SHARE SESSION REPORT

61	B190	Most Misunderstood Parts of the SRM	400+
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE

MVSP Performance	John Sullivan	LLU
PROJECT	SESSION CHAIRMAN	INST. CODE

Loma Linda U. Medical Ctr., PO Box 2000, Loma Linda, CA 92354, 714-824-4317

SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

Mr. Bernie Pierce from IBM presented the attached material.

THE MOST MISUNDERSTOOD PARTS OF THE SRM???
OR
THE PARTS MOST FREQUENTLY ASKED ABOUT

Bernie Pierce
International Business Machines Corporation
PO Box 390, D86/921
Poughkeepsie, New York 12602
(914) 298-5519

Session B190
MVS Performance Project
August 23,1983

Abstract

This presentation 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. It will be assumed that the audience 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 presentation 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 and how does page stealing work. The material will be applicable to both MVS/370 and MVS/XA.

Good morning. My name is Bernie Pierce. I work in MVS design and development in Poughkeepsie. I have concentrated on the SRM component of MVS for the past few years. Today I would like to address some of the aspects of the SRM that seem to elicit the most questions from systems programmers and systems engineers. I tried to communicate to the potential audience in the abstract that the presentation will be at a fairly detailed level. I will assume a basic knowledge of the SRM and some of its constructs, such as basic IPS controls. On the other hand, you should not need to be a tuning expert to learn something about the SRM today.

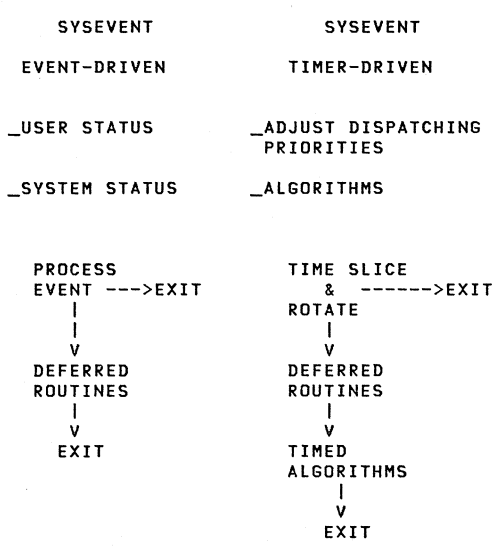
OUTLINE

- _ SRM TIMING
- _ TSO TRANSACTIONS
- _ STORAGE CONTROL
- _ MPL ADJUSTMENT
- _ SWAP CONTROL

This presentation covers several 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. I will point out changes made for MVS/SP Version 1 Release 3 and for MVS/SP Version 2 Release 1 during the presentation. I would like to point out that there will be a presentation tomorrow at 8:30 on SRM support for MVS/XA and RMF Version 3 Release 1. The SRM material presented today will not in general be covered in tomorrow's presentation. That session is number B181.

Most of the topics that I will discuss today are selectable and/or adjustable by the installation through documented externals. The outline shows the subjects to be discussed in this presentation. These subjects 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 (GC28-1029 for MVS/SP Version 1 and GC28-1149 for MVS/SP Version 2).

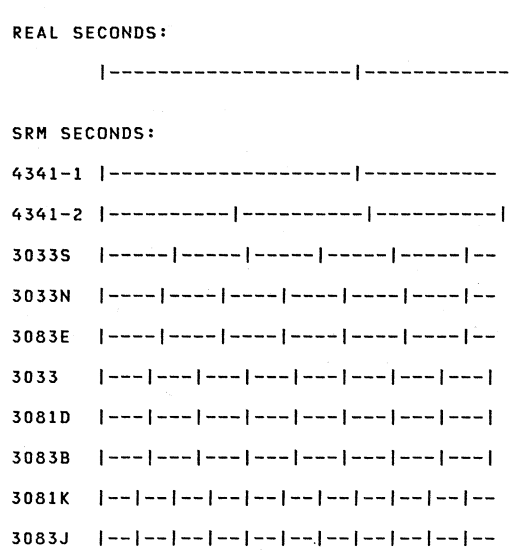
SRM INVOCATION



Understanding the implications of SRM timing requires some background on how 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 user becoming ready, or swap out completion. SRM then 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 I will be discussing today are page replacement or frame stealing and multi-programming level adjustment.

TIMING



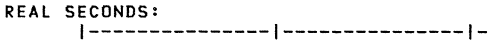
Timed events occur based either on real seconds, or on a processor adjusted second (often called an SRM second). The relationship of SRM seconds to real seconds is a function of the approximate speed of a single processor of the host machine. For instance, a 3033 SRM second is much shorter than a 4341 second. Algorithms which are driven on the basis of SRM seconds are therefore executed more frequently on a 3033 than on a 4341. Many algorithms are adjusted for the processor. For example, the algorithm that checks for period switch should run more frequently as the processor speed increases to maintain the integrity of the definition of a trivial TSO transaction. An application can consume resources at a much 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 3081D has about the

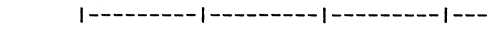
same size SRM second as a 3033, although the 3081D has over two times the capacity of the 3033. As a historical note, when MVS Release 2 was shipped, one SRM second on a 155 Model II was defined equal to a real second.

TIMING

REAL SECONDS:

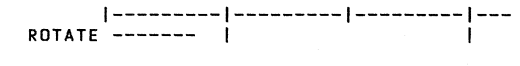


SRM SECONDS(4341-2):

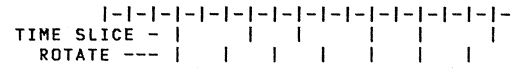


ROTATE OR TIME SLICE DISPATCHING:

TUNIT=1 (R2 , FRQ = 2)



TUNIT=5 (TSPTRN = (1,1,1,*,*))



No SRM algorithm is executed more frequently than once per SRM second. There are several routines that do run every SRM second however. Therefore the default rate of SRM timer interrupts or "timer pops" is one per SRM second. On a 3033 the default time between timer pops 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 pops 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 pops with this TUNIT specification. The SRM only sets timer pops 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 pops at all. In the example, time slice group 1 stays "up" for 3 timer units and then "down" for 2. No timer pops 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 pops. This can be done by specifying a parameter in the OPT.

TIMING

SERVICE UNITS BY PROCESSOR:

4341-2	78.9/SEC	
3033	262.5/SEC	
3083B	306.5/SEC	
3081K	368.4/SEC	
3084	368.4/SEC	PARTITIONED
3084	346.0/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

I have listed the service unit definitions for some of the processors. As you can see, 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. As you can see, 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.

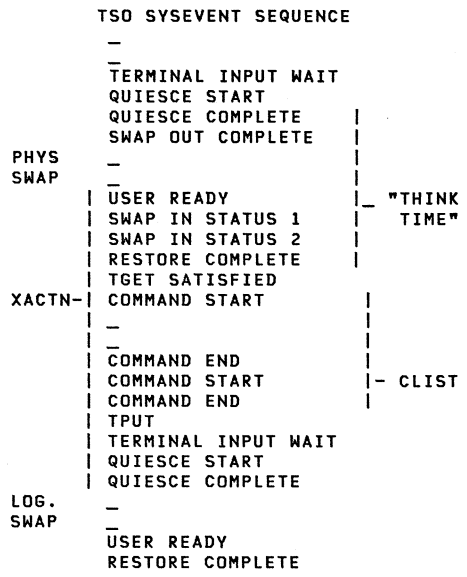
These numbers are from the Initialization and Tuning Guide. I have also shown how you can verify the value that the SRM is using on your configuration with a very simple TSO TEST session. The field names and therefore the offsets can be found in the Debugging Handbook. If you use this, you may find that 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 presentation, when I refer to times, they will be real times unless I indicate otherwise.

TSO TERMINOLOGY

- TRANSACTION
  - _ DEMAND FOR SERVICE
- RESPONSE TIME
  - _ TIME TO SATISFY DEMAND
- THINK TIME
  - _ TIME BETWEEN DEMANDS
- PHYSICAL SWAP
  - _ REMOVE FROM STORAGE
- LOGICAL SWAP
  - _ LIKE PHYSICAL BUT STOP SHORT

Before I discuss storage control with topics such as logical swapping, I would like to define some of the terms that I may use, like "think time" for instance. The first term is a transaction. I call it a demand for service. As an end user myself, I would define a transaction as the request for processing made by me when I hit the enter key on my terminal. Response time is the time for the system to process my transaction and prepare for another. Think time is the time between user requests for service that allow us to run hundreds of users on one processor; if we do our jobs right each they all think 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 us to multiprogram the storage as well as the processor. A logical swap is like a physical swap and thus the name. However we do not remove the address space from storage right away. We prepare for that eventuality but most of the time we do not do the I/O.

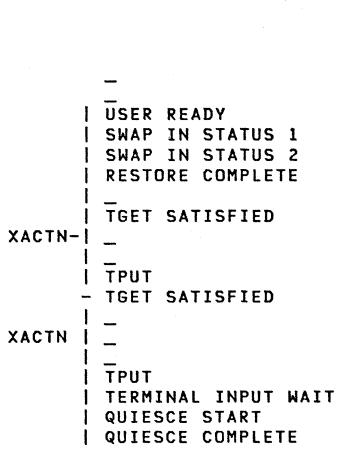


The SRM is able to define think time and response time because it hears about the progress of a TSO user's demand for service through various SYSEVENTS that are issued. In the example a SYSEVENT is issued stating that the address space may be in terminal wait. What this means is the application asked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

Think time is defined as the time from quiesce complete to restore complete. 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 SYSEVENT indicates that the program that went into terminal wait is no longer in terminal wait. The COMMAND START and COMMAND END SYSEVENTS are indicating that a clist is being processed. These SYSEVENTS allow the SRM to report on TSO command usage through RMF reports. They also allow 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 TSO/E product.

The address space again enters the terminal wait condition. This ends the transaction started by the USER READY SYSEVENT. The response time is calculated from the time of the USER READY SYSEVENT. This is the most common way to end a transaction for TSO. In this case, the address space is logically swapped instead of physically swapped. Later I will discuss the mechanics of the logical swap process. The decision was 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.



I am using this SYSEVENT sequence to show another way to start and end a TSO transaction. It also shows how you can get more transactions than swaps on the RMF workload

report. In the previous foil there was a TGET SYSEVENT following a swap in and then a TPUT SYSEVENT before entering terminal wait. In this case there is a second TGET 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 SYSEVENT is received, SRM ends the transaction initiated at USER READY and starts a new transaction which we see ending due to terminal wait as in the previous foil. Another way to get multiple transactions in the interval would be to count the commands in a CLIST as separate transactions.

TSO SYSEVENT SEQUENCE

```
          _
          _
        | USER READY
        | SWAP IN STATUS 1
        | SWAP IN STATUS 2
        | RESTORE COMPLETE
        | _
        | TGET SATISFIED
XACTN-  | _
        | _
        | TPUT
        | TERMINAL INPUT WAIT
        | QUIESCE START - WORK TO DO
        |  TURN SWAP AROUND
        | _
        | _
        | TPUT
        | TERMINAL INPUT WAIT
        | QUIESCE START
        | QUIESCE COMPLETE
```

This foil demonstrates that the TERMINAL WAIT 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 multi-tasking 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.

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 presentation is storage control. Storage control is not as well understood as processor control in my experience. First, I will explain what kind of storage control is provided by default. To really describe storage control, we would have to understand Multi-Programming Level control and logical swapping since those functions allocate storage to address spaces. We will cover those subjects later. For the moment just assume 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 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. I will discuss the concept of the unreferenced interval count and explain its role in storage control. I will show how it drives page replacement or page stealing. In the discussion of frame stealing, we will see how storage isolation can modify the global LRU algorithm of MVS.

UNREFERENCED INTERVAL COUNT (UIC)

"WHAT IS A UIC ?? "

"THE # SECS A FRAME 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 or 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. 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. 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 to start the LRU page replacement algorithm. We want to take the oldest pages and we need 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 MPL adjustment.

UNREFERENCED INTERVAL COUNT (UIC)

UPDATE INTERVAL - 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

UPDATE INTERVAL - SP 2

HIGH UIC	NON SWAPPABLES/ COMMON	SWAPPABLES	HIGH UIC
0-5	1	1	0-9
6-11	2	2	10-19
-	-	-	-
30-35	6	6	50-59
_	_	6	_
120-125	20	6	_

This foil shows the update intervals for the routine that I described earlier. It must examine the reference bit of most of the real storage pages online. Therefore the fre-

quency of the algorithm is decreased as the contention for storage decreases. The theory here is that as we steal older pages, we do not have to be as discriminating about the exact age of the pages because we will be doing it less frequently and the impact on the application will be less than when stealing more recently referenced pages. The top of the foil 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.

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

- 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?

These are some of the questions that I hope to address about the page replacement process. I have already answered the first question and have hinted at the next two at least. Frame replacement is done so that RSM will have real page frames available to satisfy page faults and swap in address spaces. RSM starts it and we'll soon see

how. Storage isolation definitely affects the process - that is what it was designed to do. It can be ignored under certain circumstances, particularly if it has been used indiscriminately.

FRAME STEALING

SYSEVENT AVQLOW AVQ<14

1 STEAL FRAMES ABOVE MAX. SIZES

2 STEAL CRITERIA IS HIGH UIC

3 STEAL EXCESS FROM LOGICALLY SWAPPED

4 PRIVATE THEN COMMON - STEAL EVENLY

5 LOWER CRITERIA

6 REPEAT, IF NEED MORE FRAMES

7 STEAL TWICE AT CRITERIA 0

7A CRITICAL - STEAL FROM LOGICALLY SWAPPED
- STEAL BELOW TARGET SIZES

8 RESCHEDULE

SYSEVENT AVQOK AVQ>18

The page replacement process is started by RSM issuing a SYSEVENT to SRM to indicate that the supply of available page frames is less than a threshold set by SRM. The threshold is normally set to 14, but it can be changed by SRM when an address space is to be swapped in and enough frames must be gathered up to accomplish the swap in. Right away storage isolation comes in to play. Before we enter the LRU part of the process, we see 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. I will describe those terms better shortly. Usually, few pages are stolen in this phase. At step 2 we start 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. I will show the relationship shortly. 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, we see if the steal criterion is below a threshold and if it is we visit the logically swapped and remove the extra pages that we may have allowed them to keep.

At step 4, frames are stolen from address spaces and the common area. By the way, SRM does not steal the pages itself; it merely indicates to 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 my or may not be stolen from. They are candidates if they own more frames than their working set target. The target will be described shortly. 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 we still can't get enough frames we consider this a critical situation. We steal any pages we can get at this point. We look for logically swapped address spaces and then we 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. RSM notifies SRM when the supply of frames exceeds a threshold, normally 18 frames. That ends the process.

PHYSICAL SWAP WORKING SET

"WHAT'S IN A WORKING SET ?"

- SP 1.1 - PAGE REFERENCE BIT ON
OR
MINIMUM WORKING SET

UIC UPDATE:

- SP 1.3 - PAGE REFERENCE BIT ON
OR
UIC UPDATE RECENTLY AND
UIC = 0
OR
MINIMUM WORKING SET

As I said 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. You can see how many pages are reallocated or what the average swap working set size is from the RMF paging activity report. You 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 specified. If you ignore storage isolation, the intent of the page reference bit test is to give 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. You can see 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, 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 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.

LOGICAL SWAPPING

- EXPLOITS AVAILABLE REAL STORAGE
- FOREGO SWAP I/O WHEN STORAGE PERMITS
- OPT PARAMETERS CAN:
  - _ TURN OFF LOGICAL SWAPPING
    OR
  - _ MAKE MORE AGGRESSIVE
- RMF REPORTS SUCCESS OF ALGORITHM

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, 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.

LOGICAL SWAPPING

"WHAT HAPPENS ON A LOGICAL SWAP?"

1 QUIESCE ADDRESS SPACE

2 PREVIOUS SHORT THINK TIME AND/OR
ENOUGH REAL STORAGE

3 LOWER DISP PRTY TO 01

4 MOVE TO WAIT QUEUE

5 LEAVE PAGES IN STORAGE

6 TRIM TO WORKING SET

7 DON'T UPDATE UIC WHILE SWAPPED

8 STEAL AS PREVIOUSLY DESCRIBED

9 DETECT LONG THINK TIME

-----> PHYSICAL SWAP

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 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. We will just get it out and will have to bring it back in. So the objective is to keep short thinkers in storage while we can afford to.

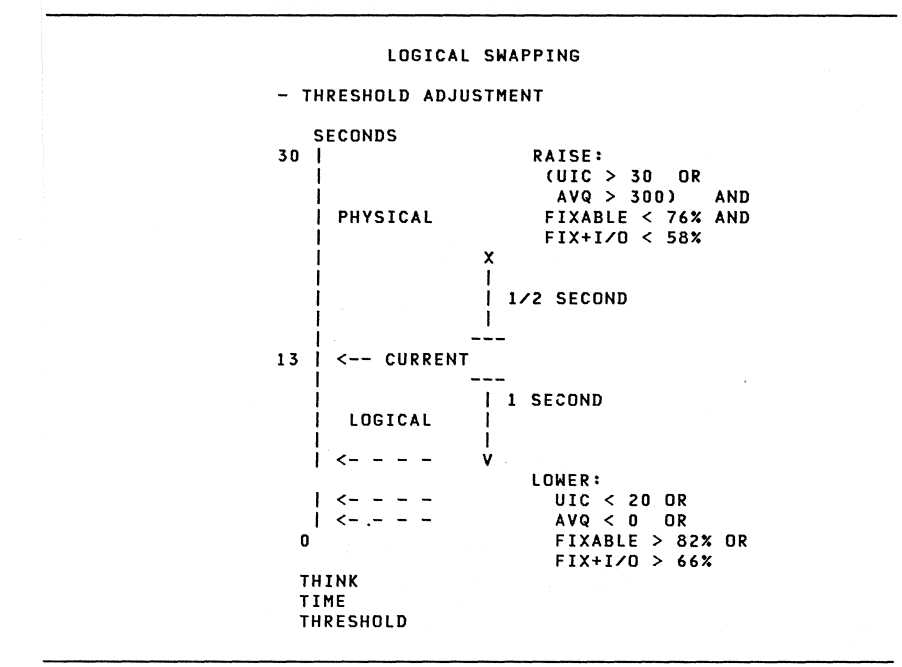
In SP 1.3, SRM allows address spaces selected for swap based on a LONG WAIT SYSEVENT or by the detected wait algorithm to be candidates for logical swap. These candidates are given lesser priority than the TSO terminal wait candidates. One reason for this is that SRM can not speculate on how long the 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 I will describe 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 reserved capacity than it really has available.

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 I said earlier 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 foil. The most important test is the average system high UIC greater than 30. This constant as well as all other constants in the tests on the foil 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 con-

ditions 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.

LOGICAL SWAP DECISION

- SP 1.1
  - _ TERMINAL WAIT ONLY (TSO)
    - THINK TIME < SYSTEM THINK TIME THRESHOLD
    - PUSH OUT WHEN THINK TIME > THINK THRESHOLD + 15 SECONDS
- SP 1.3
  - _ TERMINAL WAIT
    - THINK TIME < THRESHOLD AND UIC >= LSCTUCTL/2 (10)
    - PUSH OUT SAME AS ABOVE

This foil shows the logical swap decision in more detail than I have described previously. 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 then 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, we just wasted the storage and the address space will probably become ready and we will swap in back in in a matter of milliseconds. I cannot justify the value

of 15 seconds but I can tell you that in an unplanned experiment the value was set to 0 on our production TSO system with no real change in the logical swap statistics. I conclude 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 had to be greater than one-half the low UIC threshold for controlling the system think time threshold. The rationale for this added condition is that storage contention can change dramatically and the heuristic nature of the system think time threshold adjustment made for relatively slow changes in the storage used for logical swap.

If the system high UIC is at this level and remains at this level, logical swap will decline. The test bypasses the logical swap. If the contention is short-lived, the system high UIC will increase and logical swapping will become active again. If the situation persists, logical swapping has ended swiftly. I should add that a further condition has been added in PTF UZ60519 for 1.3 and UZ60520 for 2.1. 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 DECISION

- SP 1.3
  - _ LONG AND DETECTED WAIT
    - THINK TIME THRESHOLD > 5 AND UIC > LSCTUCTH (30)
    - PUSH OUT WHEN LOGICALLY SWAPPED LONGER THAN THE THINK TIME THRESHOLD
  - _ RMF SUPPORT VIA TRACE

The conditions for logically swapping other types of wait states are slightly different. As I said, 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. As you can see we are treating 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. You can use RMF trace if you are interested in the number of swaps avoided or you can use RMF Monitor II to check on particular applications. It is not cost effective to rewrite the Paging report to show that we have not done swaps that we shouldn't do in the first place. 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 TRIM

163

- SP1.1

  _ STEAL CRITERIA = 1
    STEAL ALL PAGES UIC >= 1

- SP 1.3

  _ TSO TERMINAL WAIT

AVE HIGH UIC	STEAL CRITERIA
0-29	0
0-29	1 RECENT UIC UPDT
** 30-39	1
40-49	2
50-59	3
–	
–	

  _ OTHER SWAPS

    - STEAL CRITERIA = 0 OR 1 AS ABOVE

** ACTUALLY LSCTUCTH - OPT UIC HIGH THRESHOLD

This foil could also be called logical swap working set to correspond with an earlier foil. 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 UIC 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 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 we can give larger working sets to avoid demand page operations and 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 we saw in the frame stealing foil. Of course, minimum working sets are honored just as with physical swap working sets.

STORAGE ISOLATION

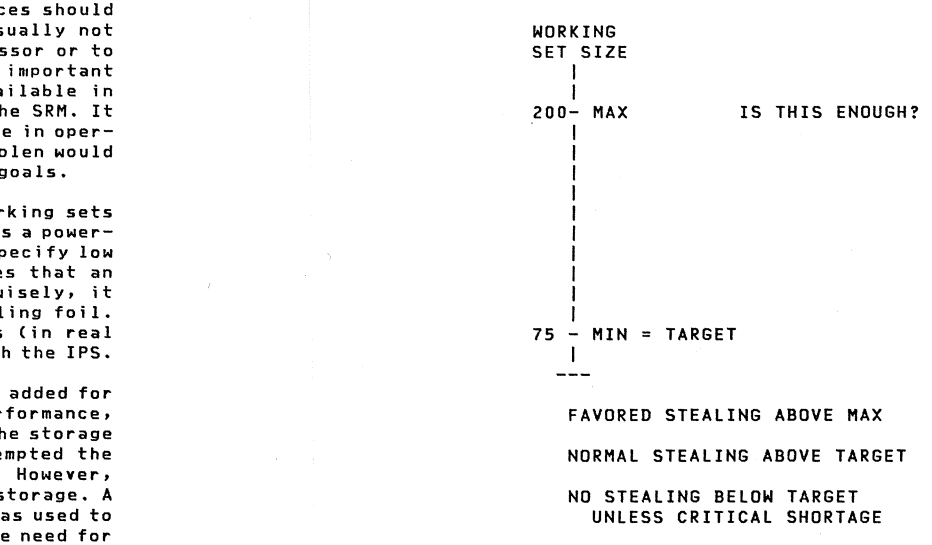
- LRU IS EGALITARIAN
- PROTECT ONLINE RESPONSE TIME
- PROTECT PRIVATE AND COMMON FRAMES
- POWERFUL TOOL - USE WISELY
- INVOKE THROUGH IPS
  - _ WORKING SET LIMITS
  - _ PAGING RATE LIMITS
  - _ BOTH
- GRS DEFAULTS TO NO STEALING

Storage isolation is one of the most powerful and effective functions offered by the SRM to meet your response time goals. As I said 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 we saw in the frame stealing foil. The installation can specify working set sizes (in real frame units), paging 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.

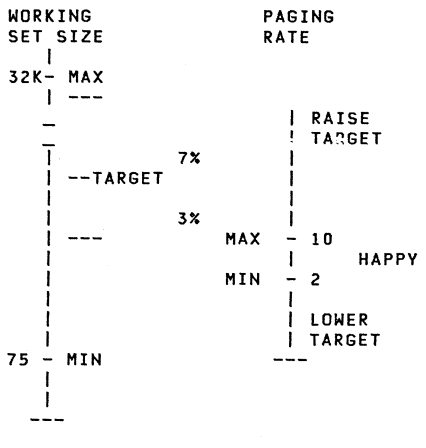
STORAGE ISOLATION



The simplest form of storage isolation is just working set control. A minimum and a maximum working set are specified. If an address space has more real frames allocated to it than the maximum, pages will be preferentially stolen from it. It has been my experience that installations have had difficulty with the maximum. If the intent of using storage isolation is to protect responsiveness, let the maximum default. 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.

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.

STORAGE ISOLATION



- FAVORED STEALING ABOVE MAX
- NORMAL STEALING ABOVE TARGET
- NO STEALING BELOW TARGET
  UNLESS CRITICAL SHORTAGE

Paging rate thresholds can also be specified for an address space or the common area. Storage isolation is a function designed to control the paging rate. This control is more direct but it is somewhat more complex due to the definition of paging rate which I will discuss shortly. When paging rate controls are in use, the target working set size is adjusted by SRM to try to keep the paging 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. If the paging rate for the address space or the common area exceeds the maximum paging rate, the target is increased by 7% to protect more storage and reduce page stealing. If the paging rate is less than the minimum rate, too much is being protected and the target is decreased by 3%.

STORAGE ISOLATION

- PAGE RATE FOR TARGET ADJUSTMENT

  - CROSS MEMORY SPACE OR COMMON

    EVERY 10 SECONDS - PAGE INS PER SECOND ELAPSED TIME

  - OTHER ADDRESS SPACES

    IF 1 SRM SECOND EXECUTION TIME AND 10 SECS RESIDENCY TIME SINCE LAST CALC - PAGE IN PER SEC EXECUTION TIME (TASK+SRB)

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 paging rate in the happy range. To do this effectively, it must know that the previous change has been tested. If I turn up the thermostat because it is freezing and 5 seconds later check the thermometer, I probably would 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 SRM increases the target working set for an address space because its paging rate was high and then recalculates a paging 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, SRM requires that most address spaces execute for 1 SRM second between paging rate computations. It is also required that the address space be resident in real storage for 10 real seconds.

The paging 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 paging 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 cer-

tainly true of the common area. The paging rate for these elements is defined in terms of page-ins per real second.

MULTI-PROGRAMMING LEVEL ADJUSTMENT

- SAMPLE EVERY SRM SECOND
- MPL CALCULATION EVERY :
  - _ 20 SECONDS IN MVS/SP VERSION 1
  - _ 106 SRM SECONDS / (#PROCS * .85)

    IN

    5 < MVS/SP 2 < 20 REAL SECS
    - 3083E 20 SECS
    - 3083J 11 SECS
    - 3081D 9 SECS
    - 3081K 7 SECS
    - 3084 5 SECS

The multiprogramming level adjustment routine is one of SRM's more well-known algorithms. I am including it 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.

MPL ADJUSTMENT

"WHY IS THE READY USER AVERAGE SO LARGE?"

- READY USER AVERAGE FOR DOMAIN MULTIPLIED BY 16 INTERNALLY
  - _ RMF TRACE REPORTS INTERNAL VALUE
  - _ DISPLAY DOMAIN ROUNDS TO INTEGER
  - _ RMF MON II ROUNDS ALSO
- MPL FOR DOMAIN WILL NOT FALL BELOW:

$$\frac{RUA + MAX(RUA)}{2} + 1$$

- RUA PULLS MPL SO MIN MPL IS IMPORTANT
- ADJUSTABLE IN OPT

The ready user average for a domain is internally scaled or multiplied by 16. This allows 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, 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. We want the MPL to accommodate the address spaces when they come ready if the real resources are available.

Unfortunately, only the ready user average alone can increase the target MPL. Therefore it is still important to have reasonable minimum MPL's for TSO domains. All the thresholds that SRM uses to determine if resources require a higher or lower MPL can be changed in the OPT.

MPL ADJUSTMENT

_ RAISE MPL IF:

AVERAGE HIGH UIC > 4

AND

AVERAGE CPU UTIL < 98%

AND

DEMAND PAGING RATE < 70 (3033UP)
OR
PAGE DELAY < 100 MS.
AND
AVERAGE CPU UTIL.
< 95%

AND

- OK TO HAVE HIGH PAGING RATE AS LONG AS PAGING SUBSYSTEM IS ABLE TO HANDLE IT

Here is the first of two foils showing the logic for increasing the MPL. The ones on this foil are the most important tests provided by SRM. The tests on the next foil 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. It may not be bad in a severely storage constrained system running batch, though. 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 I said, SRM has to handle all situations, but in the matter of MPL adjustment it can use some help from you. Throughput must often be traded off against responsiveness when resources are in contention. This algorithm tends toward throughput and if your goal is to be responsive it may allow too much demand paging when there is storage contention. You can counter this with storage isolation or by changing some of these thesholds.

MPL ADJUSTMENT

_ RAISE MPL IF:

AVE FIXABLE STOR FIXED < 82%

AND

AVE STOR FIXED OR PAGE I/O < 66%

AND

NO RECENT PAGEABLE STOR SHORTAGE

AND

AVE ASM QUEUE LENGTH < 1000

AND

PAGE FAULT RATE < 1000/SEC

AND

AVE 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 your paging configuration and your responsiveness goals you can set page fault parameters that will control the MPL.

MPL ADJUSTMENT

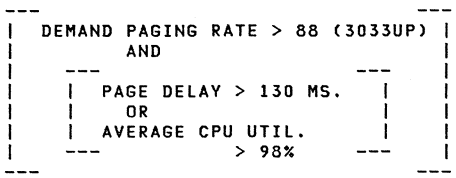
_ LOWER MPL IF:

AVERAGE HIGH UIC < 2

OR

AVERAGE CPU UTIL > 100.9%

OR



OR

- ONLY LOWER MPL DUE TO PAGING RATE IF PAGE DELAY OR CPU UTIL. ALSO INDICATES AN OVERLOAD

168

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

I should explain how the CPU utilization could be greater than 100%. CPU utilization is 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). If this persists for a long time, we have dead wood in storage. Address spaces are sitting in storage but are not being dispatched. This could be bad if the address space owns an ENQ resource. 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. SRM assumes that they will run if they are swapped in. This logic attempts to ensure that.

MPL ADJUSTMENT

_LOWER MPL IF:

AVE FIXABLE STOR FIXED > 88%

OR

AVE STOR FIXED OR PAGE I/O > 72%

OR

AVE ASM QUEUE LENGTH > 1000

OR

PAGE FAULT RATE > 1000/SEC

OR

AVE PAGE SERVICE TIME > 1000 MS

These tests are mirror images of the tests from the MPL increase foil. 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 the environments we are in today. 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.

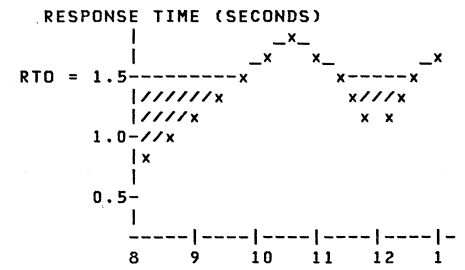
TSO RESPONSE TIME OBJECTIVE (RTO)

- FIRST PERIOD AVERAGE RESPONSE
- BENEFITS:
  - _ EASIER TO CODE THAN AOBJ
  - _ MORE CONSISTENT RESPONSE TIME
  - _ RESERVE CAPACITY
- DOMAIN DATA
  - _ COLLECTED AT TRANSACTION END
  - _ AVERAGED EVERY 20 SECS.
  - _ HISTORY 1:1
  - _ TRACE WITH RMF

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.

End users demand consistent response time. They also demand very good response time. I personally believe that they deserve excellent response time. Studies have shown dramatic improvements in productivity with subsecond response times. So I must admit that I feel a bit sheepish telling you about a function that slows down response time. I do believe that consistent response times are very important, though. If you do not have the capacity to deliver 300 millisecond response times yet, you may want to deliver a consistent one second response time. The choice is yours but there have been questions about the function which I will try to address.

TSO RESPONSE TIME OBJECTIVE



|////////| = SWAP IN DELAY

xxxxxxxxxx = RESPONSE TIME WITHOUT RTO

---------- = ACTUAL RESPONSE TIME

This foil shows how the function works. 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 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

170

You 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. No delay is imposed for second and subsequent transactions once the address space is swapped in either.

OBJECTIVES

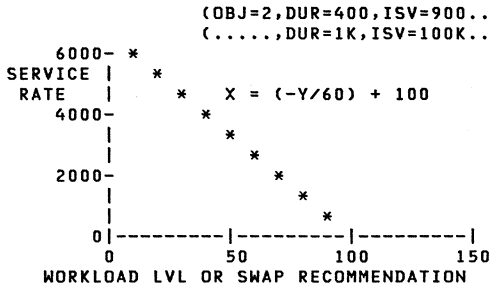
" WHAT ARE THEY ANYWAY "

- FUNCTION WITH 2 VARIABLES
  - _ Y = X
  - _ Y = -60X + 6000
  - _ X = (-Y/60) + 100
- _ MAPS A SERVICE RATE INTO A NUMBER
  - _ HIGHER SERVICE RATE - LOWER NUMBER
  - _ LOWER NUMBER - LESS PRIORITY
  - _ NUMBERS CAN BE BASIS FOR DECISIONS TO IMPLEMENT POLICY

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. I have shown 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. We tend to always think that X must be the independent variable. We 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



- WHEN IN ISV THE CUTOFF WLL IS USED
- WORKLOAD LVLS MUST DIFFER BY >= 1
  FOR EXCHANGE SWAP
- LARGE ISV PREVENTS EXCHANGE SWAPS
- LET ISV DEFAULT TO 100K IN GENERAL

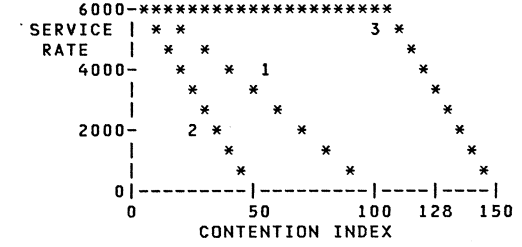
Now that I have explained what an objective is, I will explain how they are used to get swap recommendation values and how the theory is compromised for the better by something called the interval service value or ISV. I have shown the graph for the last function formula from the previous foil. Swap 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 our 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



DMN 1,...DOBJ=1

DMN 2,...DOBJ=2

DMN 3,...AOBJ=3

DMN 4,...FWKL=128

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 my 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. SRM attemps 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.

SUMMARY

- _ SRM TIMING
- _ TSO TRANSACTIONS
- _ STORAGE CONTROL
- _ MPL ADJUSTMENT
- _ SWAP CONTROL

Objectives were my final item under swap control. You can see that I saved the best for last. I am confident that everyone finally understands objectives now. There are other areas of the SRM that could have been topics in this presentation. The load balancers certainly are not well understood for example. However, there does not seem to be much interest in them and I felt the time could be used more effectively dealing with the areas that are making decisions that affect your work on a daily basis. As I

said at the beginning of the presentation, session B181 will describe the changes in SRM for MVS/XA tomorrow at 8:30.

Timing, storage control, MPL control and swap control are the areas I get asked about the most here at SHARE and through other forums. I hope that I have been able to further your understanding of the System Resource Manager.