# CENTRAL STORAGE MANAGEMENT WITH MVS/ESA

## Donald R. Deese

*Beginning with MVS/ESA SP3.1.3, IBM redesigned the logical swapping algorithms, integrated the page stealing process with logical swapping, and revised the algorithms controlling processor storage (central storage and expanded storage). This logic has been enhanced with MVS/ESA SP4.2 and SP4.3. A major goal of the changed logic is to take advantage of increasingly large amounts of processor storage and to minimize page and swap I/O operations. This paper describes the concepts and algorithms used to manage central storage with MVS/ESA and highlights major changes between MVS/XA and MVS/ESA.*

## 1.0 INTRODUCTION

This introduction provides a brief overview of the terms and concepts used by the System Resources Manager (SRM) and the Real Storage Manager (RSM) to manage the use of processor storage. A brief definition of these terms is provided so the more detailed explanations have a basic framework:

- **Processor storage** consists of (1) **central** storage and (2) **expanded** storage. Central storage is very high speed storage and is addressed at an individual byte level. Expanded storage is relatively lower speed storage and is addressed only at a page level (4096 bytes).

- **Auxiliary storage** is the storage represented by direct access storage devices (DASD). Auxiliary storage is much lower speed than is expanded storage and is addressed only by I/O operations transferring a page or several pages at a time.

- **Address space** is a term used to described a logical entity which is managed by MVS (a batch job, a TSO user, a started task, a CICS region, etc.).

- **Physical swap-out** moves an address space out of central storage to either expanded storage or to auxiliary storage.

- **Logical swapping** keeps an address space in central storage rather than physically swapping out the address space, but the SRM adjusts control blocks to indicate that the address space is in a "swapped out" status.

- **Multiprogramming set** is the set of all address spaces which are in a "swapped in" status.

- A **page** consists of 4096 contiguous bytes of program code or data. A page may reside in processor storage or reside in auxiliary storage.

- A **page frame** is a 4096-byte block of central or expanded storage. The page frame may hold a page of program code or data, or the page frame may be unused.

- **Page stealing** steals page frames from the Common area or from swapped-in address spaces.

- **Swap trim** removes page frames from an address space which is to be logically or physically swapped out.

- **Migration** moves pages from expanded storage, through central storage, to auxiliary storage. Migration is performed by the **Migrator** component of the RSM.

Programs and data may reside in central storage, expanded storage, or auxiliary storage. Not all instructions nor all data are required to be in central storage in order for a program to execute. Much of a program's instructions or its data may be in "virtual" storage and may reside in expanded storage or reside in auxiliary storage, rather than reside in central storage.

However, programs can execute instructions only if the instructions reside in central storage and instructions can reference data only if the data resides in central storage.

The main reason for different levels of storage is financial; central storage is very expensive, expanded storage is less expensive, and auxiliary storage is least expensive per megabyte of storage.

Once an address space is swapped into central storage and is executing, the amount of central storage used to support the address space may vary. The variance is based upon the size of the virtual storage associated with the address space, the amount of central storage available on the computer system, the central storage requirements of other address spaces executing concurrently, the addressing characteristics of the address space, and specifications contained in the IEAIPSxx and IEAOPTxx members of SYS1.PARMLIB.

The MVS virtual storage environment operates on the principles that:

- The central storage required by any particular address space during execution is a subset of the total central storage necessary to load the address space initially

into central storage. Much of the storage required to load an individual address space is often unused.

- The central storage that **is** regularly used is referred to as the "working set" of the address space. The working set is typically a small part of the overall central storage requirement to initially load an address space. The remaining (typically large) amount of central storage can be used by other address spaces which are loaded concurrently.

- Idle central storage should be used to prevent unnecessary I/O operations. In fact, central storage generally should be managed to maximize the use of central storage while minimizing I/O operations.

- Central storage can be "allocated" to address spaces based upon the importance of the address space. That is, the "working set" of a low priority address space can be constrained if necessary to allow more important address spaces to have adequate central storage.

- With appropriate external storage and process controls, users of a virtual environment should notice little difference between the performance of the virtual environment and the performance of a non-virtual environment.

MVS exchanges pages belonging to address spaces between central storage and auxiliary storage (and between central storage and expanded storage if expanded storage is installed). In this way, MVS allows multiple address spaces to concurrently use a finite amount of central storage.

When an address space references a page of storage which is not addressable in central storage, a "page fault" occurs[1]. The address space is unable to continue processing until the page fault is resolved. MVS will locate the page and make it available to the address space.

- The page might be in central storage awaiting a page-out operation. These pages are "reclaimed" and made available without further effort. Delays for these page fault resolutions are inconsequential.

- The page may be in expanded storage (for systems with expanded storage). These pages are moved

directly from expanded storage; the page-in time is very small (various studies have reported page-in times from expanded storage in the range of 35-75 microseconds). Delays for these page fault resolutions do not generally cause a performance problem, although system performance will suffer if the page fault rate is high.

- If the page is not in central or expanded storage, the page must be physically brought in from auxiliary storage. Delays for these page fault resolutions can range from about 2 milliseconds to over 100 milliseconds, depending upon the local page data set configuration and delays in the paging I/O subsystem.

If the page is in expanded storage or in auxiliary storage, a page frame in central storage must be available to hold the page being paged in. If insufficient page frames are free, the SRM will acquire central storage page frames. Later sections of this paper describe the techniques used to make central page frames available.

## 2.0 UNREFERENCED INTERVAL COUNT

As mentioned earlier, not all instructions nor all data are required to be in central storage in order for a program to execute. This is because programs may have routines which are executed infrequently (for example, those routines which may perform initialization functions), or which reference tables or arrays infrequently.

The SRM and RSM work together to manage the central storage requirements of address spaces. A primary way of managing central storage requirements is to identify those pages which are no longer required by address spaces as the address spaces execute.

When a program executes code or when it references data, the central storage page in which the code or data reside is marked as having been "referenced" and a hardware bit associated with the page frame is set ON[2].

The SRM periodically requests the RSM to test whether central storage frames associated with one or more address spaces or the Common area have been referenced. Conceptually, this request is made every second. The SRM makes its request less often, depending upon central storage constraints.

When requested by the SRM, the RSM checks each page associated with the address spaces (or Common area) provided by the SRM. If the page has been referenced since the last time the RSM made its test (or if there is I/O pending for the frame), the RSM clears a counter associated with the page. If the page has not been referenced since the last update interval, the RSM increments the counter.

---

[1] Taking their cue from the Mad Hatter (_Alice in Wonderland_, Carroll, Lewis M.), the designers of the SRM and the authors of the _Initialization and Tuning Guides/References_ "redefine" page fault to mean exactly what they say it is: currently, a non-VIO non-swap page-in from auxiliary storage. Page faults resolved from expanded storage are ignored by their "definition" of page faults.

Additionally, page faults resolved through reclaims of pages on their way out of central storage were included in the "definition" with MVS/XA, but were dropped from the "definition" with MVS/ESA. The rest of MVS (e.g., the Real Storage Manager) properly uses the term page faults.

---

[2] The hardware also sets a bit ON to indicate whether the page frame was changed or unchanged by the reference (that is, whether the program altered the contents of the frame).

The counter contains the count of times the RSM has made its test and the page has remained unreferenced. This count is termed the **Unreferenced Interval Count (UIC)** and has a maximum value of 255. The UIC is an important metric in many of the SRM's algorithms.

Each time the SRM requests the RSM to update the UIC for address spaces or the Common area, it examines the highest UIC for any address space or the Common area.

The highest UIC is compared with the previous high UIC, and the highest of the current or previous UIC is retained. This highest UIC is termed the **system high UIC**.

The test for highest UIC is not necessarily made for every address space or the Common area. If storage isolation is in effect for an address space or for the Common area, the storage-isolated address space or Common area are excluded from the test for highest UIC. (Storage isolation is explained later in this paper.)

Additionally, with SP4.2, address spaces monitored by the Working Set Manager may be excluded from the test for highest UIC (depending upon their status). Consequently, the system high UIC may not reflect the highest UIC of any page in central storage, but will reflect only those pages which can be stolen by the SRM's page-steal algorithms (these algorithms are described later).

Each SRM second[3], the SRM adds the system high UIC to a counter and counts the number of such additions. The SRM periodically divides the sum of the system high UICs by the count of the times the sum was incremented. The result of this division is termed the **average system high UIC**. Both the system high UIC and the average system high UIC are used to guide many of the SRM's algorithms.

If the SRM requests the RSM to perform its tests once per second, the UIC count would represent the number of seconds since a page had been unreferenced. However, processor resources are required for the RSM to make its test of each page associated with address spaces and the Common area. Further, as the length of unreferenced time gets large, the algorithms which use the UIC are not particularly enhanced by precise information about how long a page has remained unreferenced.

For example, it is unimportant to the algorithms whether a page has gone unreferenced for 160 versus 161 seconds. On the other hand, if the system high UIC is low (for example, if it is less than 5), it becomes important to provide increasingly precise information about how long a page has remained unreferenced.

To minimize the processor time required to update the UICs, the SRM does not usually request the RSM to update UICs every second. The SRM uses the current system high UIC as an indicator about how long pages have gone unreferenced. If the current system high UIC is small, the SRM requests the RSM to update the UIC frequently; as the current system high UIC becomes larger, the SRM makes less frequent requests.

The SRM schedules the UIC update differently for swappable address spaces versus non-swappable address spaces and the Common area, and schedules the UIC update differently for pre-SP4.2 versus SP4.2 systems. The basic algorithms are the same, but the constants controlling the algorithms change. In simplified form, the algorithm is:

- **If system high UIC is less than lower limit, then update UIC every second**

- **If system high UIC is greater than or equal to the lower limit, then apply formula.**

  **The formula is quite simple: if the system high UIC increases by "delta" (described below), increase UIC update interval by one second.**

- **Regardless of formula results, establish a maximum interval for the UIC update**

For pre-SP4.2, the lower limit for the system high UIC before applying the formula is 10 seconds for swappables and is 6 seconds for non-swappables and common. The "delta" for increasing the update interval is 10 seconds for swappables and is 6 seconds for non-swappables and common. The maximum interval between updates (regardless of the formula results) is 6 seconds for swappables and is 20 seconds for non-swappables and common.

With SP4.2, the lower limit for the system high UIC before applying the formula is 5 seconds for swappables and is 3 seconds for non-swappables and common. The "delta" for increasing the update interval is 5 seconds for swappables and is 3 seconds for non-swappables and common. The maximum interval between updates (regardless of the formula results) is 10 seconds for swappables and is 20 seconds for non-swappables and common.

The changes with SP4.2 will normally allow for a less-frequent updating of the UIC and correspondingly less overhead. These changes were made in response to

---

[3]The "SRM second" is based on the value specified for the RMPTTOM keyword of IEAOPTxx, as adjusted by processor speed. The *Initialization and Tuning Guides* provide the relationship between the SRM seconds to wall clock time. This number of SRM seconds per second of wall clock typically ranges from 27.6757 (for an IBM 3090 J-series) to 70.1754 for an IBM ES/9000 (Model 9021). Depending upon the algorithms involved, the SRM second may be further adjusted by the number of processors.

increasingly large amounts of central storage available with newer computer systems.

### 3.0 PHYSICAL SWAPPING

Address spaces may be swapped into and out of central storage. The original purpose of swapping was to allow system resources to be used by other Ready users when some address space (such as a TSO user) was unable to execute because it was waiting for some event (such as a terminal user to press the ENTER key). The waiting address space would be swapped out to auxiliary storage.

These swaps can be viewed as "voluntary" swaps, in that the address space is swapped out when it enters a wait condition and it temporarily cannot use system resources.

An additional refinement to the concept of swapping involves "involuntary" swaps, in which address spaces are swapped out even though they are ready to execute. Ready address spaces may be swapped out because installation management has decided to restrict the amount of system resources used by the particular type of address space, or because the SRM has decided the computer system is becoming over-utilized.

For example, batch jobs may be involuntarily swapped out if the total resources consumed by all batch jobs (or even the average resources consumed by batch jobs) exceed installation criteria.

Address spaces may be swapped out from central storage to auxiliary storage or to expanded storage. Swaps to expanded storage are done if sufficient expanded storage is available to contain the swap working set and if the ESCTSWWS controls in IEAOPTxx permit the swap to go to expanded storage.

### 4.0 LOGICAL SWAPPING

As larger amounts of processor storage became available, the concept and purpose of swapping address spaces was changed. Address spaces are no longer necessarily swapped out of central storage when the address space is waiting. If sufficient central storage is available, the waiting address space is simply "quiesced" and various control blocks are updated to reflect the "swapped out" status.

The address space remains in central storage until (1) central storage frames are needed or (2) the address space becomes Ready. If the address space becomes Ready before central storage frames are needed, a physical swap out is not required. This process of leaving a "swapped-out" address space in central storage is called a "logical swap" and is used to prevent unnecessary physical movement of pages.

Logical swapping relies on several concepts:

- Physical swaps require a relatively lengthy time, and they place a burden on the processor (and on the I/O subsystem if directed to auxiliary storage).

- Address spaces recently entering a wait state may soon become ready to execute. If possible, these address spaces should not be physically swapped out.

- Address spaces may be swapped out for reasons other than entering the wait state (e.g., they may be swapped out because the SRM has lowered the system target multiprogramming level). These address spaces are ready to execute, and they will be swapped in whenever the SRM decides to add them back into the multiprogramming set.

- Often, not all central storage is required to support the working set of address spaces in the current multiprogramming set. If central storage is not required to support the working set of address spaces, the storage can better be used to retain "swapped out" address spaces in central storage.

Logical swapping responds to these concepts by attempting to minimize the number of physical swaps. This is accomplished by using central storage to hold "swapped out" address spaces. Eligible address spaces are quiesced and removed from the dispatcher's True Ready Queue. These address spaces are not immediately swapped out to auxiliary storage or to expanded storage. They are swapped out of central storage only when central storage is needed.

Logical swapping is a balance among:

- The probability that an inactive address space will soon become ready.

- The amount of memory required to maintain the inactive address space in central storage.

- The amount of time required to physically swap an address space.

- The time and resources required to resolve page faults.

- The number of pages stolen from swapped-in address spaces.

Logical swapping originally applied only to waiting address spaces (e.g., a TSO address space waiting for a user to press ENTER on a terminal). Beginning with MVS/ESA SP3.1.3, **all** swaps (except for Transition swaps, Requested swaps, and Storage Shortage swaps) are eligible for logical swapping if sufficient central storage is available when the SRM decides whether to physically swap out an address space or logically swap the address space. However, the SRM algorithms differentiate between address spaces which are swapped out in **wait state** versus address spaces which are swapped out in a **non-wait or ready state**.

- Address spaces which are swapped out in a wait state are those which are waiting for terminal I/O (Terminal Input swaps or Terminal Output swaps), those which the SRM has detected have not executed for some period of time (Detected Wait swaps), those which have issued a WAIT macro with the LONG parameter (Long Wait swaps), and those which are waiting for Advanced Program to Program Communication services (APPC Wait swaps).

    Address spaces eligible for wait swaps have a special indicator set in the SRM's control blocks. These address spaces will not normally[4] be physically swapped out until they have remained logically swapped out for some elapsed time. This elapsed time is called the "think time" of the address space and is compared to the current "system think time".

    Both of these "think time" concepts are discussed in more detail later. Wait state swaps are treated uniquely by many of the SRM's algorithms.

- Address spaces which are swapped out in a non-wait or ready state are those address spaces which were swapped out because the SRM decided to remove them from the system's multiprogramming set and deny them access to system resources. This typically happens because the SRM has lowered the system target MPL, or because the number of ready users in some domain exceeded the target MPL for the domain.

    The non-wait (or involuntary) swaps are Unilateral swaps, Exchange on Recommendation Value swaps, and Enqueue Exchange swaps[5]. With SP4.2, these swaps also include Improve System Paging Rate swaps, Improve Central Storage Usage swaps, and Make Room swaps.

    Address spaces which are removed from the multiprogramming set in a non-wait or ready state are physically swapped out of central storage only when central storage is needed. So long as central storage is plentiful, these address spaces are kept in central storage.

- An address space could initially be swapped out for a wait state swap, but could subsequently become ready to execute (e.g., a Detected Wait swap could become ready).

    If this ready address space has been logically swapped out longer than the **current** "system think time", the SRM treats it as though it had been originally swapped out as a ready user (that is, the SRM will physically swap it out whenever central storage is needed).

    On the other hand, if the address space has not been logically swapped out longer than the **current** "system think time", the SRM will leave it logically swapped and add it to the multiprogramming set (a "swap-in" for the address space) when system conditions permit.

    The word "**current**" is placed in bold in the preceding paragraphs to emphasize that the system think time is periodically adjusted; the comparisons deal with the current system think time. One implication of this is that if the system think time should be decreased, logically swapped wait state address spaces will not be "protected" as long based on the think time.

For example, suppose a wait state address space had been logically swapped for 10 seconds and the system think time was 12 seconds. The address space would be "protected" by its think time, since the think time was less than the system think time.

A second later, the address space would be logically swapped for 11 seconds. During this second, the system think time could be lowered by 1 second, for a new system think time of 11 seconds. The address space would no longer be "protected" by its think time.

The address space could be converted from a logical swap to a physical swap **if central storage was needed** by the SRM.

There are two important concepts in the above paragraphs: (1) address spaces logically swapped for wait state status will remain logically swapped until their "think time" exceeds the current "system think time" and (2) ready address spaces will remain in central storage until the central storage is needed.

## 4.1 ADDRESS SPACE THINK TIME

The SRM associates a "previous think time" and a "current think time" with each address space which is swapped out for a wait state. The purpose of the "think time" is (1) to provide a guess as to when the address space will become ready to execute and (2) determine how long the address space currently has been logically swapped.

- The **previous** address space think time is computed differently, depending upon whether the address space is in a terminal wait state or in some other type of wait state.

---

[4] The word "normally" is used throughout the text to describe normal actions in normal operating situations. MVS will take abnormal actions in response to abnormal situations. For example, many abnormal actions will be taken if there is a shortage of fixed storage below 16 megabytes. These actions may include "shutting off" expanded storage, "shutting off" logical swapping, overriding storage isolation, etc.

This paper would become large and cumbersome if all abnormal actions were discussed in each situation. Thus, only "normal" actions and important exceptions are typically described.

[5] Non-wait swaps also include Transition swaps, Requested swaps, and Storage Shortage swaps. However, these swaps are not eligible for logical swapping.

- For terminal wait swaps, the SRM keeps track of how much time lapsed during the previous wait swap for the address space. The SRM uses the previous wait swap as an estimate of how long the terminal users will wait before again interacting with the computer.

- For other types of wait swaps (e.g., Detected Wait swaps), the SRM uses an artificial "think time" of 5 seconds as the previous think time. This value is simply an arbitrary constant which estimates how long an address space might wait for some event to occur.

This previous think time concept depends upon fairly regular behavior of a terminal user. The previous think times of a terminal user may or may not be an indicator of how long the user will "think" before again interacting with the computer.

- The **current** address space think time is simply how long the address space has been logically swapped. The SRM computes the current address space think time as the current time of day less the time the address space was logically swapped. The SRM compares the current address space think time with the current "system think time" as it makes decisions about whether an address space is logically swapped based on wait state. These decisions are described in more detail below.

## 4.2 SYSTEM THINK TIME

The SRM maintains a "system think time" which is used as a basic control mechanism in determining whether address spaces swapped out in a wait state receive special consideration.

The current "system think time" is an attempt to weigh the advantages of minimizing physical swaps against the costs of keeping "swapped out" address spaces in central storage.

From one perspective, "system think time" is an inappropriate name for the measure; the current system think time is simply a metric which is intended to show whether central storage **generally** is abundant or constrained.

- If central storage **generally** is abundant, "swapped out" address spaces should be retained in central storage rather than physically swapped out to auxiliary storage or to expanded storage.

- If central storage **generally** is constrained, "swapped out" address spaces should be physically swapped out of central storage.

Unfortunately, simply examining the number of central storage frames unassigned to ready address spaces is insufficient to determine whether central storage generally is abundant or constrained.

For example, ready address spaces might hold central storage frames but not use all the storage for long periods. These "old" pages might be better used to hold swapped out address spaces, even though the frames were assigned to ready address spaces.

As another example, an address space might complete (e.g., a batch job end) and free large amounts of central storage. This central storage might be available for a short time but another address space might start and require the central storage just freed.

In either of these examples, it would be improper to conclude that central storage was constrained or abundant based on a single metric or a single observation of the status of central storage.

The SRM uses the current system think time in two main situations: (1) upon swap-out to decide whether to turn on control bits indicating that the address space is logically swapped by think time and (2) when examining logically swapped address spaces to decide whether an address space logically swapped because of wait state is still eligible for think time protection.

- During Quiesce processing[6], the SRM tests whether a wait state address space should have its OUCBLSW (the "logical swap" bit) set ON.

This test differs depending upon whether the address space has entered a terminal wait state or has entered some other wait state (e.g., Detected Wait). .

- For terminal wait state, the OUCBLSW bit is turned ON if the address space's think time (the maximum of the current and previous think time) is less than the current system think time. A simplified version the algorithm for terminal wait address spaces is:

```
IF
   (USER THINK TIME < SYSTEM THINK TIME)
THEN
   OUCBLSW = 1
```

For example, suppose the default values were specified with IEAOPTxx and the current system think was six seconds. The OUCBLSW bit would be turned ON for an address space in a Terminal Wait if the address space's think time was less than 6 seconds.

- For other types of wait state, the OUCBLSW bit is turned ON if the current system think time is greater than 5 seconds and the average percent of fixed frame below 16 megs is less than or equal to the LSCTFET high value. A simplified version of the

---

[6] Please note that the Quiesce Complete processing has been moved to OCO code with SP4.2. Consequently, the author is unable to determine whether the algorithms have changed

algorithm for address space in other types of wait state is:

```
IF
    (SYSTEM THINK TIME > 5 SECONDS)
AND
    (PERCENT FIXED BELOW 16MEG <=LSCTFETH)
THEN
    OUCBLSW = 1
```

For example, suppose the default values were specified with IEAOPTxx and the current system think was six seconds. The OUCBLSW bit would be turned ON for an address space in other types of Wait if the system think time was greater than 5 seconds and if not more than 82 percent of the frames below 16 megabytes were fixed.

The design attempts to logically swap wait state address spaces unless central storage is seriously constrained.

- When the SRM is attempting to acquire enough frames to reach the Logical Swap Available Frame Queue target (described below), the SRM decides whether to physically swap address spaces which are logically swapped.

The algorithms use the current system think time to decide whether an address space logically swapped because of wait state (e.g., terminal wait) is still eligible for think time protection. These address spaces will be trimmed and swapped out only if they have been logically swapped for longer than the current system think time.

### 4.2.1 System Think Time Adjustment

The system think time is adjusted based on a combination of LSCTxxxx controls contained in the IEAOPTxx member of SYS1.PARMLIB, and based on tests which the SRM makes independent of the LSCTxxxx controls.

The system think time is **increased** when the SRM concludes that central storage is not a serious constraint. The system think time is increased by 0.5 seconds under the following conditions:

The average system high UIC is greater than the LSCTUCT high value, indicating that central storage is not a serious constraint.

**OR** the average Available Frame Queue is greater than the LSCTAFQ high value[7] (for MVS/XA) **OR** the average Available Frame Queue is greater than the Logical Swap Frame Queue Target (beginning with MVS/ESA), indicating that there are plenty of central storage frames available.

**AND** all the following conditions are true:

- The system target MPL has not decreased because of storage constraints since the last time the SRM attempted to adjust the system controls.

- There is no current shortage of central storage frames.

- The percent of fixed storage is less than the LSCTFET low value.

- The percent of fixed storage or storage allocated for paging I/O is less than the LSCTFTT low value.

For example, suppose the default values were specified with IEAOPTxx. The system think time would be **increased** if the average system high UIC was greater than 30 or an average of at least 150 central storage frames had been available over the previous adjustment interval, and if the system target MPL had not been decreased because of storage constraints, there was no current shortage of frames, less than 76 percent of central storage frames were fixed, and less than 58 percent of central storage frames were allocated for paging I/O.

The system think time is **decreased** when the SRM concludes that central storage is a constraint. The system think time is decreased by 1 second (or decreased by the maximum of 1 second or 25% of the current system think time with SP3.1.3) under the following conditions:

The average system high UIC is less than the LSCTUCT low value, indicating that central storage is a constraint.

**OR** the average Available Frame Queue is less than the LSCTAFQ low value[7] (for MVS/XA), or the average Available Frame Queue is less than the Logical Swap Frame Queue Target (beginning with MVS/ESA), indicating that there is an inadequate supply of central storage frames available.

**OR** the percent of fixed storage is greater than the LSCTFET high value.

**OR** the percent of fixed storage or storage allocated for paging I/O is greater than the LSCTFTT high value.

For example, suppose the default values were specified with IEAOPTxx. The system think time would be **decreased** if the average system high UIC was less than 20 or an average of less than 150 central storage frames had been available over the previous adjustment interval, or if the system target MPL had been decreased because of storage constraints, there was a current shortage of frames, more than 82 percent

---

[7]The LSCTAFQ keyword was dropped with SP3.1.3 and the SRM tests the Available Frame Queue count against a Logical Swap Frame Queue Target. The Logical Swap Frame Queue Target is described in more detail later.

of central storage frames were fixed, or more than 66 percent of central storage frames were allocated for paging I/O.

The system think time is intended to be a single indicator of whether central storage generally is a constraint to performance. If central storage is not a constraint, the system think time is increased. If central storage is a constraint, the system think time is decreased. The current system think time therefore indicates whether central storage has been constrained or abundant over the previous system think time adjustment intervals.

The increase and decrease of the system think time do not happen immediately when storage becomes plentiful or storage becomes a constraint. The adjustment occurs only when the SRM's Resource Monitor 2 (IRARMRM2) code is executed[8].

The frequency with which the IRARMRM2 routine is executed is based on the number of processors and the speed of the processors. Regardless of the speed or number of processors, the IRARMRM2 routine is executed no more frequently than once every 5 seconds (pre-SP4.2) or once every 2 seconds (with SP4.2). Thus, the system think time is adjusted no more frequently than once per 5 seconds (pre-SP4.2) or once per 2 seconds (with SP4.2).

### 4.2.2 System Think Time Limits

The **frequency** with which the system think time is adjusted cannot be controlled by system performance analysts. However, **limits** can be set on the minimum and maximum values for the system think time. These limits are imposed using the LSCTMTE keyword in IEAOPTxx. The default minimum system think time is 0 seconds. The default maximum system think time is 30 seconds for MVS/XA and was lowered to 5 seconds for MVS/ESA.

Some authors have suggested a low LSCTMTE value greater than zero or a relatively high LSCTMTE value. Both these suggestions are ill-advised. The authors may have missed the crucial points that, (1) since SP3.1.3, the SRM will leave address spaces logically swapped **until central storage frames are needed** and (2) when central storage frames are needed, **the frames must come from somewhere** (the SRM will either trim pages from logically swapped address spaces, or will steal pages from address spaces which are swapped in).

*   If a minimum system think time of greater than zero were specified, the SRM will retain logically swapped address spaces in central storage until

they have been logically swapped for the minimum specification. The address spaces will be retained in central storage, even though central storage is constrained and the SRM is forced to steal pages from active address spaces. The result can be severe performance problems and "thrashing" of page movement as pages are stolen from active address spaces.

*   If a large maximum system think time were specified, the SRM may be unable to respond to a change in the central storage demands of the workload. The system think time could increase to the maximum during periods of relatively light demand for central storage.

If the environment changes such that there is a large demand for central storage, the SRM will begin lowering the system think time. As described above, each adjustment of the system think time occurs only every Resource Monitor 2 interval. This RM2 interval normally occurs no more often than once per 5 seconds (pre-SP4.2) or once per 2 seconds (with SP4.2).

Prior to SP3.1.3, the decrease in the system think time was only 1 second per adjustment and this adjustment occurred only once per 5 seconds. If the maximum system time were specified as 60 seconds (for example), it would take 5 * 60 or 300 seconds before logical swapping could be turned off. Existing logically swapped address spaces could remain in central storage for an average of 150 seconds before being pushed out. The existing logically swapped address could remain in central storage even though central storage was constrained. The result could be extensive page stealing from active address spaces.

The SRM designers responded to this potential performance problem by modifying the system think time adjustment algorithms. With SP3.1.3, the decrease in system think time is either 1 second or 25% of the current system think time. The changed algorithm allows the SRM to more quickly reduce the system think time if central storage becomes constrained.

With SP4.2, the RM2 interval changed from a maximum of once per 5 seconds to a maximum of once per 2 seconds. Consequently, the system think time can decrease even more quickly with SP4.2, and the high value specified for the LSCTMTE keyword becomes less important.

### 5.0 AVAILABLE FRAME QUEUE

The MVS environment normally requires central storage frames under two conditions: (1) when address spaces are swapped into central storage and (2) when a page fault occurs.

*   When the SRM physically swaps in an address space, the SRM normally acquires enough central storage

---

[8] With SP4.2, this adjustment has been moved to the Resource Monitor 3 (IRARMRM3) routine, which is OCO code. However, the IRARMRM3 routine is invoked by the IRARMRM2 routine. The frequency of execution is the same for both routines.

frames to hold the working set of the address space. Sufficient central storage frames must be available to hold the working set.

- When an address space references a page which is not addressable in central storage, a "page fault" occurs. MVS must locate the page and make it available to the address space. If the page is in expanded storage or in auxiliary storage, a page frame in central storage must be available to hold the page being paged in.

  With SP4.2, additional page frames might be required if the page is part of a block of pages (block paging is described later). If a block had been formed when the page was moved out of central storage, multiple pages might be required when a page fault occurs. If the block resides in auxiliary storage, the remainder of the block will be brought into central storage. If the block resides in expanded storage, the remainder of the block will be brought into central if the Working Set Manager has turned on implicit block-in for the address space.

In either of these cases, the RSM acquires central storage frames from the Available Frame Queue (AFQ).

When address spaces release frames (for example, the address space terminates) or when the SRM physically swaps out an address space, the central storage frames held by the address space are added to the Available Frame Queue. As will be discussed below, these normal events may not provide enough frames to handle the workload's requirement for central storage. The SRM must take action to make frames available if there is a shortage of frames on the Available Frame Queue.

## 5.1 DETECTING AFQ SHORTAGE

The RSM keeps a count of the number of frames on the Available Frame Queue; the count is decreased when frames are removed from the Available Frame Queue and is increased when frames are added to the Available Frame Queue. This count (the RCEAFC variable) is used in most of the SRM's algorithms as it manages central storage.

The SRM attempts to keep enough central storage frames available to allow logical swapping, and to provide enough free central storage frames to allow speedy resolution of page faults. The SRM accomplishes this by comparing the RCEAFC count with indicators of whether adequate central storage frames are available.

The SRM uses two main indicators of whether central storage **instantaneously** is constrained or is not constrained. These are the RCEAFCLO and RCEAFCOK variables, which indicate "low" and "OK" thresholds of available page frames. When the count of available central storage frames is less than the

RCEAFCLO threshold, the SRM will schedule algorithms to "replenish" the Available Frame Queue. The replenishment algorithms will stop when the count of available central storage frames is greater than the RCEAFCOK threshold.

The RCEAFCLO and RCEAFCOK variables initially are based on the MCCAFCLO and MCCAFCOK values. The MCCAFCLO and MCCAFCOK values are constants for pre-SP4.2 and have values of 20 and 40, respectively. The MCCAFCLO and MCCAFCOK values are parameters in IEAOPTxx with SP4.2 (the low value and high value of the MCCAFCTH keyword) and have default values of 50 and 100, respectively.

The RCEAFCLO and RCEAFCOK thresholds are dynamically adjusted based on system conditions.

- In an environment without expanded storage, the initial RCEAFCLO and RCEAFCOK values can be adjusted based upon how often central storage becomes constrained. The RCEAFCLO and RCEAFCOK values can be increased when central storage is constrained. These values subsequently can be lowered if central storage is no longer constrained, but they will not be lowered below their initial values.

- The SRM can temporarily raise the RCEAFCLO and RCEAFCOK thresholds if necessary to make available more processor storage to contain the working set of a swapped-in address space. Once the address space has been swapped into central storage, the thresholds will be lowered by the amounts they were raised.

## 5.2 SCHEDULING AND REPLENISHMENT

The SRM's Available Frame Queue replenishment algorithms are controlled by the IRARMMS2 routine, which will be scheduled by the RSM or by other routines in the SRM.

- The IRARMMS2 routine is scheduled by the RSM whenever the RSM detects a shortage of available central storage frames, during normal page fault processing. The RSM detects that there is a shortage of available central storage frames when the count of free central storage frames (the value in the RCEAFC variable) falls below the RCEAFCLO threshold.

- The IRARMMS2 routine normally is scheduled (1) by the SRM when the SRM detects that a shortage exists of fixed frames or fixed frames below 16 megabytes, (2) by Quiesce processing as an address space is to be swapped out, (3) during swap-in processing if the address space which the SRM is trying to swap in will not fit into processor storage, and (4) at periodic time intervals.

  - The SRM periodically checks for a shortage of fixed frames below 16 megabytes, fixed frames in all of central storage, and the sum of fixed and Disabled

Reference (DREF) frames. IRARMMS2 is scheduled if a shortage exists in any of these areas.

- Quiesce processing occurs when the SRM is swapping out an address space. Quiesce processing schedules the IRARMMS2 routine to complete the swap-out processing for certain swaps (Request Swaps, Storage Shortage swaps, and Transition Swaps) or to perform preliminary swap trim for wait state swaps which are to be logically swapped.

- During swap-in processing, the SRM will determine whether an address space to be swapped in from auxiliary storage will fit into processor storage. If the SRM wishes to swap in an address space from auxiliary storage and the address space will not fit into processor storage, the SRM will temporarily raise the storage thresholds and schedule the IRARMMS2 routine.

  For example, suppose the SRM wishes to swap in an address space from auxiliary storage and the address space has 400 pages in the working set. If swapping in the 400 frames would reduce the count of available processor storage frames below the low thresholds, the SRM will make room for the address space.

  The SRM makes processor storage available by temporarily raising the central storage and expanded storage thresholds. The SRM will reschedule the swap-in after processor storage is available.

- The IRARMMS2 routine is scheduled at periodic time intervals (every SRM second, as adjusted for processor speed).

The IRARMMS2 routine computes a "Logical Swap Target" for the Available Frame Queue. The Logical Swap Target is the number of frames which the SRM will attempt to keep available so that logical swapping can be accomplished for an address space swapped out for wait state. The target is normally the **maximum** of (1) three times the MCCAFCLO low value or (2) the current RCEAFCOK value.

- If three times the MCCAFCLO variable is greater, the Logical Swap Available Frame Queue target is 60 frames for pre-SP4.2 (20 * 3 = 60) and the Logical Swap Available Frame Queue target is 150 frames for SP4.2 (50 * 3 = 150). These are the minimum values for the Logical Swap Available Frame Queue target. The values were increased with SP4.2 to better accommodate the typical working set of a TSO user.

- For pre-SP4.2, the initial RCEAFCOK value is 40 frames. With SP4.2, the initial values are based on the high value of the MCCAFCTH keyword in IEAOPTxx, which has a default value of 100. The RCEAFCOK threshold is dynamically adjusted as described above, so the RCEAFCOK threshold can become higher than three times the MCCAFCLO value.

The RCEAFCLO and RCEAFCOK thresholds will be adjusted by the SRM as it attempts to swap in an address space, if the address space does not fit. Depending upon how large the address space is, the adjusted RCEAFCOK threshold could be significantly larger than three times the MCCAFCLO value. Consequently, target could be larger than shown above.

## 5.3 AFQ REPLENISHMENT PROCESS

The IRARMMS2 routine attempts to acquire enough free central storage frames so the count of available central storage frames is as large as the Logical Swap Available Frame Queue Target. The difference between the current count of available central storage frames and the Logical Swap Available Frame Queue Target is the "quota" which the IRARMMS2 routine attempts to reach.

The IRARMMS2 acquires central storage frames to meet its quota in one of two ways: (1) it schedules page stealing or (2) it trims and physically swaps address spaces which previously were logically swapped (swap trim is described later in this document).

The page steal algorithms can be scheduled both before and after the swap trim algorithms.

- If the average system high UIC is 250 or larger (and fixed storage is not constrained), the SRM will schedule the IRARMPR5 routine to steal pages from active address spaces. The rationale behind this decision is that pages which have remained unreferenced in central storage for 250 seconds are not likely to be referenced again in the near future. Consequently, these old pages can be removed from active address spaces.

- If the current average system high UIC is less than 250 (or if fixed storage is constrained), the SRM will trim and swap out logically swapped address spaces. **The trim and swap out process normally[9] stops whenever the IRARMMS2 routine meets its quota of central storage frames**.

  - When selecting address spaces for swap trim, the SRM first scans the queue of logically swapped READY address spaces. These address spaces are normally batch jobs, started tasks, or long-running

---

[9]The SRM **does** continue processing to trim and swap out Transition, Request, and Storage Shortage swaps. For these swaps, the IRARMMS2 routine establishes a quota consisting of all central storage frames held by the address space experiencing the swap. Consequently, the SRM will trim and swap out the entire address space.

TSO address spaces which were swapped out based on Unilateral swaps or Exchange swaps.

Additionally, this category could include address spaces which were previously swapped out for a wait state, but which had been logically swapped so long that their think time exceeded the system think time threshold.

With SP4.2, the SRM will give the users which previously were swapped out because of a wait state a one-half second "grace" period before swapping them out. The rationale is that these users have become Ready following a wait state swap and perhaps system conditions will allow them to be swapped in shortly. These address spaces are allowed to remain on the queue of logically swapped READY address spaces until their total logically swapped time is greater than the current system think time plus one-half second. The SRM simply skips over these address spaces while processing the queue of logically swapped READY users.

- After the queue of logically swapped READY address spaces has been processed, the SRM processes the queue of logically swapped WAIT address spaces. The queue of logically swapped WAIT address spaces consists of address spaces in terminal wait, in Detected Wait, in Long Wait, or in APPC Verb wait. These address spaces will be trimmed and swapped out only if they have been logically swapped for longer than the current system think time.

- After all logically swapped users have been processed and if frames are still needed to meet the Logical Swap Available Frame Queue target, the SRM will schedule the IRARMPR5 routine to steal pages from active address spaces.

## 6.0 PAGE STEALING

The page stealing process uses the RSM to take central storage pages from active address spaces or from the Common area, send these pages to expanded storage or to auxiliary storage, and add the central storage frames to the Available Frame Queue. **The page steal process stops whenever the required number of pages are stolen**.

Page stealing is actually done by the RSM. However, page stealing is controlled by the SRM. The SRM invokes page stealing in one of two ways:

- The SRM attempts to replenish the central storage Logical Swap Available Frame Queue Target. This is the normal way in which page stealing is invoked.

- With SP4.2, the SRM may impose "self steal" on an address space. Once the SRM has designated "self

steal" for an address space, the RSM will steal pages from the designated address space if necessary to resolve page faults from the address space.

Both of these page stealing processes are discussed below.

### 6.1 PAGE STEALING: REPLENISH AFQ

When the page steal algorithms are scheduled, they have a "quota" which they are trying to reach. The quota is the number of frames which must be added to the Available Frame Queue to reach the Logical Swap Available Frame Queue Target.

The discussion presented earlier illustrated that the Logical Swap Available Frame Queue Target is at least 60 frames for pre-SP4.2 and 150 frames for SP4.2. Further, the earlier discussion described that the Available Frame Queue Replenishment algorithms (the IRARMMS2 routine) is scheduled whenever the RSM detects that the count of available frames is **less than** the RCEAFCLO threshold (initially 20 for pre-SP4.2 and 50 with SP4.2). Therefore, the minimum number of pages which will be stolen in one page-steal invocation for pre-SP4.2 is 40 pages (60 - 20 = 40) and 100 pages with SP4.2 (150 - 50 = 100).

The minimum values passed to the page steal algorithms could be smaller if pages were acquired by the swap trim process described under the Available Frame Queue Replenishment discussion. However, the page steal algorithms will steal at least MCCAFCOK - MCCAFCLO, regardless of the values passed to it. Consequently, the minimum number of pages which will be stolen in one page-steal invocation is 40 pages (pre-SP4.2) and 100 pages (with SP4.2). The minimum is established to minimize the overhead of invoking the page steal algorithms.

The minimum values could be significantly larger if page stealing is invoked by the SRM's swap-in processing (and the SRM had increased the RCEAECLO and RCEAECOK thresholds to accommodate the working set of an address space to be swapped in).

The SRM controls page stealing in the following order:

- Steal pages from an address space which is managed by the Working Set Manager (with SP4.2). The Working Set Manager keeps a Central Storage Target for managed address spaces. If a managed address space is above its Central Storage Target, pages are stolen down to the central storage target. However, recently referenced pages are not stolen at this time. If frames are still needed after stealing pages which have not been recently referenced, the SRM sets a flag and saves a pointer to the address space. Note that this algorithm is **not** the "self steal" algorithm.

- Potentially steal pages from address spaces which are storage isolated and which are above their maximum working set in central storage. Pages above the maximum protected working set in central storage are preferentially stolen if (1) expanded storage is not online, (2) expanded storage has been temporarily "turned off" because of system constraints, or (3) expanded storage is "tight"[10].

- Potentially steal pages from the Common area, if the Common area is storage isolated and if the number of central storage frames in the Common area is above the maximum working set specified for the Common area. Pages above the maximum protected working set in central storage are preferentially stolen if only (1) expanded storage is not online, (2) expanded storage has been temporarily "turned off" because of system constraints, or (3) expanded storage is "tight".

- Steal pages using a least recently used (LRU) algorithm to steal pages from address spaces and the Common area. The LRU algorithm uses the current system high UIC as the initial UIC steal criteria. The SRM directs the RSM to steal any page which has a UIC greater than or equal to the current UIC steal criteria.

    Prior to SP4.2, the RSM would steal a limited number of pages from any address space or the Common area (the limit would be either 2 or 10 pages, depending upon system conditions).

    With SP4.2, the limit on the number of pages stolen from an address space has been removed. Thus, for a given UIC value, the SRM will steal up to the target number of pages from a single address space. This change was made to facilitate block paging.

The following processing is performed for the LRU steal process:

- Steal pages from swapped-in address spaces. Pages are stolen which have a UIC greater than

or equal to the current steal criteria (this means that the pages are older than the current steal criteria). The SRM will steal as many pages as necessary to reach the target.

If the address space is storage isolated, the SRM will not steal below the target working set size unless the SRM has decided to override storage isolation because of a critical shortage of central storage.

- Steal pages from the Common area. Pages are stolen which have a UIC greater than or equal to the current steal criteria. The SRM will steal as many pages as necessary to reach the target. If the Common area is storage isolated, the SRM will not steal below the target working set size unless the SRM has decided to override storage isolation because of a critical shortage of central storage.

- After all address spaces and the Common area have been processed, the SRM will lower the UIC steal criteria.

If the current steal criteria is greater than 4, then the UIC steal criteria is not lowered by 1, since there is no need to loop through the private/Common area one UIC at a time. Rather, the UIC steal criteria is decremented by the following algorithm:

$$decrement = \frac{2 * (current\ steal\ criteria - 5)}{5}$$

For example, if the current steal criteria were 200, the steal criteria would be:

$$decrement = \frac{2 * (200 - 5)}{5} = 41$$

and the new steal criteria would be 159. This algorithm has the advantage of quickly removing old pages from central storage, while minimizing calls to the RSM steal algorithms.

- With SP4.2, special processing may be done if the current UIC steal criteria is less than 5. If the current UIC steal criteria is less than 5, the SRM checks whether the current UIC steal criteria is less than or equal to the high value of the RCCUICT keyword in IEAOPTxx[11].

---

[10]The concept of expanded storage being "tight" changed between MVS/XA and MVS/ESA SP4.2.

With MVS/XA, the SRM considers expanded storage to be "tight" when the number of available frames is less than three times the MCCAECLO variable. The MCCAECLO variable is constant with a value of 50. With MVS/XA, the SRM considers expanded storage to be tight if less than 150 expanded storage frames are available.

With MVS/ESA SP4.2, the SRM considers expanded storage to be "tight" when the number of available frames is less than the RCEAECOK value plus three times the MCCAECOK variable (which is the high value of the MCCAECTH keyword in IEAOPTxx). The default MCCAECOK variable has a value of 300. With MVS/ESA SP4.2, the SRM considers expanded storage to be tight if less than 1200 expanded storage frames are available.

---

[11]The RCCUICT low and high values are normally used to indicate whether central storage is constrained or abundant based on the average high system UIC, respectively. The SRM will lower the system target MPL if the system high UIC is less than the RCCUICT low value The SRM may raise the system target MPL if the system high UIC is greater than the RCCUICT high value and other conditions permit.

The default value for the RCCUICT high value is 4, so this logic would apply unless the default value was overridden.

If this test is met, the SRM concludes that there is not an abundance of central storage. The SRM will reprocess a managed address space if there is not an abundance of central storage and if pages are still needed.

Recall that if a managed address space was above its Central Storage Target, the first step of the page steal algorithm would steal pages from the address space if the pages had **not** been recently referenced.

However, if the SRM concludes that there is not an abundance of central storage, the SRM will enforce the Central Storage Target for a managed address space, even if recently referenced pages must be stolen. That is, if an address space is above its target in central storage, then pages are stolen from the address space (down to its target), even though the pages might have been recently referenced. If this step should be taken, the SRM will ensure that at least 10 pages are stolen from the address space.

- If pages are still needed, the SRM will loop through the LRU algorithm until sufficient frames are available.

- Once the Working Set Manager is managing an address space, it can impose a Central Storage Target. If an address space is above this target, frames can be preferentially stolen (as described earlier). Before the Working Set Manager selects an address space to **manage** by imposing a Central Storage Target, the Working Set Manager can decide that additional storage would be beneficial to an address space which is being **monitored**.

The Working Set Manager will attempt to allow the address space to reach an "OK" condition, in which it has a sufficient number of central storage frames to productively execute without experiencing a high page fault rate. Pages are not normally stolen from an address space trying to reach the "OK" status.

If the page steal algorithms have stolen all unreferenced frames with a UIC of zero and more pages are needed, the algorithms consider this to be a critical situation.

If an address space trying to reach the "OK" status had been skipped by the page steal algorithms, the address space will be processed. Pages will stolen from the address space which was skipped, down to a UIC of zero.

- If frames are still needed to replenish the Available Frame Queue, the SRM will override the "protected" working set of any address space protected by storage isolation. Frames will be stolen from

storage isolated address spaces or the Common area. The page steal algorithms will start with the highest UIC of all storage isolated address spaces or the Common area. The algorithms will follow the normal LRU page steal processing described above, except that storage isolation will be ignored.

## 6.2 PAGE STEALING: SELF STEAL WITH WORKING SET MANAGER

The Working Set Manager introduced with SP4.2 created a new category of page stealing: the "self steal" category for an address space which is being managed[12] by the Working Set Manager. The Working Set Manager can impose a Central Storage Target on a **managed** address space. When a Central Storage Target is imposed on an address space, any page faults **generated** by the address space will cause the RSM to check whether the number of central storage frames held by the address space exceeds the Central Storage Target.

- If the number of central storage frames held by the address space does not exceed the Central Storage Target, a page is acquired from the Available Frame Queue to resolve the page fault. This processing is identical to that performed for an address space which is not managed by the Working Set Manager.

- If the number of central storage frames does exceed the Central Storage Target, the RSM will steal a page **from the managed address space**. This "self steal" is performed immediately, rather than going through the SRM's normal Available Frame Queue replenishment algorithms.

The RSM will use the SRM's Fast Interface Processor (IRARMFIP) to select a destination for the stolen page (that is, the SRM will decide whether the stolen page should be sent to expanded storage or to auxiliary storage).

The effect of the self-steal algorithm is to restrict the page stealing to the address space causing the page faults. Since this address space has already been selected for working set management, the Working Set Manager has determined that adding central storage frames to the address space will not improve the performance of the address space (or the SRM could have decided that adding central storage frames to the address space will increase overall system page movement).

## 7.0 STORAGE ISOLATION

Storage isolation is a technique by which a certain number of pages associated with an address space (or with the Common area) are normally protected from page stealing or from swap trim (swap trim is discussed

---

[12] Many address spaces may be "monitored" by the Working Space Manager. Only one address space at a time will be "managed" by the Working Set Manager, so "self steal" applies to a single address space.

later). Storage isolation is implemented at the performance group period level by the PWSS keyword in IEAIPSxx, and for the Common area by the CWSS keyword in IEAIPSxx.

When the PWSS (or CWSS) keyword is used, a **minimum** working set size can be specified and a **maximum** working set size can be specified. Conceptually, the minimum working set is the minimum number of frames which an address space should be allowed to have allocated. Conceptually, the maximum working set is the maximum number of frames which an address space should be allowed to retain unless there is no demand for processor storage. The word "conceptually" is used because storage isolation works differently depending upon whether expanded storage is installed.

The storage isolation concept and algorithms were designed before expanded storage was available. Many of the concepts are applicable only to protect **central** storage frames from being stolen. The implementation of storage isolation changes dramatically if expanded storage is online.

## 7.1 STORAGE ISOLATION WITHOUT EXPANDED STORAGE

In an environment without expanded storage, the minimum working set specification represents the target (or "protected") **central storage** frames associated with an address space or the Common area.

During the page stealing (or swap trim) operation, the SRM will not normally steal pages from an address space (or the Common area) if the stolen pages would lower the number of pages below the target working set. The SRM will ignore storage isolation protection if there is a critical shortage of storage.

The maximum protected working set specifies an upper bound of pages which the address space (or the Common area) is allowed to keep when pages must be stolen (or trimmed). As described earlier in the Page Steal Algorithms, the SRM will determine whether any address spaces (or the Common area) holds more pages than the maximum protected working set size specified. If any address spaces (or the Common area) hold pages in excess of the maximum protected working set size, these pages are stolen before the LRU steal algorithm is implemented.

## 7.2 STORAGE ISOLATION WITH EXPANDED STORAGE

If there is expanded storage online, the SRM uses the total **processor storage** frames (the combined central storage frames and expanded storage frames) as the metric against which the number of frames is compared.

- The SRM will not normally steal pages if the count of total processor storage frames held by the address space would decrease below the target working set size. The SRM will preferentially steal pages **only** if the count of central storage frames exceeds the maximum protected working set size in central storage **and** if expanded storage is constrained. If expanded storage is not constrained, the SRM will **not** preferentially steal pages from storage isolated address spaces.

Pages stolen from storage isolated address spaces (or the Common area) are normally directed to expanded storage (without regard to the ESCTxxx criteria specified in IEAOPTxx)[13].

- The SRM will implement purge migration[14] if the count of processor storage frames is greater than the maximum protected working set and if expanded storage frames are needed.

Purge migration migrates pages from expanded storage to auxiliary storage without regard to how long the pages have resided in expanded storage. Purge migration will migrate as many pages from the address space (or the Common area) as necessary to satisfy the demand for expanded storage frames. However, purge migration will not migrate pages where the total processor storage frames would be less than the target working set size.

## 7.3 DYNAMIC ADJUSTMENT OF THE TARGET WORKING SET

The PWSS (or CWSS) keywords can be specified to control the minimum and maximum protected working set. Additionally, the target (protected) working set size can be dynamically adjusted, based on the page-in rate from auxiliary storage.

The PPGRT or PPGRTR keywords in IEAIPSxx can be used to control the dynamic adjustment of the target working set for address spaces associated with a performance group period. The CPGRT keyword can be used to control the dynamic adjustment of the target working set for the Common area.

If dynamic adjustment of the target working set is **not** specified, the target (or protected) working set is always at the minimum value specified for the PWSS keyword. If dynamic adjustment **is** specified, the target (or

---

[13]Note that the MVS/ESA SP4.2 and MVS/ESA SP4.3 *Initialization and Tuning References* are incorrect with regard to the ESCTSTC keyword. Pages stolen from storage-isolated address spaces are sent to expanded storage **regardless** of whether 32767 is specified as the criteria age. There is no way to prevent pages stolen from storage-isolated address spaces from being sent to expanded storage.

[14]Expanded storage migration is described in a companion paper ("Expanded Storage Management with MVS/ESA").

protected) working set is dynamically adjusted based upon the page-in rate.

The page-in rate is calculated either every 10 seconds of execution time (if PPGRT or CPGRT were specified) or every 10 seconds of address space residency time (if PPGRTR were specified).

- When the page-in rate from auxiliary storage falls below the value specified as the first parameter in the dynamic adjustment control keyword, the target working set is decreased by 3% of the existing protected working set. The reason for this action is that paging has fallen to a level such that sufficient private pages are in memory to satisfy memory demands. The implication is that memory demand does not require the target working set to be as large, and too much is being protected.

- When the page-in rate from auxiliary storage rises above the value specified as the second parameter in the dynamic adjustment control keyword, the target working set is increased. The reason for this action is that paging has risen to a level such that sufficient pages are **not** in memory to satisfy memory demands. The implication is that memory demand requires the target working set to be larger to reduce page replacement.

The SRM normally increases the target working set by 7% when the paging rate rises above the value of the second parameter[15].

The dynamic adjustment algorithms allow the target working set size to be dynamically adjusted, based upon varying memory demands. Central storage (or processor storage if expanded storage is online) is protected when needed to minimize paging from auxiliary storage, but storage is made available when not needed.

### 8.0 SWAP TRIM

As described earlier, the working set of an address space consists of those pages which the address space regularly uses as it executes. All other pages are non-working set pages. MVS removes non-working set pages from an address space using either page steal algorithms (discussed earlier) or swap trim algorithms.

Swap trim removes non-working set pages from address spaces which have been placed in a swapped out" status. Swap trim is similar to page steal (and the RSM page steal routines actually removes the pages). However, swap trim differs from page steal in that the page steal algorithms are applied to address spaces which are

swapped in; swap trim algorithms are applied to address spaces which are to be swapped out.

Swap trim occurs (1) when the SRM performs Quiesce processing for certain swaps which are destined to be physically swapped out, (2) when the SRM performs Quiesce processing for wait state swaps, and (3) when the SRM needs to replenish the Available Frame Queue from logically swapped address spaces.

- Quiesce processing schedules the Available Frame Queue Replenishment routine (IRARMMS2) to complete the swap-out processing for certain swaps (Request Swaps, Storage Shortage swaps, and Transition Swaps). The IRARMRM2 routine schedules swap trim to trim these address spaces down to their working set, since these address spaces are to be immediately swapped out to expanded storage or to auxiliary storage.

- Quiesce processing schedules the IRARMMS2 routine to perform swap trim processing for logically swapped wait state swaps (Terminal Input swaps or Terminal Output swaps, Detected Wait swaps, Long Wait swaps, and APPC wait swaps). Initial swap trim is immediately performed for these swaps, since they are to remain logically swapped until their think time exceeds the current system think time. The working set for these address spaces is "enhanced" as described below.

- The previous "AFQ Replenishment Process" section describes the order in which logically swapped address spaces are processed by swap trim when the SRM needs to replenish the Available Frame Queue. Address spaces in these categories include (for example) READY batch jobs which were swapped out for a Unilateral swap.

  These READY address spaces are not trimmed by Quiesce processing unless the Available Frame Queue needs to be replenished. They are allowed to keep all frames held upon swap-out. On the other hand, these address spaces are immediately selected for trim and physical swap-out whenever the Available Frame Queue needs to be replenished.

The distinction of working set pages and non-working set pages depends upon the type of address space and overall system conditions. The SRM calculates a "steal UIC" value for the RSM. The RSM will trim (steal) pages from an address space, in descending UIC order, down to (and including) the "steal UIC" value provided by the SRM.

The "steal UIC" value calculated by the SRM depends on the type of address space, whether the trim is being done by Quiesce processing for a wait state address space (rather than in preparation for physical swap-out), and whether the swap of the address space is targeted for auxiliary storage or for expanded storage.

---

[15]The target (protected) working set is increased by 14% for Fast Workload Acceptance (FWA) address spaces. FWA is requested by an address space associated with an alternative subsystem in an extended recovery facility (XRF) environment. FWA address spaces are given special treatment by many SRM algorithms.

- The minimum "steal UIC" is 0. This indicates that the RSM will trim (steal) pages from the address space, including pages with a UIC of 0. Except for the special cases (such as the "large jobs" and storage isolated address spaces over their target working set, discussed below), the RSM will not steal pages with the "referenced bit" turned ON. Additionally, the RSM will not steal LSQA pages or fixed pages.

    These conditions (UIC of 0 and referenced bit ON, LSQA pages, and fixed pages) define the strict working set of the address space.

    - The "steal UIC" is increased if the demand for central storage indicates that central storage is not constrained. The SRM computes a "delta" increase if the average system high UIC is greater than the high value of the LSCTUICT keyword in IEAOPTxx. The default high value for the LSCTUICT keyword is 4, so if the average system high UIC is greater than 4, the SRM will compute a delta increase in the "steal UIC". The delta increase is computed as:

        $$delta = \frac{Average\ high\ UIC - LSCTUICT\ high\ value}{10}$$

        For example, if the average system high UIC were 96 and the default LSCTUICT keyword values were in effect, the above algorithm would yield:

        $$delta = \frac{96 - 4}{10} = 9$$

        In this example, 9 would be added to the "steal UIC" and an address space would be allowed to keep all frames with a UIC of 8, regardless of whether the pages were recently referenced.

- The "steal UIC" is set to at least 1 for TSO users, regardless of central storage conditions. That is, the working set for TSO users includes pages with a UIC of 0, regardless of whether the pages were recently referenced.

- Address spaces which are being trimmed for physical swap-out have their "steal UIC" increased by 2 if the swaps are targeted for expanded storage[16]. This allows a working set to include pages with a UIC of at least 1, regardless of whether the pages were recently referenced.

---

[16]Note that with SP4.2, this increase is applied only to address spaces which (1) have fewer central storage frames than the MCCMAXSW large job limit or (2) have been trimmed down to the MCCMAXSW large job limit. The MCCMAXSW large job limit is available as a keyword in IEAOPTxx.

When the SRM selects an address space for trimming (and possible swap-out), it establishes a count of "frames needed" from the address space. The RSM normally will be directed to trim only as many non-working set pages from an address space as necessary to satisfy the "frames needed" count. The SRM establishes this count of "frames needed" using several algorithms:

- Under most circumstances, the count of "frames needed" will be only the number of frames needed to meet the Logical Swap Available Frame Queue target. Transition/Requested/Storage Shortage swaps are considered "special case" swaps. For these swaps, the SRM sets the "frames needed" count to be all the central storage frames held by the address space (as a result, the address space is trimmed to its working set and then swapped out).

- If the address space is protected by storage isolation, the SRM computes the "frames needed" by subtracting the address space's target working set from the number of central storage frames held by the address space. The SRM will not trim pages below the target working set from an address space protected by storage isolation.

- If an address space holds few central storage frames, the SRM will consider the address space a "small" job and will not schedule swap trim. If an address space holds 96 or fewer frames, the SRM will consider the entire number of frames to be the working set of the address space and will not trim pages from the address space.

- With SP4.2, if the address space holds more than the value specified for the MCCMAXSW keyword in IEAOPTxx, the SRM will consider the address space a "large" job and will perform special trimming. The default value for the MCCMAXSW keyword is 512 frames. If an address space holds more than 512 frames, the SRM will trim the address space down to 512 frames. However, the SRM will not trim an address space to less than the "protected" target working set if the address space is protected by storage isolation.

    The SRM will first trim pages which have not been recently referenced and, if necessary to meet the large job limit, the SRM will trim pages which were recently referenced. (As mentioned above, the SRM will not trim below the target working set for storage isolated address spaces.)

## 9.0 SWAP WORKING SET

The result from the swap trim process is the swap working set of an address space. The swap working set consists of, as a minimum, the strict working set of the address space. However, the swap working set may be significantly larger than the strict working set if central

storage conditions permit (that is, the "steal UIC" might have been increased significantly).

The swap working set is swapped out to auxiliary storage or to expanded storage.

- If the swap is directed to auxiliary storage, the entire swap working set is swapped out in what is termed a "single stage" swap. Single stage swaps will be directed to swap data sets if swap data sets have been defined and if the swap data sets have sufficient space; otherwise, the single stage swaps will be directed to local page data sets. When the address space is swapped back into central storage, the entire swap set will be swapped in.

- If the swap is targeted for expanded storage[17], the swap working set is split into a "primary" working set and a "secondary" working set. The primary working set consists of all fixed pages, all LSQA pages, and one page from each segment (a segment is one megabyte of memory). The secondary working set consists of all other pages in the swap working set.

When the address space is swapped back into central storage, only the primary working set is swapped in from expanded storage. Pages in the secondary working set are simply converted to non-working set pages and they are page-faulted into central storage as needed[18].

## 10.0 BLOCK PAGING

Block paging is a technique by which pages are moved between central storage and either expanded storage or auxiliary storage as a block of pages. The block is **formed** when the pages are moved from central storage. The block (or some pages in the block) are moved **into** central storage when a page fault occurs on any page in the block. The advantage of this process is that many pages may be brought into central storage at one time, rather than having each page be brought in to resolve its individual page fault.

Prior to SP4.2, block paging was used for swap working sets and for VIO windows. Although the process was not termed "block paging" at that time, the process of forming blocks of pages (swap working set or VIO window) was used to more efficiently move logically-related pages.

---

[17] Note that an address space may be "targeted" for expanded storage based on the ESCTSWWS criteria. By the time the address space has been trimmed and is ready for swap-out, the SRM tests whether sufficient expanded storage exists to hold the working set.
The "target" destination will be overridden and the address space will be swapped to auxiliary storage if insufficient expanded storage exists at the time of actual swap-out.

[18] Part or all of the secondary working set may be migrated to auxiliary storage and the primary working set may be migrated to auxiliary storage. The companion paper "Expanded Storage Management with MVS/ESA" describes these situations.

Two types of block paging were introduced with SP4.2: **explicit** block paging and **implicit** block paging.

## 10.1 EXPLICIT BLOCK PAGING

Explicit block paging is used by application developers in situations when the developers can predict the reference pattern of data. The developers can provide information to the RSM which describes the way in which the data is referenced.

The information provided to the RSM simply describes multiple pages as logical groups of pages. Additionally, the information can describe how the data will be referenced (by rows or by columns, forward direction or backward direction, etc.).

For example, developers might know that once a particular matrix is being processed, the entire matrix will be processed. Further, the developers might know whether the matrix is processed by rows or columns. If the matrix spanned many pages of virtual storage, virtual storage referencing might be significantly improved by loading all rows or columns of a matrix as a block, rather than having each page loaded in response to an individual page fault.

For large matrices, this knowledge about the reference pattern could be conveyed to the RSM such that the matrix (or part of the matrix) is maintained as a logical entity.

The reference pattern is described using the REFPAT macro for assembler programs (or by callable services for programs written in a high level language).

The REFPAT macro is used to INSTALL and REMOVE a reference pattern, and to describe referencing characteristics (row versus column, forward versus backward, whether the data area is part of an address space or a data space, etc.).

Once a reference pattern has been defined to the RSM, the RSM treats the virtual storage defined by the macro as a logical entity, or block.

- When the RSM selects a page from the block as part of the page steal processing, the RSM will steal the entire block of pages. The entire block will be moved to expanded storage or to auxiliary storage.

- When a page fault occurs for any page in the block, the RSM will move the entire block into central storage.

Explicit block paging can avoid multiple page faults for pages in the block, and avoid the overhead caused by the multiple page faults. Explicit block paging is done only when a block is explicitly defined to the RSM.

## 10.2 IMPLICIT BLOCK PAGING

An **implicit** block is formed by the RSM, without control by a user. An implicit block consists of all pages with the

same UIC and which are contiguous in virtual storage. Implicit blocks are always **formed** by the RSM as it steals (or trims) pages.

As the RSM is stealing or trimming pages from an address space, pages with the same UIC and which are contiguous in virtual storage are formed as a block. The RSM will move the pages out of central storage as a block of pages.

The size of the block of pages depends upon how many pages are stolen or trimmed from an address space during one invocation of the Page Steal or Swap Trim algorithms, how many pages are at the same UIC, and how many of these pages are contiguous in virtual storage[19].

• The Page Steal Routine (IRARMPR5) sets the target number of pages to steal as a function of the number of frames required to replenish the Available Frame Queue (that it, it takes as many frames as are required to reach the RCEAFCOK value). The RSM uses the target to steal pages based upon UIC.

• The Address Space Trim Routine (IRARMPR9) sets the target number of pages to be trimmed from an address space which is being trimmed for swapout. The target number of frames varies, depending upon whether the job is too large to swap (the number of frames is above the MCCMAXSW threshold as specified in IEAOPTxx), varies depending upon whether the address space is storage isolated, etc.

If the block is moved to auxiliary storage (either directly or migrated from expanded storage), the RSM requests that the contiguous slot algorithm be used. Thus, the Auxiliary Storage Manager will place the block in contiguous page slots in auxiliary storage if contiguous pages slots are available.

When a page fault occurs for some page in a block, there is a high probability that other pages in the block will be referenced. If other pages at same UIC are referenced, page fault resolution can be avoided if the block is brought into central storage.

Page fault resolution for a page which is part of a block depends upon whether the block resides in auxiliary storage or in expanded storage.

• **Pages in auxiliary storage**. If the page resides in auxiliary storage, the page and the remainder of the block will be brought into central storage. That is,

the page being faulted on will be brought in. Following that page, the ASM will bring in the remainder of the block (by increasing virtual storage address).

• **Pages in expanded storage**. If the page resides in expanded storage, processing depends upon whether the address space is monitored by the Working Set Manager.

  • **The address space is not monitored**. Only the page causing the page fault is moved into central storage.

  • **The address space is monitored**. The SRM implements implicit block page-in when the Working Set Manager begins to monitor an address space. The page causing a page fault and the remainder of the block is moved from expanded storage to central storage. The SRM keeps track of the "hit ratio" of whether pages in the block are referenced again. If the hit ratio is not sufficiently high, implicit block page-in is discontinued for the address space. Once implicit block page-in is discontinued, only the page causing a page fault is moved into central storage.

The rationale for having a different approach depending upon whether the block resides in auxiliary storage or expanded storage relates to the overhead of bringing in a page to satisfy a page fault.

• The overhead in bringing in a page from auxiliary storage is high. Minimizing this overhead justifies always performing implicit block page-in from auxiliary storage, even though other pages in the block may not always be referenced.

• The overhead for bringing in a page from expanded storage is low. Consequently, implicit block page-in will be done for blocks residing in expanded storage only if the address space is being monitored by the Working Set Manager and if the hit ratio is high enough to warrant implicit block page-in.

Notice that a block is loaded from the referenced page to higher virtual storage addresses. When the block is formed (based on either the Page Steal or Swap Trim processing), all pages at the given UIC are contained in the block. However, a subset of the block may be loaded when the block is paged in.

The rationale for this logic is the belief that most processing (of arrays, for example) operates in a forward basis with respect to the virtual address. That is, element #1 of an array would be processed, followed by element #2, element #3, and so forth. Consequently, a page fault on a page in the middle of a block will cause implicit block paging to bring in the page being page-faulted and all pages with higher virtual addresses.

---

[19]There are limits on the number of pages in the block. If the block is moved to auxiliary storage, MVS breaks large blocks into page groups of approximately 30 pages per page group. With SP4.2, the maximum size of the block moved to expanded storage is dynamically adjusted based upon the "hit ratio" of pages in the block.

## 11.0 WORKING SET MANAGER

The Working Set Manager was introduced with SP4.2, but most code associated with the Working Set Manager has been spirited away to Object Code Only (OCO). Unfortunately, analysts must glean what we can from presentations given by IBM, from comments and logic which appear in non-OCO code, from observing on-line monitors, and from analyzing RMF data. Alas.

The intent of the Working Set Manager is to manage large and relatively long-running jobs which cause significant page movement overhead. The Working Set Manager addresses the problem of non-productive CPU time spent moving pages between central storage and expanded storage[20].

The Working Set Manager monitors "non-productive" CPU time, "discretionary central storage" availability, and "idle page frame seconds" to guide its decisions.

- Productive CPU time is CPU time spent under SRB or TCB control of an address space, except for the master address space. Non-productive CPU time is considered all other CPU time. This time will mostly be CPU time spent related to page movement.

- Discretionary central storage is a count of central storage frames which are available or which can easily be made available if central storage becomes constrained. Discretionary central storage consists of the following categories:

  - Central storage frames which currently are available (they are on the Available Frame Queue count).

  - All central storage frames which are held by non-wait state logically swapped address spaces, or held by wait state logically swapped address spaces which have been logically swapped longer than the current system think time.

Page delays caused by the paging subsystem utilization levels would normally cause serious performance problems long before the CPU time spent resolving page faults to auxiliary storage became a performance issue. The main problem the Working Set Manager addresses is page movement between central storage and expanded storage.

Notice that discretionary central storage does **not** include frames held by wait state logically swapped address spaces which are protected by think time; that is, the address spaces have not been logically swapped longer than the current system think time. Since the address spaces have not been logically swapped longer than the current system think time, their frames are not considered discretionary.

- All central storage frames which are not protected by storage isolation but which are inactive (that is, the pages have a UIC greater than or equal to 250).

- Idle page frame seconds is a measure of the length of time when discretionary frames have been held by logically swapped address spaces. The idle page frame seconds is computed by multiplying the number of central storage frames held by logically swapped address spaces not protected by think time (as described above) times the length of time the address space has been logically swapped.

The result gives a time dimension to the discretionary central storage concept.

The SRM will periodically examine system conditions (whether the processors are over- or under-utilized, whether central storage is over- or under-utilized, whether expanded storage is over- or under-utilized, etc.).

Depending upon system conditions, as matched against installation guidance, the SRM will adjust various aspects of its control of the environment. These adjustments occur only when the SRM's Resource Monitor 2 (IRARMRM2) code is executed[21].

The frequency with which the IRARMRM2 routine is executed is based on the number of processors and the speed of the processors. Regardless of the speed or number of processors, the IRARMRM2 routine is executed no more frequently than once every 5 seconds (pre-SP4.2) or once every 2 seconds (with SP4.2).

Based on decisions which are made by the Working Set Manager (the IRARMRM3 routine), the SRM may decide to **monitor** address spaces or may decide to **manage** a selected address space.

### 11.1 MONITORED ADDRESS SPACES

During the system adjustment processing, the SRM determines whether over 5% of the CPU is spent in non-productive page movement. When over 5% of the CPU is spent in non-productive page movement, the Working Set Manager will begin collecting detailed data about the page movement associated with selected address spaces.

---

[20]CPU time spent moving pages to and from auxiliary storage is not normally a problem. This is because of auxiliary storage limits. Few paging subsystems can support a paging rate large enough to cause serious CPU problems.

[21]With SP4.2, the adjustments have been moved to the IRARMRM3 routine, which is OCO code. However, the IRARMRM3 routine is invoked by the IRARMRM2 routine. The frequency of execution is the same for both routines. The IRARMRM3 routine contains most of the Working Set Manager algorithms. Other SRM routines contain code which implements decisions made in the IRARMRM3 routine.

An address space is selected for detailed monitoring under the following conditions:

- If no address space is already being monitored, the SRM computes the amount of non-productive CPU time attributed to the address space. There is no direct measurement of the non-productive CPU time, so the SRM computes an estimate of the non-productive CPU time.

  For each address space, the SRM maintains information describing page movement associated with the address space: the number of page faults experienced by the address space and whether the page faults were resolved from expanded storage or auxiliary storage), the number of swaps experienced by the address space and the number of working set pages swapped, etc. Based on this information, the SRM computes a count of all pages moved as part of a block.

  The SRM multiplies the number of pages moved as part of a block by a constant which represents the approximate number of CPU service units required to move a page as part of a block. This constant is 0.05 CPU service units. Since the constant is expressed in CPU service units, the constant is independent of processor power. The result is an approximation of the amount of non-productive CPU time required because of page movement.

  An address space is selected for monitoring if the "non-productive" CPU time accounts for over 5% of the address space's productive CPU time.

- If one or more address spaces are being monitored, an address space will be added to the list of monitored address spaces only if the address space's allocated central storage frames is greater than the average number of central storage frames allocated to address spaces already being monitored.

The effect of the above two algorithms is to monitor only those address spaces which are experiencing heavy page movement and which are allocated large amounts of central storage.

For each monitored address space, the Working Set Manager will collect additional information. Based on the information collected, the Working Set Manager may implement different controls to attempt to reduce the non-productive CPU time.

- The Real Storage Manager will collect additional information to describe the number of pages held by the address space, at different UICs. This information is distributed into four variables associated with each address space. Each variable

contains a count of the number of pages at the particular UIC. Similar counters are maintained for the system as a whole, for all address spaces protected by storage isolation, and for the Common area. The Working Set Manager uses the counts of pages at various UICs in its assessment of the effect of working set management actions.

- The Working Set Manger will attempt to reduce the page fault rate by imposing implicit block page-in from expanded storage. The discussion of Block Paging describes the implications of implicit Block Paging from expanded storage.

Based on the data collected via the address space monitoring, the Working Set Manager will make decisions about the monitored address spaces.

If the CPU is over-utilized (the average processor busy is greater than the high value of the RCCCPUT keyword in IEAOPTxx), a monitored address is selected to swap out[22]. These swaps are accomplished by (1) lowering the target MPLs of the domain to which the address space is assigned so that an address space will be swapped out of the domain, and (2) giving a Working Set Manager Recommendation Value to the monitored address space such that the monitored address space will be selected for swap-out.

The Working Set Manager Recommendation Value overrides the normal Swap Recommendation Value associated with address spaces. Consequently, the address space identified for swap-out will be selected for swap-out, regardless of its Swap Recommendation Value. These swaps are the "Improve Central Storage Usage" swaps[23].

The SRM will periodically examine system conditions and will swap in the address space if system conditions change so that the SRM decides that the address space can safely be swapped in. An address space swapped out for this reason will not be left swapped out indefinitely,

---

[22]The address space is not necessarily swapped out of central storage. The address space will be logically swapped as would any other address space.

[23]There has been some confusion in the literature regarding the difference between the "Improve Central Storage Usage" swaps and the "Improve System Paging Rate" swaps introduced with SP4.2.

The "Improve System Paging Rate" swaps are **not** a result of the Working Set Manager managing individual address spaces, but are simply a logical subset of the Unilateral swaps previously identified by the SRM. "Improve System Paging Rate" result from the SRM decreasing the system target MPL because the paging rate to auxiliary storage became greater than the RCCPTRT high value in IEAOPTxx.

When the system target MPL is decreased, the SRM decreases the target MPL for some domain (the domain is selected based on its contention index), and an address space is swapped out. In fact, the address space selected for swap out may not be significantly contributing to page faults.

even if the SRM does not decide that system conditions have changed sufficiently to allow the address space to be swapped in.

After the address space has been swapped out for a period of time (30 seconds for TSO address spaces and 10 minutes for other address spaces), the SRM will swap in the address space. If necessary, the SRM will select a swapped-in address space to swap out and "make room" for the swapped out address space. These swaps are the "Make Room" swaps.

The address spaces which is to be swapped out for a Make Room swap is selected from the domain to which the address space to be swapped in is assigned. In effect, the Make Room swaps are identical to the "Exchange on Recommendation Value" swaps. The only difference is what causes the exchange to take place.

- With the Make Room swap, the swap is because a swapped-out address space has been swapped out too long.

- With the Exchange on Recommendation Value swaps, the swap is because two address spaces meet the criteria for an exchange.

## 11.2 MANAGED ADDRESS SPACE

If the CPU is under-utilized (the average processor busy is less than the low value of the RCCCPUT keyword in IEAOPTxx) but the non-productive CPU time is still too high, the Working Set Manager will select an address space to manage.

The Working Set Manager will identify the monitored address space with the highest ratio of non-productive CPU time to central storage frames assigned. The Working Set Manager will attempt to reduce page movement associated with this address space by allocating additional central storage to the address space. In order to accomplish this, the Working Set Manager will temporarily protect the address space from having pages stolen.

This action is done so the Working Set Manager can determine whether additional pages will allow the address space to productively process without experiencing a large number of page faults. When this protection is in effect, the page steal algorithms will skip the address space when stealing pages from address spaces.

By skipping the address space during the page steal processing, the Working Set Manager allows the address space to acquire and retain additional pages. Only one address space at a time will have this temporary page steal protection.

After observing the effect of allocating additional central storage frames to the address space, the Working Set Manager will decide whether adding more frames helped reduce the page movement rate (both for the system as a whole and for the address space).

- The Working Set Manager will assign a Central Storage Target on the address space being monitored. The discussion of Page Steal algorithms describes the implications of a Central Storage Target[24].

- The Working Set Manager can direct the RSM to implement "self-steal" to handle resolution of page faults. When "self-steal" is in effect, the RSM will steal pages from the address space being monitored if necessary to resolve page faults caused by the address space, **and** if the number of central storage frames held by the address space is greater than its Central Storage Target. This "self-steal" is done instead of the RSM acquiring a central storage frame from the Available Frame Queue.

The "self steal" method has the effect of containing the paging movement to the address space causing page movement.

## 11.3 EFFECT OF WORKING SET MANAGER

The Working Set Manager does not monitor or manage all address spaces. It does not monitor non-swappable address spaces, privileged address spaces, nor address spaces which are storage isolated.

Additionally, the Working Set Manager requires elapsed time to monitor system conditions, make changes, and observe the effect of the changes. The Working Set Manager will not have any effect for short-running jobs or interactive TSO transactions.

The Working Set Manager will not be particularly useful for environments consisting mostly of non-swappable address spaces (such as DB2, CICS, etc.), TSO interactive users, and short test batch. This statement is true simply because the Working Set Manager will not monitor (or manage) these workloads or because the individual address spaces do not execute long enough for the Working Set Manager to make appropriate decisions.

---

[24]Briefly, unreferenced pages above the target are stolen and after all address spaces are processed at the current system high UIC, referenced pages above the target may be stolen before the steal UIC criteria is lowered (referenced pages are stolen only if the current UIC steal criteria is low). Pages are not normally stolen if the number of pages would be reduced to less than the target.

In this regard, the Central Storage Target is similar to having storage isolation assigned to the address space, with identical minimum and maximum storage isolation values. One major difference is that pages stolen from storage isolated address spaces have their pages normally sent to expanded storage without regard to the ESCTSTC specifications, while pages stolen from managed address spaces are sent to expanded storage based on the ESCTSTC specifications.

The Working Set Manager may significantly improve overall system performance for installations which have long-running jobs which require large amounts of central storage (for example, jobs which process large arrays).

## 12. SUMMARY

This paper has explained the basic techniques and algorithms used by the SRM to manage central storage: physical swapping, logical swapping, think time and think time adjustment, page stealing and storage isolation, swap trim and swap working set, block paging, and working set management. A companion paper ("Expanded Storage Management with MVS/ESA") explains the techniques used by the SRM to manage expanded storage.

## REFERENCES

*MVS/ESA SP4.2 Initialization and Tuning Guide*, IBM document GC28-1634-3

*MVS/ESA SP4.2 Initialization and Tuning Reference*, IBM document GC28-1635-3

*MVS/ESA SP4.3 Initialization and Tuning Guide*, IBM document GC28-1634-4

*MVS/ESA SP4.3 Initialization and Tuning Reference*, IBM document GC28-1635-4

*MVS/ESA SP2.2 Working Set Management and Block Paging Presentation Guide*, IBM document GG66-3204

*MVS/ESA Component Diagnosis and Logic: System Resources Manager*, IBM document LY28-1592

MVS/XA SP2.2 Source Code, IBM Microfiche LJB2-9573

MVS/ESA SP4.2 Source Code, IBM Microfiche LJB2-9605