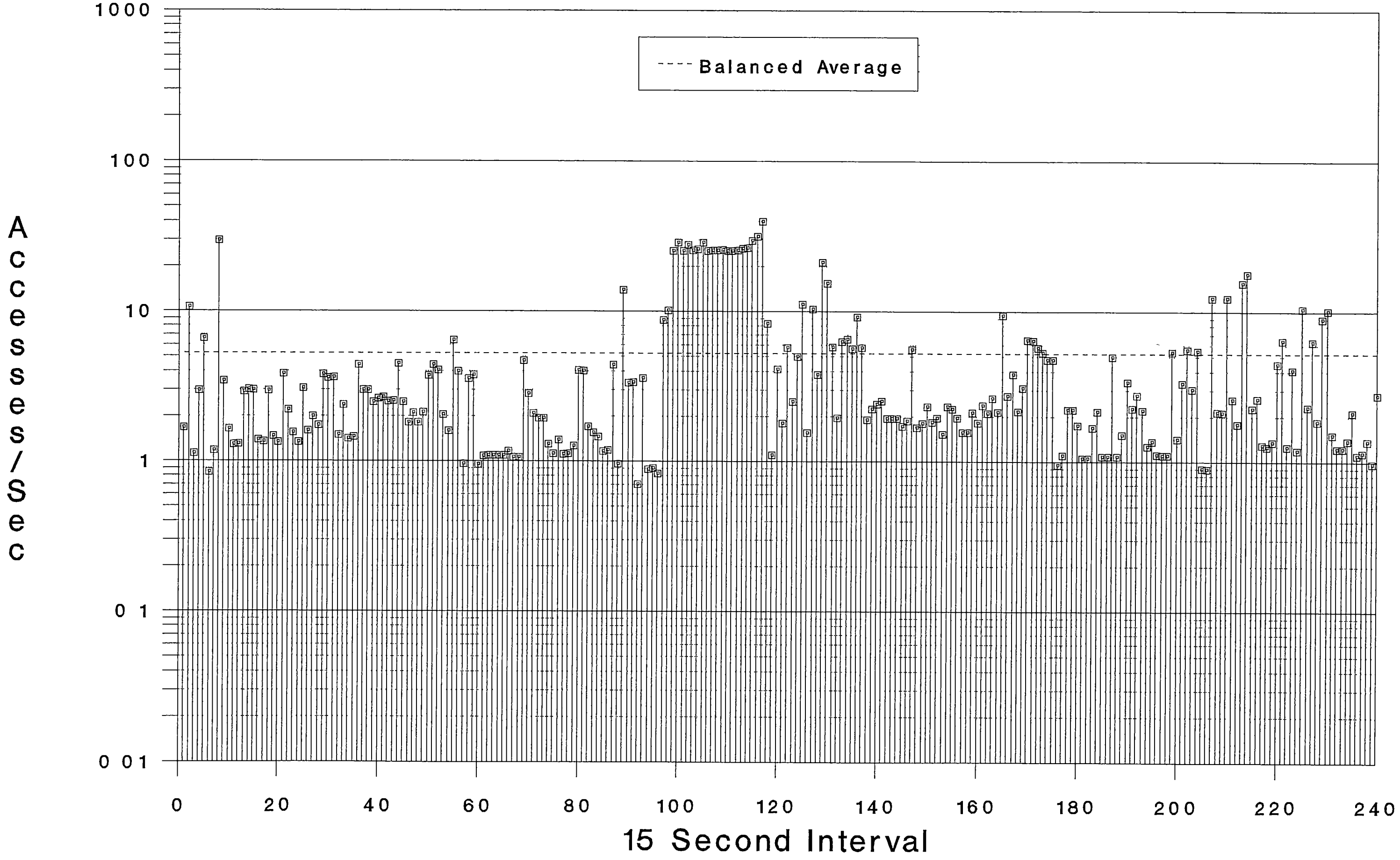
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH002 -- UNBALANCED

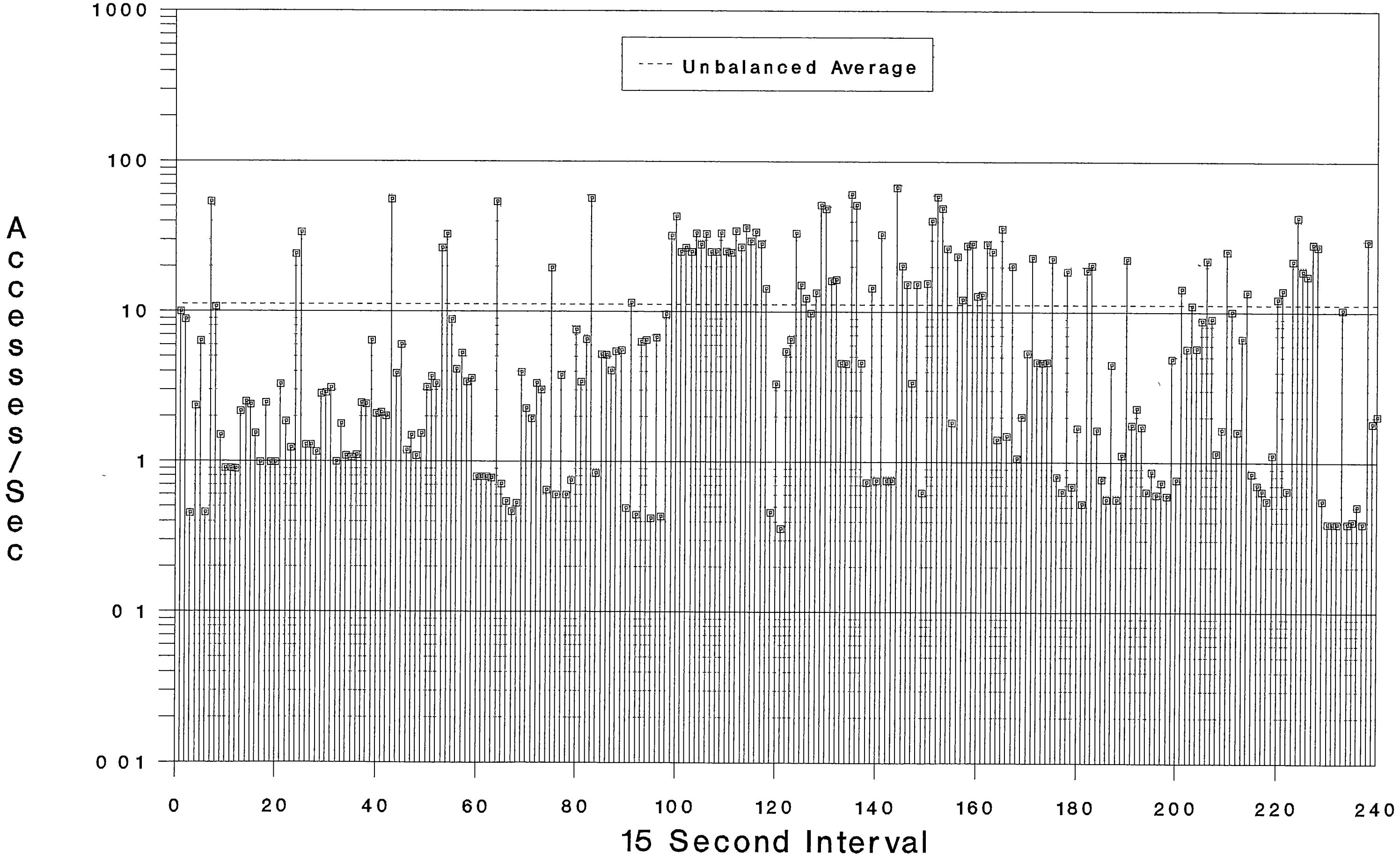


ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH002 -- BALANCED

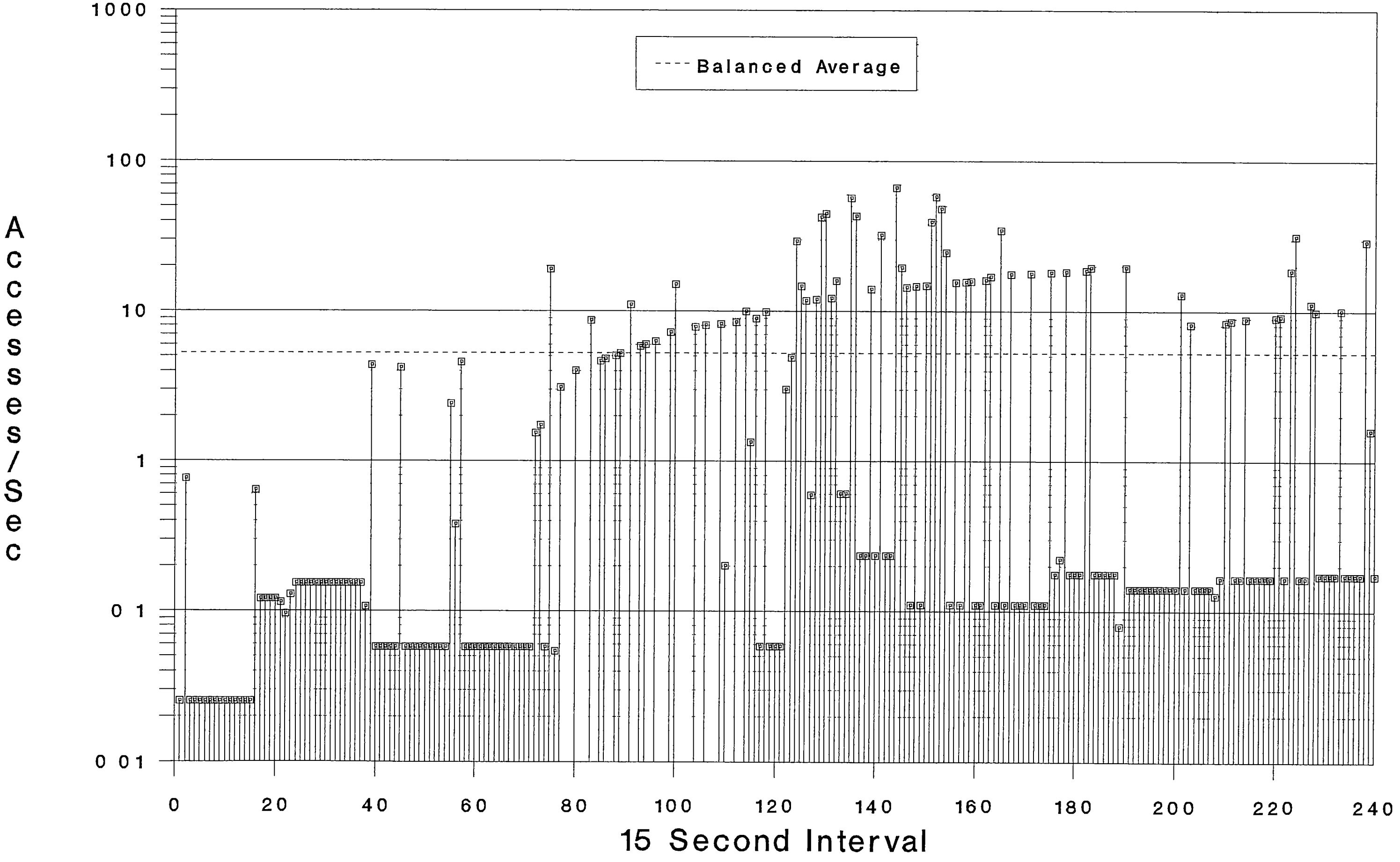


ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH003 -- UNBALANCED

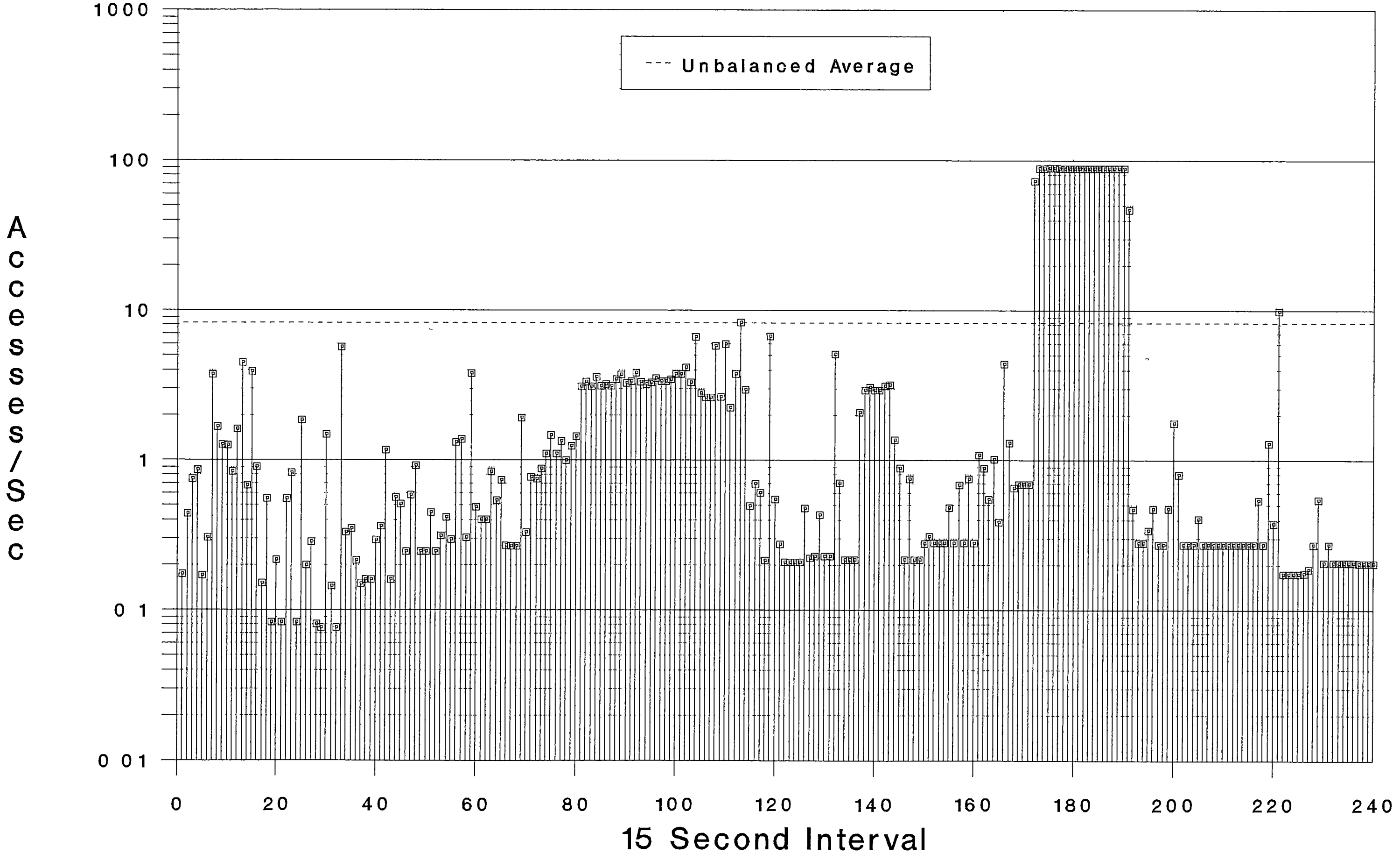


ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH003 -- BALANCED

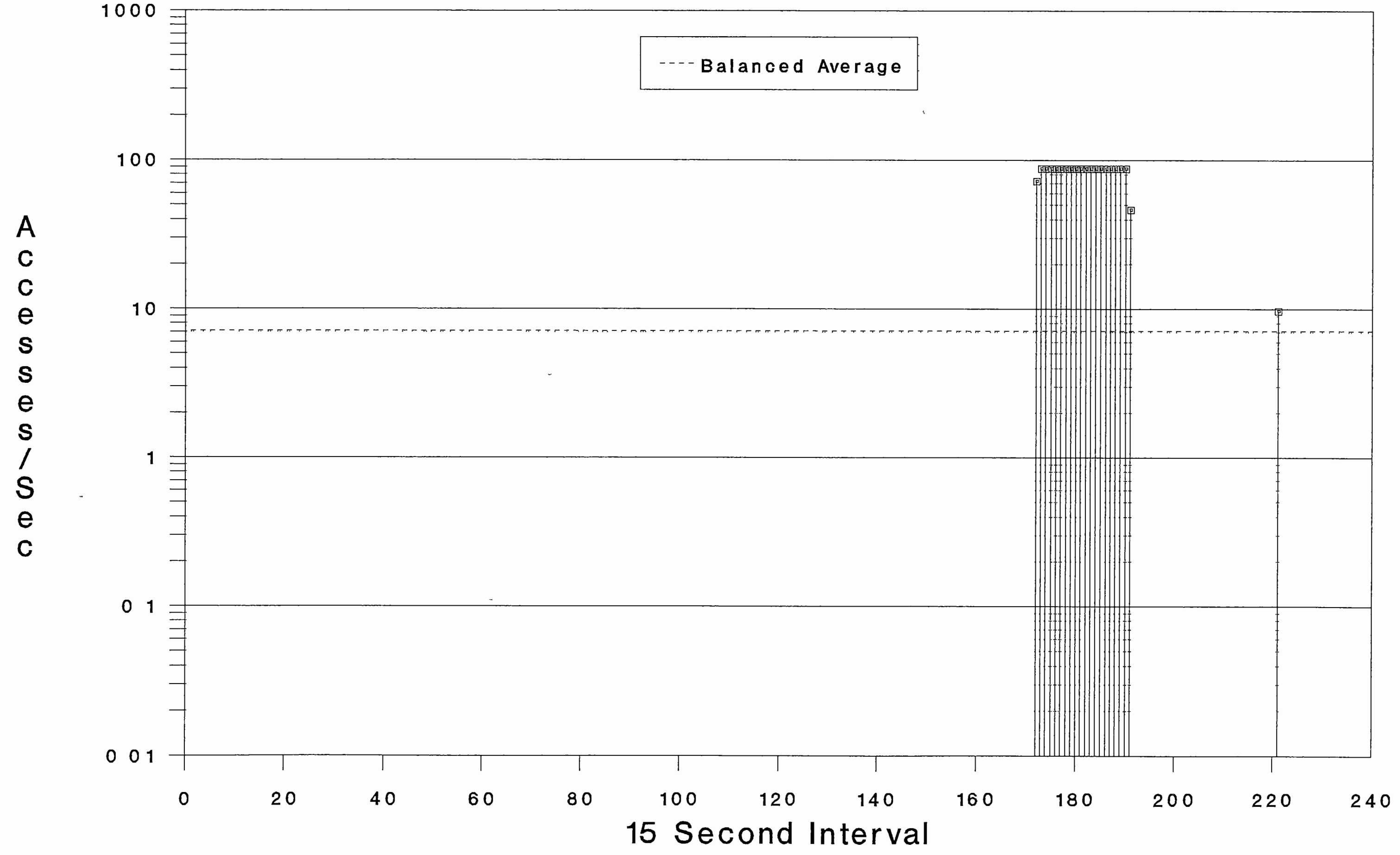


ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH004 -- UNBALANCED

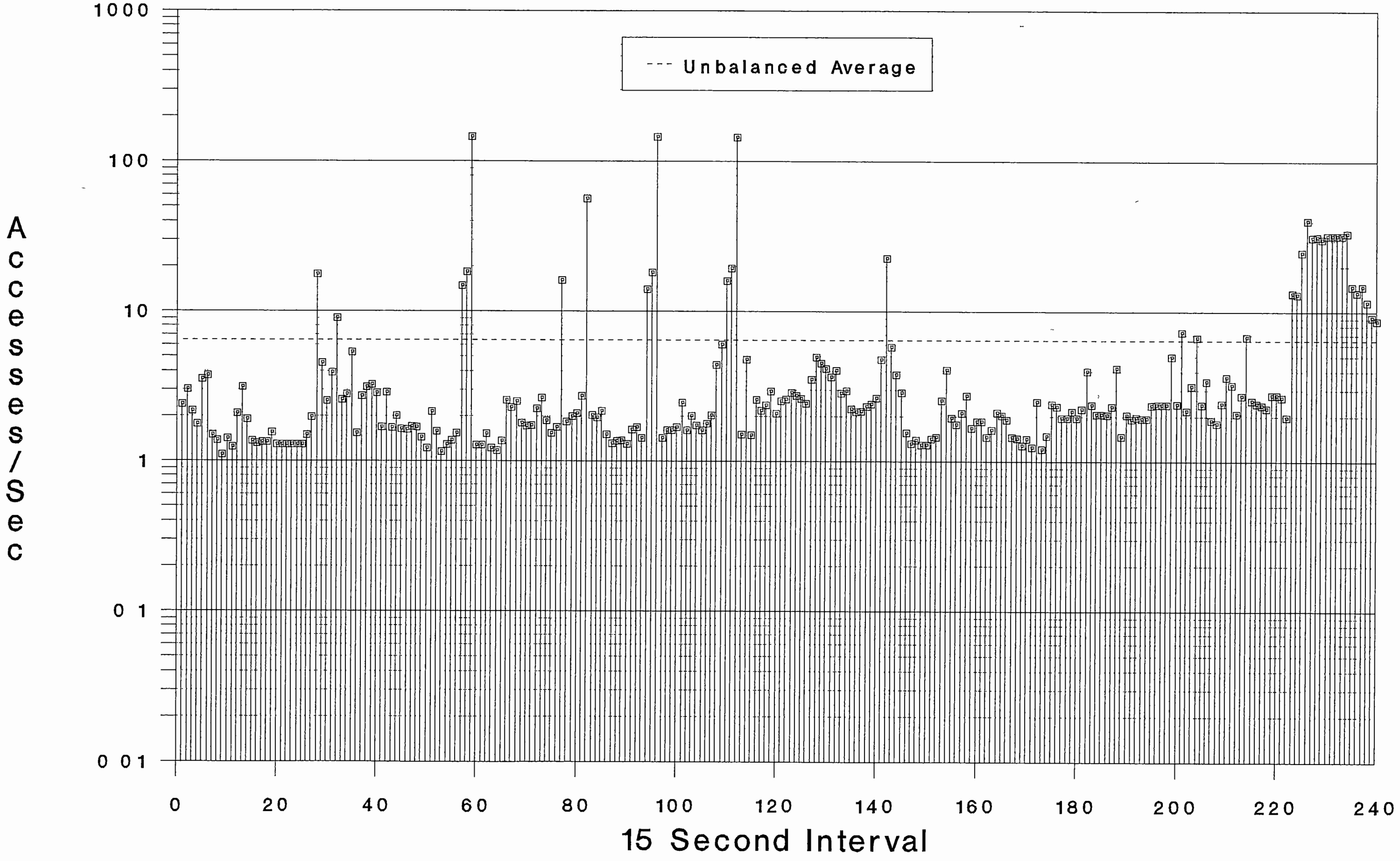


ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH004 -- BALANCED

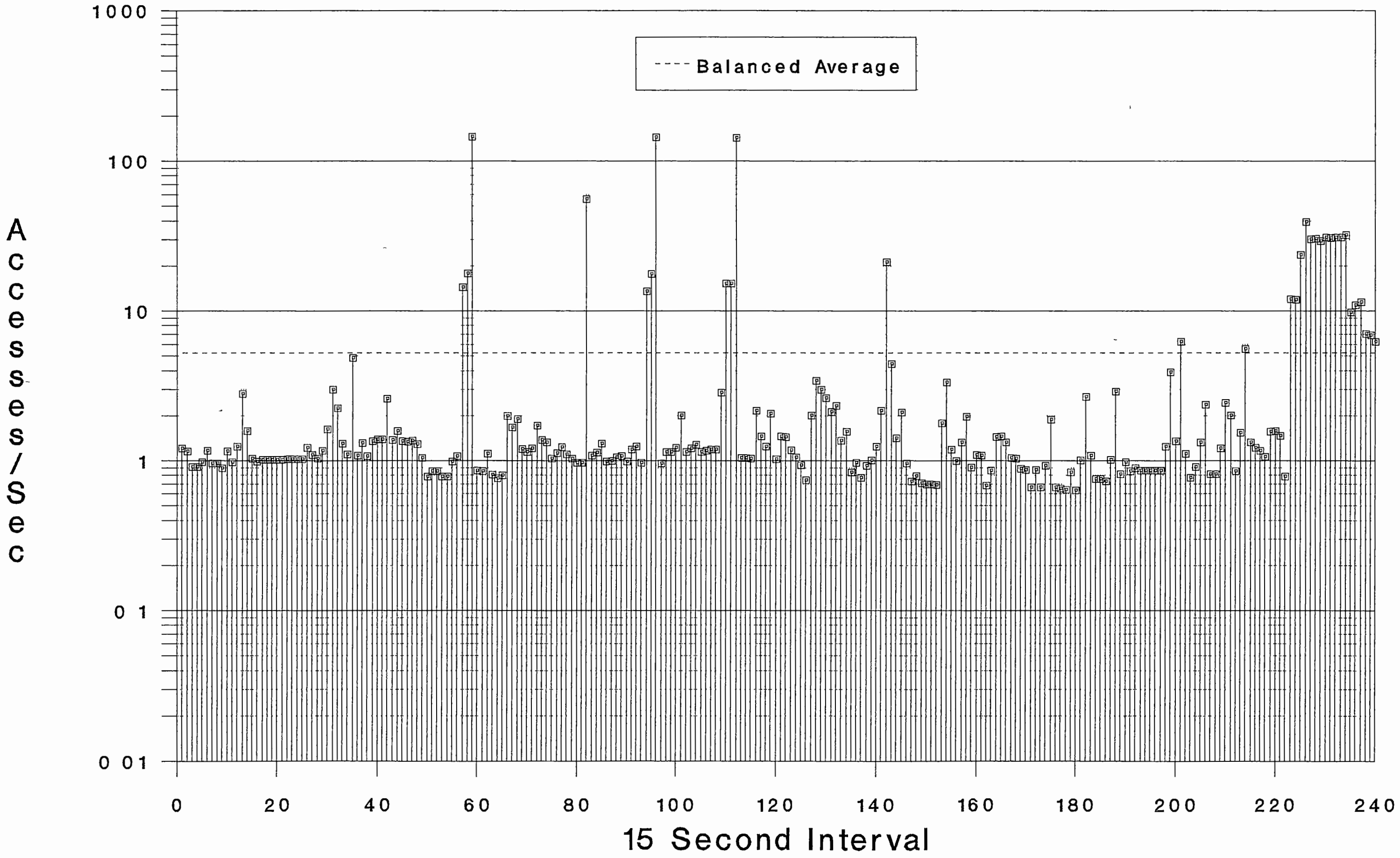


ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH005 -- UNBALANCED



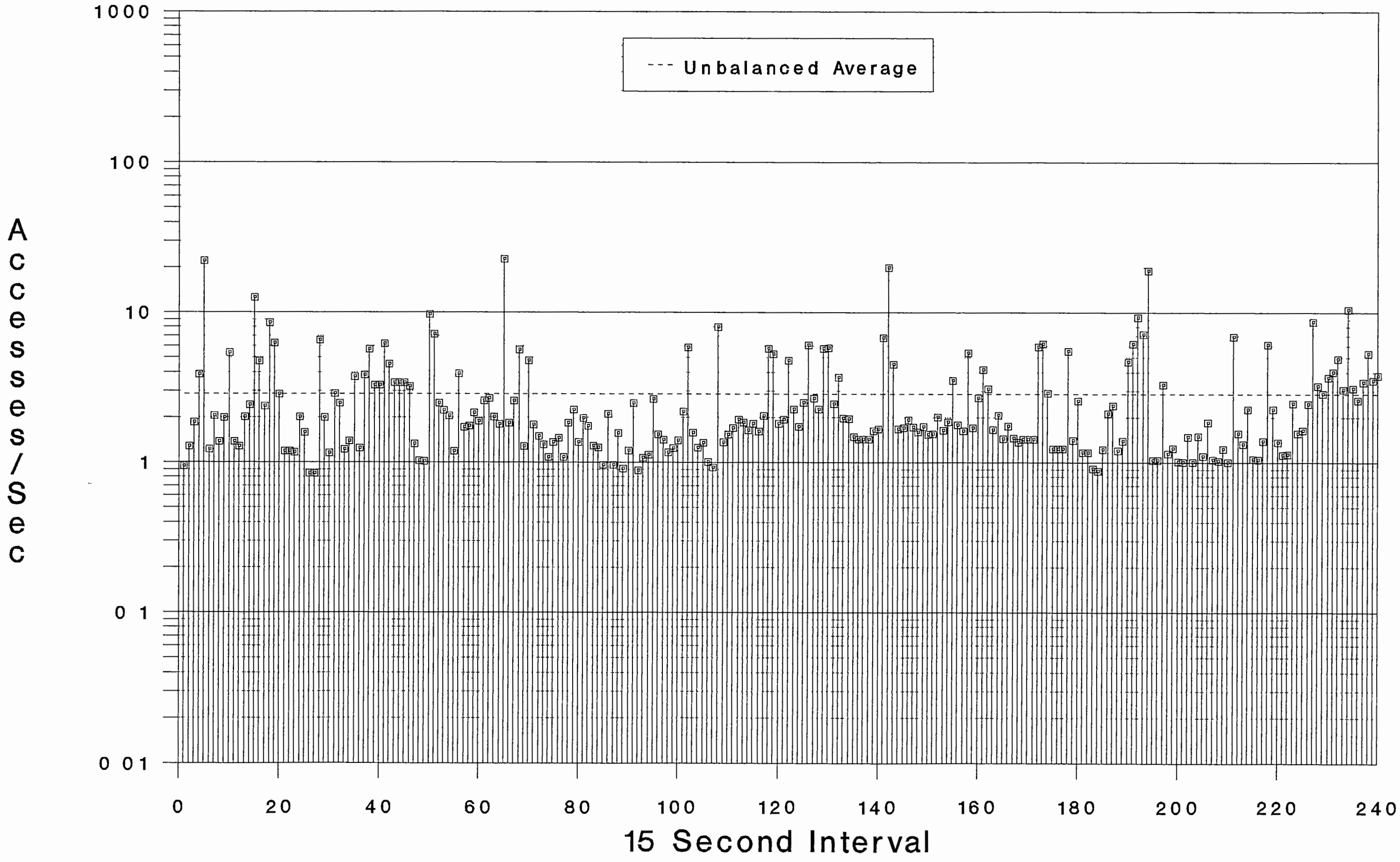
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH005 -- BALANCED



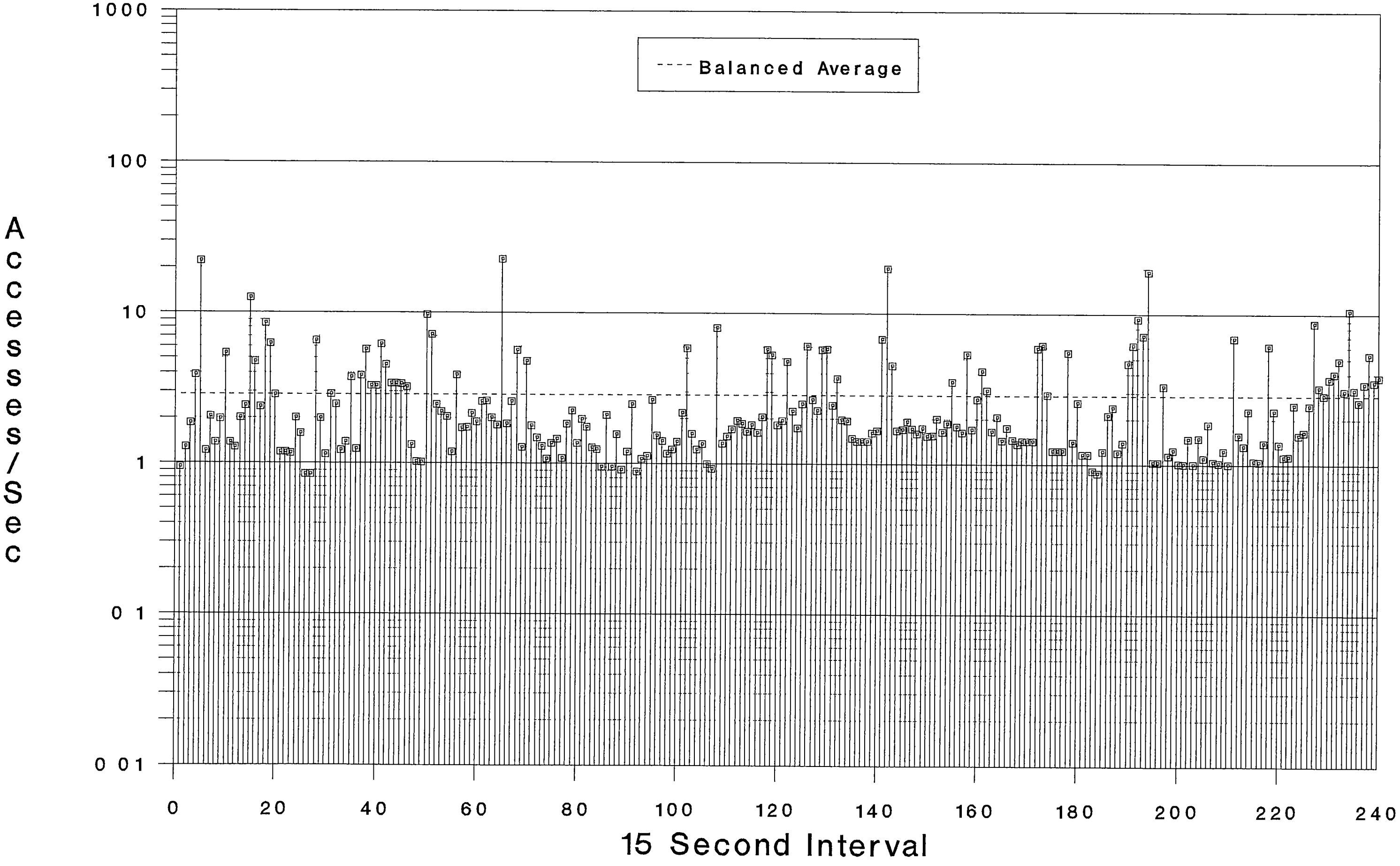
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH006 -- UNBALANCED



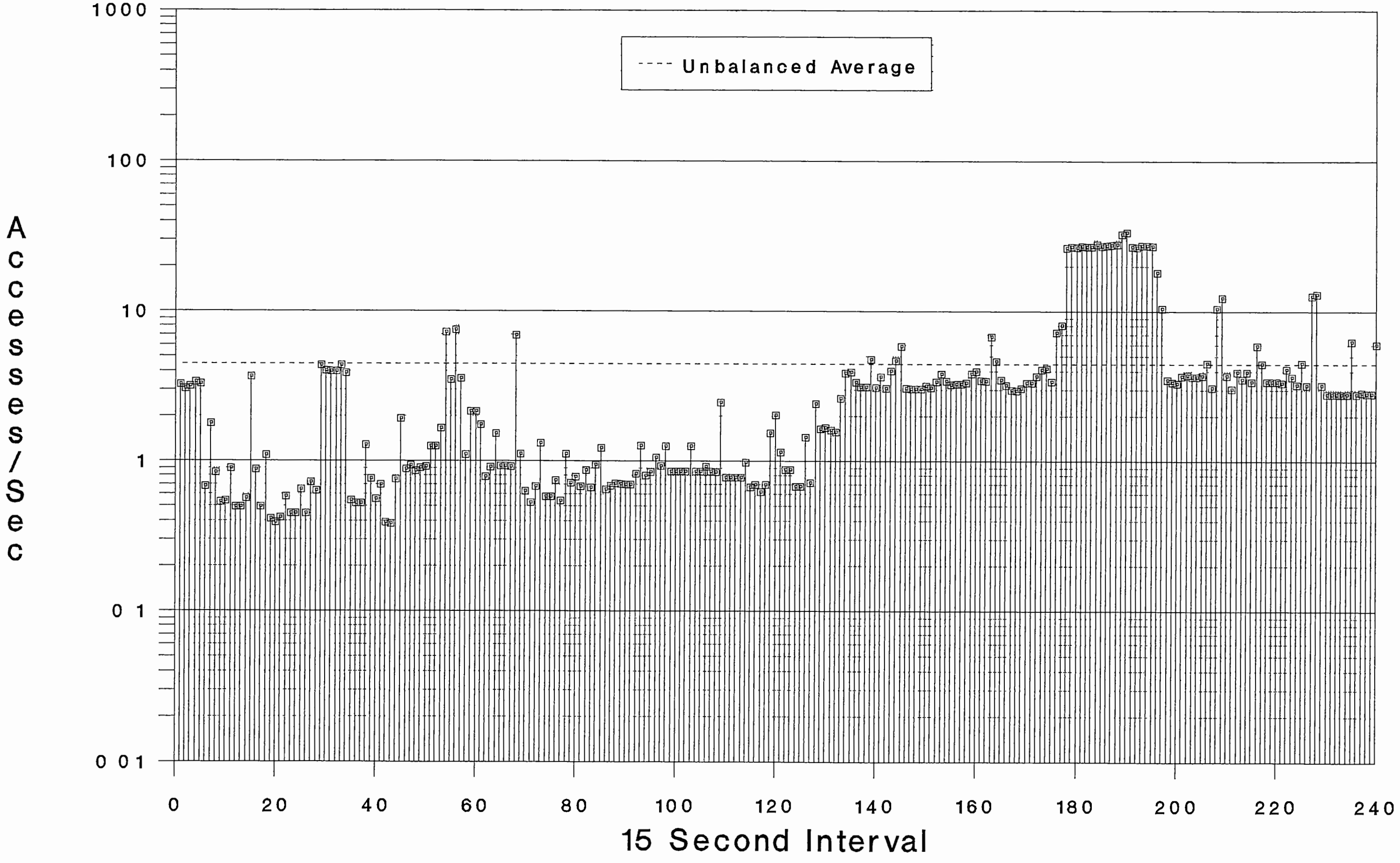
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH006 -- BALANCED



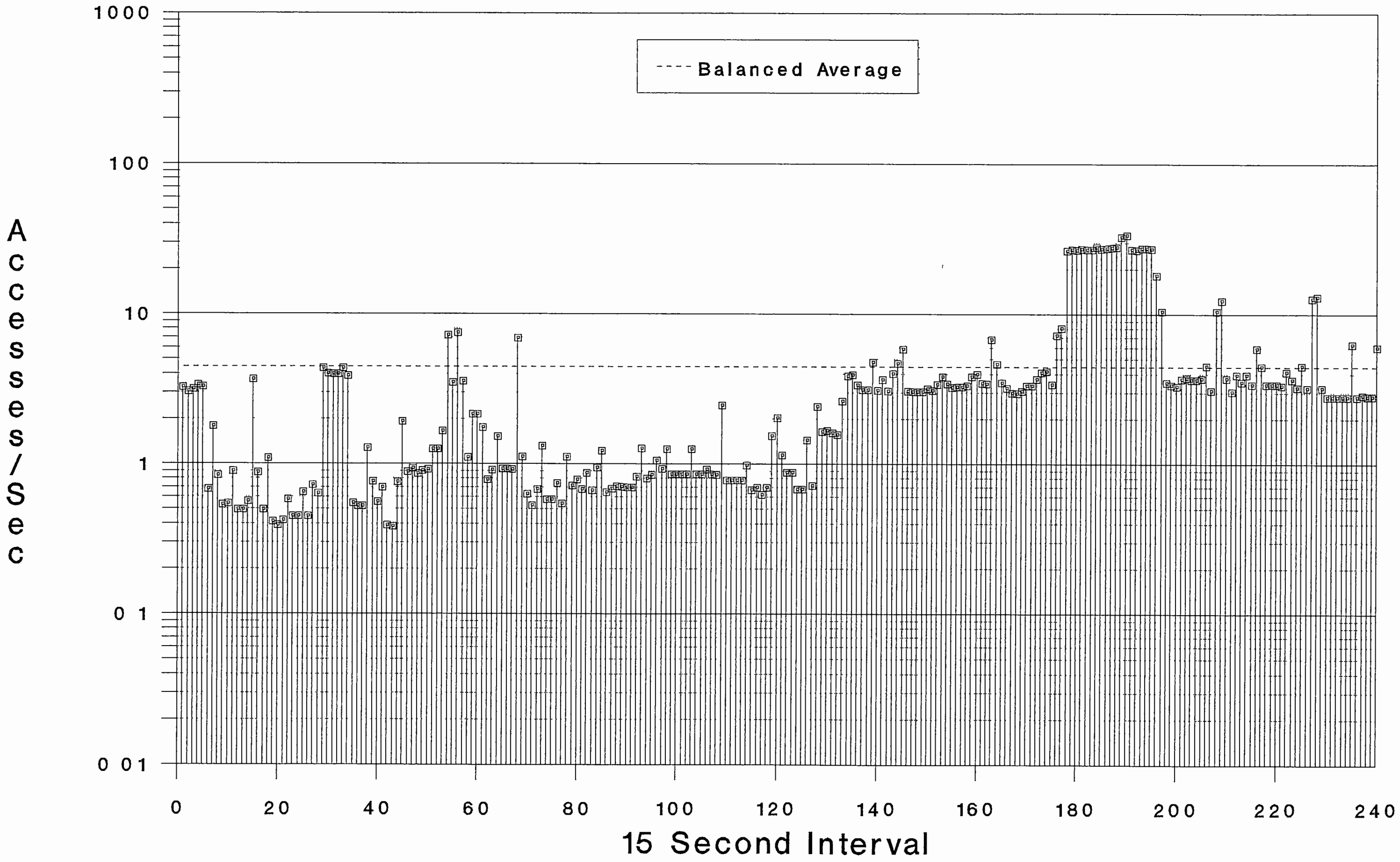
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH007 -- UNBALANCED



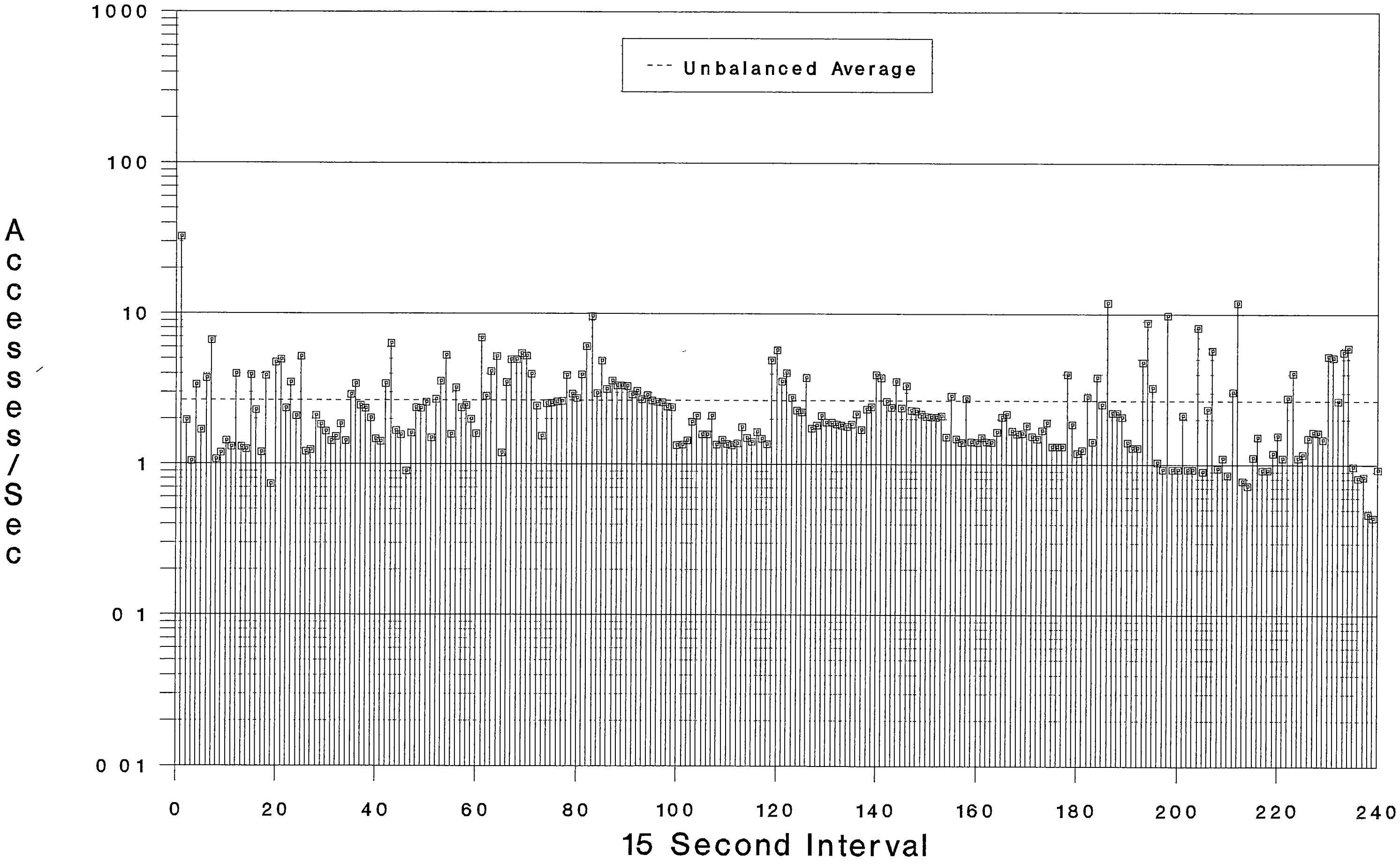
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH007 -- BALANCED



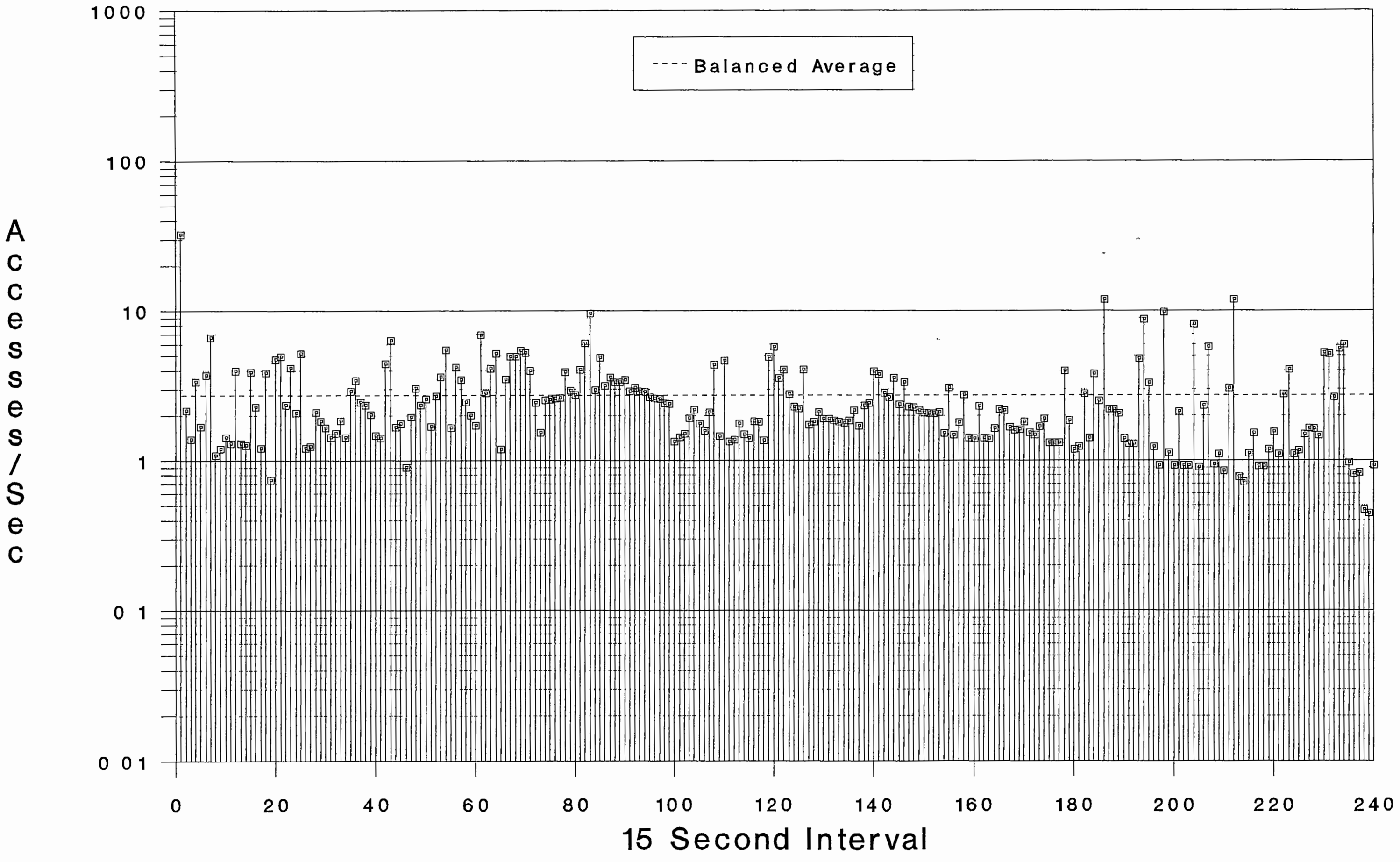
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH008 -- UNBALANCED



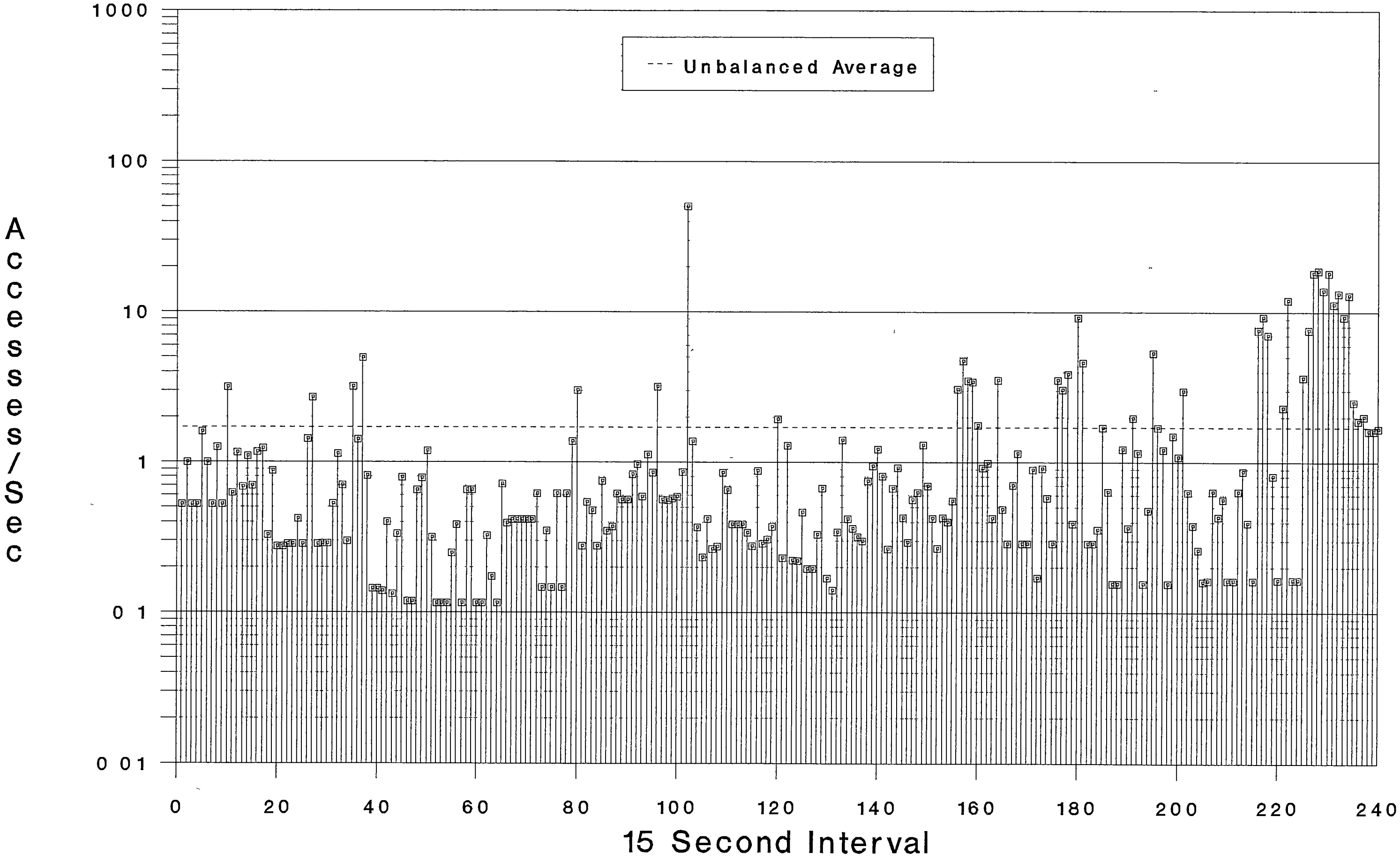
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH008 -- BALANCED



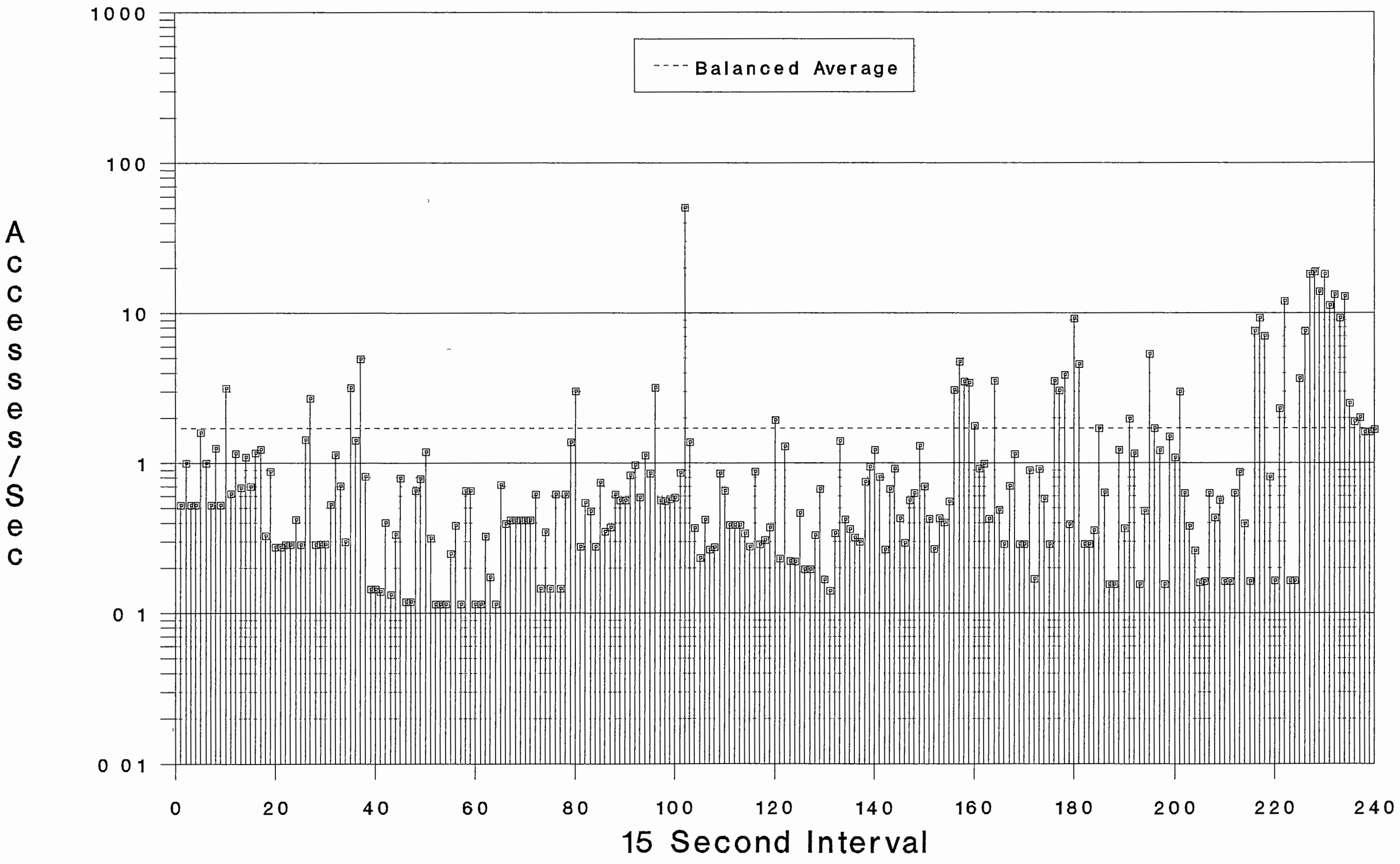
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH009 -- UNBALANCED



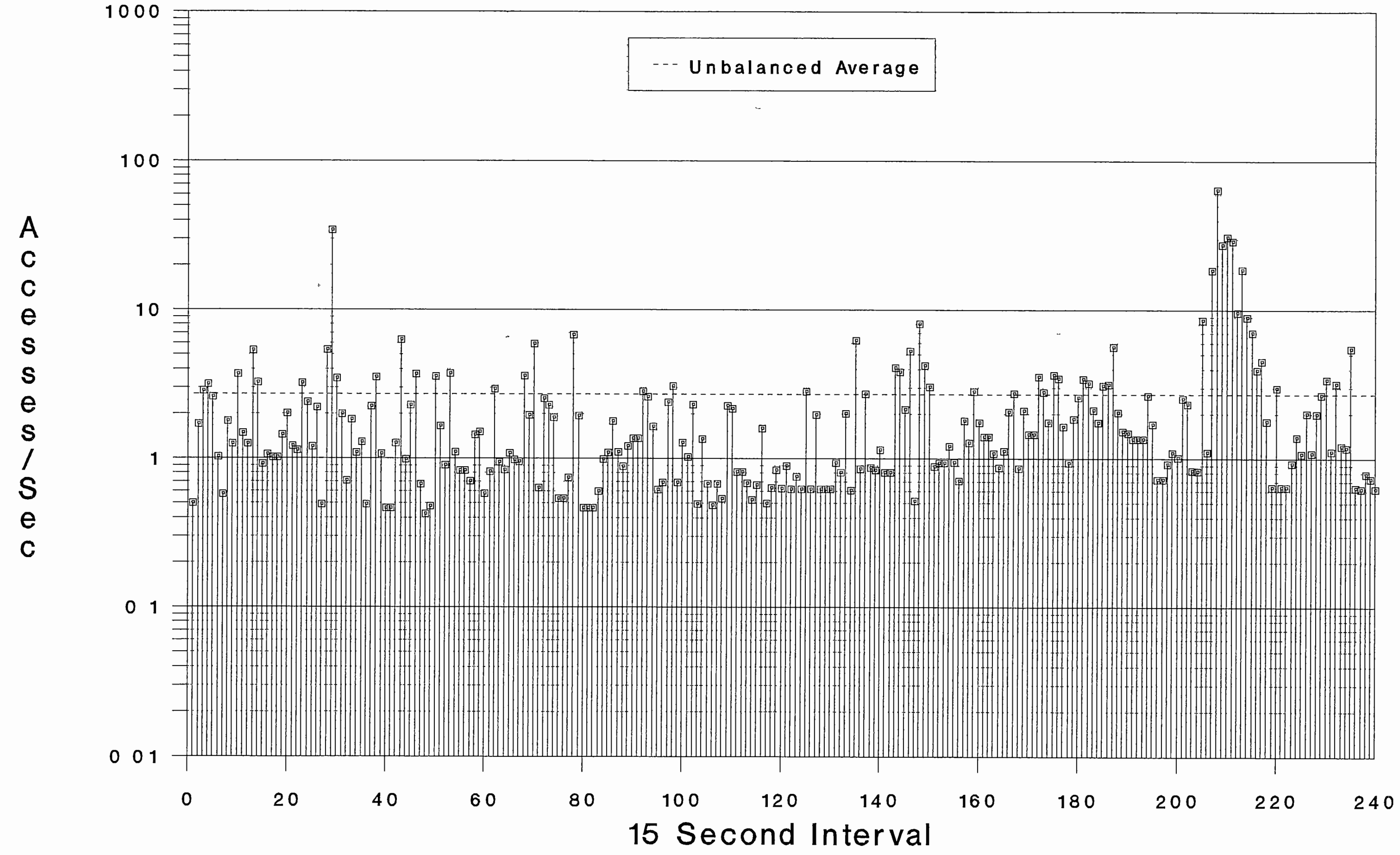
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH009 -- BALANCED

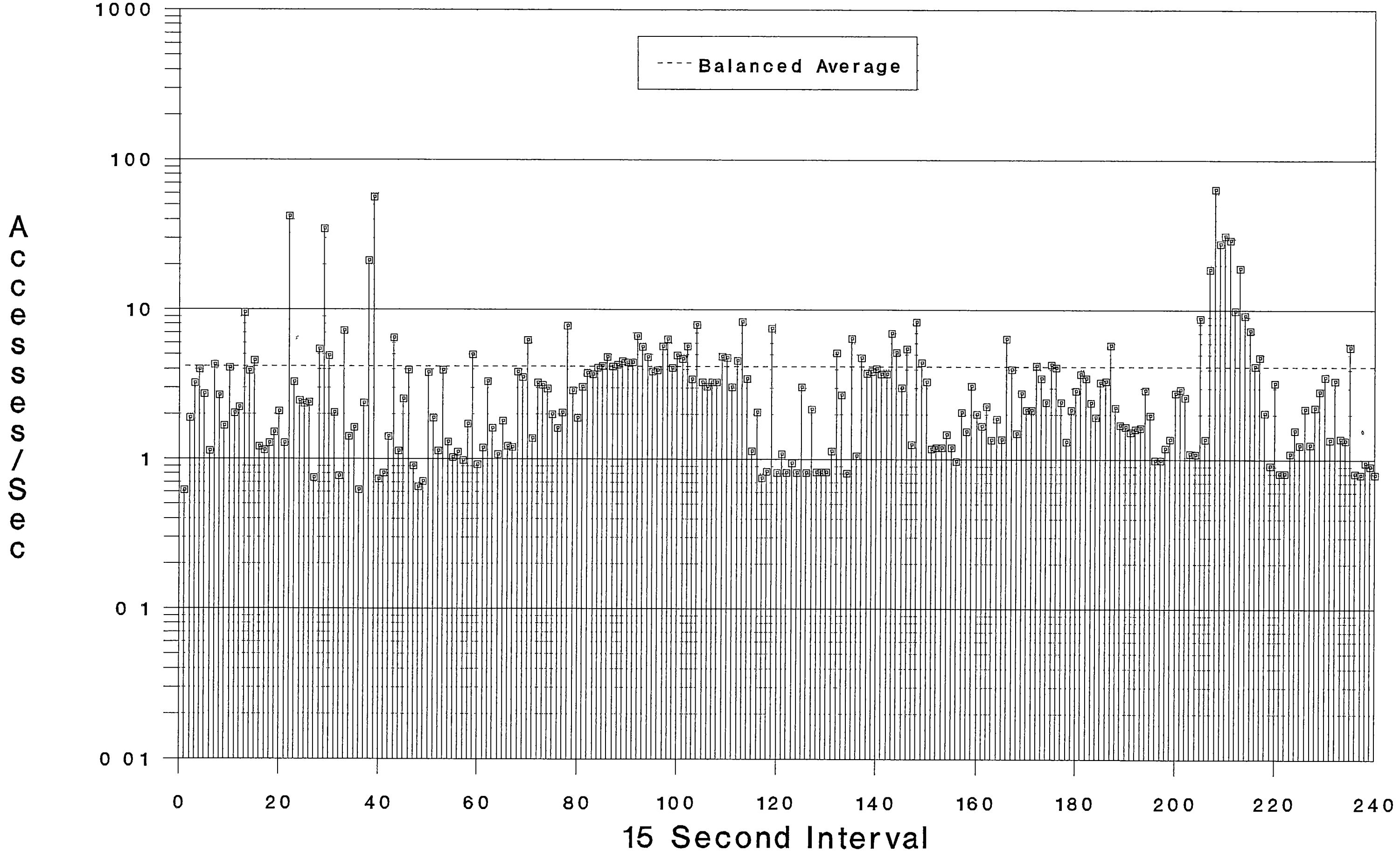


ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH010 -- UNBALANCED

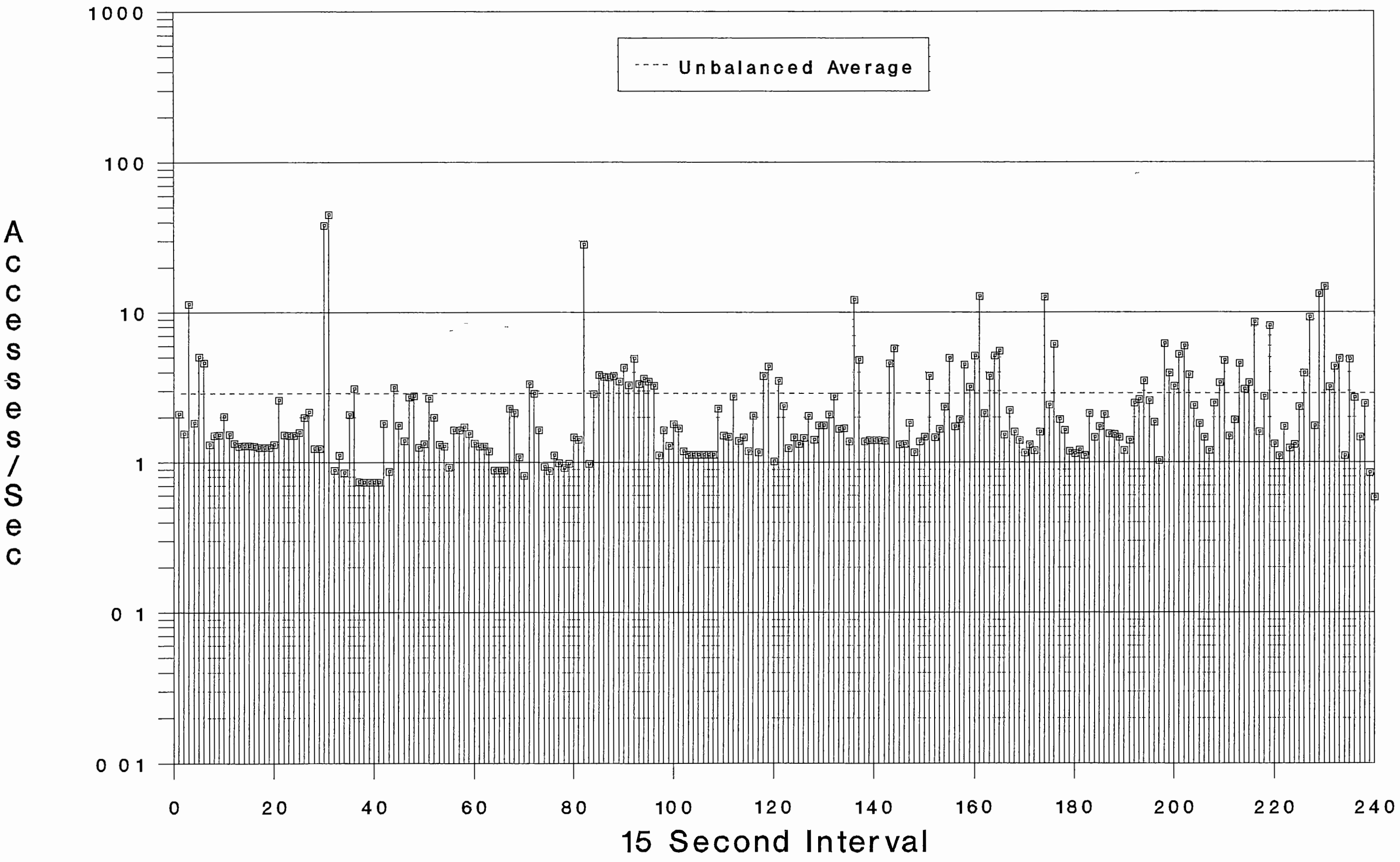


ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH010 -- BALANCED

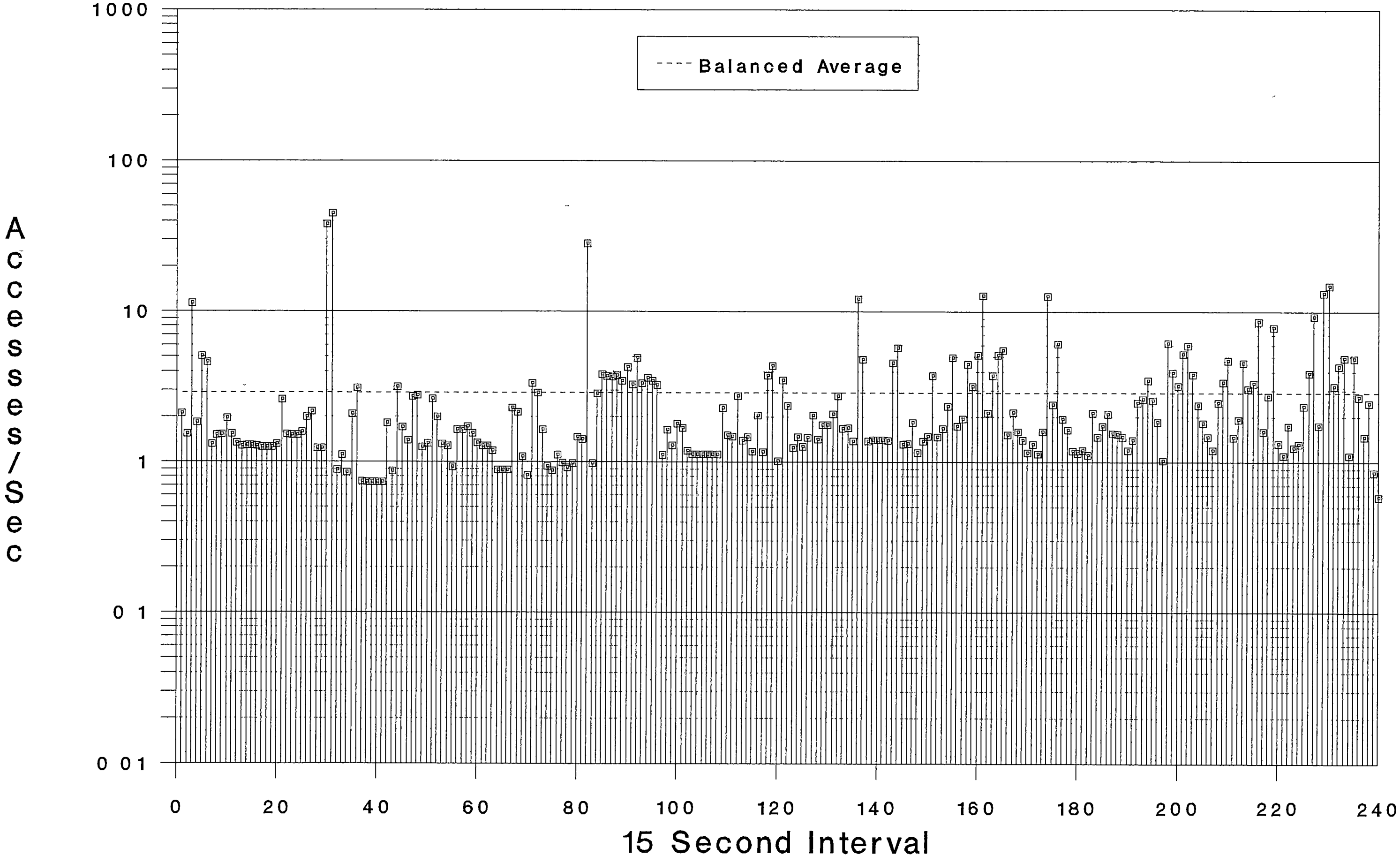


ACCESSES/SECOND - 15:00 TO 16:00 PM

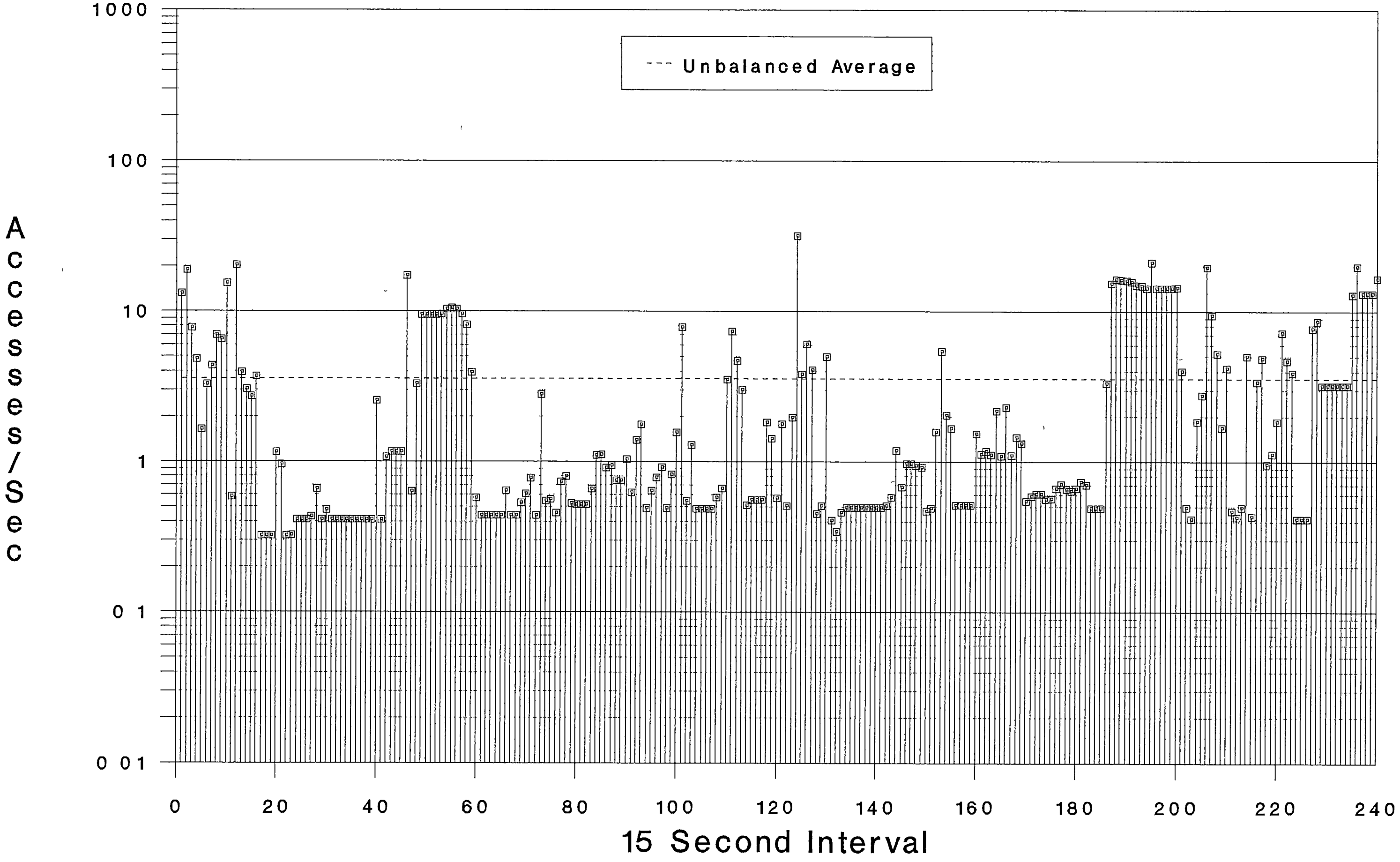
TSH011 -- UNBALANCED



ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH011 -- BALANCED

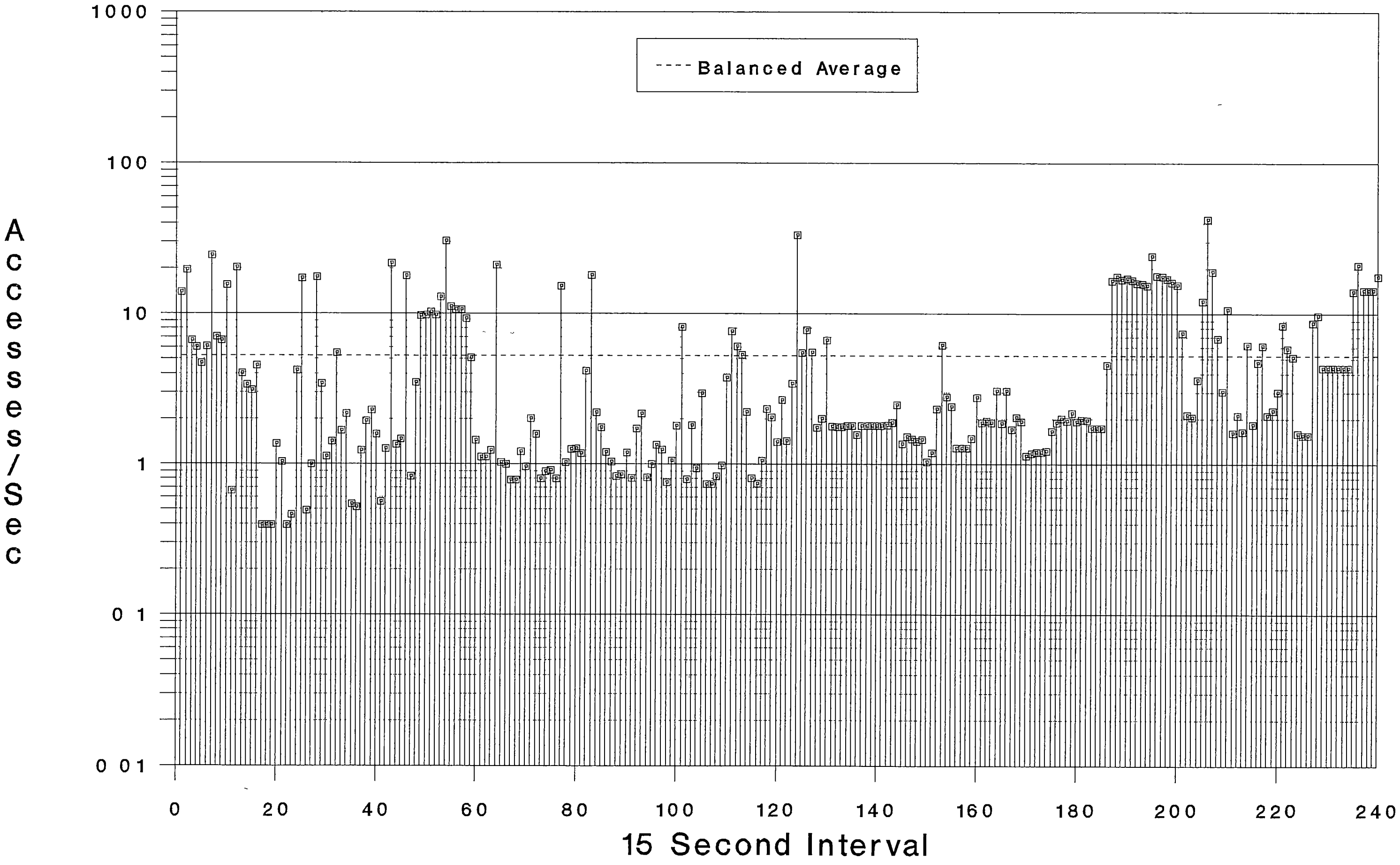


ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH013 -- UNBALANCED

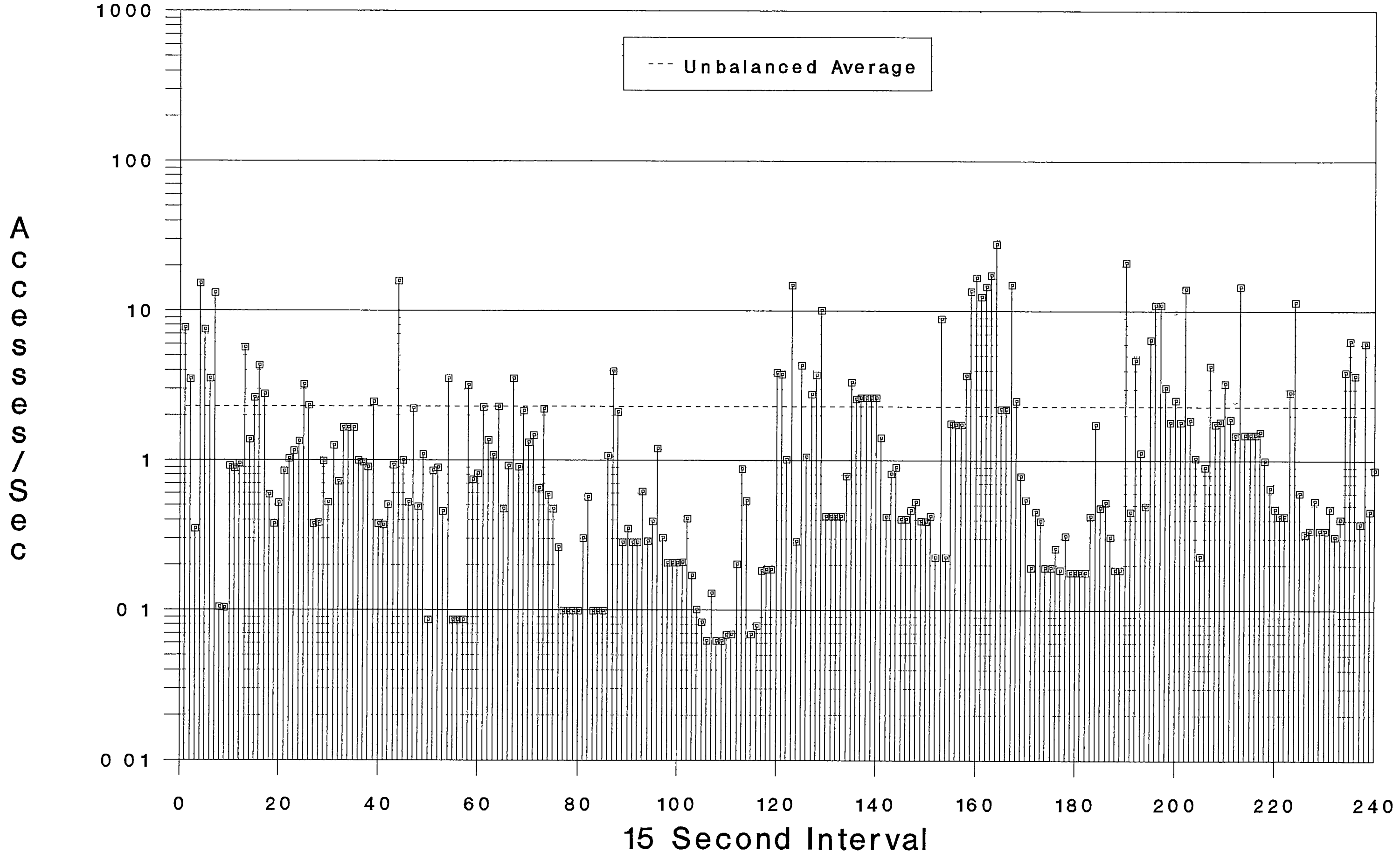


ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH013 -- BALANCED

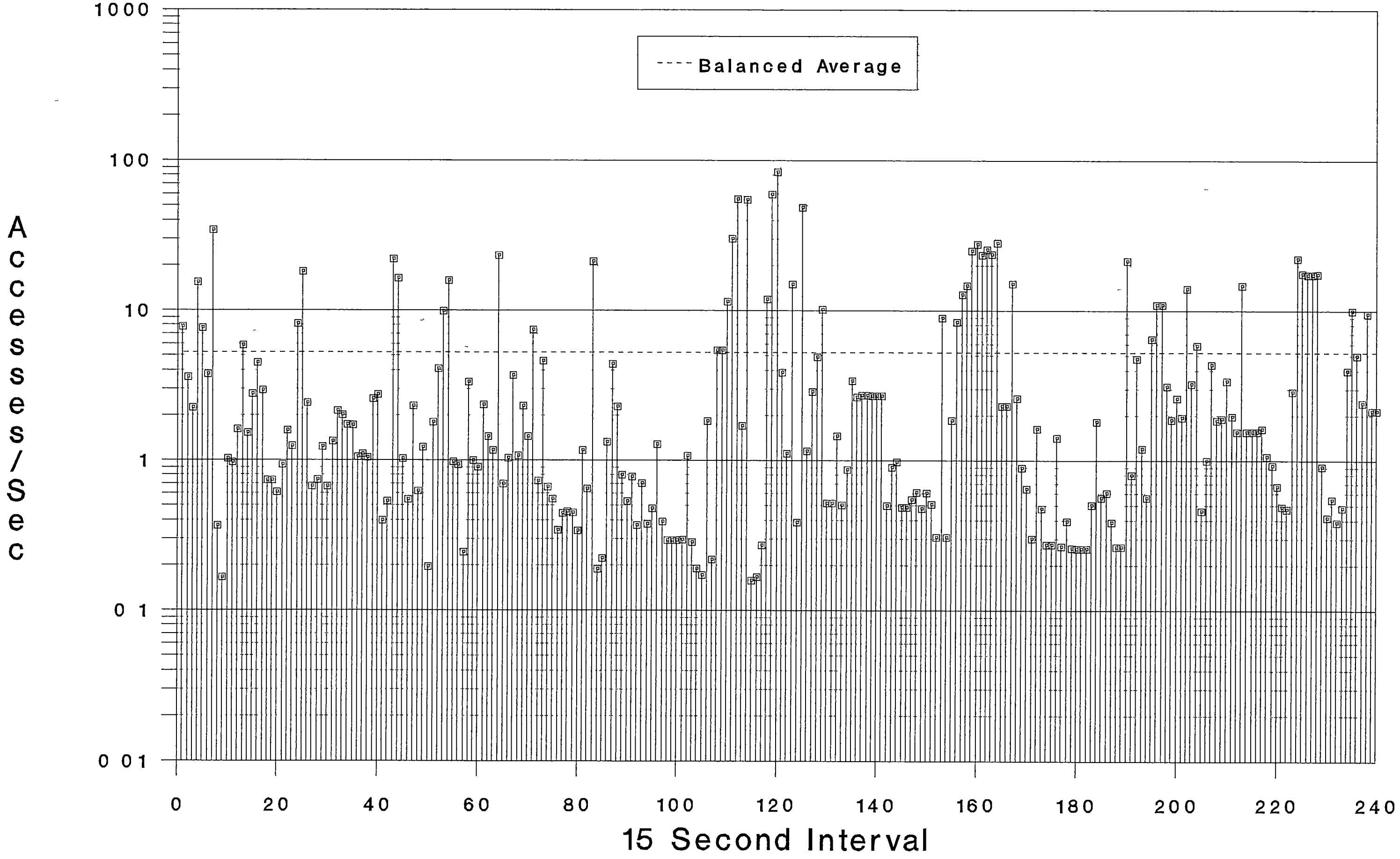


ACCESSES/SECOND - 15:00 TO 16:00 PM
TSH014 -- UNBALANCED



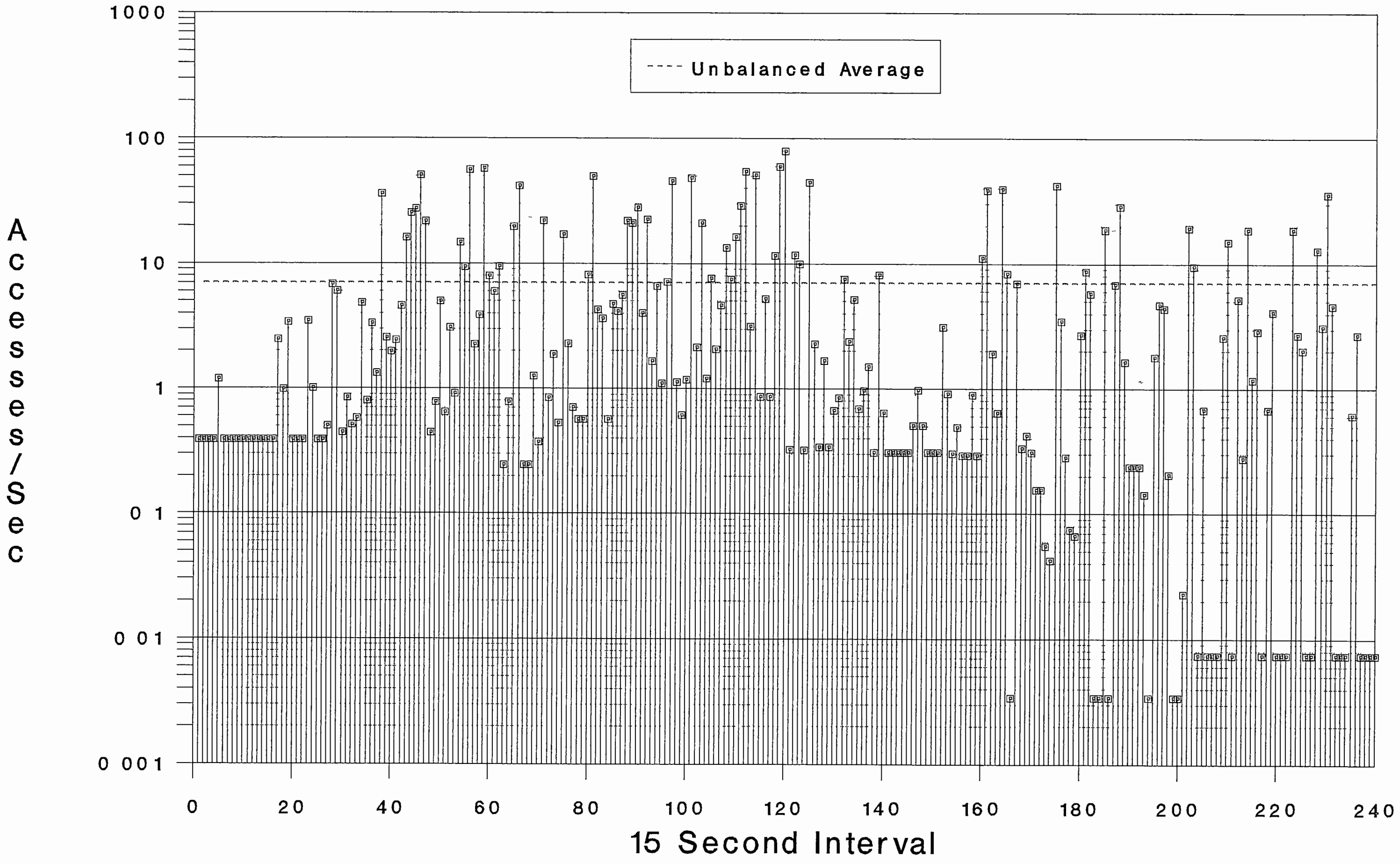
ACCESSES/SECOND - 15:00 TO 16:00 PM

TSH014 -- BALANCED



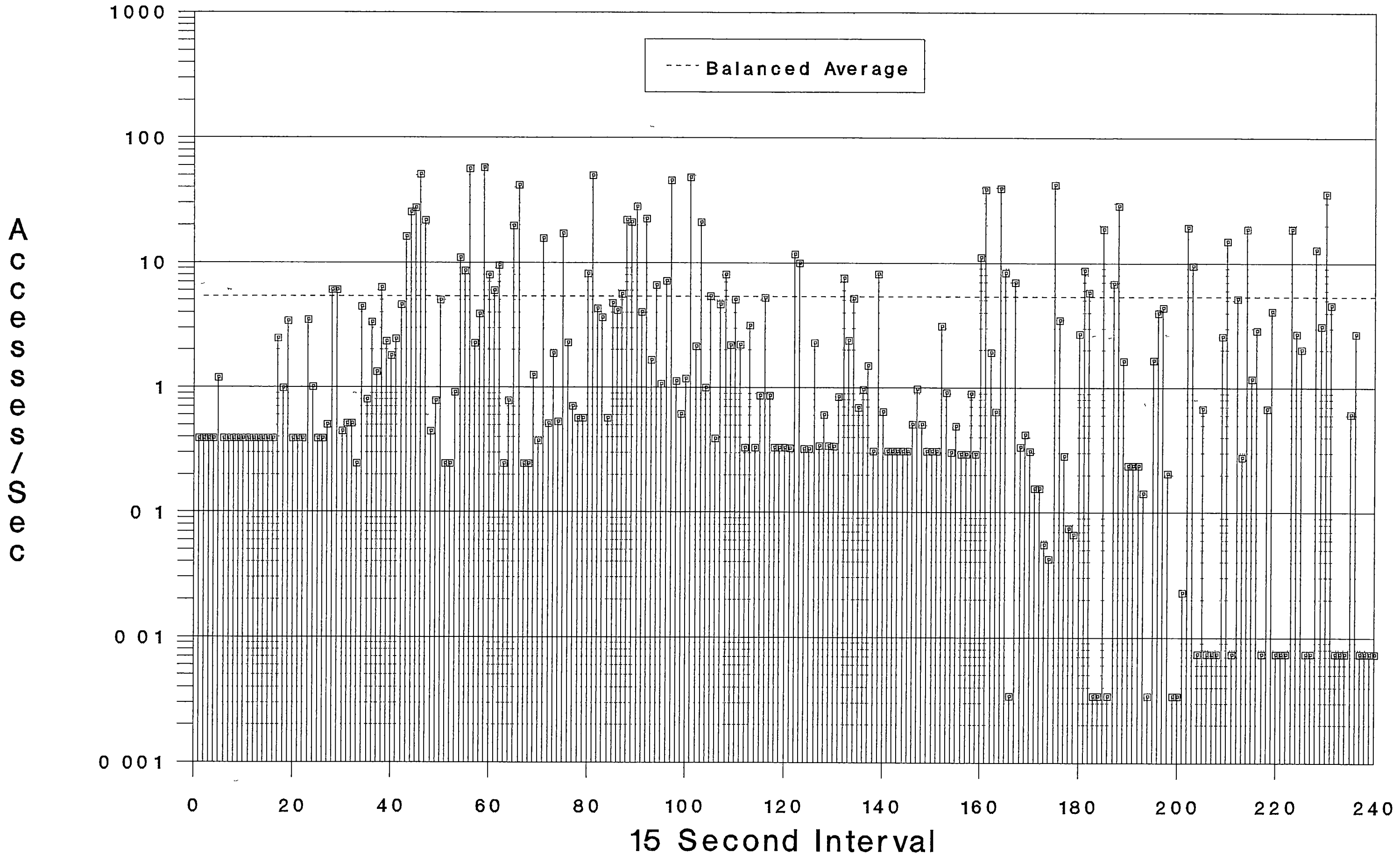
ACCESSES/SECOND - 15:00 TO 16:00 PM

USDD01 -- UNBALANCED



ACCESSES/SECOND - 15:00 TO 16:00 PM

USDD01 -- BALANCED



VITA

Roger R. Hill

Candidate for the Degree of

Master of Science

Thesis: USING STORAGE FACTORS TO BALANCE STORAGE SUBSYSTEM LOADS

Major Field: Computer Science

Biographical: Born in Ogden, Utah, July 30, 1953, the son of Reed A. and Helen U. Hill

Education: Received Bachelor of Science Degree in Civil Engineering in April 1979 from Brigham Young University; completed requirements for the Master of Science degree at Oklahoma State University in May, 1992.

Professional Experience: Software development, maintanence, and support for IBM 3090 and IBM PC applications, Research & Engineering Department, Conoco Inc., Ponca City, Oklahoma, January 1981 thru the present.