IBM

Washington Systems Center

Technical Bulletin

An MVS SRM Discussion

T. F. Bauer

GG66-0201-00
April 1985

Washington Systems Center
Gaithersburg, MD
Technical Bulletin

An MVS System Resource Manager (SRM) Discussion

B. R. Pierce

Edited by: T. F. Bauer

GG66-0201
August 1985

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis **without any warranty either expressed or implied.** The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to: Washington Systems Center, 18100 Frederick Pike, Gaithersburg, MD 20879.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Abstract

This paper will address areas of the System Resource Manager that seem to generate the most questions in forums, such as, the MVS Performance Project at SHARE. In fact, the paper is based on a presentation that has been given several times at both GUIDE and SHARE Meetings. The figures are representations of the overhead projector foils used in some of these presentations.

It will be assumed that the reader has some interest in performance management and has read or attempted to read the SRM section of the Initialization and Tuning Guide at least once. The paper will not address how to build an IPS or how to set OPT constants. It will provide information such as:

- What defines a TSO transaction?
- How does storage isolation really work?
- What is in a working set for logical or physical swaps?
- How does page stealing work?

The information will be applicable to both MVS/370 and MVS/XA.

Contents

Chapter 1. Introduction

SRM TOPICS

- SRM TIMING
- TSO TRANSACTIONS
- STORAGE CONTROL
- MPL ADJUSTMENT
- SWAP CONTROL
- I/O CONCEPTS AND CONTROLS
- LOAD BALANCERS

This paper will address areas of the MVS System Resource Manager (SRM) that seem to elicit the most questions from systems programmers and systems engineers. The paper is not a tutorial or a primer on the internal workings of the SRM. Rather, it is a discussion of concepts of some parts of the SRM that should be helpful to a systems programmer in order to tune and maintain an MVS system. The paper will assume a basic knowledge of the SRM and some of its constructs, such as basic IPS controls. On the other hand, the reader does not need to be a tuning expert to learn something about the SRM from this paper.

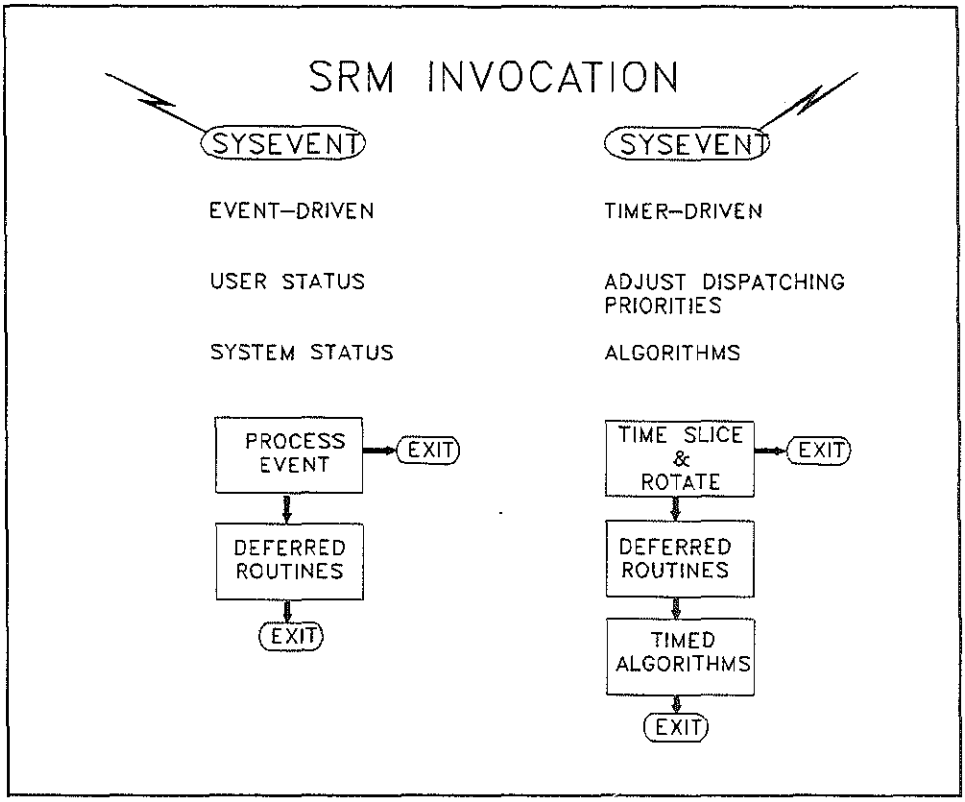
This paper covers several release levels of the SRM. It is based on MVS System Extensions Release 2 and MVS System Product Version 1 Release 1, which are nearly functionally equivalent from an SRM point of view. Changes made for MVS/SP Version 1 Release 3 and for MVS/SP Version 2 Release 1 will be highlighted in the paper.

The above outline shows the topics that will be discussed in this paper. Most of the topics that will be discussed are selectable and or adjustable by the installation through documented externals. The topics will be presented at a conceptual level; very little emphasis will be placed on keywords or syntax. The best source for that information is the *Initialization and Tuning Guide* [1].

---

1 For MVS/SP Version 1, see *OS/VS2 MVS System Programming Library: Initialization and Tuning Guide, GC28-1029.* For MVS/SP Version 2, see *MVS/Extended Architecture System Programming Library: Initialization and Tuning, GC28-1149.*

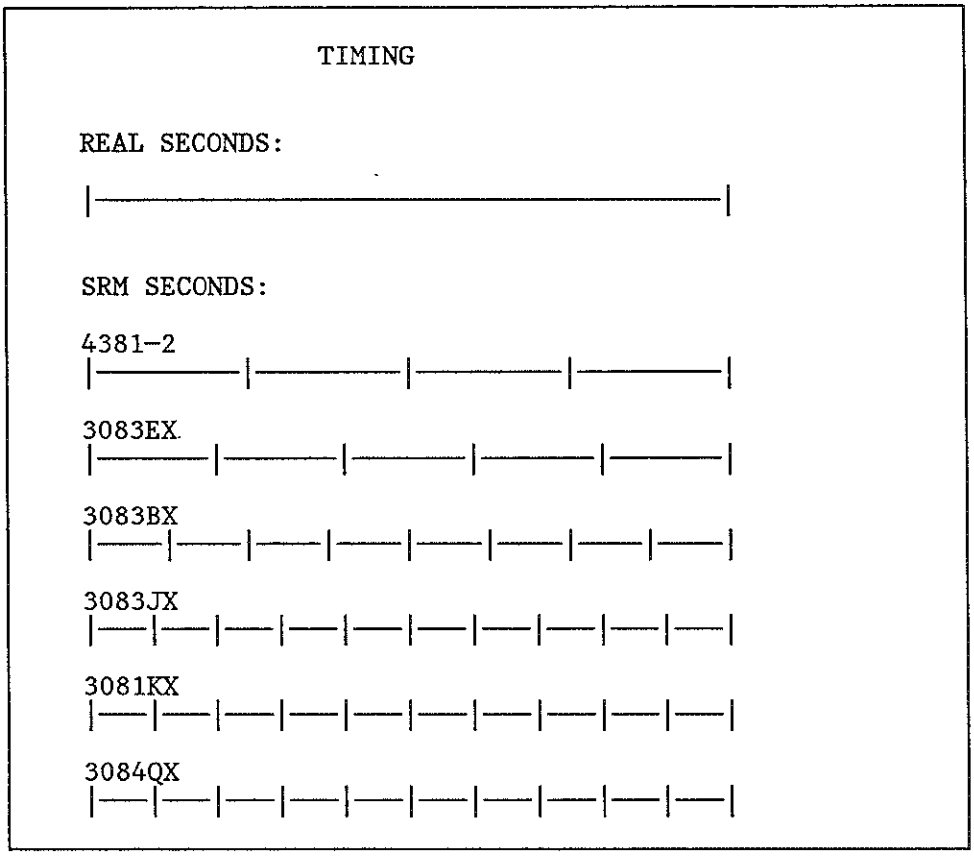
Chapter 2. SRM Timing



Understanding the implications of SRM timing requires some background on how the SRM is invoked. The invocation is via a SYSEVENT macro, which can be either event-driven or timer-driven. An event-driven SYSEVENT is issued as a result of a system or user status change, such as a TSO user becoming ready, or a swap out completing. The SRM processes the event and exits if the SRM lock is not held. The SRM lock is obtained on entry if the environment allows it to be obtained. If the lock is held, then any pending SRM routines requiring the lock - such as page replacement - will be run before exiting.

Timer-driven SYSEVENTs occur to adjust dispatching priorities or to run various SRM routines called "algorithms". The purpose of these algorithms is to check on or sample the utilization and status of the system workload and resources. Two examples of algorithms that will be discussed are page replacement or frame stealing and multiprogramming level adjustment.

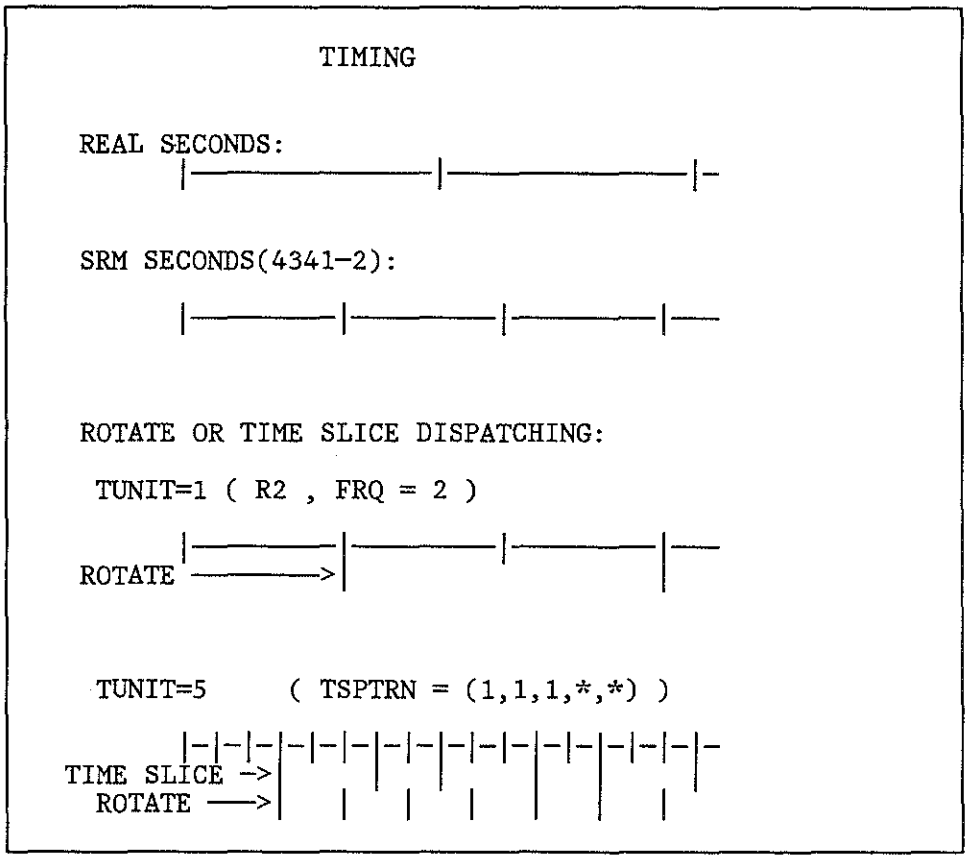
SRM Seconds



Timed events occur based either on real seconds or on a processor adjusted second, often called an SRM second. The relationship of SRM seconds to real seconds is a function of the approximate speed of a single processor of the host machine. For instance, a 3081 SRM second is much shorter than a 4381 second. Algorithms which are driven on the basis of SRM seconds are, therefore, executed more frequently on a 3081 than on a 4381. Many algorithms are adjusted for the processor. For example, the algorithm that checks for an IPS period switch should run more frequently as the processor speed increases, in order to maintain the integrity of the definition of a trivial TSO transaction. Obviously, an application can consume resources at a higher rate on the faster machine.

Most algorithms are adjusted based on the approximate speed of a single processor. The rationale is that very few applications can use more than one processor at a time effectively. Therefore, resource consumption will be limited to something less than one processor for any given address space. This explains why a 3081KX has about the same size SRM second as a 3083JX, although the 3081KX has almost two times the capacity of the 3081JX. As a historical note, when MVS Release 2 was shipped, one SRM second on a Model 155 II was defined to be equal to a real second.

Time Slicing



No SRM algorithm is executed more frequently than once per SRM second. However, there are several routines that do run every SRM second. Therefore, the default rate of SRM timer interrupts is one per SRM second. On a 3033, the default time between timer interrupts is 160 milliseconds. It is possible for an installation to speed up or slow down this frequency with IPS and OPT parameters. The IPS TUNIT keyword can be used to increase the frequency of timer interrupts to adjust dispatching priorities via the rotate or time slice dispatching control functions. Both of these functions operate on the basis of SRM timer units. If the TUNIT specification is 1, or defaults to 1, an SRM timer unit equals one SRM second. In the first rotate example, the frequency for rotate priority 2 is 2 timer units.Since a timer unit equals an SRM second, the priorities will be rotated every 2 SRM seconds.

The second example is slightly more complex and involves both time slicing and rotating. The frequency of rotation is still 2 timer units. However, the length of a timer unit has changed because TUNIT = 5 was specified. This means there are 5 timer units per SRM second. Therefore, on a 3033, the length of a timer unit is 32 milliseconds. This does not mean that there will be 5 times as many timer interrupts with this TUNIT specification. The SRM only sets timer interrupts when there is something to do. If there were no jobs associated with the performance groups using rotate and time slicing, there would be no extra timer interrupts at all. In the example, time slice group 1 stays "up" for 3 timer units and then "down" for 2. No timer interrupts are set by the time slice routine until it has some function to perform.

In a few environments, such as an MVS test system under VM, it may be appropriate to alter the frequency of timed routines. This can be done by increasing the time between timer interrupts. This is accomplished by specifying a parameter in the OPT.

Processor Service Units

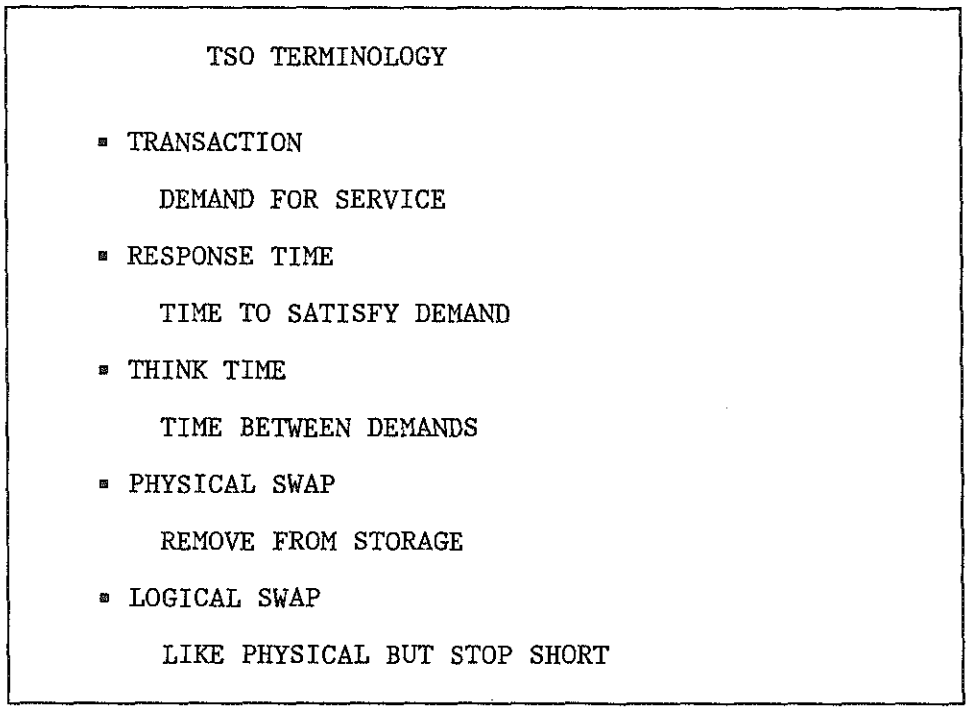
```
                    TIMING

SERVICE UNITS BY PROCESSOR:

          4381-2    181.0/SEC

          3083EX    213.2/SEC

          3083JX    420.0/SEC

          3081KX    399.0/SEC

          3084QX    399.0/SEC    PARTITIONED

          3084QX    373.8/SEC    SINGLE IMAGE

TO VERIFY THE VALUE ON ANY PROCESSOR:

     RMCT - SRM CVT (<-- CVTOPCTP)

     FIELD RMCTADJC / 16 = TIME FOR

      1 CPU OR SRB S.U. IN MICROSECONDS
```

Above are listed the service unit definitions for some of the IBM processors. As shown, neither MVS/370 or MVS/XA supports all of these processors. The point is that where both levels support a processor, such as the 3081K, the definition of CPU and SRB service units is the same. Another point is that, in 370 mode, the SRM made no distinction between a multiprocessor and a uniprocessor. This is no longer true in XA mode. The 3084, in single image mode, has a slightly lower number of service units per second of processor time. This adjustment is made because, with tightly coupled multiprocessing, there is bound to be some contention for storage and other resources, so that any unit of work may require slightly more processor time. This is not to say that less work will be accomplished by the multiprocessor, than the same number of processors configured as uniprocessors. There are many benefits, such as load balancing and shared resources, that make multiprocessors good price performers.

The above numbers are from the *Initialization and Tuning Guide*. Starting with RMF 3.3, the Workload Activity report shows the number of service units per CPU second. For systems prior to RMF 3.3, the installation can verify the value that the SRM is using on the current configuration with a very simple TSO TEST session. The field names and, therefore, the offsets can be found in the Debugging Handbook. If this technique is used, the installation may find that some of the values listed are not perfectly accurate. This is because the values are obtained using a calculator and are not quite as accurate as on a real machine.

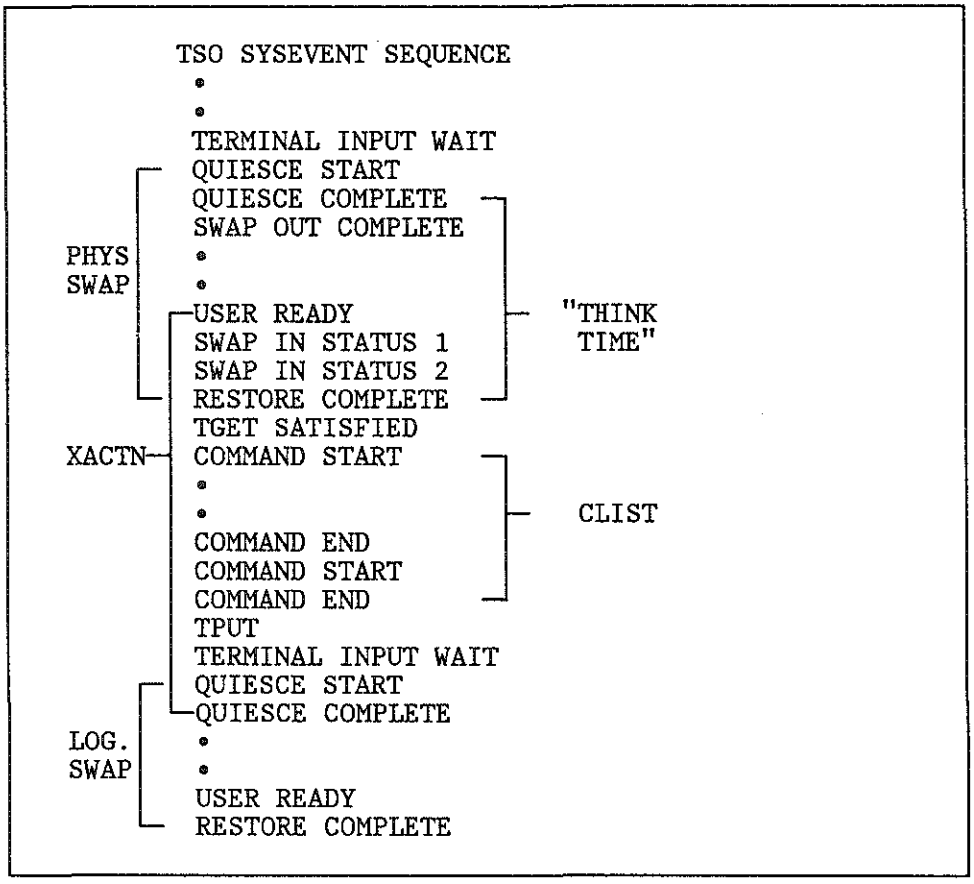
This concludes the section on SRM timing. For the rest of the paper, when references are made to times, the times will be real times unless indicated otherwise.

Chapter 3. TSO



Before the discussion of storage control, with topics such as logical swapping, some of the terms should be defined. The first term is a transaction. It might also be called "a demand for service". An end user might define a transaction as the request for processing made when the enter key is pressed on the terminal. Response time is the time for the system to process one transaction and prepare for another. Think time is the time between user requests for service that allows MVS to run hundreds of users on one processor. If MVS does its job right, each user thinks they have the machine to themselves. With programs like ISPF, the end user may be very busy typing, but to the system or SRM, he or she is thinking. A physical swap is disconnecting the user's address space from real storage. This allows MVS to multiprogram the storage as well as the processor. A logical swap is like a physical swap and thus the name. However, MVS does not remove the address space from storage right away. MVS prepares for that eventuality but most of the time the I/O is not performed.

A TSO Transaction



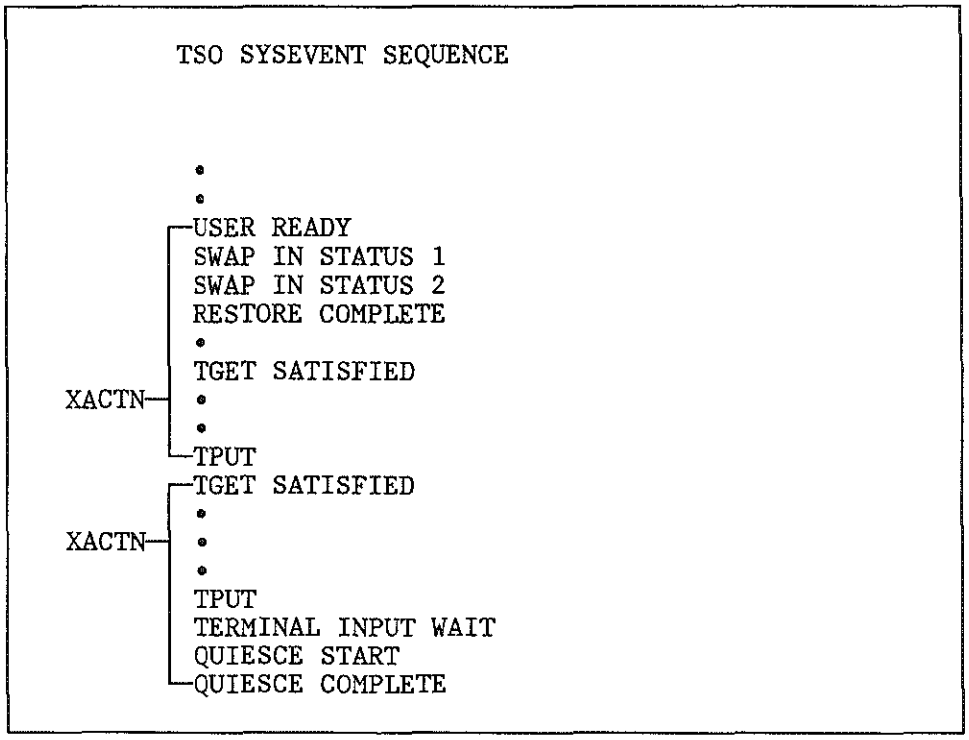
The SRM is able to define think time and response time because it hears about the progress of a TSO user's demand for service through various SYSEVENTs that are issued. In the example, a SYSEVENT is issued stating that the address space may be in terminal wait. What this means is that the application asked for input, and there was no input to process. MVS quiesces the address space to determine that there really is no work to do. In this case, MVS physically swaps out the address space to reclaim the real storage that it owned. When some processing is later scheduled in the address space, usually by the end user hitting enter, the SRM is informed by a user ready (USERRDY) SYSEVENT. The SRM requests that the Real Storage Manager (RSM) swap in the address space, and it is restored by the Region Control Task (RCT). The SRM defines the start of the transaction at the USERRDY SYSEVENT.

Think time is defined as the time from the quiesce complete (QSCECMP) SYSEVENT, to the restore complete (RSTORCMP) SYSEVENT. A performance analyst would define think time differently. The SRM will be using the think time in comparisons, however, and needs to include any delay that may have been imposed directly or indirectly.

A TGET satisfied (TGETTPUT) SYSEVENT indicates that the program that went into terminal wait is no longer in terminal wait. The command start (CMDSTART) and command end (CMDEND) SYSEVENTs indicate that a CLIST is being processed. These SYSEVENTs allow the SRM to report on TSO command usage through RMF reports. They also allow the SRM to count each command in a CLIST as a separate transaction, if the installation chooses that option. In this case, the installation has not chosen that option. These SYSEVENTs are issued by the TSO Command Package or the TSO/E Program Product.

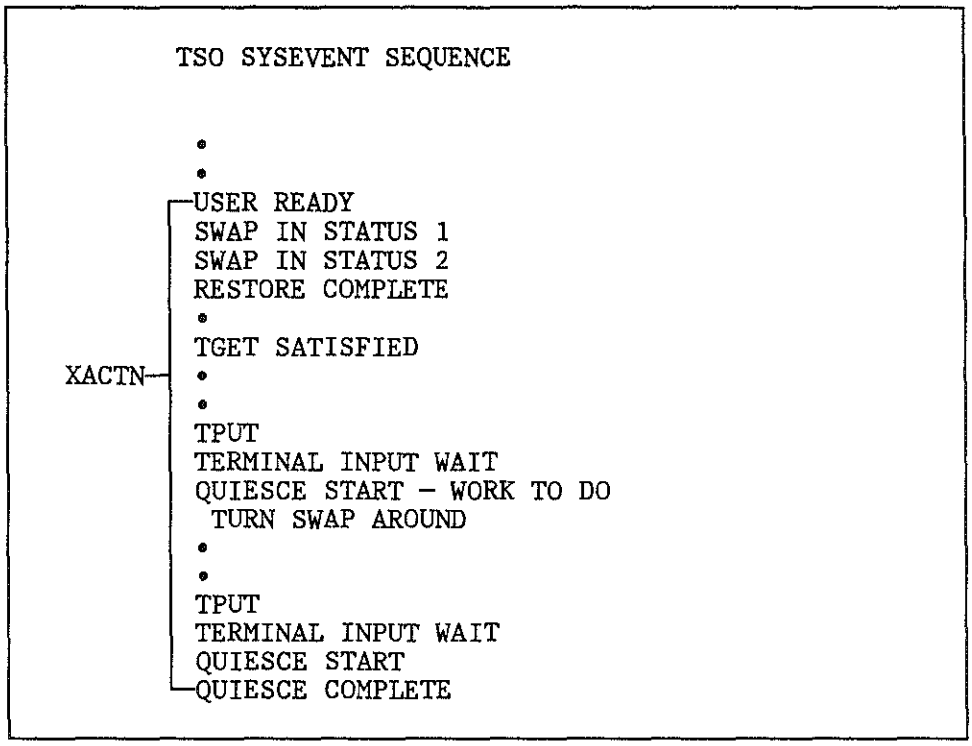
The address space again enters the terminal wait condition. This ends the transaction started by the USERRDY SYSEVENT. The response time is calculated from the time of the USERRDY SYSEVENT. This is the most common way to end a TSO transaction. In this case, the address space is logically swapped instead of physically swapped. Later, the mechanics of the logical swap process will be discussed. A decision is made, based on the measured think time and some other metrics that determine the capability of the complex, to keep an address space with this think time in real storage.

Multiple TSO Transactions



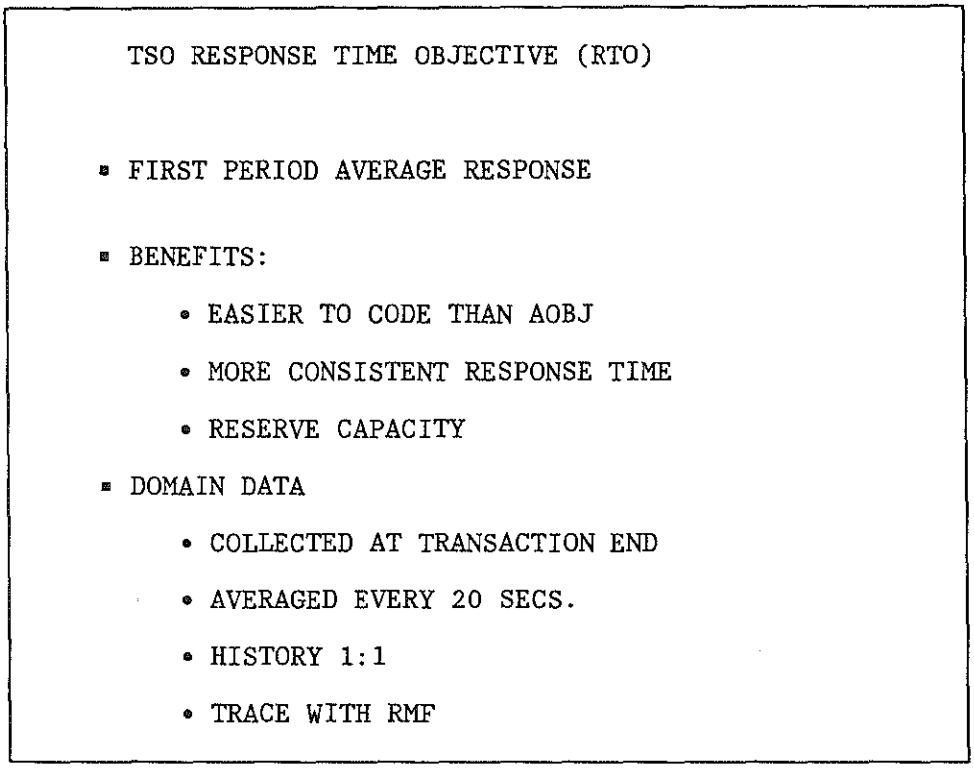
The above figure depicts a SYSEVENT sequence to show another way to start and end a TSO transaction. It also shows how an RMF workload report can have more transactions than swaps. In the previous figure, there was a TGET satisfied SYSEVENT, following a swap-in, and then a TPUT satisfied SYSEVENT, before entering terminal wait. In this case, there is a second TGET satisfied SYSEVENT. This means that there was a command for the application to process without the terminal user's intervention. This can happen if the user has stacked commands using the field mark capability of a 3277, for example. When the second TGET satisfied SYSEVENT is received, the SRM ends the transaction initiated at USERRDY and starts a new transaction which later ends due to a TERMWAIT SYSEVENT, as in the previous figure. Another way to get multiple transactions in the interval would be to count the commands in a CLIST as separate transactions.

A TSO Transaction with Multitasking



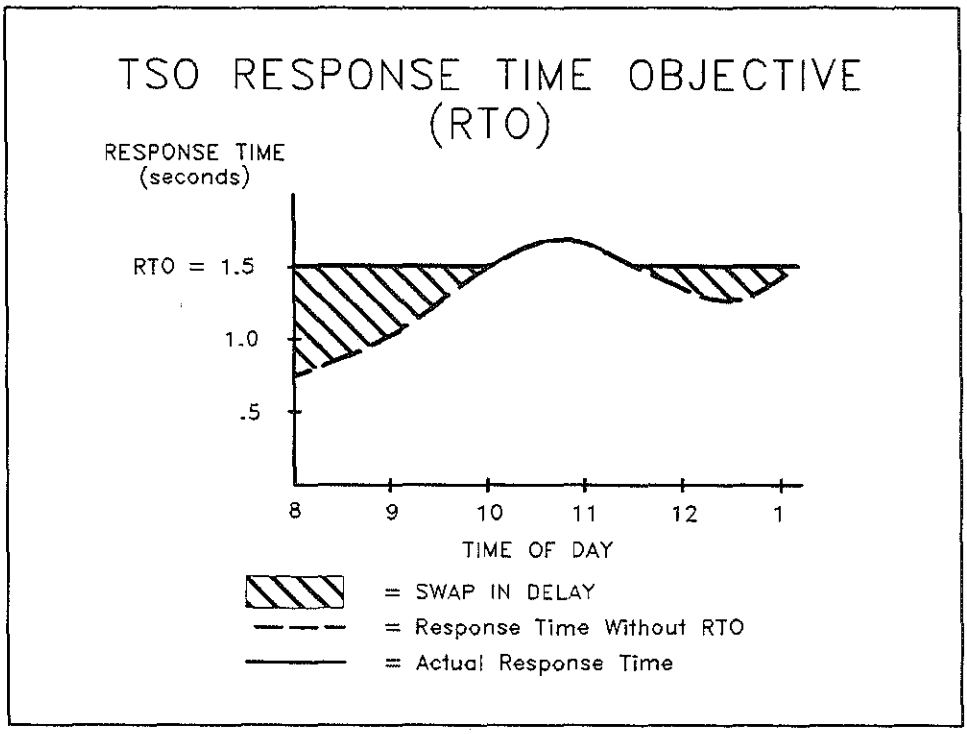
The above figure demonstrates that the TERMWAIT SYSEVENT may not always cause a TERMINAL WAIT swap and a transaction end. The SYSEVENT only indicates that some application may be in terminal wait. The quiesce process is necessary to verify the wait. If quiesce finds other activity in the address space, for example, a ready task or I/O queued or in process, the address space is not considered in a wait and the transaction is not ended. This could be possible with multitasking applications under TSO. In that environment, the definition of transactions and response times for those transactions would be unlikely to agree with the perceptions of the end user.

TSO Response Time Objective (RTO)



The response time objective option was designed to help make TSO response time more consistent, regardless of the load on the system resources. It does this by imposing a delay on the front of most TSO transactions, if necessary, to meet the response time goal for first period or trivial TSO response time. This can reserve capacity for the future or just smooth out fluctuations in response time as seen by users.

End users demand very good response time. Studies have shown dramatic improvements in productivity with subsecond response times. However, some installations believe that consistent response times are equally important. If the installation does not have the capacity to deliver 300 millisecond response times at all times of the day, it may choose to deliver a consistent one second response time. The choice is with the installation; however, in the past there have been questions about the function which the following discussion will address.



This figure shows how the function works. The SRM calculates the time from the end of any RTO imposed delay until the end of the transaction. The data is kept by domain, not by performance group, in order to get more samples. This data is averaged at the same time MPL's are adjusted. Based on these times, a delay is imposed to new transactions before they are swapped in. They may be in the logical or physical swap state. If transactions are not completing within the desired time, as in the 10 to 11:30 time frame, no delay is imposed.

TSO RESPONSE TIME OBJECTIVE (RTO)

"IS EVERY TRANSACTION SUBJECT TO DELAY?"

- DELAYS INITIAL SWAP-IN EXCEPT FOR
  - OUTPUT TERMINAL WAIT SWAP
  - ENQ RESOURCE HOLDER
  - SWAP IN FOR CANCEL
- NOTE NO DELAY FOR 2ND XACN DURING SWAPPED IN INTERVAL

Installations should not expect that the response times reported by RMF for performance groups with RTO will match the specified response time objective closely at all times. All transactions are not delayed. Address spaces swapped for output terminal wait are not delayed. Generally, this would be a small percentage of ending transactions. If there were a significant percentage of these swaps, the effectiveness of RTO could be impaired. Also, address spaces impacting other address spaces, due to ENQ resources, and address spaces being swapped in for cancel, are not delayed. In addition, no delay is imposed for the second and subsequent transactions, once the address space is swapped in initially.

Chapter 4. Storage Control

STORAGE CONTROL

"HOW IS STORAGE CONTROLLED BY BASE SRM?"

"HOW MUCH CAN I CONTROL STORAGE?"

TOPICS

- UNREFERENCED INTERVAL COUNT
- FRAME STEALING
- PHYSICAL SWAP WORKING SET
- LOGICAL SWAPPING
- STORAGE ISOLATION

The next, and by far the largest, topic in this paper is storage control. Storage control does not seem to be as well understood as processor control. First, an explanation of the kinds of storage control provided by default is in order. To really describe storage control, it's helpful to understand multiprogramming level control and logical swapping, since those functions allocate storage to address spaces. Those subjects will be covered later. For the moment, just assume that there is real storage, and that there are applications that need varying amounts of that storage to do the work that they were designed to do.

The default control is to assign the storage in page frame units to the applications, as they reference virtual pages not currently in a page frame. The supply of available page frames is not inexhaustible, of course. The page replacement policy is to pick the pages that have been least recently referenced and remove them from the page frames, so that they can be reallocated. The selected pages are written to page data sets, if they have been altered since they were last written to a page data set. Therefore, MVS defaults to a global least recently used (LRU) page replacement algorithm.

A global LRU algorithm works quite well when all applications are of equal importance, or if the resource requires very little control due to its abundance. There are many cases where the resource is not in abundance, and the applications are far from equal in importance to the installation. In these cases, storage isolation can be a powerful tool to aid the SRM in the distribution of the storage resource.

Unreferenced Interval Count (UIC)

UNREFERENCED INTERVAL COUNT (UIC)

Q. "WHAT IS A UIC ?? "

A. "THE # OF SECS A FRAME IS UNREFERENCED."

- LOGICAL PART OF A PAGE FRAME
- SET TO 0 FOR ALL FRAMES AT ALLOC
- SYSTEM HIGH UIC
  - AGE OF OLDEST PAGE NOT SUBJECT TO STORAGE ISOLATION
  - DRIVES STEAL ALGORITHM
- AVERAGE IS CONTENTION INDICATOR
  - LOGICAL SWAPPING
  - STORAGE LOAD BALANCER
  - MPL ADJUSTMENT

The unreferenced interval count (UIC) is a logical entity associated with an occupied page frame. It is a number indicating the approximate number of real seconds that the page frame has not been referenced. The SRM can tell when the page frame is referenced because there is a physical entity, the page reference bit, that is turned on by the hardware when any part of the page frame is referenced. When a page frame is allocated to an address space at page fault or swap-in time, the reference bit is set off and the UIC is set to zero. The RSM periodically interrogates the bit and in the process turns the bit off. If the bit is on, the UIC is set to zero. If the bit is off, the UIC is increased by the amount of time since the last interrogation.

The larger UIC's are associated with the pages that have not been referenced recently. The system high UIC is the largest of these values. The system high UIC will be used as the starting selection value for the LRU page replacement algorithm. The SRM wants to take the oldest pages and needs to know where to start. The storage isolation function allows an application to retain real page frames that might otherwise be stolen. Therefore, the age of pages controlled by storage isolation is not included in the system high UIC because the pages might not be eligible to steal and the steal routine would be wasting its time looking for these old pages.

The average system high UIC is used as the primary indicator of the degree of contention for real storage. Obviously, a large value means that pages can be unreferenced for a long time without being stolen. Therefore, storage must be plentiful. The contention indicator is used to control logical swapping, the storage load balancer, and the MPL adjustment.

UIC Update Intervals for MVS/SP 1

UNREFERENCED INTERVAL COUNT (UIC)

UPDATE INTERVAL - MVS/SP 1

	HIGH UIC	NON SWAPPABLES/ COMMON	SWAPPABLES
HIGHER	0-2	1	1
	3-5	2	1
CONTENTION	•	•	1
	•	•	1
LOWER	16 & UP	6	2

The above figure shows the update intervals for the UIC update routine described earlier. The routine must examine the reference bit of most of the real storage pages online. Therefore, the frequency of the algorithm is decreased as the contention for storage decreases. The theory here is that as the SRM steals older pages, it does not have to be as discriminating about the exact age of the pages because it will be doing it less frequently. Also, the impact on the application will be less than when stealing more recently referenced pages. The top of the figure shows the frequency for MVS/SP Version 1.

For nonswappable address spaces and the common area (LPA and CSA), the update interval varies from 1 to 6 seconds, as the system high UIC varies from zero to 16 seconds. For swappable address spaces, the update interval is 1 or 2 seconds, depending on the system high UIC. The range is smaller here because the reference patterns are needed to determine what pages should be reassigned to the address space, if it is swapped.

UIC Update Intervals for MVS/SP 2

UNREFERENCED INTERVAL COUNT (UIC)

UPDATE INTERVAL – MVS/SP 2

	HIGH UIC	NON SWAPPABLES/ COMMON
HIGHER	0–5	1
	6–11	2
CONTENTION	•	•
	30–35	6
LOWER	•	6
	120–125	20

	HIGH UIC	SWAPPABLES
HIGHER	0–9	1
	10–19	2
CONTENTION	•	•
	50–59	6
LOWER	•	6
	•	6

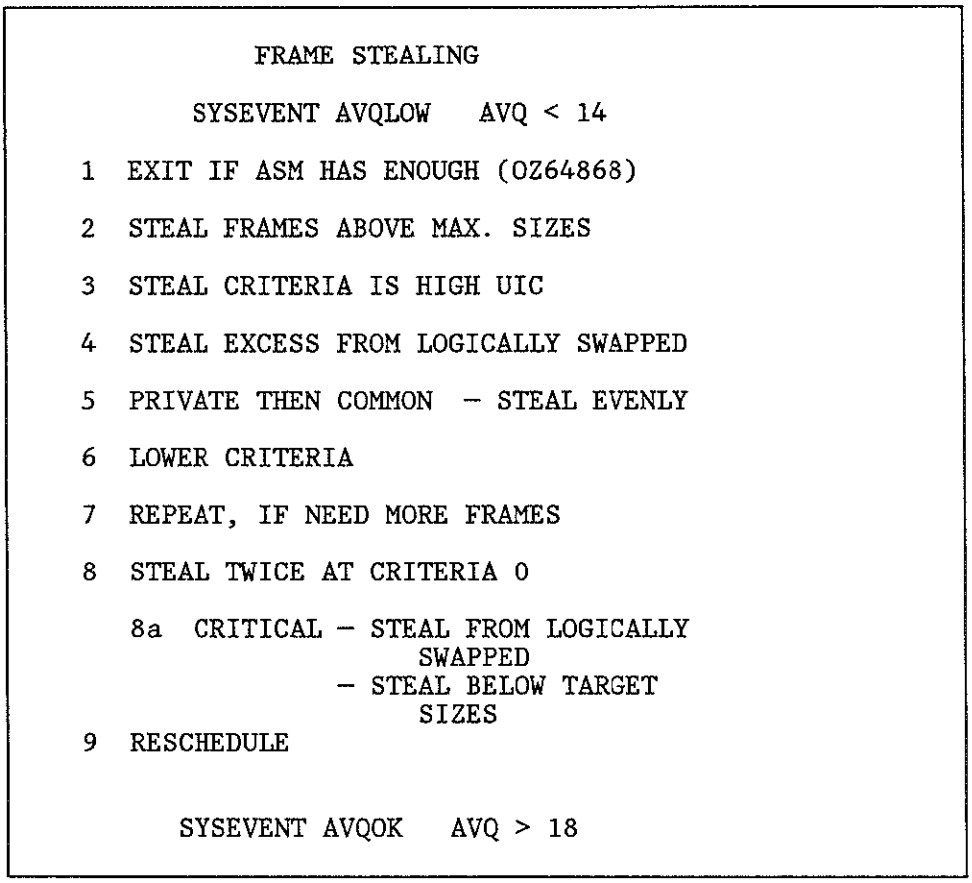
MVS/SP Version 2 was designed to manage much larger real storage than Version 1. Therefore, the frequency of the UIC update routine is different. In this case, the interval can vary from 1 to 20 seconds for the nonswappable address spaces and the common area. The interval is a regular step function of the system high UIC. This is also true for the interval for swappable address spaces, although the upper bound is 6 seconds. This automatically allows larger swap working sets when storage is available. Larger swap working sets have been found to be advantageous in most environments that are response oriented.

Frame Stealing

FRAME STEALING

- WHY IS IT DONE?
- WHEN IS IT STARTED?
- HOW DOES STORAGE ISOLATION AFFECT IT?
- WHEN IS STORAGE ISOLATION IGNORED?
- ARE LOGICALLY SWAPPED USERS STOLEN FROM?

Above are some of the questions that this discussion intends to address about the page replacement process. Frame replacement is done so that the RSM will have real page frames available to satisfy page faults and to swap in address spaces. The RSM starts it and a later discussion will show how. Storage isolation definitely affects the process - that is what it was designed to do. However, it can be ignored under certain circumstances, particularly if it has been used indiscriminately.



The page replacement process is started by the RSM issuing a AVQLOW SYSEVENT to the SRM to indicate that the supply of available page frames is less than a threshold set by the SRM. The threshold is normally set to 14, but it can be changed by the SRM when an address space is to be swapped in, and enough frames must be gathered up to accomplish the swap-in.

The first step on this figure was added in an APAR fix as indicated. This step determines if any previously stolen pages are still in the process of being paged out by the Auxiliary Storage Manager. Any frames containing such pages are counted toward the goal for this pass. The reason for this addition is that there may be bursts of page faults, particularly for programs starting a new phase of processing, requiring large amounts of virtual and only processor execution time to reference the virtual.

Prior to the APAR change, page replacement would be performed each time the available frame queue was entirely depleted. There was no check for previously stolen pages. The philosophy was that the SRM does not want to delay an application for lack of real frames. On each pass, it was likely that some unchanged pages were stolen which could immediately replenish the available frame queue. This would allow the hypothetical program to reference more storage and start the steal process again.

This problem has become significant of late because the processors have become more and more powerful allowing such a program to reference a significant amount of storage in a very short time relative to the paging subsystems capability to deliver the stolen pages and reclaim the frames. Analysis showed that these types of uses of real storage are more common to batch applications and very uncommon to online subsystems like IMS and CICS. The more aggressive stealing, to prevent storage shortages, could only hurt the more response oriented applications, to favor less important applications. Therefore, the check was added and the SRM will now allow the available frame queue to be empty, rather than continuing to steal in the hope of getting unchanged pages.

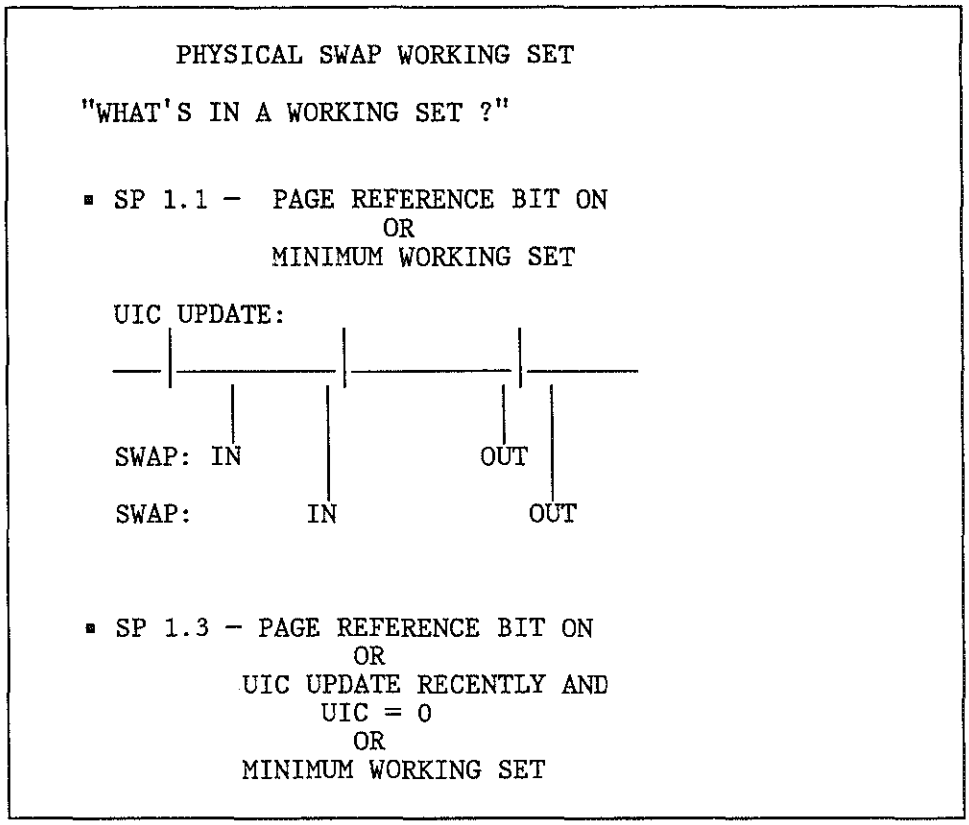
When stealing is necessary, storage isolation may come in to play. Before the SRM enters the LRU part of the process, it sees if any address spaces or the common area have more pages than they should have, according to the maximum working set sizes that have been specified. These terms will be described better shortly. Usually, few pages are stolen in this phase. At step 2, the SRM starts the LRU process by setting the criterion for this pass to the system high UIC.

Step 3 was added in SP 1.3. The working set size of a logically swapped address space is a function of the system high UIC in SP 1.3. The idea is to reduce page faults and improve response time when storage is available. However, the demand for real storage can change dramatically in a short period of time. Therefore at each pass in this algorithm, the SRM checks to see if the steal criterion is below a threshold and if it is, it visits the logically swapped address spaces and removes the extra pages that may have been allowed to keep.

At step 4, frames are stolen from address spaces and the common area. The SRM does not steal the pages itself; it merely indicates to the RSM the address space and the steal criterion. The steal is done evenly from all candidates by stealing no more than a constant number from each candidate before checking others. The constant is 2 in SP 1 and is 10 or 4 for SP 2 depending on the criterion. At this step, storage isolated address spaces may or may not be stolen from. They are candidates if they own more frames than their working set target. If they have more than their target, but their frames are not as old as the criterion, they will not be stolen from, just like any other address space.

This process is repeated at lower and lower criteria, if necessary, until enough frames have been designated to replenish the available frame queue. If the steal criterion must be reduced to zero to get enough frames, a second pass is made. This is because a frame will never be stolen with the reference bit on. The process of examining a frame to steal also turns off the reference bit. The second pass picks up those frames. If the SRM still can't get enough frames, it considers this a critical situation. At this point, it will steal any pages it can get. It will look for logically swapped address spaces and then ignore storage isolation target working set sizes and do a straight LRU on all of storage. Critical situations should not be common occurrences, but they can occur if storage isolation is used indiscriminately. This is a powerful function and must be used wisely.

The page replacement algorithm always reschedules itself so that it can see how the supply of available frames is and initiate more steals if necessary. The RSM notifies the SRM when the supply of frames exceeds a threshold, normally 18 frames. That ends the process.

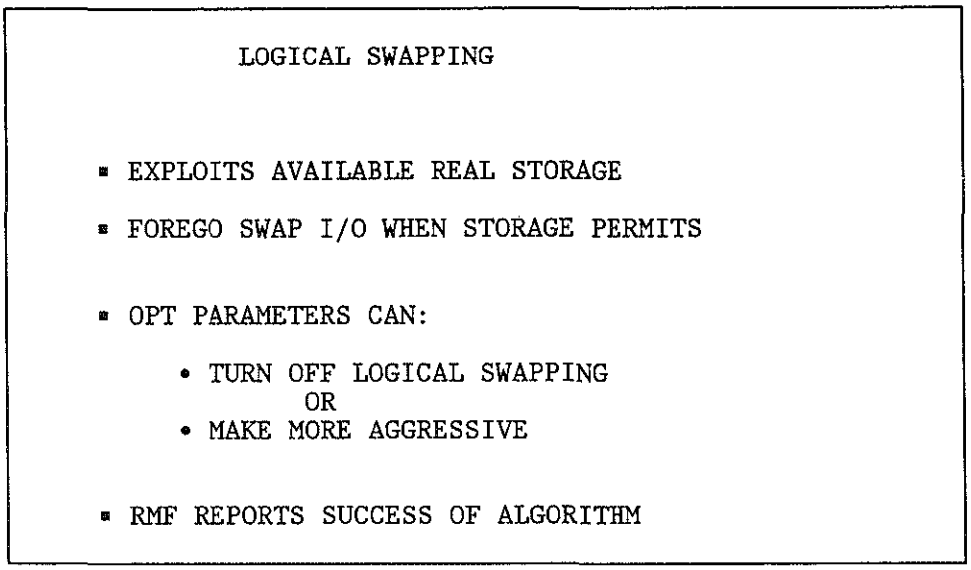


As stated earlier, the reference patterns of swappable address spaces must be determined to decide which pages should be allocated to page frames, if they are swapped out and back into storage. The installation can see how many pages are reallocated or what the average swap working set size is from the RMF paging activity report. The installation can also see the swap working set size for an address space from the RMF Monitor II Address Space State Data or ASD report. The field name is WS IN and contains the working set size from the last physical swap.

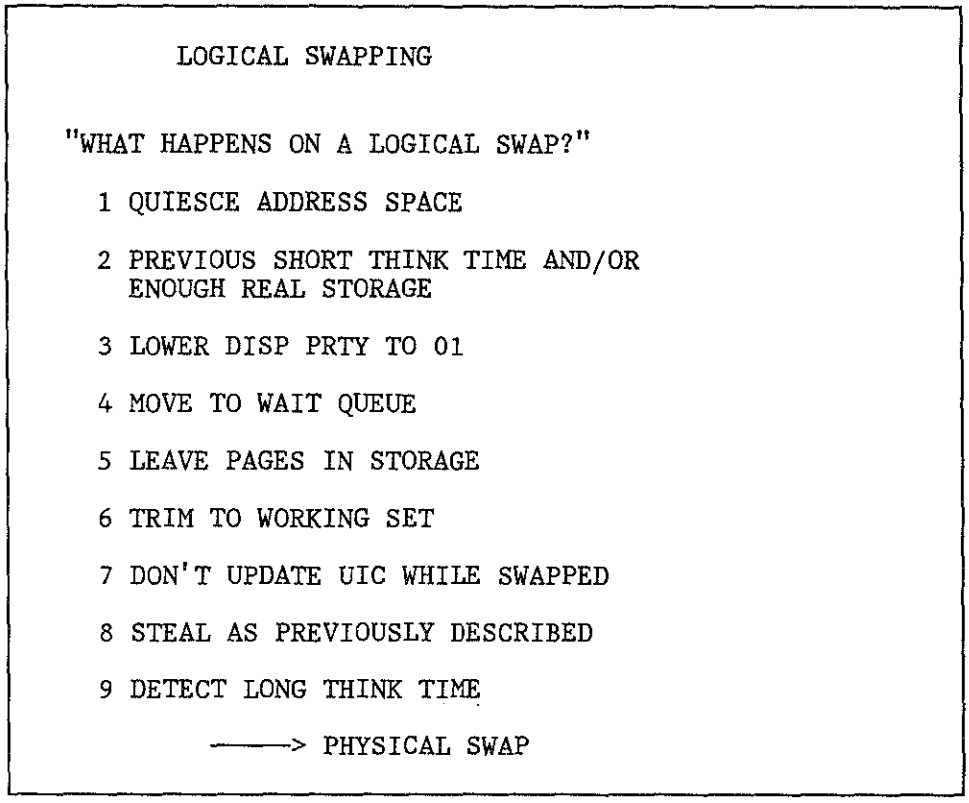
Prior to SP 1.3, a page was considered to be in the swap working set if the page reference bit was on. It could be in the working set, if the page reference bit was off, only if the address space was storage isolated and eliminating the page would reduce the working set below the minimum working set specification. If storage isolation is ignored, the intent of the page reference bit test is to leave the most recently referenced pages in the working set. Since the routine that turns off the reference bits runs every one or two seconds, the working set should include pages referenced within the last 1 to 2 seconds. The routine runs at a variable frequency, but when it runs, it processes all swappable address spaces. It can be seen that for the 2 swap sequences shown, the working sets will be different. The second address space will have very few pages with the reference bit on. This explains the success of using storage isolation with the extended swap IUP. The swap operation is improved and storage isolation increases the working set and makes it more consistent, thereby reducing page faults and improving response time, which can decrease the net storage needed for TSO.

In SP 1.3, the swap working set should be more consistent because the test was changed. When the UIC update routine is executed, it will skip address spaces that were swapped in during the last half of the interval since it ran. This saves processing time and allows the address space time to establish its reference pattern. When an address space will be swapped out, pages that were recently referenced, but which may have the reference bit off, are included in the working set. Recently referenced includes pages where the reference bit was turned off within one half of an update interval. This change will increase the size of the average working set because it is more consistent.

The reduced frequency of the UIC update process in SP Version 2 will increase the working sets even more in environments with large real storage. Both changes result in fewer demand page operations and improved TSO response time. Storage isolation can still be used to increase swap working sets to reduce demand page operations. Of course, the paging subsystem has to be capable of handling the increased swap load that will result.



The purpose of logical swapping is to exploit real storage to reduce the use of I/O and processor resources. If conditions are suitable, instead of physically swapping out address spaces, the SRM will decide to leave them in storage, but in a not-executable status. Through parameters in the OPT PARMLIB member, the installation can disable logical swapping or adjust the aggressiveness of the logical swap decision. RMF provides data in the Paging Activity report on the effectiveness of logical swapping for TSO terminal wait conditions.



Before a logical swap can occur, an address space must be quiesced. In all cases, a logical swap will not occur if the quiesce operation shows the address space ready to use resources. That is, only waiting address spaces are candidates for logical swap. Prior to SP 1.3, only TSO address spaces in terminal wait were logical swap candidates. For these address spaces, the SRM has a think time to indicate the approximate time that the address space will be in wait. This can be used to decide if the address space should stay in storage. Swapping out "short thinkers", when storage is available, is not a good policy. MVS will just get it out and will have to bring it back in. So the objective is to keep short thinkers in storage whenever the system can afford to.

In SP 1.3, the SRM allows address spaces selected for swap based on a long wait (NIOWAIT) SYSEVENT or by the detected wait algorithm to be candidates for logical swap. These candidates are given lesser priority than the TSO terminal wait candidates. One reason for this is that the SRM cannot speculate on how long the address spaces will be waiting. These candidates will only be logically swapped, if there are not enough TSO terminal wait candidates to use storage.

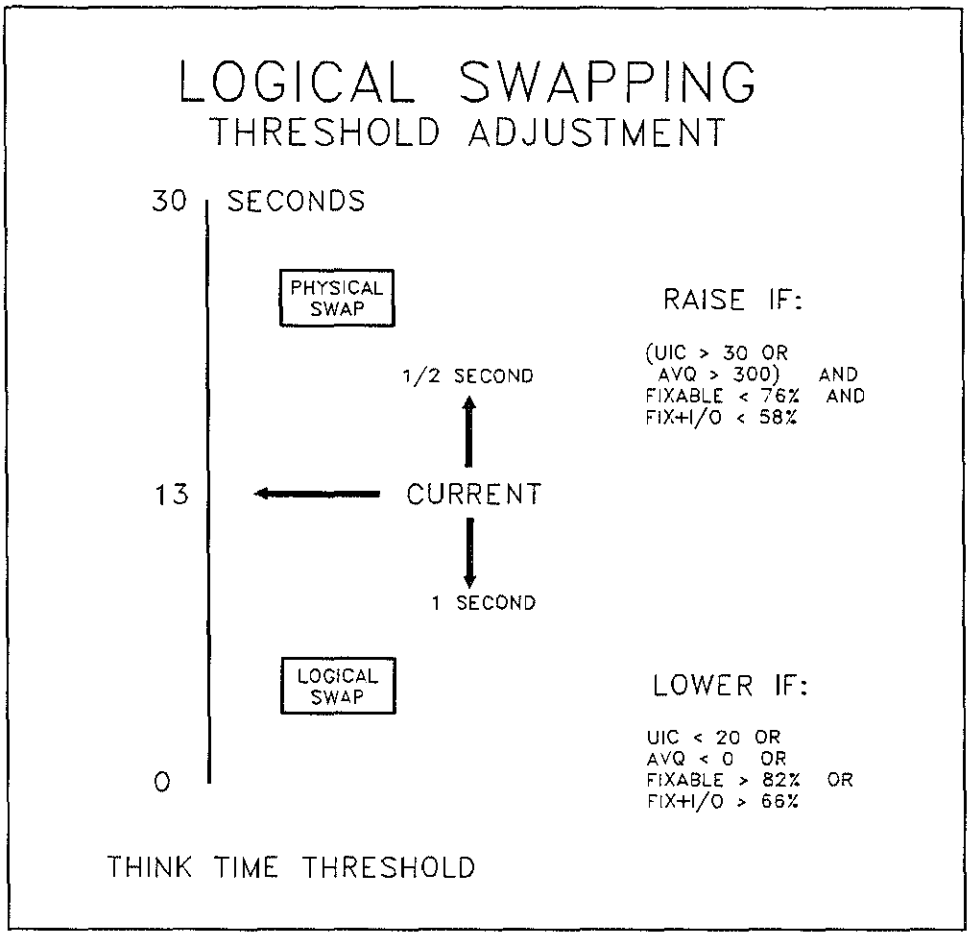
If the address space meets the criterion which will be described in more detail shortly, the dispatching priority of the address space is reduced to priority 1 and the address space is put on the SRM wait queue. Most of the pages currently in real storage frames are left in storage after a trim is done.

The decision to leave the address space on the dispatching queue has some implications for capacity planning people. The address space is placed at priority 1 below most other work. The dispatching priority controls in the IPS can insure that no real work is at priority zero or one. Even if some work is at zero or one, the logically swapped address spaces have no work to do. However, the dispatcher will encounter these address spaces some of the time as it looks for the real work to dispatch.

When the dispatcher cannot find work to dispatch, it enables for interrupts and scans the entire chain of address spaces in priority order. Enabling allows events to be signalled which usually make address spaces ready, such as I/O interrupts. The only alternative would be to put the processor in a wait state immediately, so the enabled scan is easily justified. However, this means that the dispatcher will examine all the logically swapped address spaces twice before entering the wait state. This can significantly change the amount of uncaptured system time depending on the number of logically swapped address spaces and the processor utilization. A lightly loaded processor, running TSO with logical swapping, may appear to have less reserve capacity than it really has.

The above discussion is only relevant to systems prior to MVS/SP 2.1.3. Starting with MVS/SP 2.1.3, logically swapped users will no longer be examined by the dispatcher.

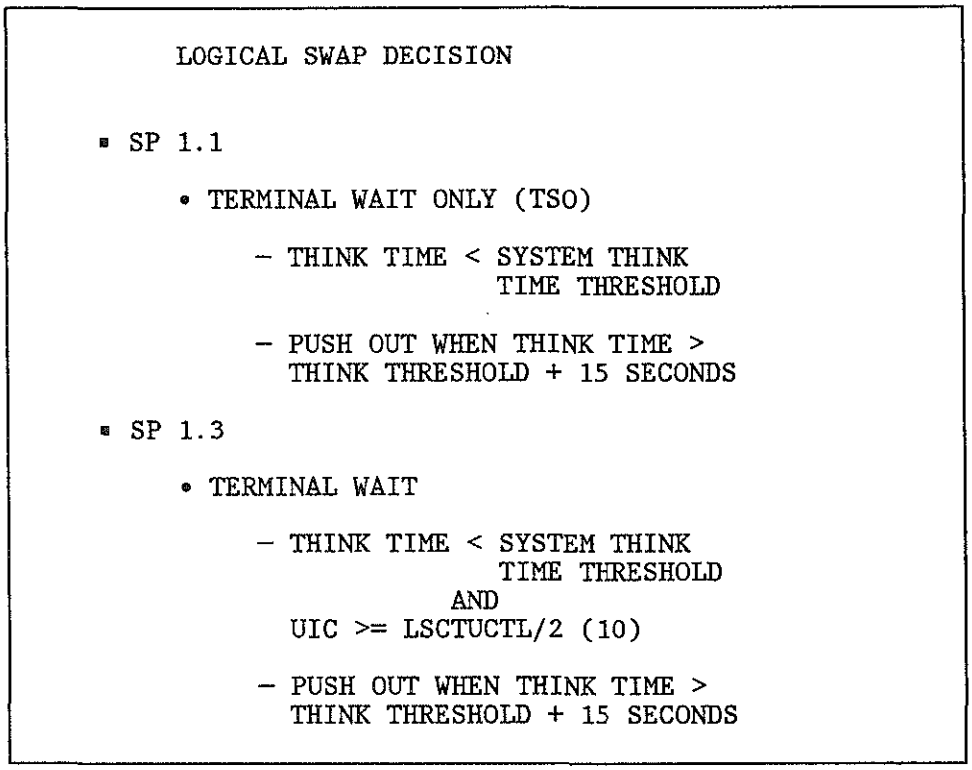
While an address space is logically swapped, it is not subject to UIC updating, because it is not referencing any pages. A routine runs periodically to physically swap out address spaces that do not become active as expected. These are called detected long think time swaps in RMF.



Logical swapping is driven by the logical swap think time threshold, which is controlled by the average system high UIC. The intent of the system think time threshold is to tell if there is storage to support logical swapping and how that storage can best be used when it is available. The threshold is in milliseconds and is bounded by limits that can be changed in the OPT. The defaults are 0 and 30 seconds. When the threshold is zero, no address space will be logically swapped. When the threshold is 500, TSO address spaces with a think time of one half second or less will be logically swapped.

The threshold increases, based on the tests in the upper right corner of the figure. The most important test is the average system high UIC greater than 30. This constant, as well as all other constants in the tests in the figure, can be changed through the OPT. The second test is for more than 300 frames on the available frame queue, on average. The next two tests for fixed storage are for very special conditions that are very unlikely to occur and therefore, will not be considered further.

When the tests are met, the threshold is increased by 500 milliseconds. When the tests in the lower right hand corner are met, the threshold is decreased by 1000 milliseconds. Based on these tests, the system think time threshold should stabilize, when the average system high UIC stabilizes in the range of 20 to 30.



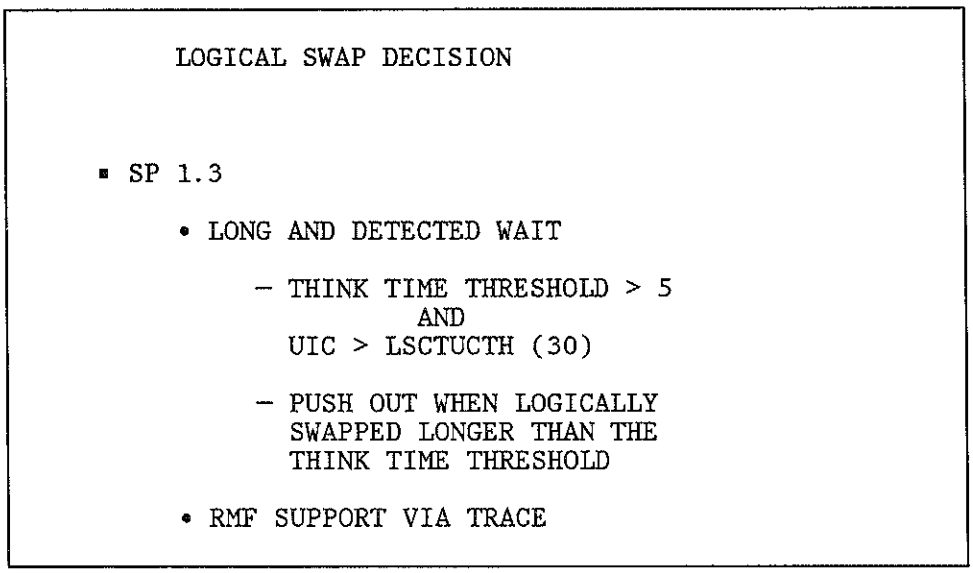
This above figure shows the logical swap decision in more detail. In SP 1.1, if the previous think time is less than the system think time threshold, the address space is logically swapped. The address space is later physically swapped, if it has been logically swapped for a time greater than the current system think time threshold, plus an internal grace period of 15 seconds.

The rationale for a grace period is that previous think times are merely indicators of future think times. If an address space is logically swapped because it had a 2 second think time and it is pushed out of storage when 2 seconds are up, the storage has just been wasted and the address space will probably become ready and have to be swapped back in a few milliseconds later. It's not clear that the value of 15 seconds is the perfect number. Once, in an experiment, the value was set to 0 on a production TSO system with no real change in the logical swap statistics. This seems to support the conclusion that previous think times are good indicators of future think times.

In SP 1.3, a second condition was added to the logical swap test. The current system high UIC has to be greater than one-half the low UIC threshold for controlling the system think time threshold. If the condition is not met, the logical swap is bypassed. The rationale for this added condition is that storage contention can change dramatically; however, the heuristic nature of the system think time threshold adjustment makes for relatively slow changes in the storage used for logical swap. If the system high UIC falls to a low level and remains at that level, logical swap will eventually decline. If the situation persists, logical swapping has ended swiftly. If the contention is short-lived, the system high UIC will increase and logical swapping will become active again.

A further condition has been added in the PTFs for APAR OZ59747. This condition is that if the think time is less than the minimum system think time as set in the OPT, the address space is always logically swapped regardless of the system high UIC.

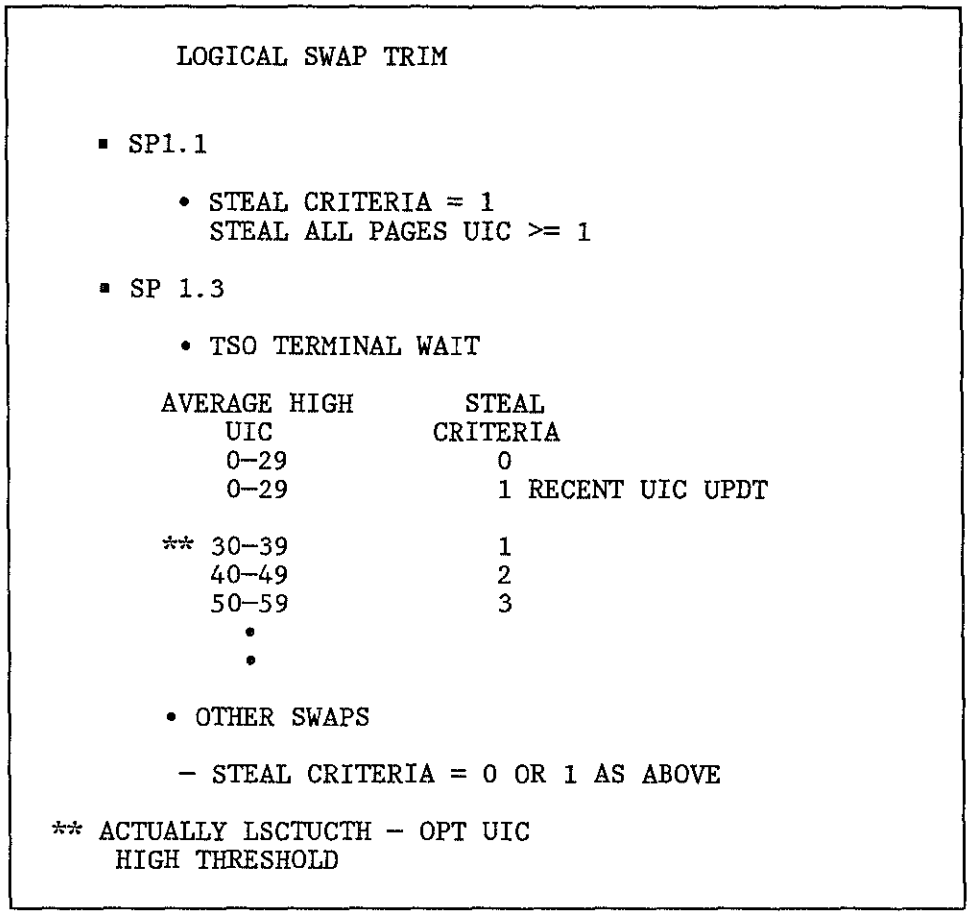
Logical Swap For a Long Wait



The conditions for logically swapping other types of wait states are slightly different. The SRM has no think time to see if the address space is a good candidate. Therefore, the test is made solely on the basis of the contention for real storage. The system think time threshold must be greater than 5 so that terminal wait swaps are being avoided. Also the system high UIC must be greater than the high UIC threshold (default is 30). This means that the system think time threshold will be increasing. Since the SRM regards these types of swaps as less important than the terminal wait swaps, they get no grace period.

RMF does not report on these types of swaps in the Paging Activity Report. RMF trace can provide the number of users logically swapped for long or detected wait and RMF Monitor II can be used to check on particular applications. It is true that long and detected waits can be symptoms of problems. These problems are invariably ENQ contention problems which can be identified in the RMF ENQ reports.

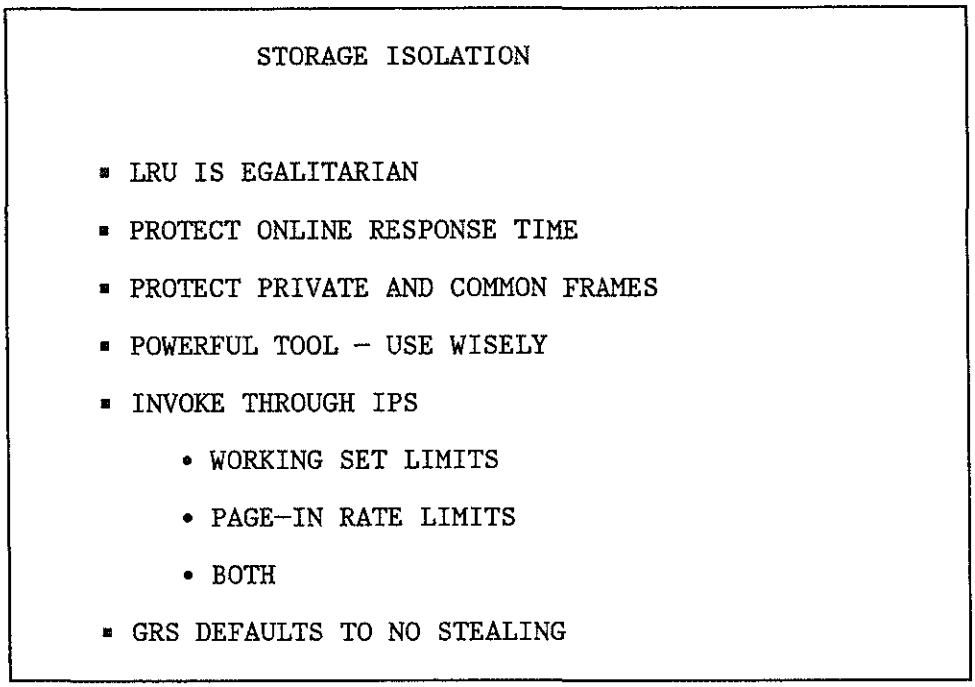
Logical Swap Trimming



This figure could also be called logical swap working set to correspond with an earlier figure. Every address space entering the logical swap state is examined to see if real storage should be reclaimed by removing pages that have not been recently referenced. In SP 1.1 all pages with a UIC of 1 or greater would be stolen from the address space. These pages are counted as page-outs if I/O is necessary. Note that this is still more generous than the working set for a physical swap. This allows pages with a UIC of 0 and the reference bit off in the working set. These pages became a part of the physical swap working set in SP 1.3.

In SP 1.3, the SRM allows larger working sets to logically swapped address spaces in terminal wait. The steal criterion is a function of the system high UIC. It increases as a step function when the high UIC is greater than 30. The rationale is that larger working sets are allowed to avoid demand page operations and to further improve response time to terminal users. This use of storage is discretionary, however, and the larger working sets will be trimmed down if storage contention increases, as shown in the frame stealing figure. Of course, minimum working sets are honored just as with physical swap working sets.

Storage Isolation

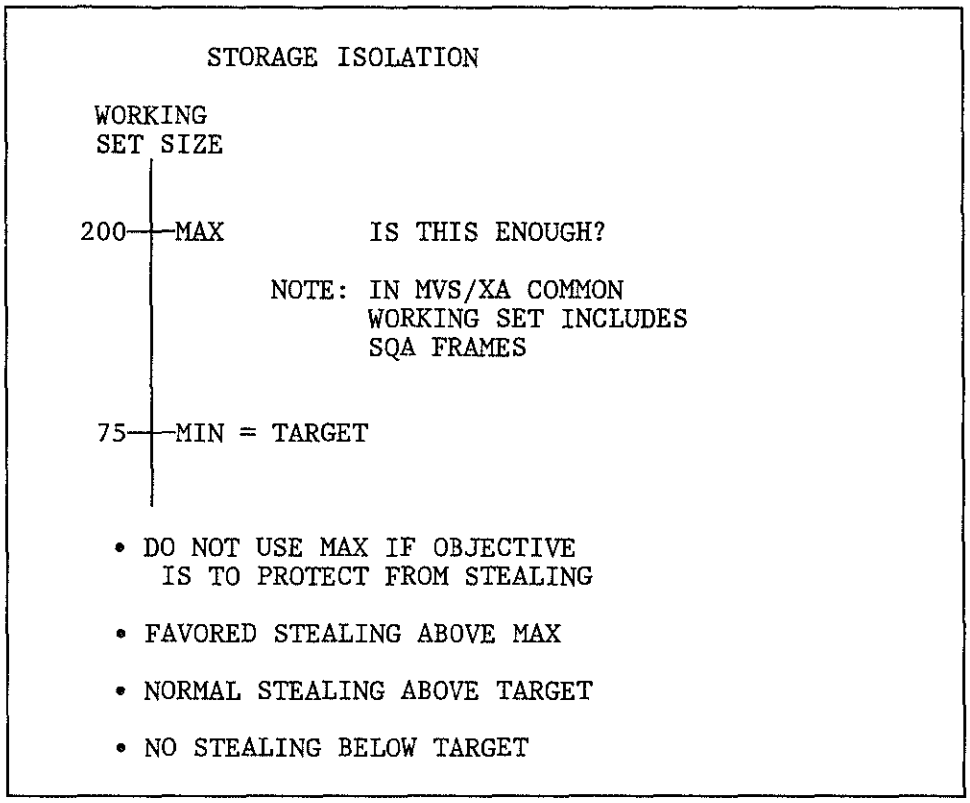


Storage isolation is one of the most powerful and effective functions offered by the SRM to meet response time goals. As stated earlier, the LRU page replacement algorithm basic to MVS assumes all address spaces should be treated equally. All address spaces are usually not treated equally in getting access to the processor or to I/O resources though. Sometimes the most important address spaces need to have virtual pages available in real storage that would be considered old by the SRM. It does not matter that the pages are old, if the page-in operations that would result from the pages being stolen would cause the application to miss its response time goals.

Storage isolation can protect the critical working sets of both address spaces and the common area. It is a powerful function, allowing a system programmer to specify low level parameters like the number of real frames that an address space should have. If it is not used wisely, it can be ineffective as shown in the frame stealing figure. The installation can specify working set sizes (in real frame units), page-in rate limits, or both through the IPS.

In SP 1.3, a special use of storage isolation was added for the GRS address space. To ensure adequate performance, the designers of GRS considered fixing much of the storage that would be referenced. This would have exempted the storage from page replacement at all times. However, there was no functional reason for fixing the storage. A page-in could be tolerated. Storage isolation was used to keep the pages unfixed but only steal them if the need for frames was critical. This control can be changed by including GRS in the IEAICSxx PARMLIB member.

Target Working Set Size Limits



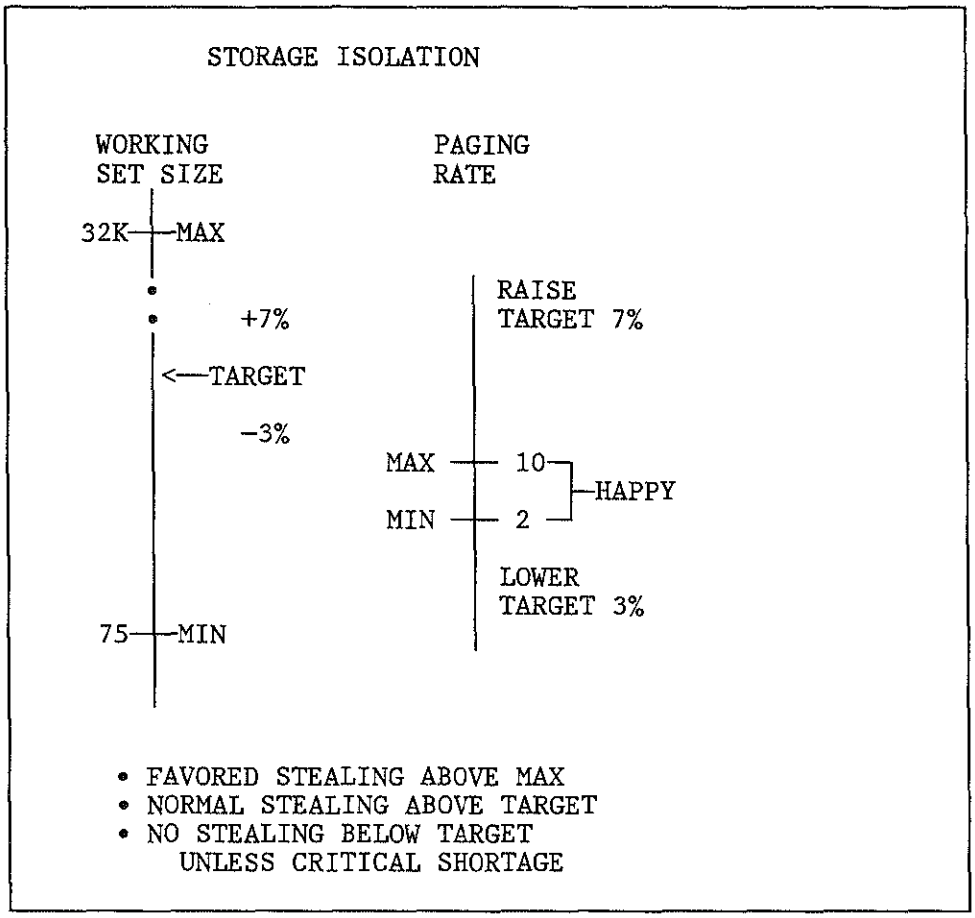
The simplest form of storage isolation is just working set control. A minimum and a maximum working set can be specified. If an address space has more real frames allocated to it than the maximum, pages will be preferentially stolen from it. Some installations have had difficulty when specifying the maximum. If the intent of using storage isolation is to protect responsiveness, let the maximum default. The default maximum is larger than the amount of real storage online. There are cases where the maximum is important such as for test applications on a production machine. The test application can be prevented from impacting the production work by limiting the amount of storage that it can hold. This is accomplished by preferentially stealing pages from an address space or the common area if more frames are currently allocated to that entity than its maximum working set size, every time page replacement is invoked.

The minimum working set is the protection. Frames will not be stolen from the address space, if the result would reduce the allocation below the minimum, except for critical situations. Actually, it is the target working set size that is used by page replacement but in this simple case the target is equal to the minimum. If the address space owns more than the target, it is a candidate for frame stealing just like any other address space in the LRU phase.

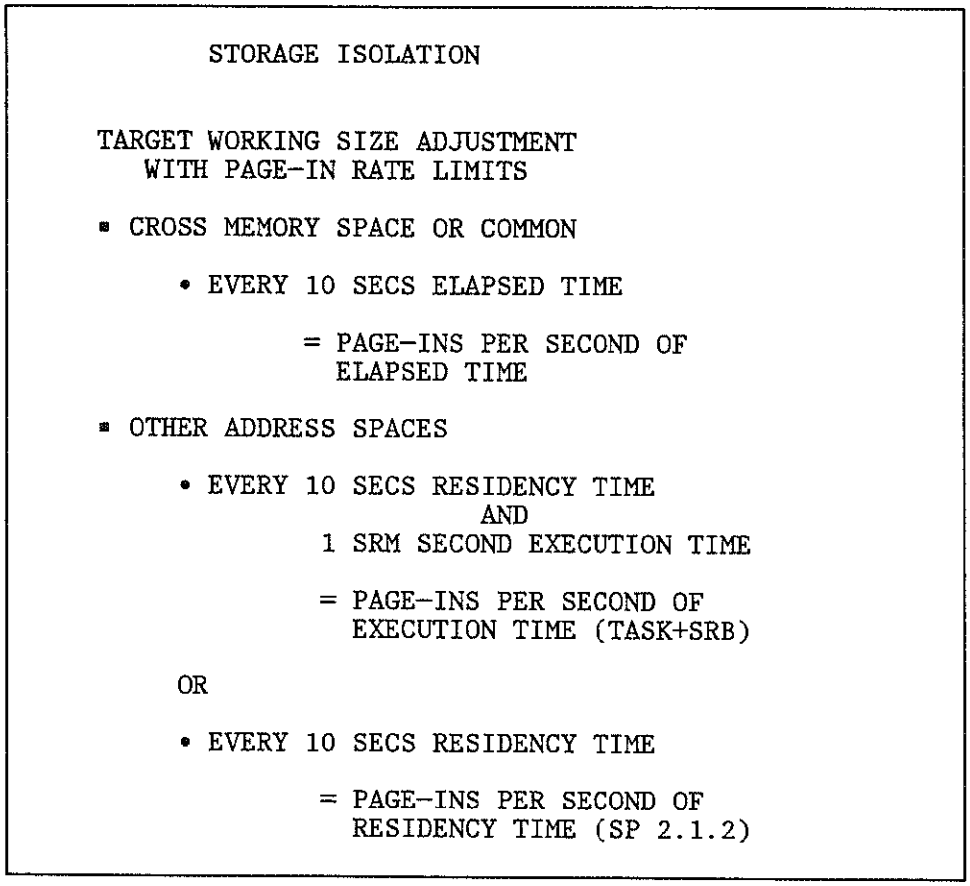
The lesson of letting the maximum working set size default was learned quite painfully by some early users of MVS/XA. If a maximum working set was set for the common area using MVS/370 data, and the same maximum was specified when MVS/XA was installed, less than desirable results were sometimes achieved. In MVS/370, the maximum specification might have been determined using the number of CSA and LPA frames from the RMF monitor I or II reports. In MVS/XA, the common area frame count available to the SRM includes the SQA frames as well as the CSA and LPA. This was not immediately well known to the SRM people and this change was not documented. This oversight will be corrected in the *Conversion Notebook* and the *Initialization and Tuning Guide*.

The only reasonable way to control the common area page-in rate is by using the common area page-in rate as the control parameter, possibly subject to a minimum working set to handle low utilization periods. Therefore, if this control was used, the MVS/XA change would not have affected the system very negatively. The installation might wonder why the target was so large compared to the common working set in MVS/370, however, it would not be contending with very high common page-in rates.

Page-in Rate Limits



Page-in rate thresholds can also be specified for an address space or the common area. Storage isolation is a function designed to control the page-in rate. This control is more direct but it is somewhat more complex due to the definition of page-in rate which is discussed later. When page-in rate controls are in use, the target working set size is adjusted by the SRM to try to keep the page-in rate within the minimum and maximum specified. The target is bounded by a minimum and a maximum. If working set sizes are specified, they are the minimum and maximum. If no working set sizes are specified, 0 and 32K are used as the minimum and maximum. If the page-in rate for the address space or the common area exceeds the maximum page-in rate, the target is increased by 7% to protect more storage and reduce page stealing. If the page-in rate is less than the minimum rate, too much is being protected and the target is decreased by 3%.



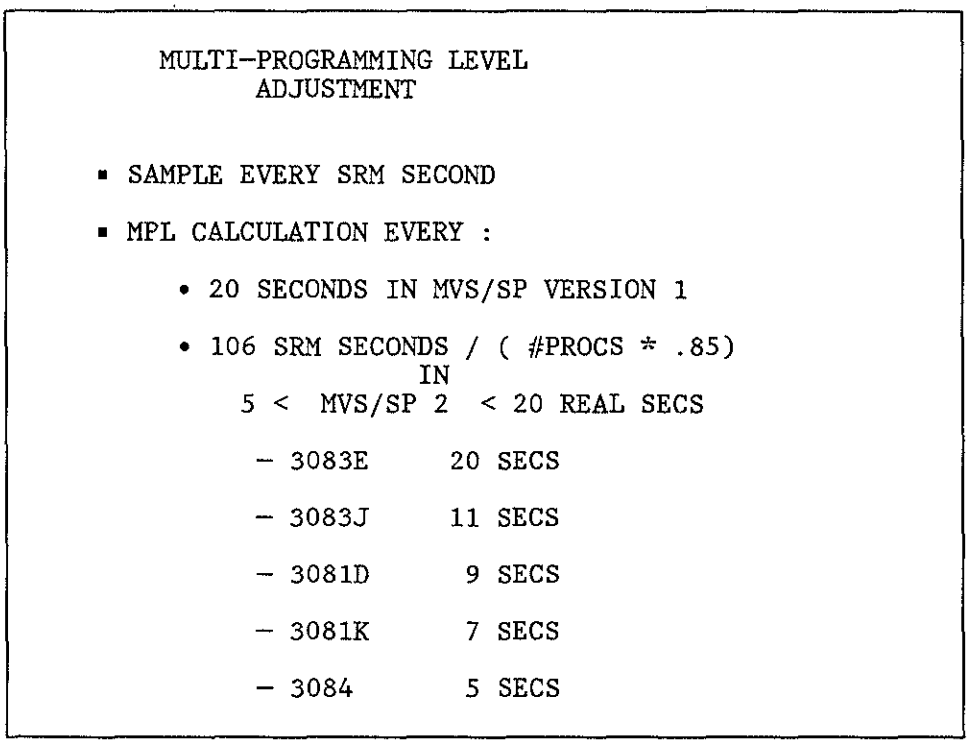
The algorithm just defined is a heuristic algorithm. It makes a change based on the result of a previous change until the desired result is achieved, namely a page-in rate in the happy range. To do this effectively, it must know that the previous change has been tested. If a person turns up the thermostat because it is freezing and 5 seconds later checks the thermometer, the person would probably conclude that it is still freezing. That does not mean that turning up the thermostat will not be effective in making the room more comfortable.

If the SRM increases the target working set for an address space because its page-in rate was high and then recalculates a page-in rate before the address space has executed for some reasonable amount of time, it would not get good feedback on the effectiveness of the previous change. For this reason, the SRM has required that most address spaces execute for 1 SRM second between page-in rate computations. It is also required that the address space be resident in real storage for 10 real seconds.

Prior to SP 2.1.2, the page-in rate calculated is the rate of page-in operations per second of execution time for all address spaces except cross memory address spaces such as GRS and ALLOCAS. Nonswappable address spaces like CICS may be idle for periods of time. The page-in rate definition should be more representative of page operations per transaction processed than the more common definition of page-in operations per real second. Cross memory address spaces usually have very little execution time associated with them but may have many storage references. This is certainly true of the common area. The page-in rate for these elements is defined in terms of page-ins per real second.

In SP 2.1.2, the capability to calculate the page-in rate based on residency time rather than on execution time was added as an option. This will mean that while an address space is swapped in, its page-in rate will be calculated and the target working set will be subject to change even if it has not executed. This may allow the working set to erode during inactive periods. This will allow the storage to be used by other applications at a cost of increased page faulting to get the original application going again. Of course, a realistic minimum working set can limit the erosion, but this requires additional analysis and maintenance. Nevertheless, this may be desirable for nonswappable applications with long inactive periods. For swappable applications, similar support is provided since the minimum working set size is used to form the swap-in working set, not the target working set size. Therefore, the working set may erode when the address space is swapped for detected wait.

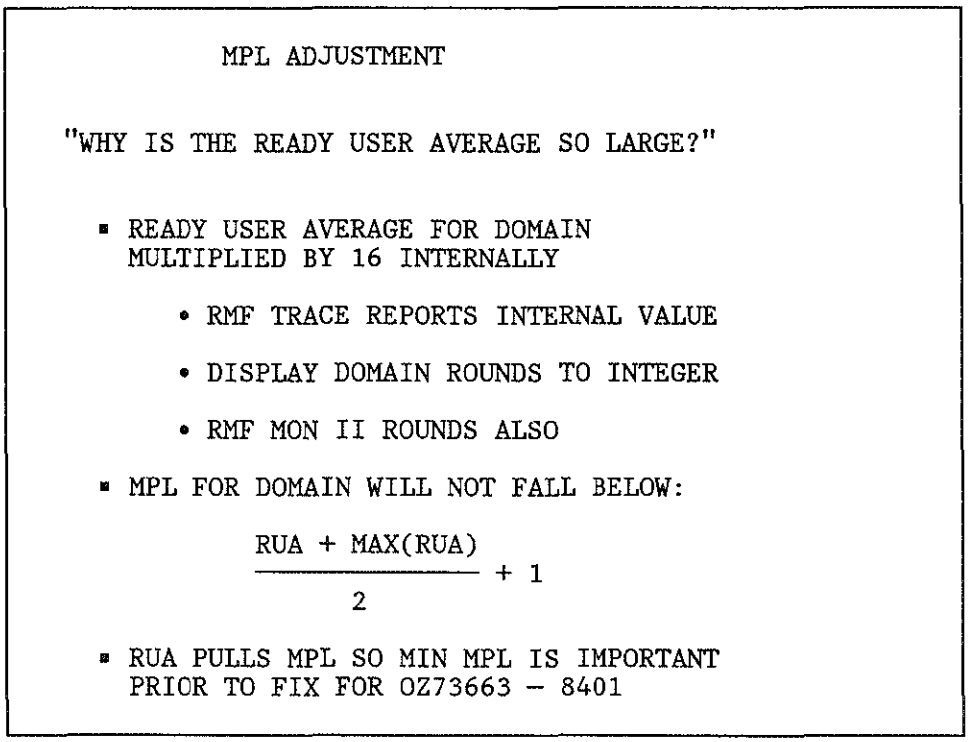
Chapter 5. Multi-Programming Level (MPL) Adjustment



The multiprogramming level adjustment routine is one of the SRM's more well-known algorithms. It is included here for completeness. The algorithm controls the number of swappable address spaces allowed in real storage and eligible to be dispatched. The algorithm is sensitive to the degree of contention for system resources such as the processor(s) and real storage. Various indicators are sampled every SRM second.

At regular intervals, the samples are summarized and a decision is made to change the target MPL by 1 or leave it as it is. That interval is every 20 seconds in SP Version 1. In SP Version 2, the interval is a function of the total processing capability of the complex. The interval will not be greater than 20 seconds or less than 5 seconds. When managing large processors, such as the 3084, a significant amount of resource is wasted if the MPL only increases by 1 every 20 seconds.

Ready User Average (RUA)

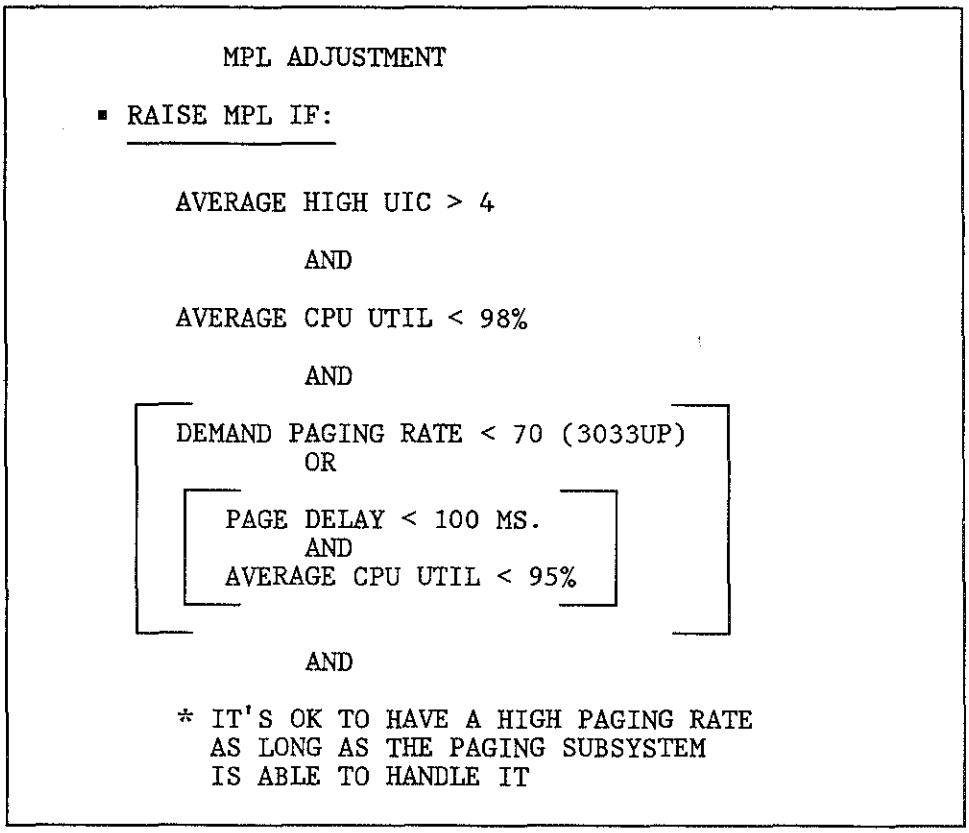


The ready user average (RUA) for a domain is internally scaled or multiplied by 16. This allows the SRM to essentially keep a fraction and then round the average up. This would not be of any great interest except that the RMF TRACE report simply shows the contents of a field. There is no data formatting done. Therefore a value of 88 means that there were 5.5 ready users on average. The display domain operator facility and the RMF Monitor II Domain report round the internal value.

In addition to the average number of ready users, the SRM keeps the maximum number from the interval. The MPL will not decline below the value shown, if the domain has a high enough contention index. This support is especially important for TSO domains that have less stable ready user averages than a batch domain. It is desirable to have the MPL accommodate the address spaces when they come ready, if the real resources are available.

Unfortunately, only the RUA alone can increase the target MPL, until the fix for APAR OZ73663 is applied, which is available on tape 8401. Therefore, it is still important to have reasonable minimum MPL's for TSO domains. The SRM should be better able to select the MPL for TSO domains after the PTF is applied. The SRM will use the average of the RUA rounded up and the maximum number of ready users as shown above. All the thresholds that the SRM uses to determine if resources require a higher or lower MPL, can be changed in the OPT.

Raising the MPL



This is the first of two figures showing the logic for increasing the MPL. The tests in this figure are the most important tests provided by the SRM. The tests in the next figure are unlikely to play any major role in the decision due to the threshold values chosen. To increase the MPL, the average high UIC must be greater than four. This is a low value when compared to the UIC values of 20 and 30 used for logical swap. A system high UIC of four is not a good value in a response oriented system; however, it may not be bad in a severely storage constrained system running batch. The SRM has to handle all environments.

Demand paging rate would, in most cases, control the MPL before UIC would. Demand paging rate thresholds are sensitive to the number of processors and their model and version code. If demand paging is less than the threshold, the MPL can increase. If it exceeds the threshold, but processor utilization is not too high, and page delay time is not too high, the MPL can increase. The rationale is that demand paging uses processor resources, but if the processor is not near 100%, there is no loss. Demand paging impacts applications though, and if the time to service a page operation is high, the MPL should not increase.

As stated earlier, the SRM has to handle all situations, but in the matter of MPL adjustment it can use some help from the installation. Throughput must often be traded off against responsiveness, when resources are in contention. The algorithm tends toward throughput and if the installation's goal is to be responsive, it may allow too much demand paging when there is storage contention. The installation can counter this with storage isolation or by changing some of these thresholds.

Raising the MPL (Part 2)

MPL ADJUSTMENT

- RAISE MPL IF:

AVERAGE FIXABLE STOR FIXED < 82%

AND

AVERAGE STOR FIXED OR PAGE I/O < 66%

AND

NO RECENT PAGEABLE STOR SHORTAGE

AND

AVERAGE ASM QUEUE LENGTH < 1000

AND

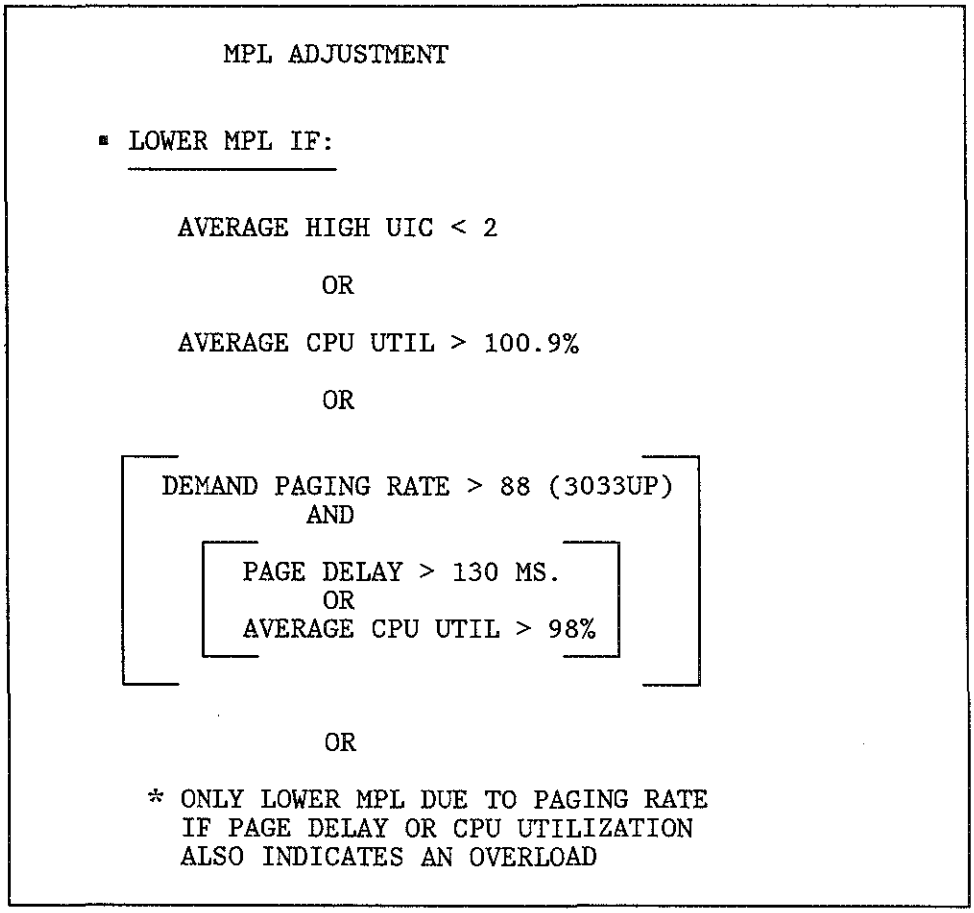
PAGE FAULT RATE < 1000/SEC

AND

AVERAGE PAGE SERVICE TIME < 1000 MS

The easiest threshold to change is the page fault rate threshold. This control has been advocated in a couple of Washington System Center Publications, most notably *An MVS Tuning Perspective*. Based on the paging configuration and the installation's responsiveness goals, page fault parameters can be set that will control the MPL.

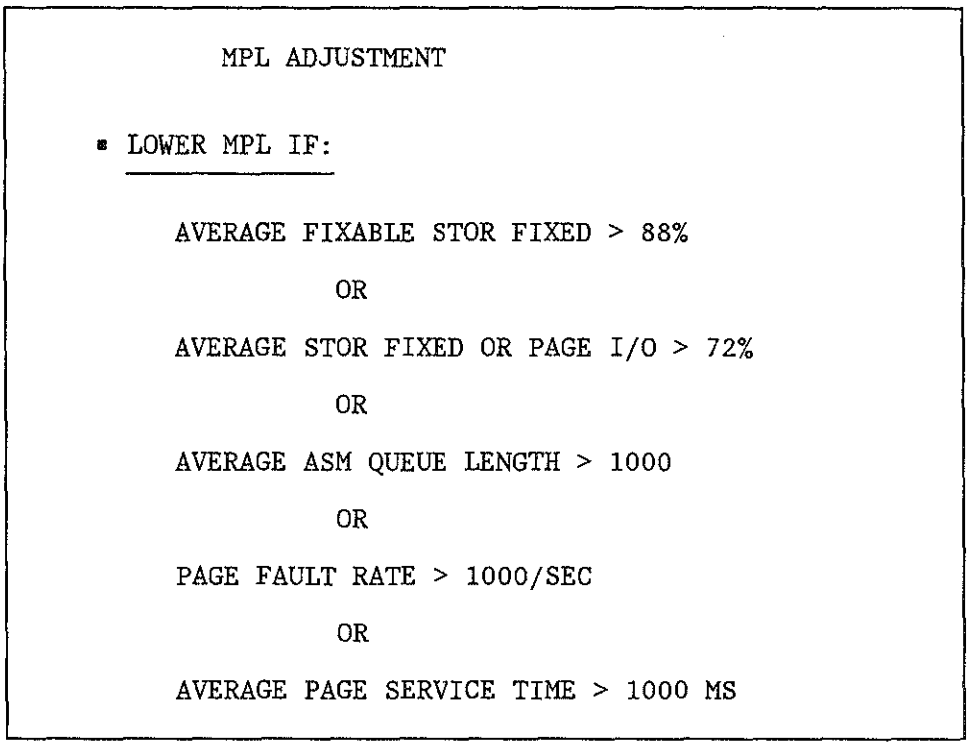
Lowering the MPL



Above is the logic that decides if the MPL should be lowered. "Or" logic is used rather than the "and" logic of the previous figures. If any one of the conditions is met, some resource is overutilized and the MPL should be decreased.

It may not be obvious how the CPU utilization can be greater than 100%. CPU utilization can reach 101% if the processor does not enter the wait state during some interval and some ready address space does not get dispatched during that interval (3 SRM seconds). Address spaces are sitting in storage but are not being dispatched. Should this persists for a long period, it could be bad if the address space owns an ENQ resource. The SRM has support to see that such address spaces stay in real storage, and they may get a dispatching priority boost if they are in a mean time to wait group. The SRM assumes that they will run if they are swapped in. This logic attempts to ensure that.

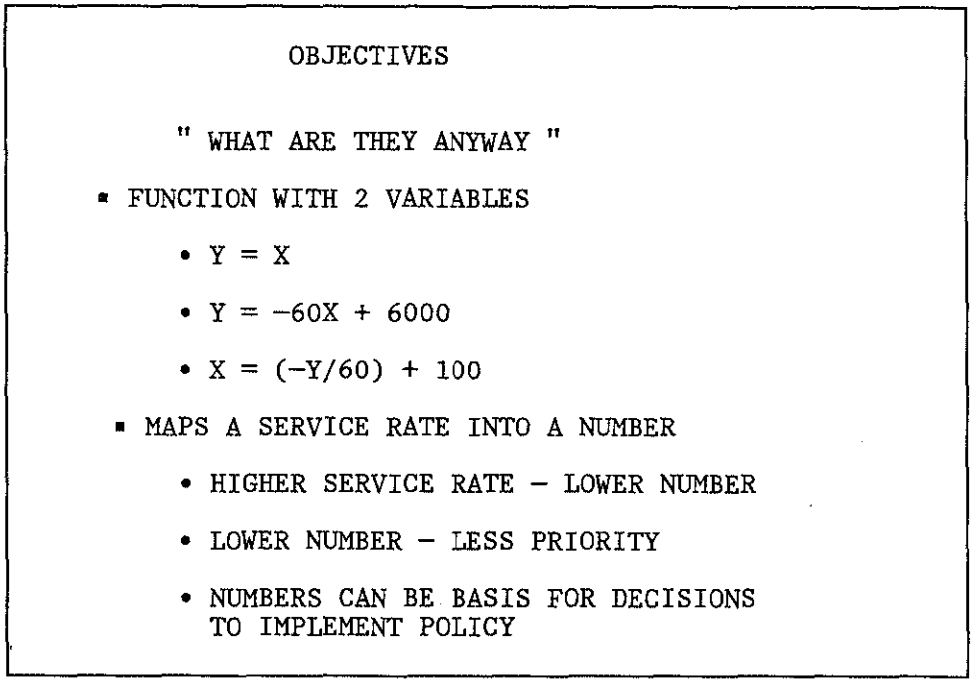
Lowering the MPL (Part 2)



These tests are mirror images of the tests from the MPL increase figure. Again, the conditions being tested are very unlikely events due to the choice of the thresholds. For instance, the test at the top of the page is for greater than 88% of the storage below 16 megabytes fixed. This condition could occur, but is not too likely in today's typical environments. The latter 3 conditions should not occur. The thresholds were chosen specifically to cause the test to be ineffective because a value could not be chosen that would be applicable to all environments.

Chapter 6. SWAP Control

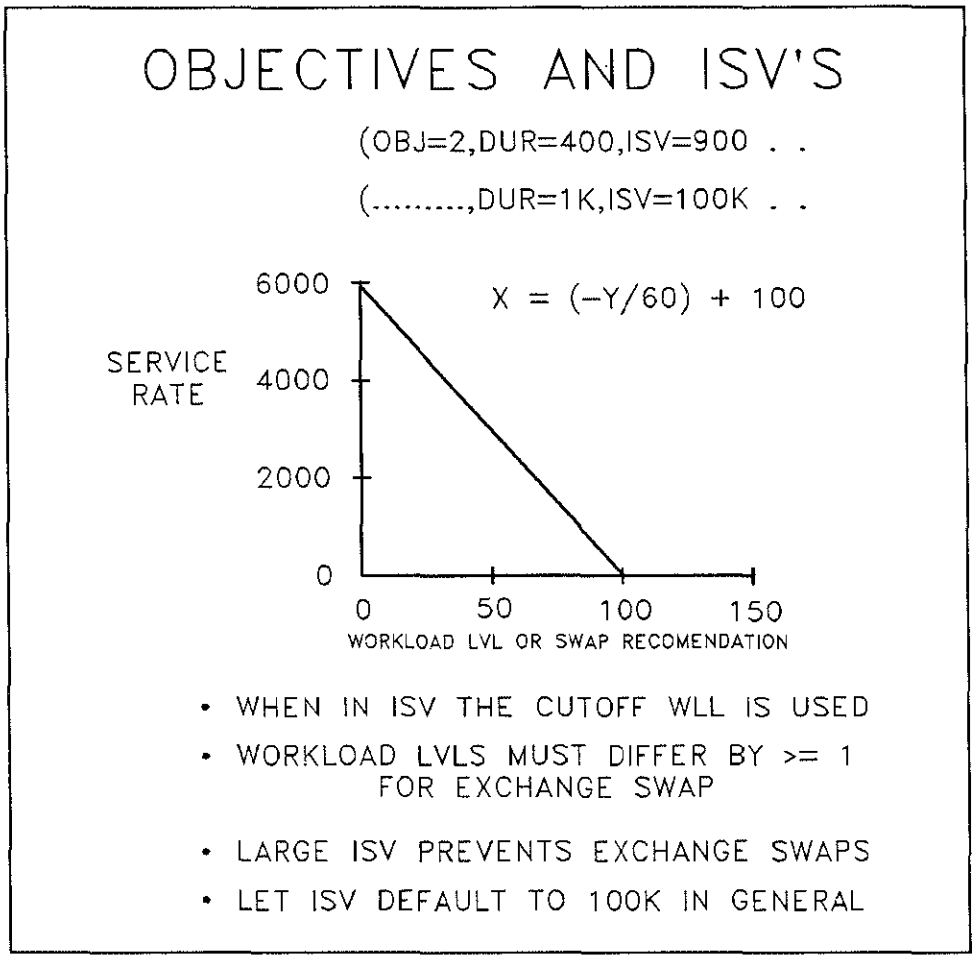
Objectives



No SRM presentation would be complete without a discussion of objectives. What is an objective anyway? An objective defines a mapping function. It is a function with two variables, one dependent and one independent. Shown in the figure are a few functions which are not necessarily objectives. The first, Y = X, is an extremely simple function. The second function is more complex. The third is the same as the second but with X expressed in terms of Y. People tend to always think that X must be the independent variable. For purposes of this discussion, one must be flexible enough to let Y be the independent variable.

The objective maps a service rate into a number. With the rules defined for specifying the objective functions, the higher the service rate, the lower the number from the function. The numbers are used as priorities. The lower the number, the lower the priority. The numbers can be used to implement a policy such as which address space should be swapped in or which domain should get an MPL increase or decrease.

Objectives and ISV's

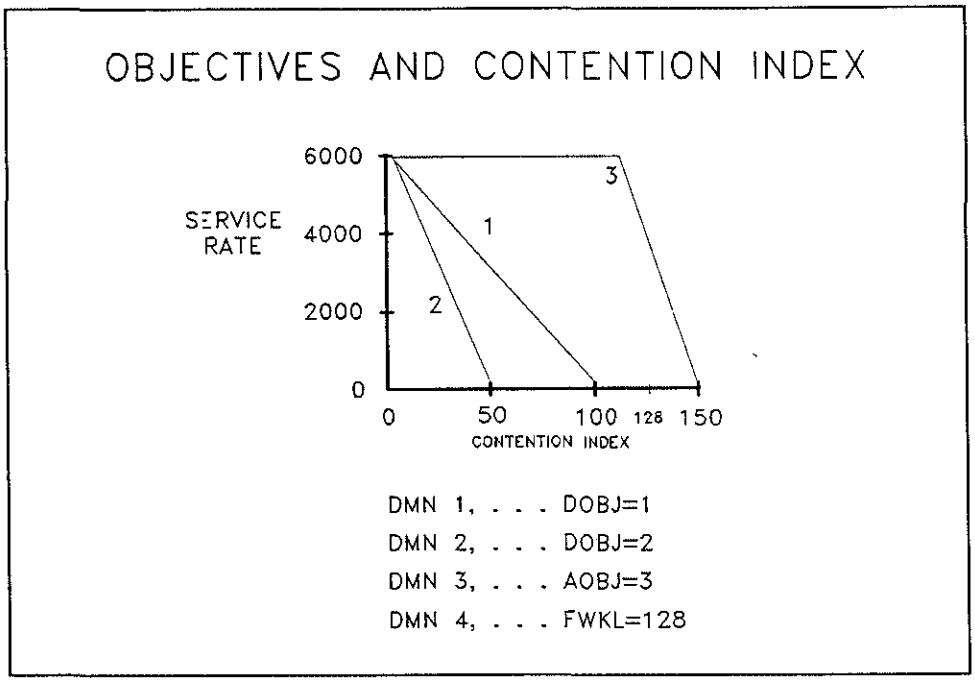


It is now time to explain how objectives are used to get swap recommendation values and how the theory is compromised for the better by something called the interval service value or ISV. Shown above is the graph for the last function formula from the previous figure. Swap recommendation values for two address spaces would be calculated from the objective given the service rates of the two address spaces. High recommendation values indicate that the address space should be in storage. The intent is to equally distribute the resources among equally important address spaces. But swapping address spaces to accomplish equal service rates can be very expensive if the service interval is too short.

The ISV takes care of this. The ISV is a number of service units that must be accumulated since the last swap-in before the real service rate is used in the objective formula. Until the interval service is accumulated, 0 is used as the service rate. The resulting swap recommendation value is 100 in this example. This is referred to as the "cutoff workload level". Therefore, if two address spaces share a common objective and the one swapped in is in its ISV, no exchange swap will be done. They could even be associated with different objectives and no exchange swap would occur if the cutoff workload levels were different by 1.

Large ISV's prevent unwanted exchange swaps. The best way to ensure a large ISV is not to specify one. It will default to 100K service units. Some of IBM's publications have implied that one should carefully choose the ISV based on the duration of preceding performance group periods to control exchange swapping. This is not necessary. An ISV larger than needed does not carry over into a new period. It is easier and better to let it default.

Objectives and Contention Index



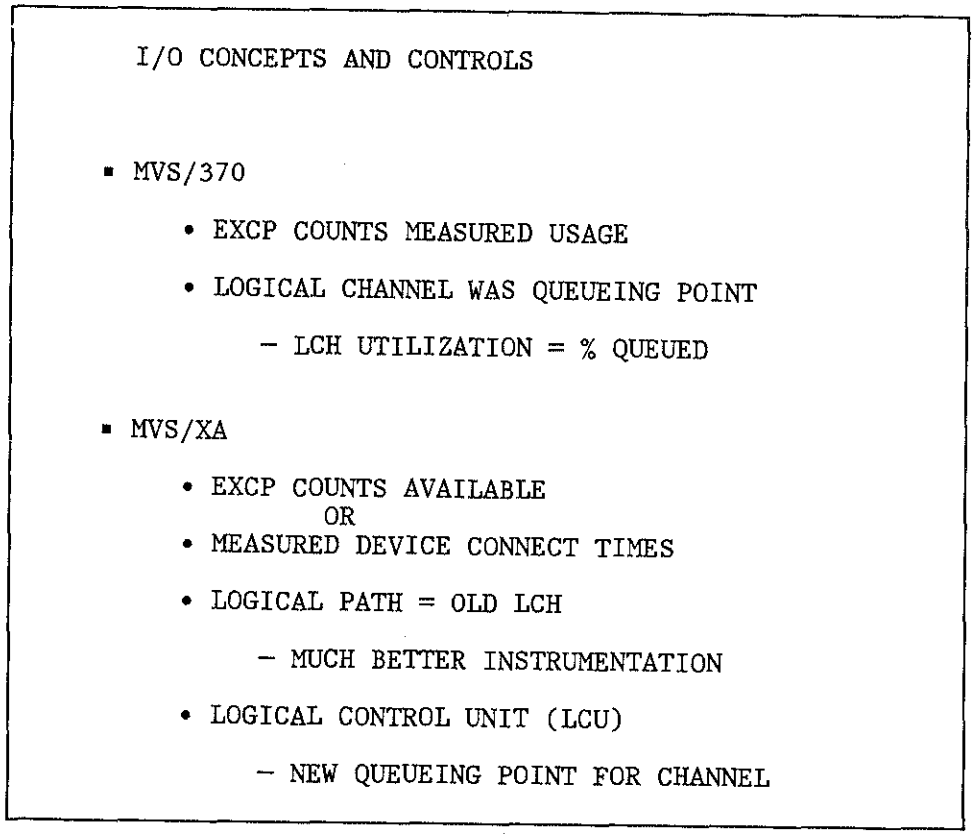
The second use for objectives is to establish a contention index for a domain based on the service provided to the domain. Domain weights can be used to develop contention indices also. Many installations have found objectives to be a better way to state their policy.

For a domain with a DOBJ specified, the service rates of all address spaces associated with the domain are added to get the domain service rate. This is the input to the objective function. The output is the contention index for the domain. In the above example, domain 1 is clearly favored over domain 2. For any given service rate, domain 1 will have a higher contention index and will be receive an MPL increase. The SRM attempts to equalize the contention index of all domains. This means that the policy statement made here is that domain 1 should get twice the service rate of domain 2, since objective 1 has one half the slope of objective 2.

For a domain with an AOBJ, the average service rate of all address spaces associated with the domain is the input to the objective. Usually AOBJ's are functions like the example here. The policy statement is that each address space should get 6000 service units per second in that domain. If not, it will be the domain with the highest contention index. If each address space gets more than 6000 service units per second, it will have a very low contention index.

The final piece is a control independent of service rates. The fixed workload level or FWKL specifies a fixed contention index. The name was a poor choice, but FCI for fixed contention index is probably not much better. This is usually used to say some domain is most important or least important.

Chapter 7. I/O Concepts and Controls



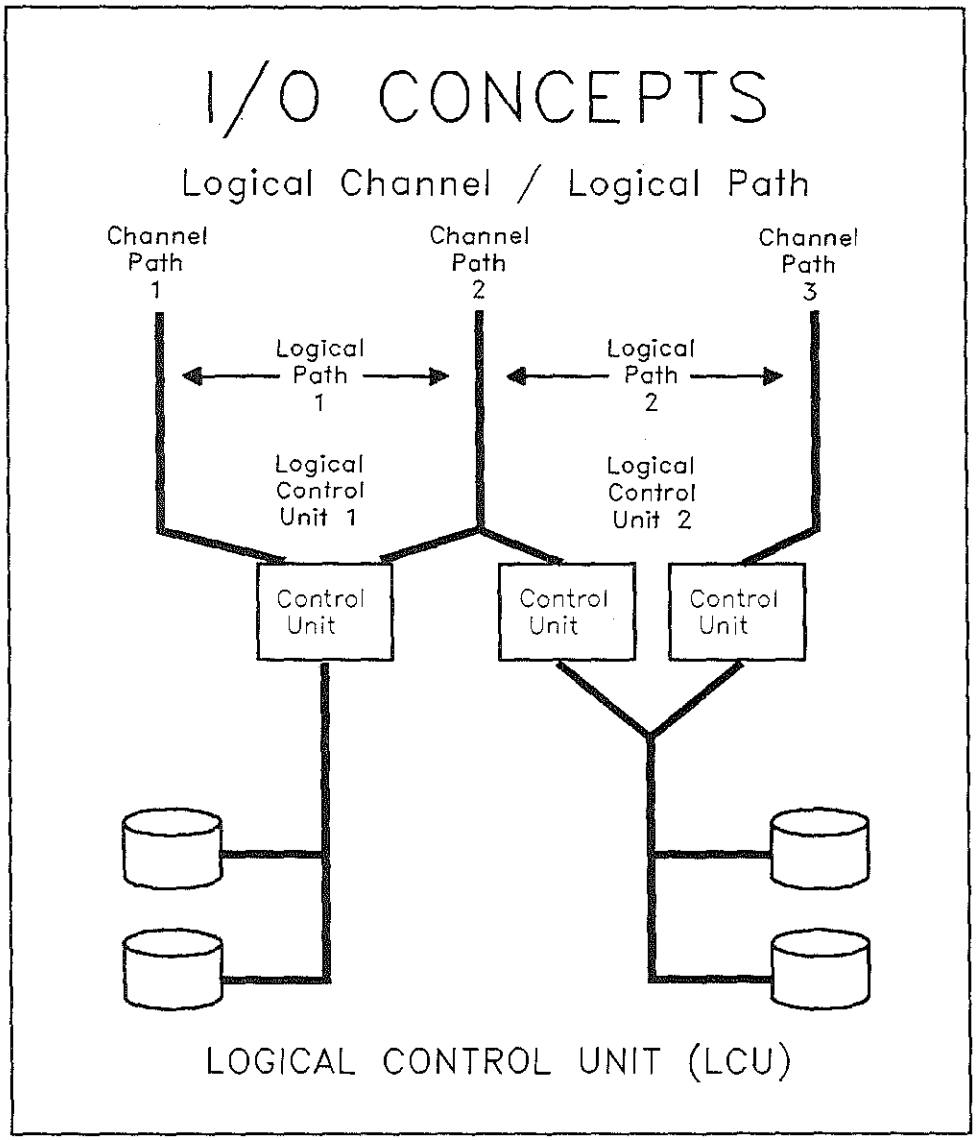
The next topic is I/O concepts and controls and is one area that was significantly changed in MVS/XA. The SRM's I/O constructs are different in MVS/XA than in 370. MVS/370 counted the number of I/O blocks transferred by an address space. This is often called the EXCP count but the title is incorrect. If chained scheduling is used, the number of invocations of EXCP will not be equal to the so called EXCP count. Block count is a better metric since the count is used as an indicator of load placed on the I/O subsystem. The SRM used the block count to derive the I/O service used by the address space. The block count was also used by I/O load balancing to determine the demand that the address space placed on the logical channel(s).

The logical channel is easier to draw than to describe. It is the set of all physical channels serving a device or group of devices. A physical channel may belong to more than one logical channel, as will be seen in the next figure. In MVS/370, requests that could not be started immediately were queued on the logical channel serving the device. IOS kept counts of the number of requests that were queued due to channel busy, device busy, control unit busy and logical busy. The SRM used the percent of requests queued as an indicator of contention for the I/O subsystem. This was used to make decisions in nonspecific device allocation requests and for I/O load balancing, if that option was requested.

In MVS/XA, the SRM has significantly better I/O instrumentation to make decisions. The I/O block counts are still available and can still be used to determine I/O service. The channel subsystem now measures the time that an I/O request was connected to a channel path transferring commands or data. This information is stored with each I/O interrupt and accumulated in the ASCB and TCT-TIOT. The SRM can optionally use the device connect time to determine I/O service. Connect time is more representative of the I/O resource required by an application than the number of I/O requests.

The logical channel was an IOS construct and is not part of MVS/XA. The channel now queues all I/O requests that cannot be started right away. Requests for a device that already have an active request are queued to the UCB and started when the active request completes. The logical path is an SRM construct which is basically the same as the old logical channel. The logical path is the point of contention to the SRM. It is used by device allocation and I/O load balancing in a way similar to the way the logical channel was used in MVS/370. The details will follow.

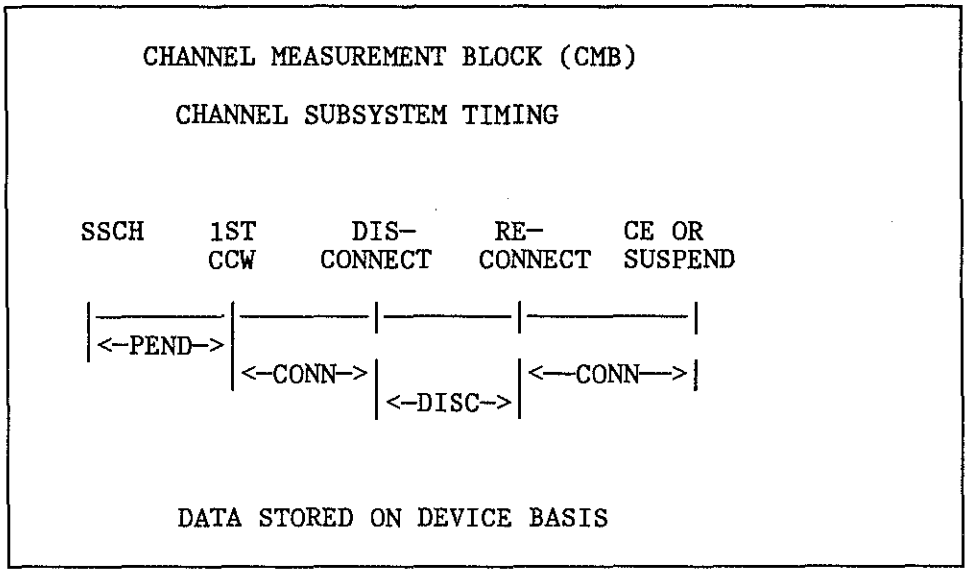
The logical control unit (LCU) is a construct of the channel subsystem of the 308X and 3090 processor family. The logical control unit is the logical representation of a control unit or group of control units with one or more devices in common. Any I/O request for any device attached to any control unit in the group is queued at the logical control unit if it cannot be started immediately. The description of the logical control unit is here for completeness. The SRM does not acquire the LCU data that is kept by the channel. However, for almost all configurations, the LCU and logical path constructs subset the devices in exactly the same way.



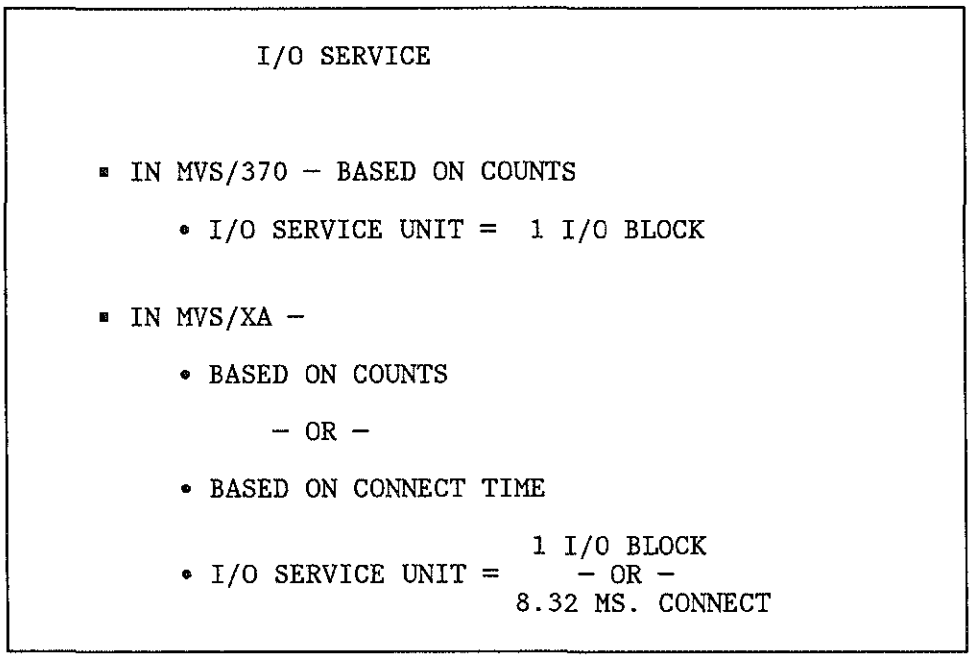
This is an attempt to illustrate the different constructs just described. The devices connected to the first control unit on the left can be accessed by channel paths 1 and 2. Therefore, channel paths 1 and 2 constitute a logical path or a logical channel in MVS/370. The SRM is focusing on the channel paths as its measure of I/O contention by using the logical path. This allows the SRM to view the multiple paths to a device as one logical path and the probability that the logical path will be available is used to make decisions such as which device to allocate to a new request. This is analogous to the MVS/370 use of the logical channel.

The devices served by the first control unit are served by no other control unit. Therefore the physical control unit constitutes a logical control unit as well. The devices connected to the two control units on the right are connected by the control units to channel paths 2 and 3. Therefore channel paths 2 and 3 make another logical path or logical channel (MVS/370). The two control units constitute a logical control unit since they serve common devices. As shown, logical paths and logical control units are really defined by the devices and the paths from the devices to the processor. In both examples, the logical path construct subsets the devices into the same sets as the logical control unit construct. This would be true for almost all reasonable configurations.

Channel Measurement Block



This figure shows all the instrumentation that the channel subsystem provides for each I/O request. The timings are stored at I/O completion into a channel measurement block (CMB) which is obtained by the SRM at system initialization. The SRM obtains storage for CMB entries for all tape and DASD devices. Additional entries can be obtained for use by performance monitors such as RMF by using the CMB system initialization parameter described in the *Initialization and Tuning Guide*. The channel subsystem measures the pending time, the connect time and the disconnect time of each I/O request. As indicated, pending time measures the time from the start subchannel until the request is actually initiated to the device. The connect time has been defined previously. The disconnect time is all the rest and is due to seek time, latency and RPS misses for DASD. The SRM uses the connect time and the pending time.



As in MVS/370, I/O service can be based on block counts. MVS/XA provides an additional measure of I/O service, total device connect time for the user. A new keyword in IPS PARMLIB member specifies whether I/O service is to be computed using block counts or device connect time. In the unlikely event of a malfunction of the channel measurement facility, device connect time cannot be measured. Instead, block counts are used to compute I/O service.

When time is selected to determine I/O service, an I/O service unit is defined as 8.32 milliseconds of connect time. The number 8.32 ms is one half the revolution time of most current DASD devices. This would mean that datasets with half track blocking should result in the same service units with either definition of service. Of course, the installation can use the I/O coefficient of the IPS to adjust the service definition if they desire comparable service values but have a different average blocksize. The I/O service calculated from device connect time is a more accurate measure of the user's contribution to the device and I/O path loading, than is the count of user I/O operations to the device. MVS/XA can now distinguish between cheap I/Os and expensive I/Os.

Channel Path Sampling

CHANNEL PATH SAMPLING IN MVS/XA

STORE CHANNEL PATH STATUS (STCPS)

- SAMPLES ALL CHANNEL PATHS AT ONCE
- DEVELOP PHYSICAL PATH UTILIZATION
- DEVELOP LOGICAL PATH UTILIZATION

	LOGICAL PATH	
	CHANNEL PATH	CHANNEL PATH
BUSY	3/15	5/15
PROBABILITY THIS CHANNEL PATH BUSY	.20	.33
PROBABILITY BOTH CHANNEL PATHS BUSY	.20 X .33 = .066	

The Store Channel Path Status (STCPS) instruction is a new instruction of the 370-XA architecture. It is the equivalent of 256 Test Channel instructions. It returns a 256 bit string. If the Nth channel path is active when the instruction is processed, the corresponding Nth bit is turned on. Thus, the status of all channel paths is obtained at once.

SRM issues the STCPS instruction about five times per second to sample the channel paths. Every three seconds, it calculates the ratio of the times it found the path busy to the samples taken for each path. This is the probability of finding that channel path busy.

The probability of finding the individual channel paths busy is used to calculate the logical path utilization. For DASD with the dynamic path reconnect (DPR) feature, a channel program started on one path can be completed on any other channel path in the logical path. The probability of finding all the paths busy is the probability of a delay in starting a channel program or more importantly, of a missed re-connect after a set sector.

Suppose, for example, there were 2 channel paths in a logical path. A sampling interval found the first channel path was busy 3 times in 15 samples, and the second channel path was busy 5 times in 15 samples. The probability of finding the first channel path busy is 3/15 or 20%. The probability of finding the second channel path busy is 5/15 or 33%. The mathematical probability of finding both paths busy is:

```
3/15 x 5/15    or    .20 X .33 = .0660
```

Thus the logical path utilization is .066 or 6.6% busy. The logical path utilization is the probability that a request will be delayed due to channel path busy.

Of course, fifteen samples would not provide a very reliable indication of the utilization of a channel path. The SRM keeps historical information to determine a more reliable measure of utilization. The definition of utilization does not change, however. It is the sampled probability that the channel path will be busy.

Logical Path Utilization

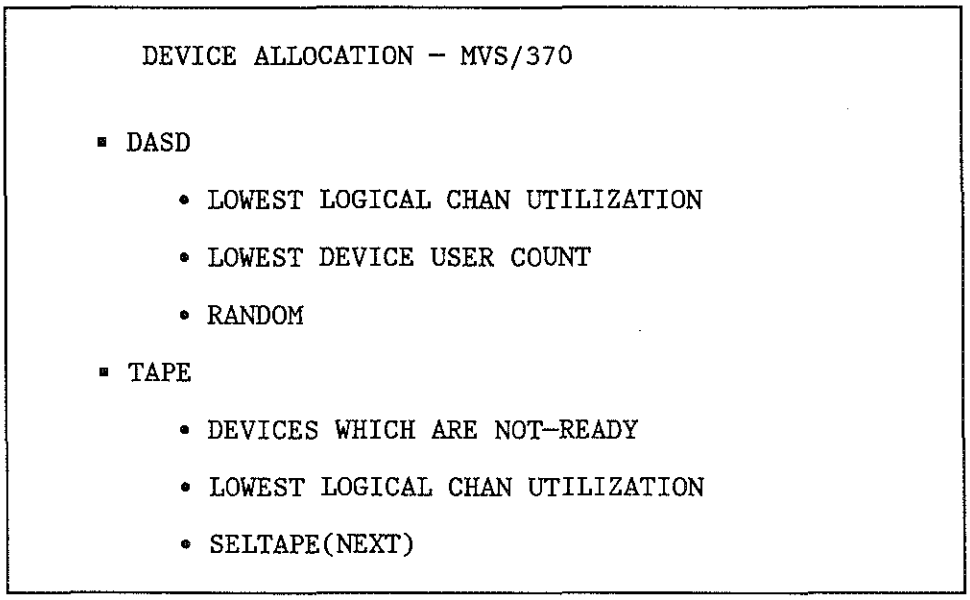
- LOG. PATH UTIL. – TAPE AND DPR DASD
  - = PROBABILITY ALL PATHS BUSY
  - = 6.6% IN THE EXAMPLE
- LOG. PATH UTIL. – NON DPR DASD
  - – AVERAGE OF
    - (1) PROB. ALL PATHS BUSY AT START
    - (2) AVG. PROB. PATH BUSY TO RECON.

$$= \frac{(.20 \times .33) + \frac{(.20 + .33)}{2}}{2}$$

$$= .1625 = 16.25\%$$

Without the DPR feature, the channel program must reconnect on the channel path where it started. In this case, the probability of finding all paths busy to start is averaged with the average probability of finding that same channel path busy to reconnect. In the example, (.20 x .33) is the probability of finding a busy path to start. (.20 + .33)/2 is the average probability of finding that same path to finish. The average of these two factors, .1625 or 16.25%, is the logical path utilization.

Device Allocation in MVS/370

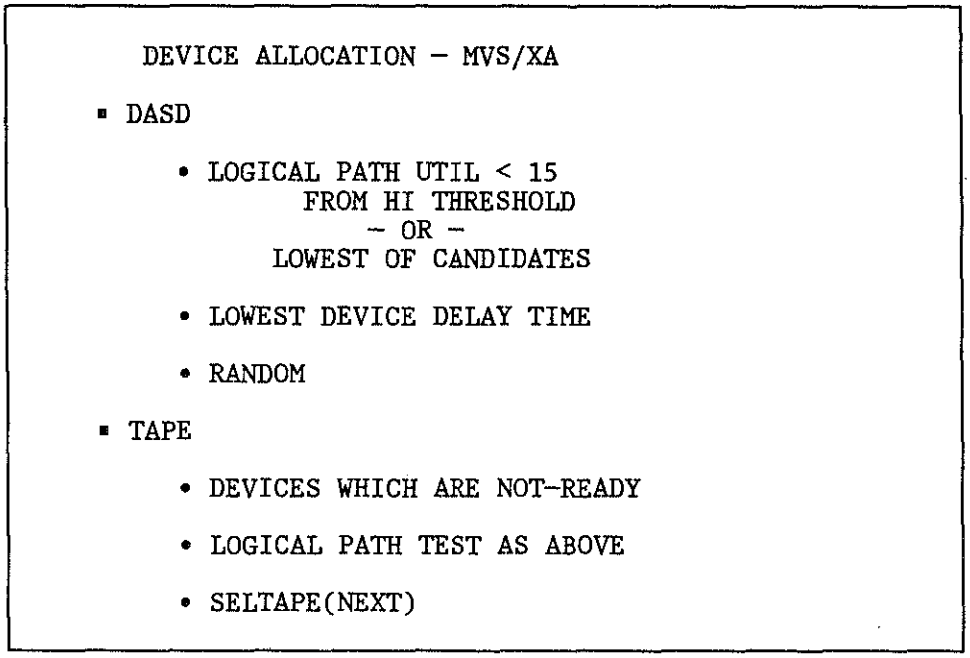


The constructs of logical channel utilization and logical path utilization have now been defined. The constructs are used by two functions in the SRM. They are nonspecific device allocation and I/O load balancing. If a dataset is allocated on tape or DASD, with no volume specified or implied, the SRM is called to select a device based on overall system parameters. The SRM attempts to pick a device, based on the load on the I/O subsystem so that a device will not be selected that may result in delays, not only to the new requestor, but also to current users of the I/O subsystem. For tape requests, some additional control can be exercised by the system programmer through the OPT PARMLIB member.

In MVS/370, the SRM will eliminate all candidate devices that are not on the logical channel(s) with the lowest logical channel utilization. Since logical channel utilization is the probability of a delay at SIO time, the SRM is trying to pick devices on paths that are not highly used already. From the remaining candidates, the SRM will choose the device(s) with the lowest allocated user count. This attempts to bias toward devices that are not highly utilized already. If more than one candidate remains, the SRM will choose one at random.

For tape requests, the SRM eliminates ready candidates. The presumption is that they are pre-mounted. The logical channel test is the same. From the remaining candidates, the SRM chooses the candidate with the next higher device name than the last tape allocated. For example, if tape 680 was the last tape allocated and 681, 689 and 580 were equally acceptable candidates, 681 would be chosen. This rule can be replaced by one of three other rules, specified via the OPT. The others are LOWEST, FIRST and RANDOM. If FIRST is specified, the first entry in the list is chosen. If lowest is specified, the lowest device name of the remaining candidates is chosen. The RANDOM specification is self describing.

Device Allocation in MVS/XA



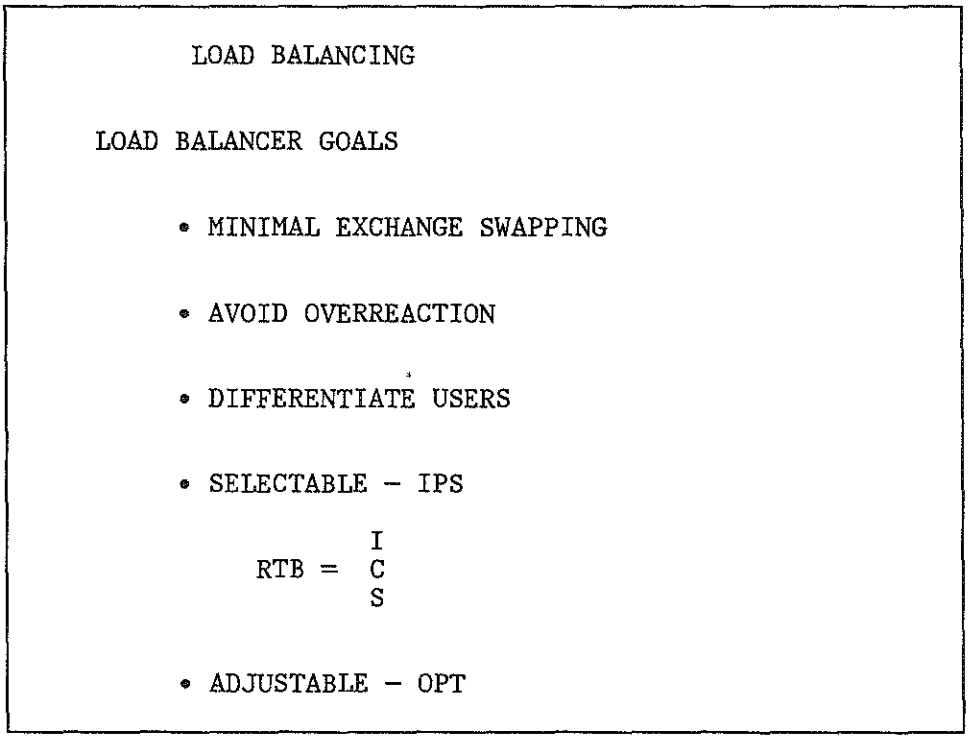
The SRM device allocation continues to operate as it did in MVS/370. In MVS/XA, however, the decision is based on Logical Path utilization and device delay time, rather than Logical Channel utilization and the allocated user count. Logical path utilization is a much better metric than logical channel utilization. Logical path utilization takes into account the probability of a delay at DASD reconnect time, which is the largest factor in the total I/O time as the path utilization increases. Device delay provides real information about potential devices to be selected. The allocated user count could be very high with very little contention on the device. Device delay is queued time due to previous request(s) and the measured pending time from the channel subsystem.

For DASD allocations, select the devices with the lowest Logical Path utilization relative to the high threshold for that type of device. If any logical path utilization is 15 less than the high threshold for that type of device, include all the devices on logical paths with utilization below that value. If there are no such devices, select the devices on logical path(s) with the lowest utilization. The high thresholds will be defined in the section on I/O load balancing.

From the remaining candidates, select the devices with the lowest delay time, based on I/O time and queue time. If more than one candidate remains, select a device at random from those remaining.

The tape selection process is the same as in MVS/370, except that the logical path test has replaced the logical channel test.

Chapter 8. Load Balancing



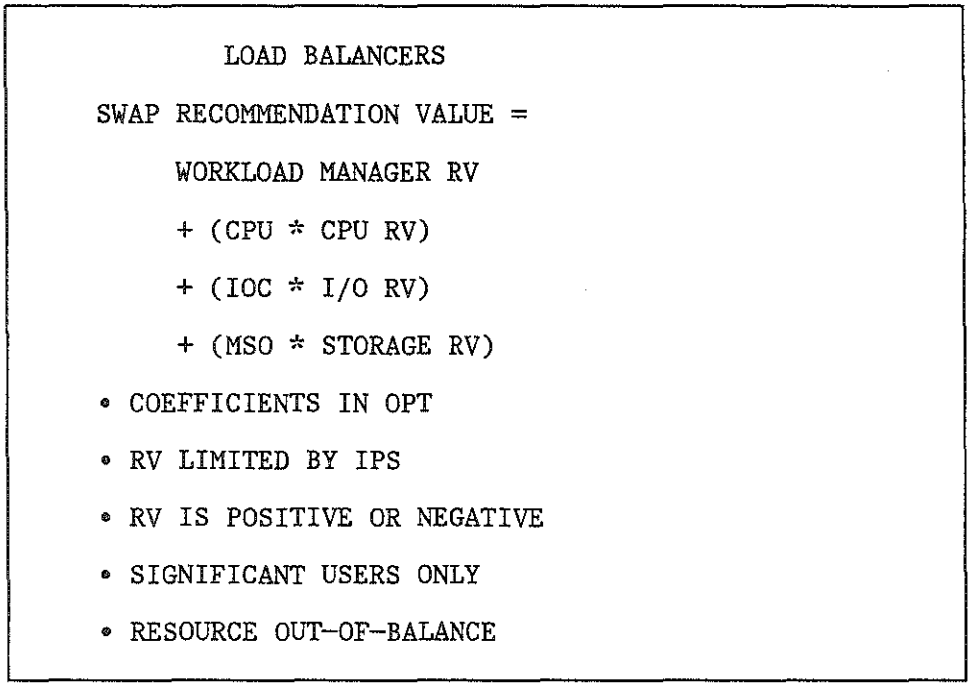
The SRM provides CPU, I/O and storage load balancers. The goal of each is to keep the use of the corresponding hardware resource "in balance". That is, the resource should not be overutilized or a significant point of contention and, therefore, potential delays. Also, it should not be underutilized and, therefore, a wasted opportunity for additional throughput. The algorithms seek to accomplish the goal by biasing swap decisions of eligible address spaces that use the hardware resources more intently than other eligible address spaces. During periods of high contention for the resource, the load balancer recommends a swap out of the address space(s) that demand that resource more than others. During periods of low contention, the load balancer recommends a swap-in of that same address space.

The load balancers are ineffective in managing a resource that is always out of balance. The best example would probably be storage. If insufficient storage is available with a resulting high demand paging rate, the storage load balancer would recommend that address spaces using the most real storage be swapped out. The problem might be that there are no periods of low contention for that resource; therefore, the work can never be swapped back in and completed. Similarly, if storage is plentiful, the load balancer will favor the jobs using the most real storage over other jobs. The load balancers are useful when a resource may see temporary periods of high contention and periods of lower contention. The idea is to let the work that needs a somewhat scarce resource have access to the resource when its demands can best be accommodated.

The use of the swap mechanism to alleviate resource shortages is prudent only if the frequency of swaps in low and the correct choice of swaps is made. Therefore, the load balancers use long term trends, remember recent past actions, and differentiate among the users of the resources.

The load balancers can be selected by performance group period in the IPS. Any combination of the load balancers can be selected. Of course, the best candidates for load balancers are the performance group periods representing discretionary work, like fourth period TSO. In the OPT PARMLIB member, the individual load balancers can be controlled by setting the coefficients which control the magnitude of the swap recommendations and by changing many of the parameters and key values that control the actions of the functions.

Swap Recommendation Value



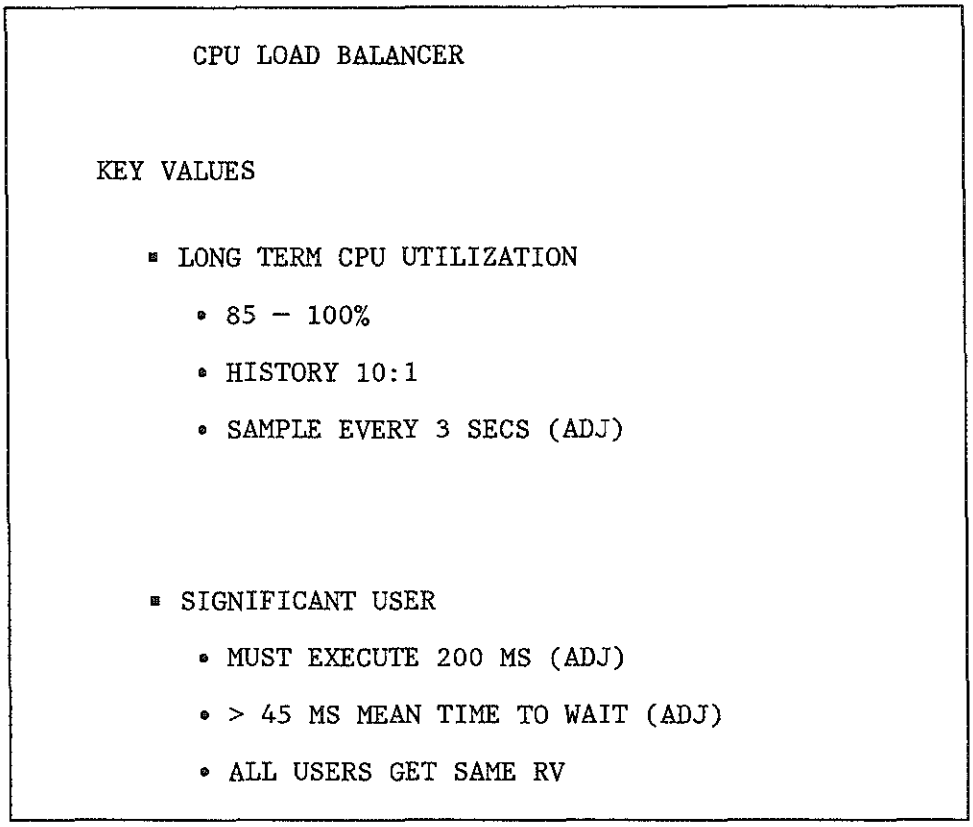
Every address space has a swap recommendation associated with it. The swap recommendation is the sum of up to four recommendation values, three of which are optional. The workload manager recommendation value is not optional. It is the recommendation value previously described in the section on objectives. If the CPU load balancer is selected in the IPS performance group period for the address space, and the CPU coefficient is non-zero in the OPT, the CPU load balancer will contribute a recommendation value as well. Similarly, the I/O and storage load balancers may contribute to the recommendation value. Address spaces with the largest recommendation values are to be swapped in, within the target MPL for their respective domains.

An important control on the load balancers is the selection of the coefficients in the OPT. The recommendation produced by each load balancer is limited to plus or minus one fifth of the maximum workload manager recommendation value associated with the IPS in effect. Therefore, the selection of the OPT coefficient can be used to weight the load balancer recommendation value(s) relative to the workload manager value.

The load balancer could override the workload manager if the coefficient selected was large enough. The coefficient could be set so that no matter what the load balancer recommended, the address space would be swapped back in because its service rate would decay to a point where its workload manager value would prevail.

The load balancers provide a nonzero recommendation value, only for the address spaces that are a significant user of the resource, and only when the resource is overutilized or underutilized, currently or in the recent past.

CPU Load Balancer

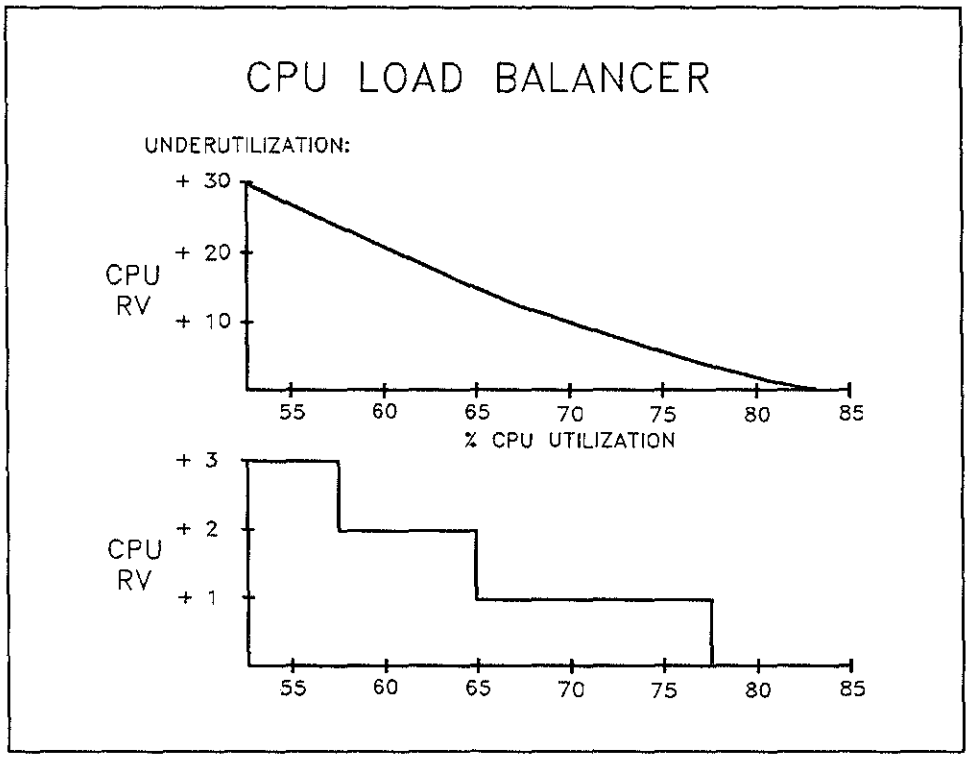


The CPU load balancer uses long term CPU utilization as the indication of contention for the processor(s). The processor is overutilized if the utilization exceeds 100%. The processor resource is considered underutilized if the utilization is less than 85%. The long term utilization is calculated by determining the utilization every three SRM seconds and averaging the most recent finding with the old long term utilization with a weight of 10 on the old and 1 on the new. This tends to dampen short lived changes in processor utilization so that overreaction will not occur.

An address space is considered a significant user of the processor resource if it has executed for 200 SRM milliseconds since the last swap-in and its mean time to wait is greater than 45 SRM milliseconds. This load balancer does not differentiate among the significant users. All significant users get the same recommendation value based on the long term CPU utilization.

The thresholds to determine under and overutilization and a significant user may be changed by OPT parameters.

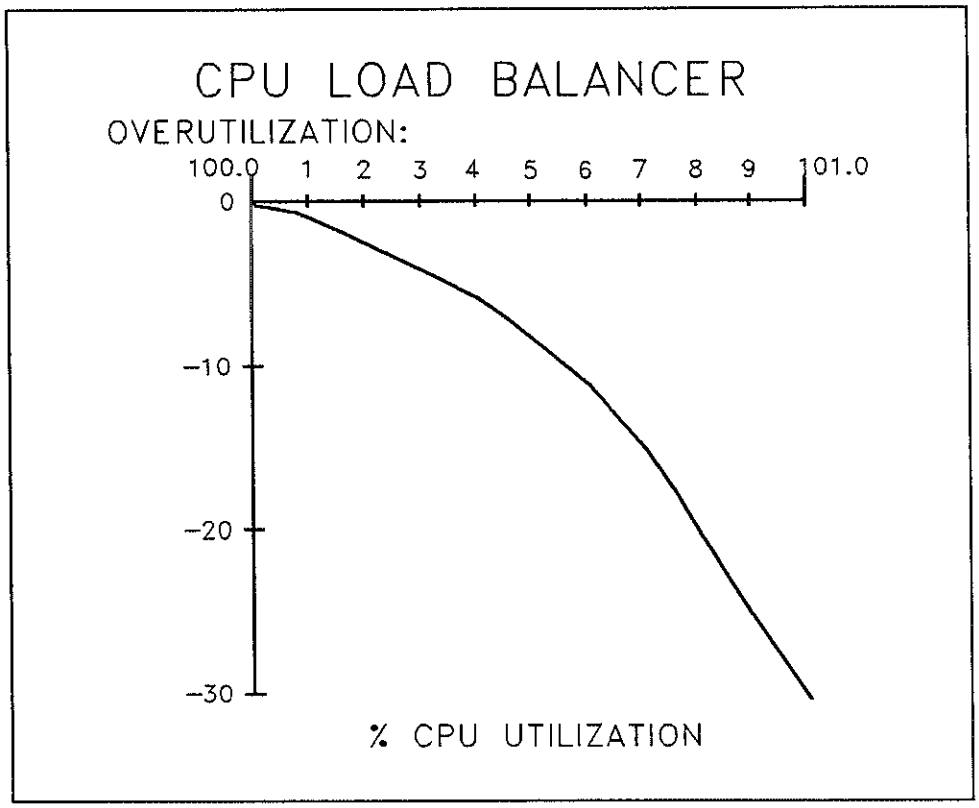
CPU Underutilization



This figure shows the load balancer recommendation value as a function of long term CPU utilization for two different IPS's. It shows how the load balancer is sensitive to the degree to which the processor resource is underutilized and also how the choice of the objectives may limit the load balancer. In the top graph, the IPS has a maximum workload manager recommendation value of 150. The load balancer is allowed to give a recommendation value of plus or minus 30. The recommendation value produced is one at about 82.5% and increases exponentially to 30 at 55%.

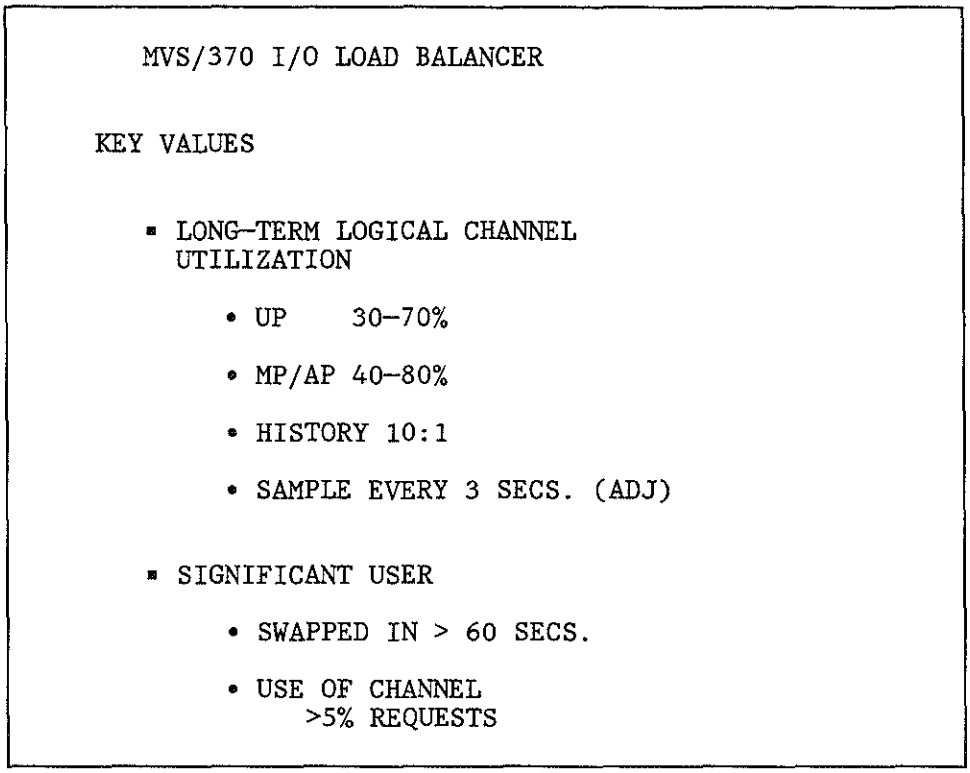
In the lower graph, the load balancer is limited to plus or minus 3 because the IPS has a limit on the workload manager recommendation value of 15. The recommendation value becomes nonzero at about 77.5% with this restriction, and the load balancer can only generate three possible positive recommendation values. The load balancers are all designed to round recommendation values up to deal with this specific restriction. All the load balancers operate more effectively when they are allowed a wider range of possible values.

CPU Overutilization



This figure shows the load balancer recommendation values for the overutilized case. Although the range of utilizations is smaller, the intent is the same. The recommendation value increases exponentially as the processor utilization exceeds the high threshold.

MVS/370 I/O Load Balancer

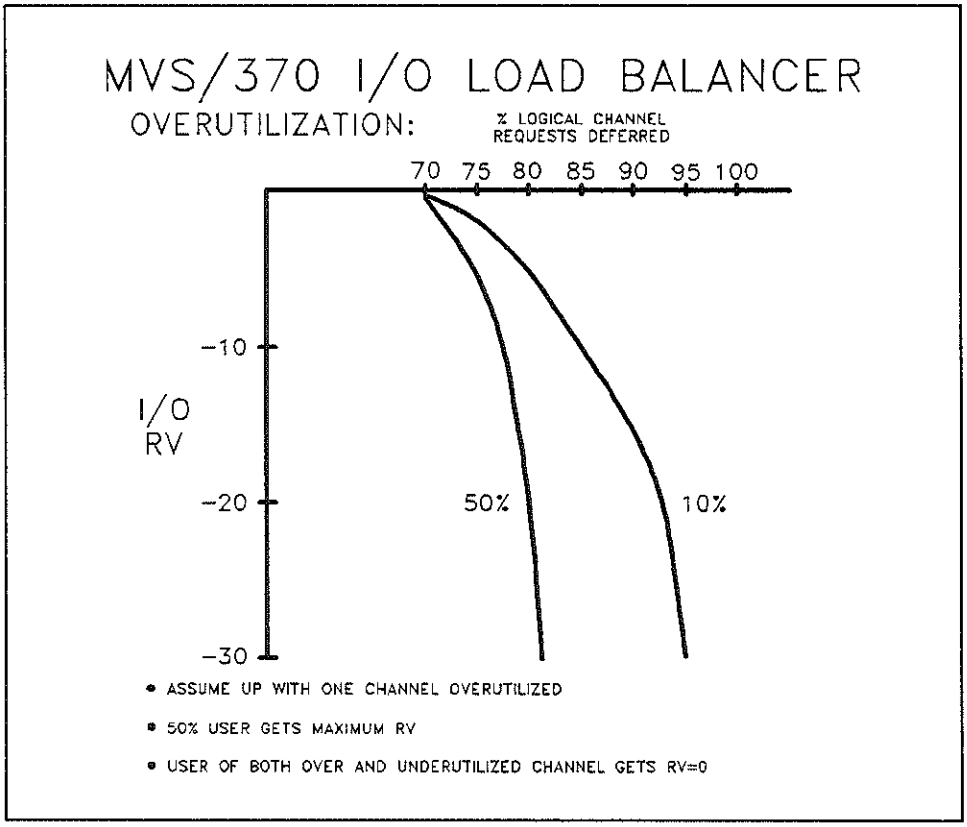


In MVS/370, the I/O load balancer uses long term logical channel utilization as the figure of merit for I/O. The percent of requests queued on the LCH is calculated every three SRM seconds and averaged with the current long term utilization to form a new one. The thresholds are set to 30 and 70% for a uniprocessor. The thresholds are set to 40 and 80% for an MP, AP or dyadic system. The reason for the higher thresholds with more than one processor is that some requests must be queued to get them started on the other processor even though they are logically started right away.

To be considered a significant user of a logical channel, an address space must account for at least five percent of the I/O's on the LCH and must have been swapped in for at least 60 seconds to show a consistency of use. The I/O load balancer does differentiate among the significant users of a logical channel as will be seen in the next figure.

The thresholds that determine under and overutilization, and the threshold to determine a significant user of I/O, can be changed by parameters in the OPT PARMLIB member.

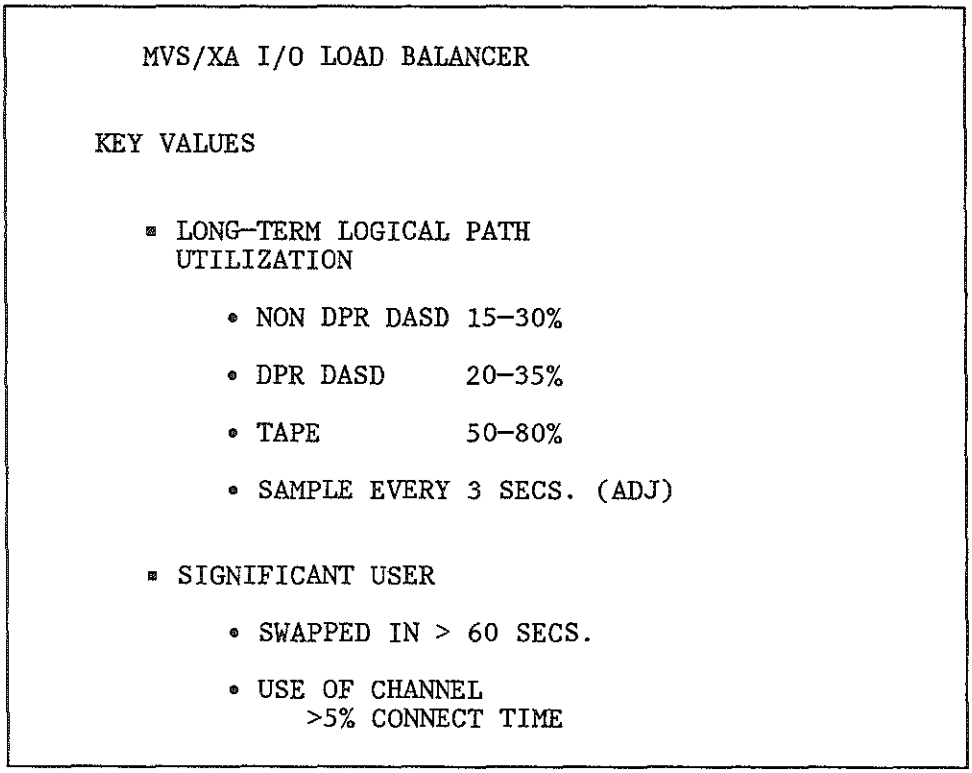
MVS/370 I/O Overutilization



This figure shows the recommendation values produced by the I/O load balancer assuming a uniprocessor and one channel overutilized. Recommendations are shown for two hypothetical users of the channel. One accounts for ten percent of the I/O's and the other accounts for fifty percent. If an address space accounts for more than fifty percent of the I/O's, it is treated as if it accounted for the complement of its percent. That is, if it accounted for sixty five percent of the I/O's, it would be treated the same as a thirty five percent user. The rationale is that swapping out an address space that accounts for more than fifty percent of the activity will probably only throw the utilization to the underutilized state. In other words, this guy is so big it is probably better to leave him alone. There is insufficient granularity among users.

The I/O recommendation value increases exponentially in absolute value as the utilization exceeds the threshold, just as the CPU load balancer recommendation value did. Also, if an address space is a significant user of both an overutilized logical channel and an underutilized logical channel, it gets a zero recommendation value. This says there is no value to balance one logical channel at the expense of another.

MVS/XA I/O Load Balancer



The SRM I/O load balancing function in MVS/XA functions very much like it did in MVS/370. The logical path is the I/O construct to be "balanced" and the long term logical path utilization is the figure of merit for the path. The measured connect time on the path is used to determine the significant users of the logical path. This is a much better indicator of the load placed on the I/O subsystem than the count of I/O requests on the logical channel.

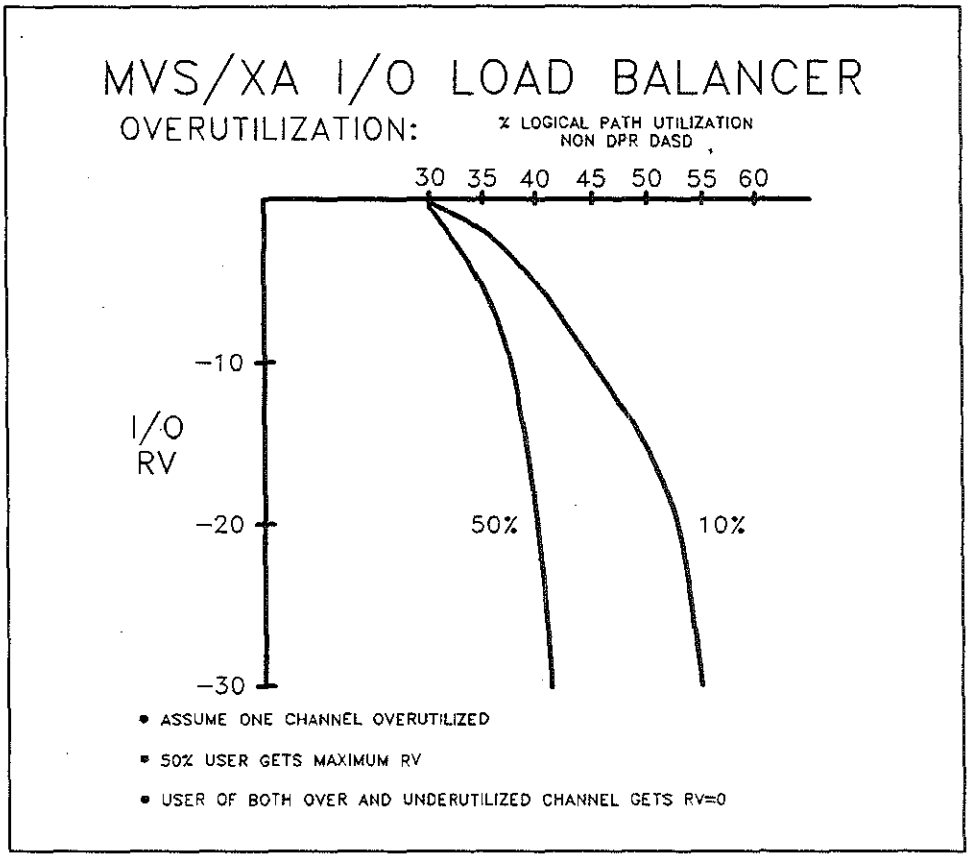
The numbers shown are the low and high thresholds for the three types of logical paths. The thresholds for DASD without the DPR feature are lower than those for DASD with the feature because the penalty for a missed reconnect is very high (one revolution). A typical channel program requires at least two reconnects. The reconnects must be done to one and only one path without the feature. Therefore, the thresholds are more conservative for DASD without the DPR feature.

If a logical path has devices from two of these categories, the SRM uses the threshold corresponding to the device type with the smallest high threshold. For example, if a logical path had DASD with and without the DPR feature, the logical path would be associated with the more conservative thresholds of the non-DPR logical path type. Remember that the high thresholds are also used in device allocation.

The determination of a significant user of the logical path is very similar to the MVS/370 implementation except that the measured connect time on the path must be at least five percent of the total connect time on the logical path.

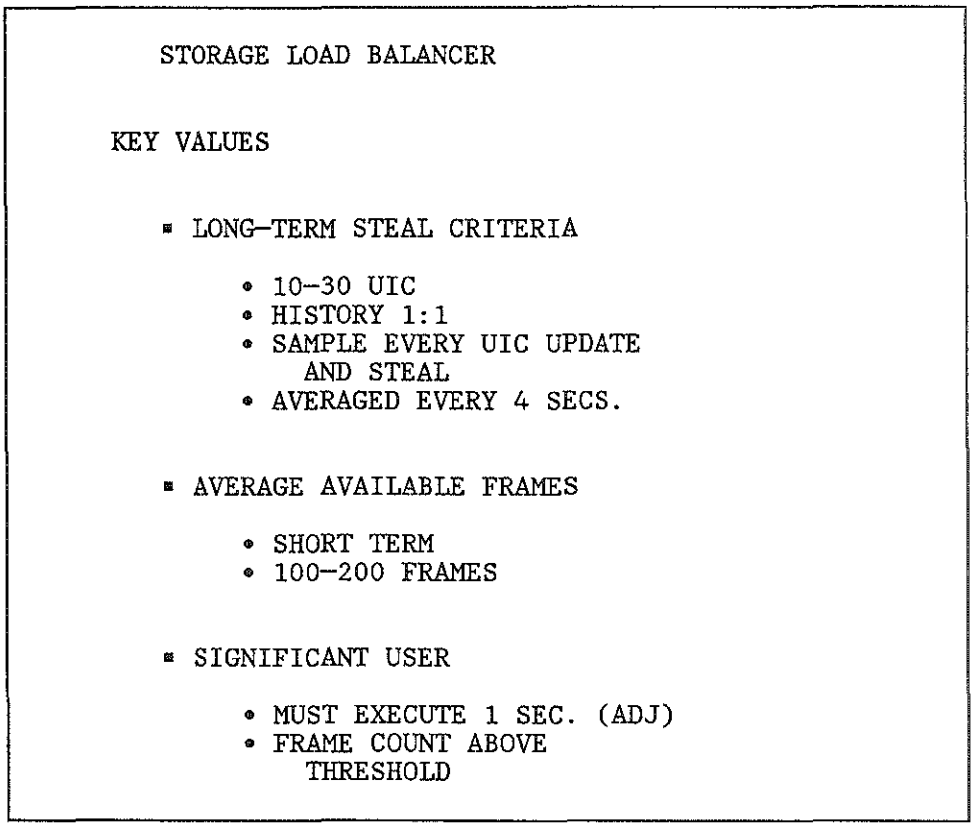
The thresholds that determine under and overutilization and the threshold to determine a significant user of I/O can be changed by parameters in the OPT PARMLIB member.

MVS/XA I/O Overutilization



This figure shows the I/O load balancer recommendation value produced by the load balancer again for two hypothetical users of a logical path to DASD without the DPR feature. Except for the scale, the graph is almost identical to the MVS/370 graph. The graph would be identical for either of the other types of logical paths except for the horizontal axis points.

Storage Load Balancer

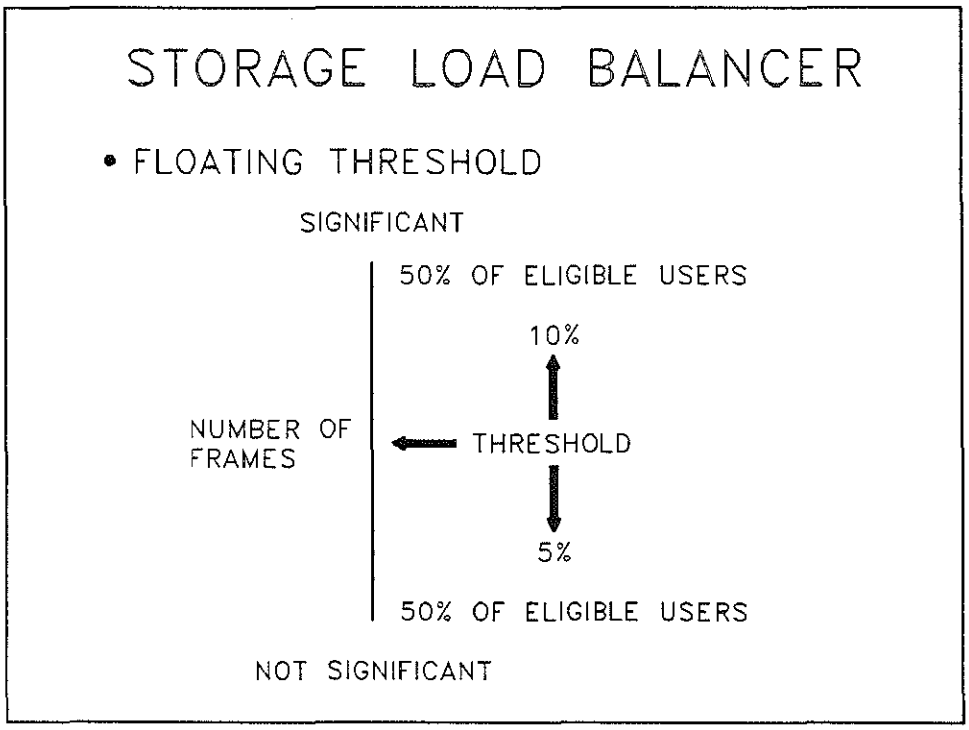


The storage load balancer uses two indicators of the contention for real storage to determine that the resource is overutilized or underutilized. The primary factor is the long term steal criteria or system high UIC as it is also called. The thresholds associated with the long term steal criteria are ten and thirty. The steal criteria is sampled when UIC's are updated and when page replacement is invoked. Every four real seconds a new long term steal criteria is calculated. In this case, the most recent four seconds are considered equally with the long term criteria. The rationale for weighting the current short term so much is that four real seconds are used as opposed to three SRM seconds for the other two load balancers.

The second figure of merit for storage is the number of frames on the available frame queue. Obviously, if a megabyte of frames are on the available frame queue, storage is not constrained. Theoretically, the steal criteria could be a small number, even with this many available frames, based on the page reference characteristics of the applications. In practice, this is very unlikely in MVS, unless storage isolation has been widely used, causing the steal criteria to be unrepresentative of the page life expectancy. The four second average available frame count must be less than 100 and the long term steal criteria must be less than 10 for storage to be considered overutilized. If the long term steal criteria is greater than 30, or if the average available frame queue length is greater than 200, storage is considered underutilized.

To be considered a significant user of storage, an address space must have executed for at least one SRM second since the last swap-in and must have more real storage allocated than a threshold which will be explained in the next figure.

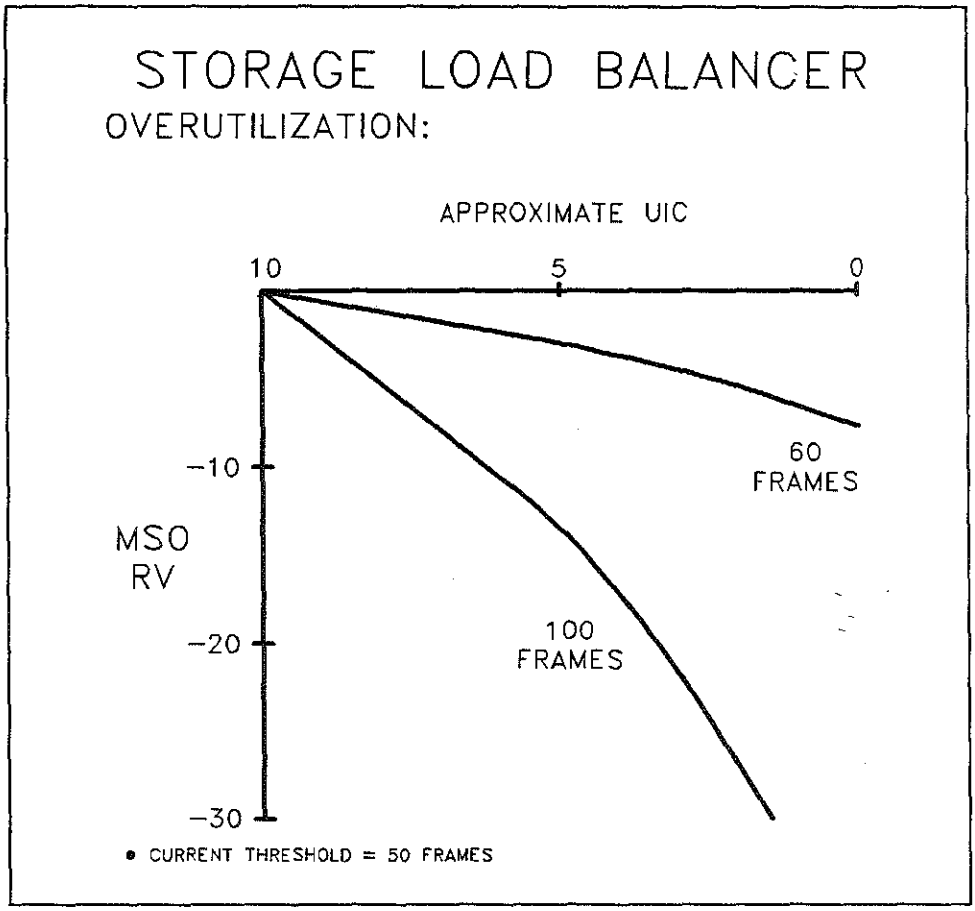
Storage Load Balancer Threshold



The requirements for virtual and real storage can vary significantly by application and over time as the application and even the operating system itself change. Therefore, it is very difficult to pick one number to determine a significant amount of real storage. The storage load balancer does not attempt to pick the number. It picks the number based on the storage used by address spaces eligible for storage load balancing. It dynamically establishes a threshold so that fifty percent of the eligible address spaces will be considered significant users of storage. The threshold is increased by ten percent and decreased by five percent until the median frame count is established.

There is a minimum frame count of forty frames for the threshold. The parameters shown here, as well as the thresholds from the previous figure, can be changed in the OPT.

Storage Overutilization



This figure shows the storage load balancer recommendation value generated for two address spaces when storage is considered overutilized. The current threshold is 50 frames to be considered a significant user of storage. This says that half of the eligible address spaces have more than fifty frames on the average and the others have less than fifty frames. The storage load balancer recommendation value also increases in absolute value exponentially as the long term steal criteria decreases below the low threshold.

Index

T

U

V

W

Numerics

READER'S COMMENT FORM

Title: An MVS System Resource Manager (SRM)
Discussion
Washington Systems Center
Technical Bulletin GG66-0201

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Please state your occupation: _____

Comments:

Please mail to: Ted Bauer
Washington Systems Center
18100 Frederick Pike
Gaithersburg, MD 20879

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

Mr. T. F. Bauer
NAD Washington Systems Center
IBM Corporation
18100 Frederick Pike
Gaithersburg, MD 20879

Fold and tape | Please Do Not Staple | Fold and tape

IBM

GG66-0201-00

An MVS SRM Discussion

Printed in U.S.A.

GG66-0201-00

IBM