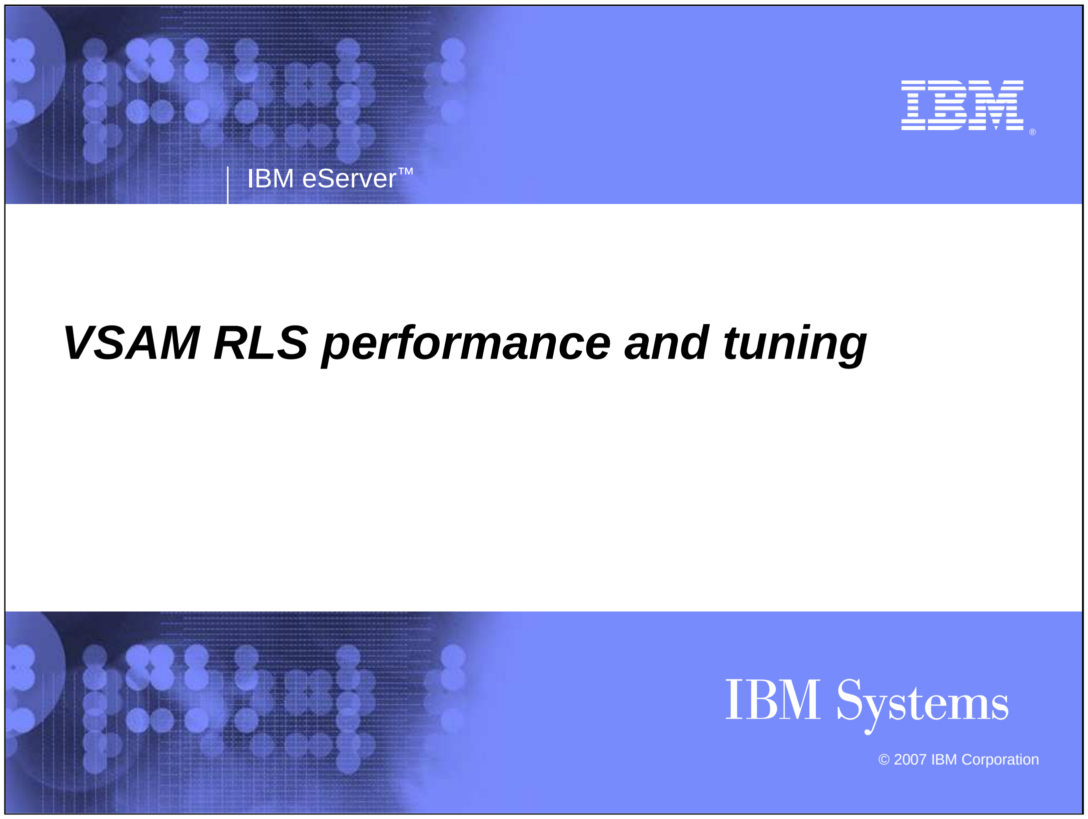


IBM eServer™

VSAM RLS performance and tuning

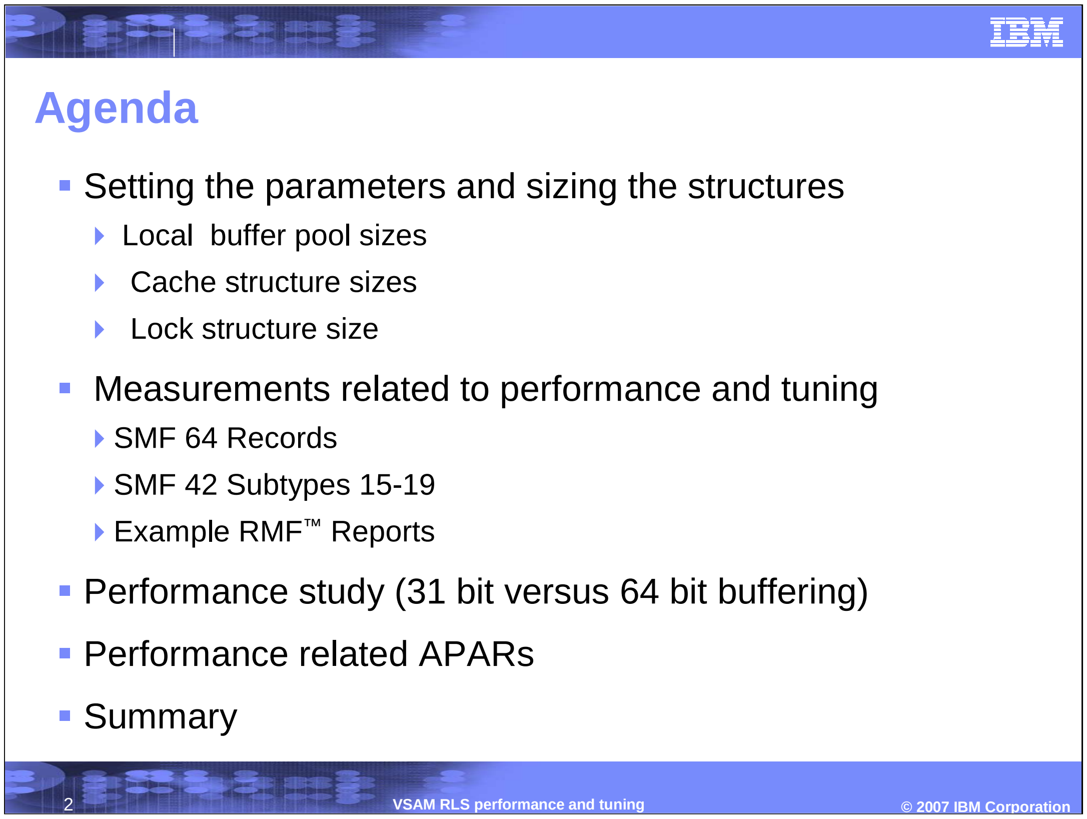
IBM Systems

This is the VSAM RLS Performance and Tuning presentation.

You should view the IBM Education Module “*VSAM RLS Performance and Tuning Overview*” before this module.

The Overview provides you with an understanding of the VSAM RLS I/O path, the terminologies that you will need to know.

It includes specific example of the I/O path. The various parameters are also presented.



Agenda

- Setting the parameters and sizing the structures
  - Local buffer pool sizes
  - Cache structure sizes
  - Lock structure size
- Measurements related to performance and tuning
  - SMF 64 Records
  - SMF 42 Subtypes 15-19
  - Example RMF™ Reports
- Performance study (31 bit versus 64 bit buffering)
- Performance related APARs
- Summary

2 VSAM RLS performance and tuning © 2007 IBM Corporation

The subject of module is VSAM RLS performance and tuning. The most important thing is to look at what is called the performance path in RLS also referred to as VSAM RLS I/O path. This is where you need to focus on for performance and tuning.

This module will give you specific examples of how you would set those parameters.

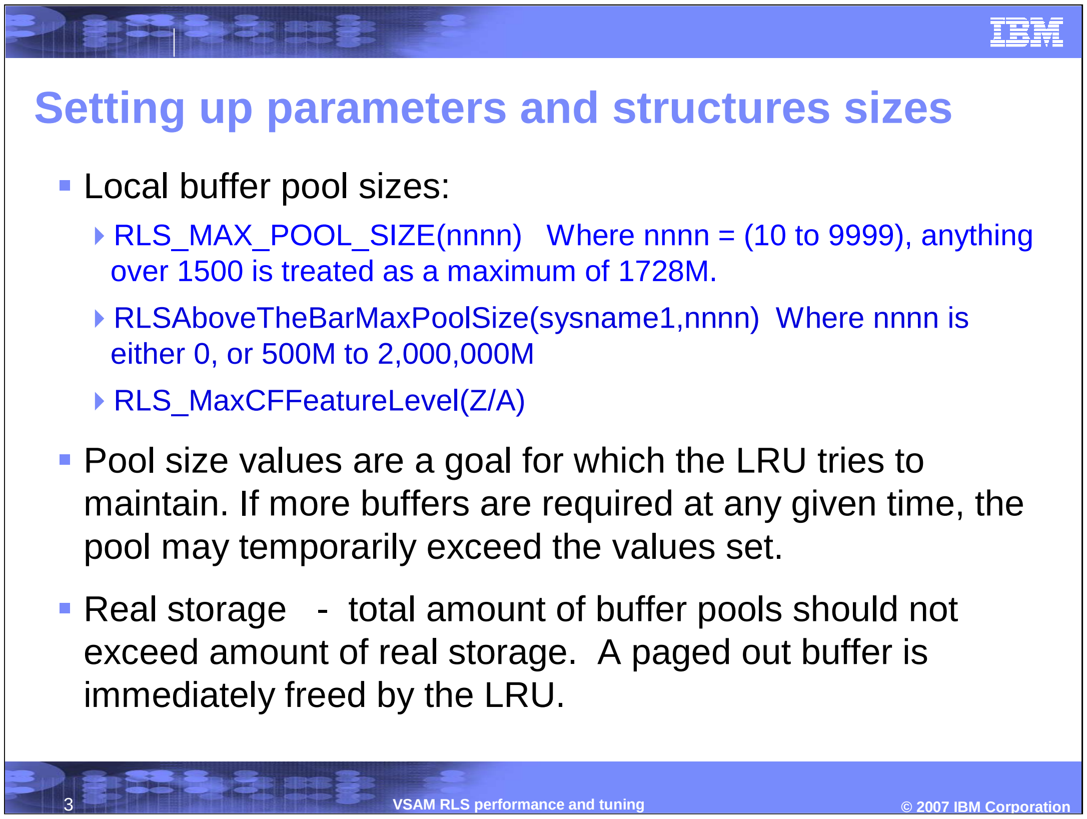
There are also some structures that exist in the coupling facility for RLS. You will need to know how to size those structures correctly.

You will learn about sizing a local buffer pool, the cache structure, and the lock structure and how they relate to performance and tuning.

Once your system is set up, and you have all your parameters set, then you need to know how to measure the performance. Setting up the parameters are important. There are mechanisms for measuring performance. You can use SMF type records, such as SMF 64 and SMF 42 records. RMF also provides some online reports that are very helpful. You will see a couple of examples.

You will see an actual real life performance study which will help you see how this all fit together. In this particular study,

There's a comparison using 31bit buffering vs. 64bit buffering, this shows you how buffering, in a larger buffer pool, can help with performance improvements. There are some recent APARs that are related to performance that you should get for your system before you get into doing any of this performance tuning.



Setting up parameters and structures sizes

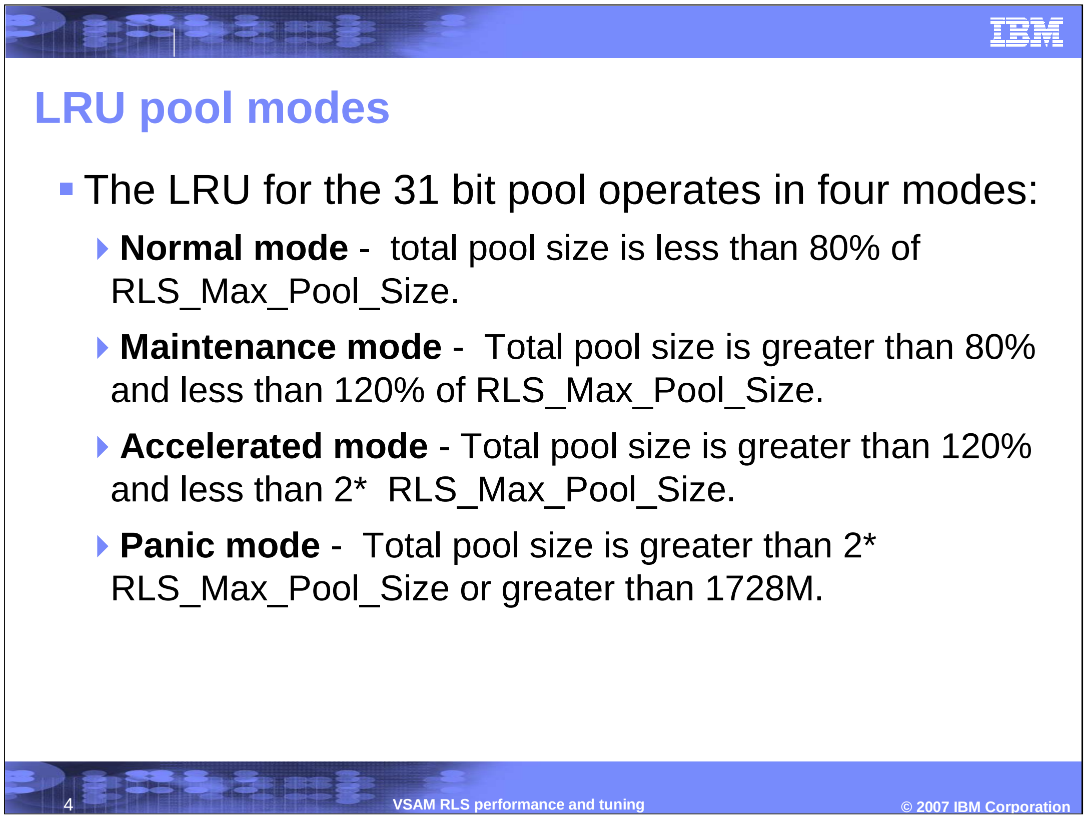
- Local buffer pool sizes:
  - RLS_MAX_POOL_SIZE(nnnn)   Where nnnn = (10 to 9999), anything over 1500 is treated as a maximum of 1728M.
  - RLSAboveTheBarMaxPoolSize(sysname1,nnnn) Where nnnn is either 0, or 500M to 2,000,000M
  - RLS_MaxCFFeatureLevel(Z/A)
- Pool size values are a goal for which the LRU tries to maintain. If more buffers are required at any given time, the pool may temporarily exceed the values set.
- Real storage   - total amount of buffer pools should not exceed amount of real storage. A paged out buffer is immediately freed by the LRU.

3 VSAM RLS performance and tuning © 2007 IBM Corporation

When you are setting these parameters and sizing these structures, there are the things that you want to consider when you do it. The first thing that you want to start with when it comes to getting these parameters set up to the sizing local buffer pool correctly. In this example, there are the two buffer pools.

For the 31 bit pool, the goal for this pool is set by the RLS_MAX_POOL_SIZE parameter, which ranges from 10 MB or you could put in 9999 to indicate the maximum. When SMS sees a 9999 in the parameter, it will set it to 1500 MB. If there is anything over that 1500, the limit of 1.7 Gig will be set. If the 1.7 Gig limit is exceeded, then RLS will react very strongly to that and trying to reduce the buffer pool dramatically.

For RLSAboveTheBarMaxPoolSize, you can specify this parameter for each system in the sysplex. You can specify 0 to indicate no 64 bit pool, or a minimum of 500 MB, and you can actually go up to 2 millions MB, which is 2 terabytes. It is not recommended that you go that high at this point. There are some criteria that you have to consider when you are building your buffer pool, which are shown below. If a buffer is specified that is larger than the amount of available real storage, a paging situation will occur quickly. If buffers are paged out, the LRU will free them immediately. This will defeat the purpose. For this reason, a buffer pool should be sized within the amount of the available real storage. Since the specified sizes of the pools are goal sizes, the BMF LRU will maintain those sizes but BMF may exceed those sizes if needed. Once the pool exceeds its specified size, BMF will try to reduce this pool down by accelerate the criteria on what buffer are allowed to stay on that pool. In such case, BMF will be throwing buffer out of the pool that may recently be read into the pool. If they get throwing out right away, there will be no buffer hit on them, and then RLS will end up going back out to the cache or the DASD. The goal is to try to figure out how much storage will be allocated on the system. The pool size should not exceed the available real storage. A large enough pool should be allocated to allow a buffer hit; however, the size should not exceed the limit to cause the LRU to begin throwing buffer out. This situation can be avoided and will be demonstrated a little bit more after you learned how the LRU works.

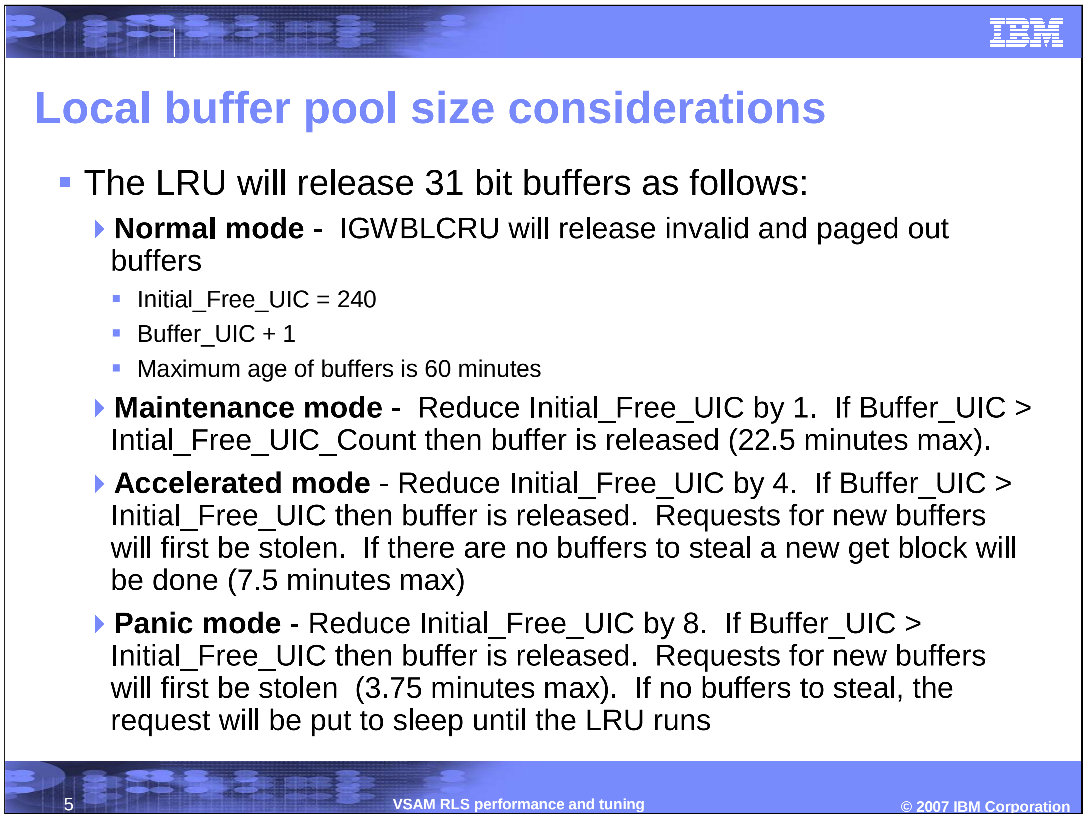


LRU pool modes

- The LRU for the 31 bit pool operates in four modes:
  - **Normal mode** - total pool size is less than 80% of RLS_Max_Pool_Size.
  - **Maintenance mode** - Total pool size is greater than 80% and less than 120% of RLS_Max_Pool_Size.
  - **Accelerated mode** - Total pool size is greater than 120% and less than 2* RLS_Max_Pool_Size.
  - **Panic mode** - Total pool size is greater than 2* RLS_Max_Pool_Size or greater than 1728M.

4 VSAM RLS performance and tuning © 2007 IBM Corporation

There are two LRU's. One LRU controls the 31 bit pool, and another LRU controls the 64 bit pool. For the 31 bit pool LRU, there are four running modes: Normal Mode, Maintenance Mode, Accelerated Mode, and Panic Mode. The mode changes depending on the pool sizes relative to the parameter you specified for RLS max pool size. It's in the Normal Mode if the total buffer is less than 80% of what you specified in the RLS Max Pool Size. If RLS Max Pool Size is set to 100 MB for the 31 bit pool and the buffer in the pool is greater than 80MB, then the mode will be switched from the normal mode to the maintenance mode. The maintenance mode is when the pool size got over the 80% but less than 120% of what you specified. Once more than 120% but less than 2 times of the RLS Max Pool Size, the mode will be changed to the accelerated mode. It will be in panic mode if it's greater than 2 times of the RLS Max Pool Size or if it hit that hard line at 1.7 Gig.



Local buffer pool size considerations

- The LRU will release 31 bit buffers as follows:
  - **Normal mode** - IGWBLCRU will release invalid and paged out buffers
    - Initial_Free_UIC = 240
    - Buffer_UIC + 1
    - Maximum age of buffers is 60 minutes
  - **Maintenance mode** - Reduce Initial_Free_UIC by 1. If Buffer_UIC > Intial_Free_UIC_Count then buffer is released (22.5 minutes max).
  - **Accelerated mode** - Reduce Initial_Free_UIC by 4. If Buffer_UIC > Initial_Free_UIC then buffer is released. Requests for new buffers will first be stolen. If there are no buffers to steal a new get block will be done (7.5 minutes max)
  - **Panic mode** - Reduce Initial_Free_UIC by 8. If Buffer_UIC > Initial_Free_UIC then buffer is released. Requests for new buffers will first be stolen (3.75 minutes max). If no buffers to steal, the request will be put to sleep until the LRU runs

5 VSAM RLS performance and tuning © 2007 IBM Corporation

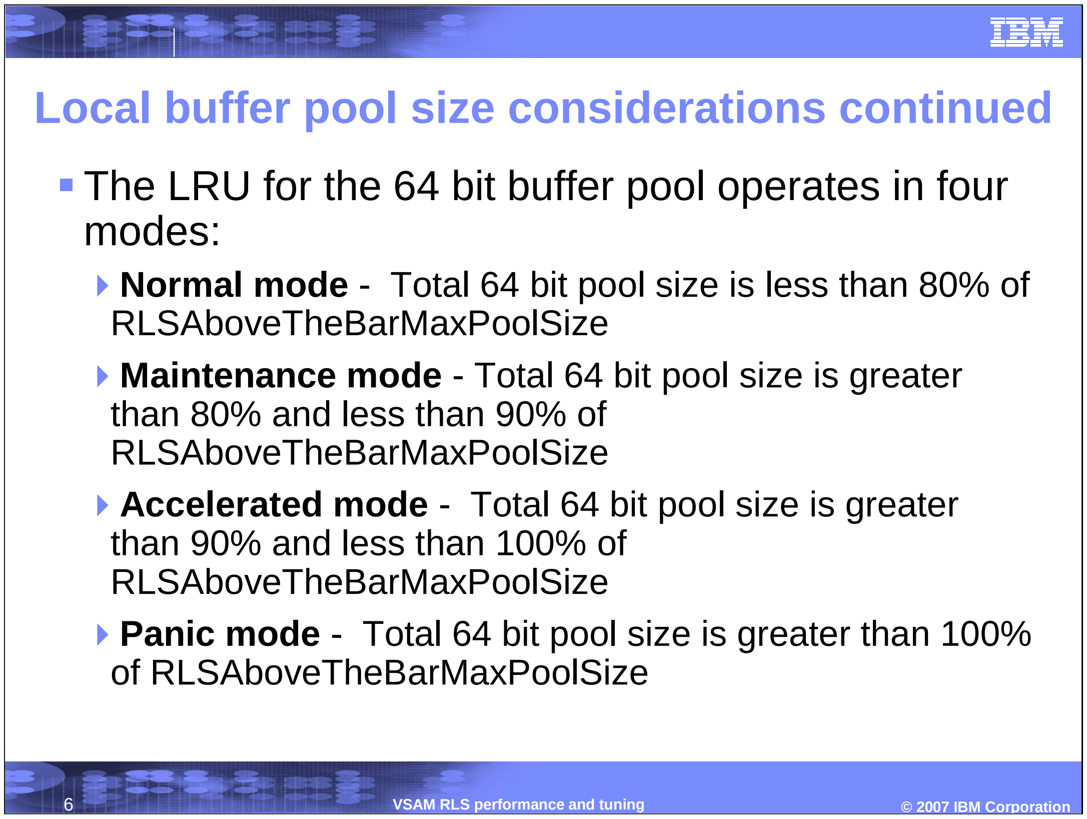
In normal mode, all buffers will be searched to remove all the ones that are invalid. A buffer becomes invalid because it was modified or it was paged out. LRU will throw those buffers out. Then, The initial criteria will be set to this constant of 240. This is an upper limit of how old the buffers are going to stay in the pool. LRU will search through buffers and age them all. The unreferenced interval count or the UIC will be incremented by one. A buffer will be read into the pool with the UIC set to 0. Each time the LRU wakes up, the buffer will increased by one, and the value will be compared with the 240 number. When the UIC gets to 240, the LRU will remove the buffer from the pool. In this example, the LRU is running every 15 seconds. The maximum age for every buffer is up to 60 minutes. For any buffer that you read in that stays valid, you have one hour to re-reference them in order to get a buffer hit on them. That's the normal mode.

If the buffer pool is managed to get larger than 80% of the specified goal, then the LRU will reduce the criteria by one. In stead of 240, the LRU will start searching for the buffer that are 239 and counts all. The maximum age on that is about 22 minutes.

Likewise, if the LRU goes to the accelerated mode, it will start to reduce the count by four. In that case, buffers can only be 7.5 minutes in the pool. Once the LRU gets to accelerated mode, it will do things a little bit differently. Instead of getting a new buffer request out to the pool, and LRU is running in accelerated mode, the LRU will be trying to steal one of the existing buffers. To keep the buffer pool from growing larger and getting this LRU into a worse state, the LRU will be more picky about which buffers will be thrown out.

In panic mode. The LRU will select buffers to throw out with more strict policy by reducing the count of the UIC by 8. In that case, the buffers will stay about 3.75 minutes here. Once the LRU is in panic mode, the LRU will first steal buffer before try to increase the the size of the pool.

In summary, a buffer can stay in a pool for one hour. You need to consider how often

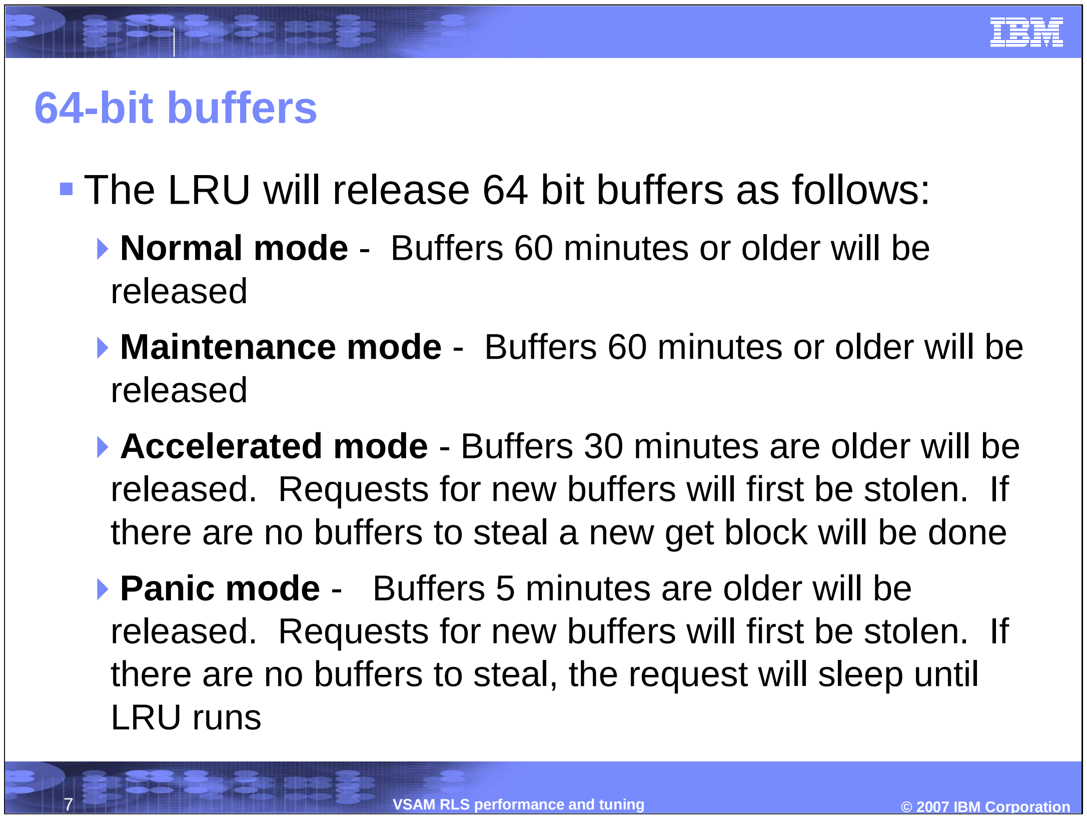


Local buffer pool size considerations continued

- The LRU for the 64 bit buffer pool operates in four modes:
  - **Normal mode** - Total 64 bit pool size is less than 80% of RLSAboveTheBarMaxPoolSize
  - **Maintenance mode** - Total 64 bit pool size is greater than 80% and less than 90% of RLSAboveTheBarMaxPoolSize
  - **Accelerated mode** - Total 64 bit pool size is greater than 90% and less than 100% of RLSAboveTheBarMaxPoolSize
  - **Panic mode** - Total 64 bit pool size is greater than 100% of RLSAboveTheBarMaxPoolSize

6 VSAM RLS performance and tuning © 2007 IBM Corporation

For the 64 bit pool, since it is a huge big pool, the LRU can be less picky on how fast things are growing. However, the LRU has the same kind of deal. There are four modes and the same logic as in the 31 bit pool.



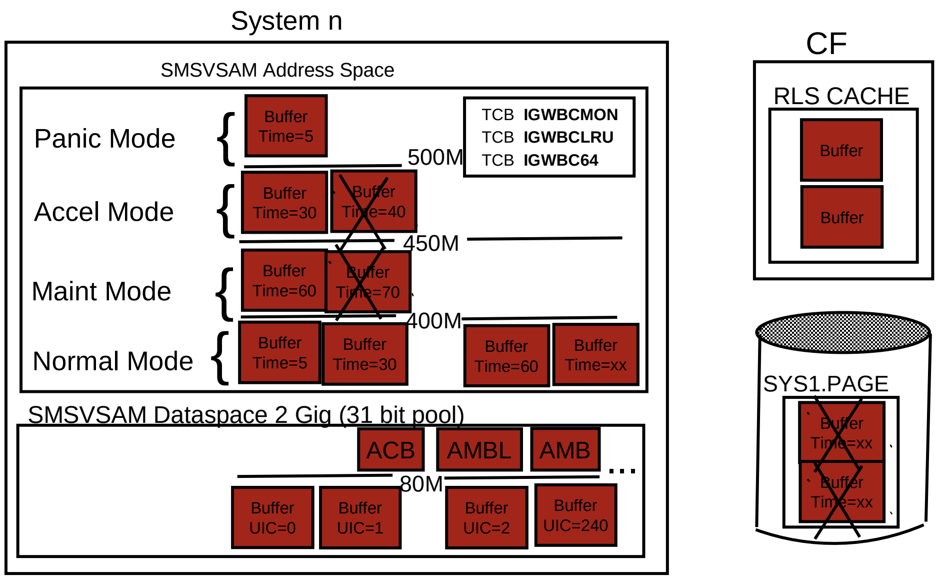
64-bit buffers

- The LRU will release 64 bit buffers as follows:
  - **Normal mode** - Buffers 60 minutes or older will be released
  - **Maintenance mode** - Buffers 60 minutes or older will be released
  - **Accelerated mode** - Buffers 30 minutes are older will be released. Requests for new buffers will first be stolen. If there are no buffers to steal a new get block will be done
  - **Panic mode** - Buffers 5 minutes are older will be released. Requests for new buffers will first be stolen. If there are no buffers to steal, the request will sleep until LRU runs

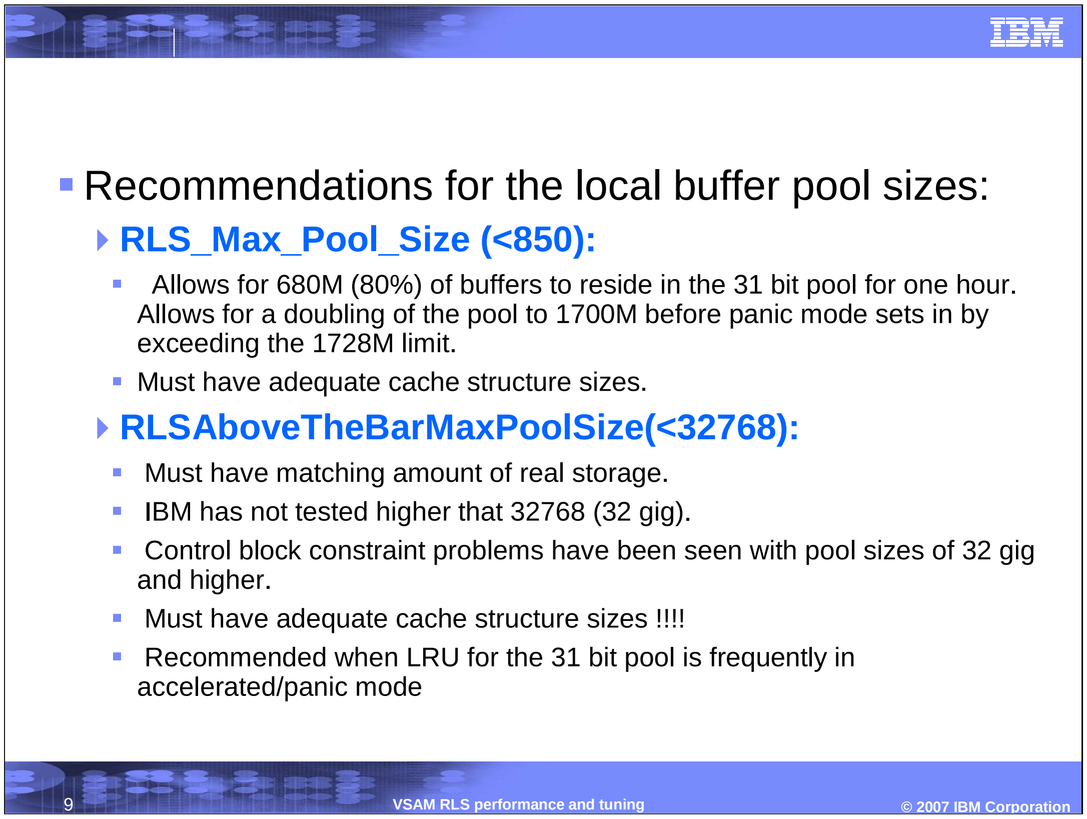
7 VSAM RLS performance and tuning © 2007 IBM Corporation

There is only a subtle difference between 31 bit and 64 bit. The 31 bit LRU was using this count and aging the buffer by one and then compare it with the LRU's criteria. In the 64 bit pool, the count is not used. Instead, timestamps are used, which is a lot easier. The buffers are time-stamped and compared against the current time to determine the time a buffer spent in the pool.

RLSAboveTheBarMaxPoolSize(500) RLS_Max_Pool_Size(100)



(tungsten) This figure graphically shows the previous discussion. For example, the RLS Above The Bar Max Pool Size is set to 500 Meg, which is the minimum size that you can have. The RLS Max Pool Size parameter is set to 100. The some data are using 64 bit pool, and some are using 31 bit pool. This is a picture of what the buffer pool would look like, and what's the LRU is doing. In the top right corner, a couple of tasks or TCB's are shown. Those are the name of the LRU that are running to monitor the 64 bit pool and the 31 bit pool. There is the other TCB called BCMON. This TCB monitors both pools to see what size they are. This TCB also set the parameters to indicate the normal mode, the maintenance mode, and panic mode for those two TCB's. They wakes up every 15 seconds and looks at what mode they are in. They are going to throw out buffers based off the size of the pool and how old these buffers are. With a 500 MB 64 bit pool, 80% of the pool is 400 MB. As long as the buffer total stays below the 400 MB, these buffers can stay in the pool up to one hour. Once those buffers start to exceed the 80% value into the accelerated mode or panic mode, the LRU will begin reducing its criteria and begin throwing out the older buffers. It's the same for the dataspace where a count is used.



- Recommendations for the local buffer pool sizes:
  - **RLS_Max_Pool_Size (<850):**
    - Allows for 680M (80%) of buffers to reside in the 31 bit pool for one hour. Allows for a doubling of the pool to 1700M before panic mode sets in by exceeding the 1728M limit.
    - Must have adequate cache structure sizes.
  - **RLSAboveTheBarMaxPoolSize(<32768):**
    - Must have matching amount of real storage.
    - IBM has not tested higher that 32768 (32 gig).
    - Control block constraint problems have been seen with pool sizes of 32 gig and higher.
    - Must have adequate cache structure sizes !!!!
    - Recommended when LRU for the 31 bit pool is frequently in accelerated/panic mode

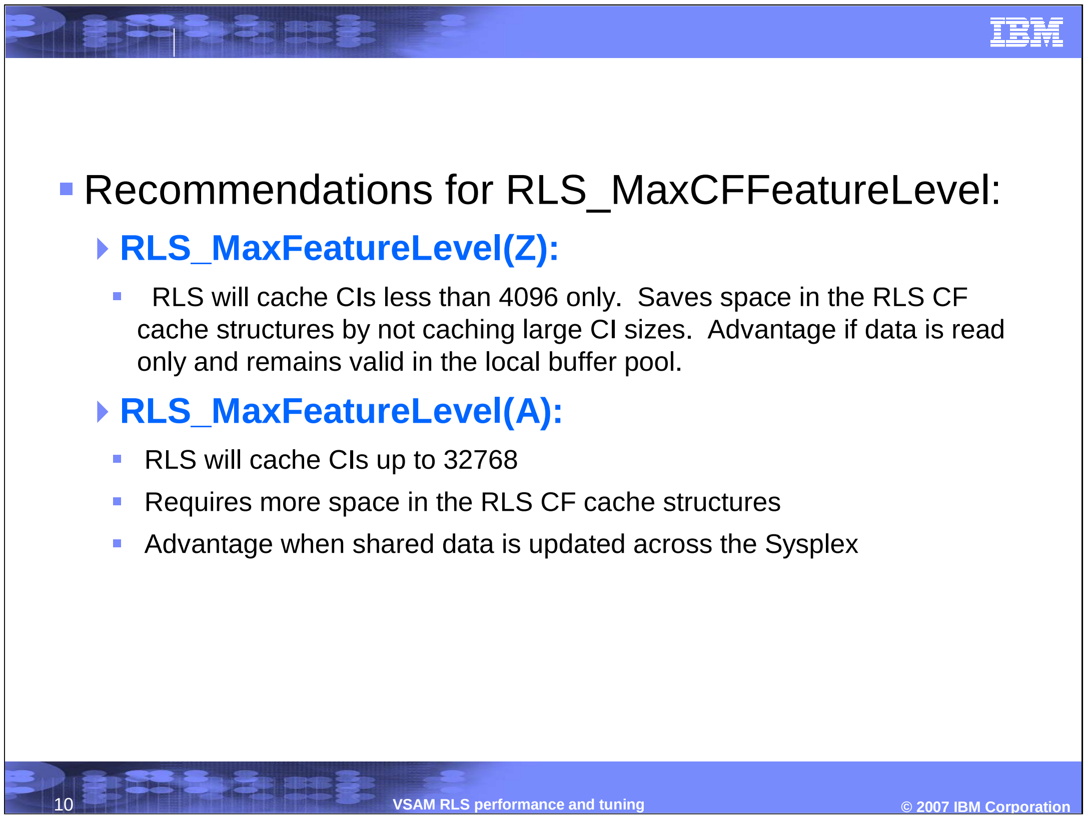
9 VSAM RLS performance and tuning © 2007 IBM Corporation

Here are some recommendations for the local buffer pool sizes.

For the RLS Max Pool Size, The goal is to keep as much data as possible in the pool for an hour. Assume the amount of RLS data that for your datasets is more than 2 gigabytes, which is the usual case. The 31 bit pool can still be used with the high limit of 1728. As soon as the buffer total hits it, the LRU switches to panic mode and all the buffers will be throwing out. Therefore, the buffer total should be put somewhere that can utilize the LRU to your advantage. If you get more than 2 gigabytes of data out there and you are using a 31 bit pool, the pool should be set it around 850 MB. This setting would allow you to keep 80% of 850 MB, or 650 MB, worth of data in the 31 bit pool for one hour. It allows the pool to double in size before the LRU gets to panic mode. When the buffer total is doubled from the 850 MB, you get up to 1700 MB just below that limit of 1728 MB, where the LRU will go to panic mode. This is the best optimal parameter that you can set.

For the Above The Bar Pool Size, it should be set less than 32 gigabytes. Since no test has been done for pool over 32 gigabytes yet. For the 64 bit pool, it is similar to the 31 bit pool that you need to have at least the matching amount of real storage available or you will get into paging situation. To get a big pool of 32 gigabytes, you need at least 32 gigabytes of real storage on your system. Ideally you should have more available real storage. You don't want to have exactly equal. In order for you to get up to these big pool sizes, you will need to have enough real storage. There are reasons for keeping the pool size below 32 gigabytes. There are control blocks that map these buffers that are still in the 31 bit storage. There can be constraint problems to these control blocks to support such a big pool. There have been constraint problems to exceed the 32 gig right now. The cache structure size must be matching your pool sizes. With a big pool, you need a have really big cache structures. Therefore, it is not recommended to exceed the 32 gigabytes limit.

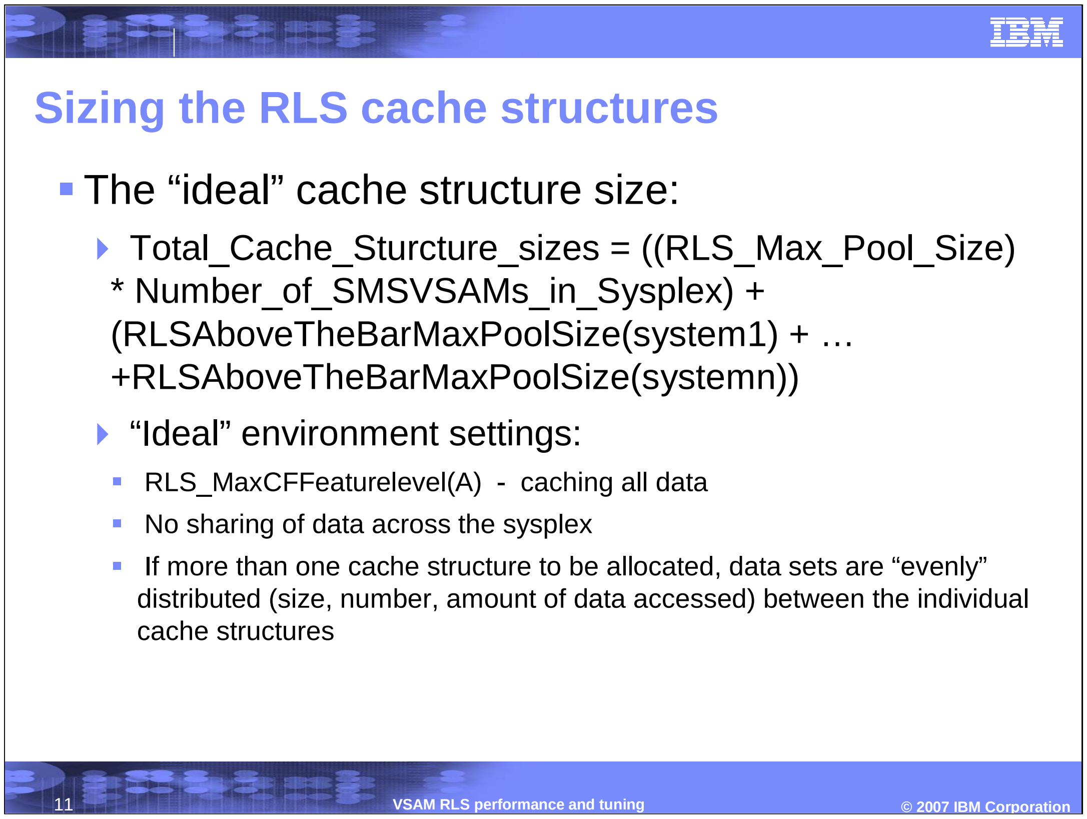
If now you have a case where you constantly exceeding your limit of the 31 bit pool, then maybe that's the time you want to consider using the 64 bit pool.



- Recommendations for RLS_MaxCFFeatureLevel:
  - **RLS_MaxFeatureLevel(Z):**
    - RLS will cache CIs less than 4096 only. Saves space in the RLS CF cache structures by not caching large CI sizes. Advantage if data is read only and remains valid in the local buffer pool.
  - **RLS_MaxFeatureLevel(A):**
    - RLS will cache CIs up to 32768
    - Requires more space in the RLS CF cache structures
    - Advantage when shared data is updated across the Sysplex

10 VSAM RLS performance and tuning © 2007 IBM Corporation

Here are the recommendations for RLS_MaxCFFeatureLevel parameter. The feature level Z was first came out, which limits the cache CI that are in the coupling facility to 4 KB or less. If you can afford the space in the coupling facility, then you can turn on the feature level A, and you can cache all the buffers with size up to 32 KB. Caching the data helps avoiding the possibility of having to go to the DASD. Therefore, caching is certainly going to help in shortening that I/O path, but with a cost of available space in the coupling facility.

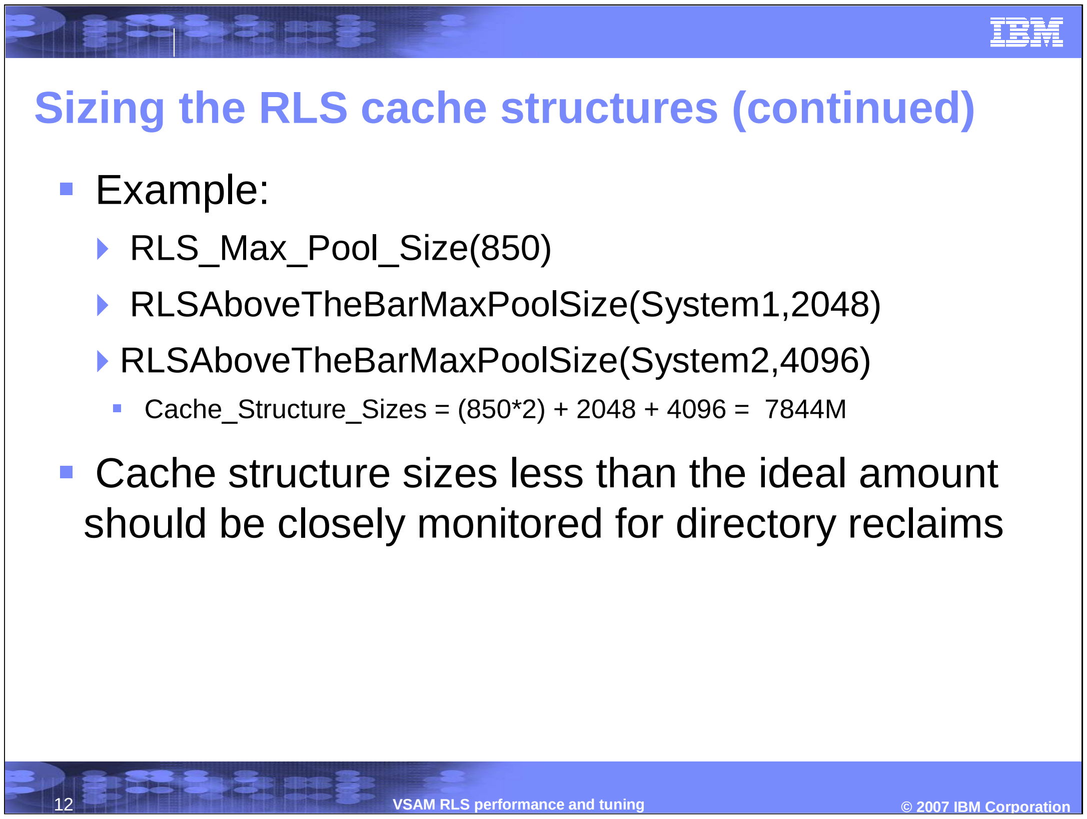


Sizing the RLS cache structures

- The “ideal” cache structure size:
  - Total_Cache_Sturcture_sizes = ((RLS_Max_Pool_Size) * Number_of_SMSVSAMs_in_Sysplex) + (RLSAboveTheBarMaxPoolSize(system1) + … +RLSAboveTheBarMaxPoolSize(systemn))
  - “Ideal” environment settings:
    - RLS_MaxCFFeaturelevel(A) - caching all data
    - No sharing of data across the sysplex
    - If more than one cache structure to be allocated, data sets are “evenly” distributed (size, number, amount of data accessed) between the individual cache structures

11 VSAM RLS performance and tuning © 2007 IBM Corporation

This slide shows how to size the cache structure. There is a direct relationship between the buffer pool size and the cache structure size. You have to figure out how big of a buffer pool size desired first. For example, if you decided to keep the 500MB buffer in the buffer pool for one hour, you'll need at least 500 MB for the total pool size in the sysplex. This would be the ideal situation. Then you will need to size your cache structure off of that, because if your cache structure is not big enough, then the buffers are going to become invalid. This situation occurs because there must be the corresponding directory element in the cache structure that matched the buffer you have in the buffer pool. So if you limit the cache structure, you’re going to tell the LRU that the buffer is invalid, and the LRU will throw that buffer out. The goal is to get valid buffers in our pool for 60 minutes. It would be defeating the purpose by having too small of a cache structure. There is a little formula here for you to use. Basically, what it says is that you add up all your buffer pools and that’s how big the total of your cache structures should be. For example, if there are two cache structures, together they should be equal to the sum of all your buffer pools in the sysplex. This is the best case scenario.

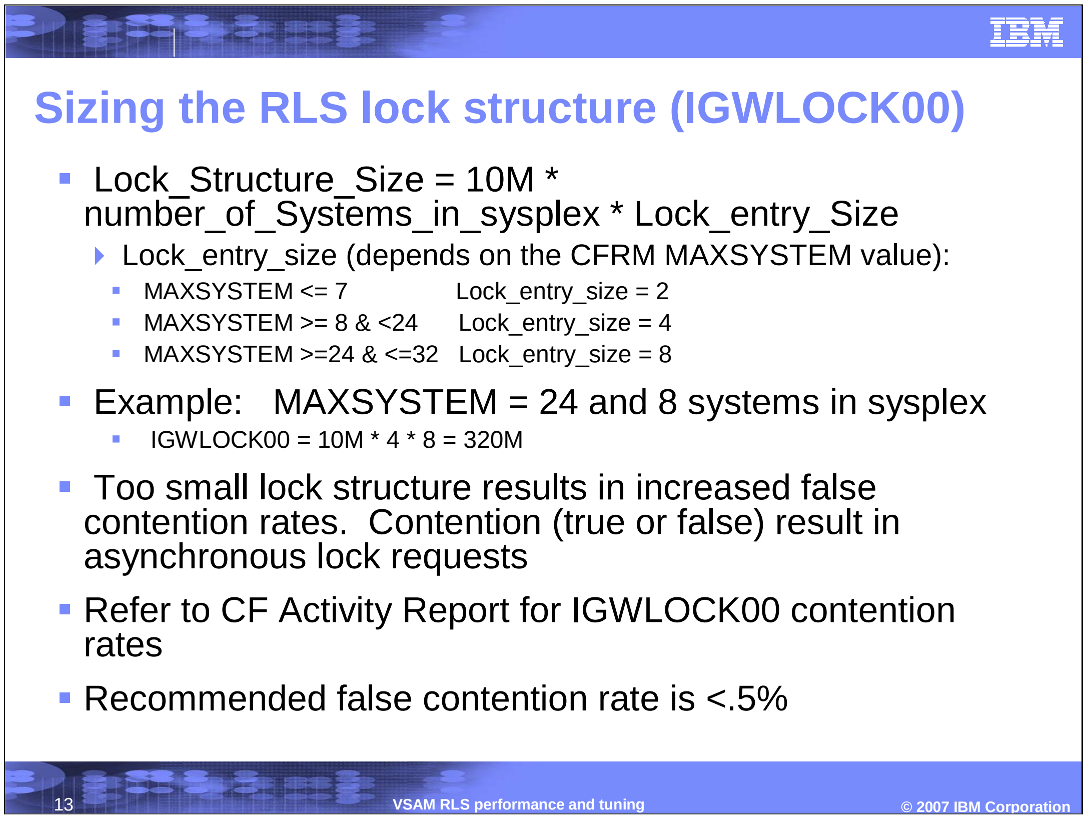


Sizing the RLS cache structures (continued)

- Example:
  - RLS_Max_Pool_Size(850)
  - RLSAboveTheBarMaxPoolSize(System1,2048)
  - RLSAboveTheBarMaxPoolSize(System2,4096)
    - Cache_Structure_Sizes = (850*2) + 2048 + 4096 = 7844M
- Cache structure sizes less than the ideal amount should be closely monitored for directory reclaims

12 VSAM RLS performance and tuning © 2007 IBM Corporation

Here is just an example. If you have a 850 MB of buffer in the 31 bit pools, and you have 2 systems in your sysplex. One of them is going to have a 2 GB 64 bit pool. One of them is going to have 4 GB 64 bit pool. The total size that you would need for you cache structures should be the sum of all the buffer pools, which is 7.8 GB.



Sizing the RLS lock structure (IGWLOCK00)

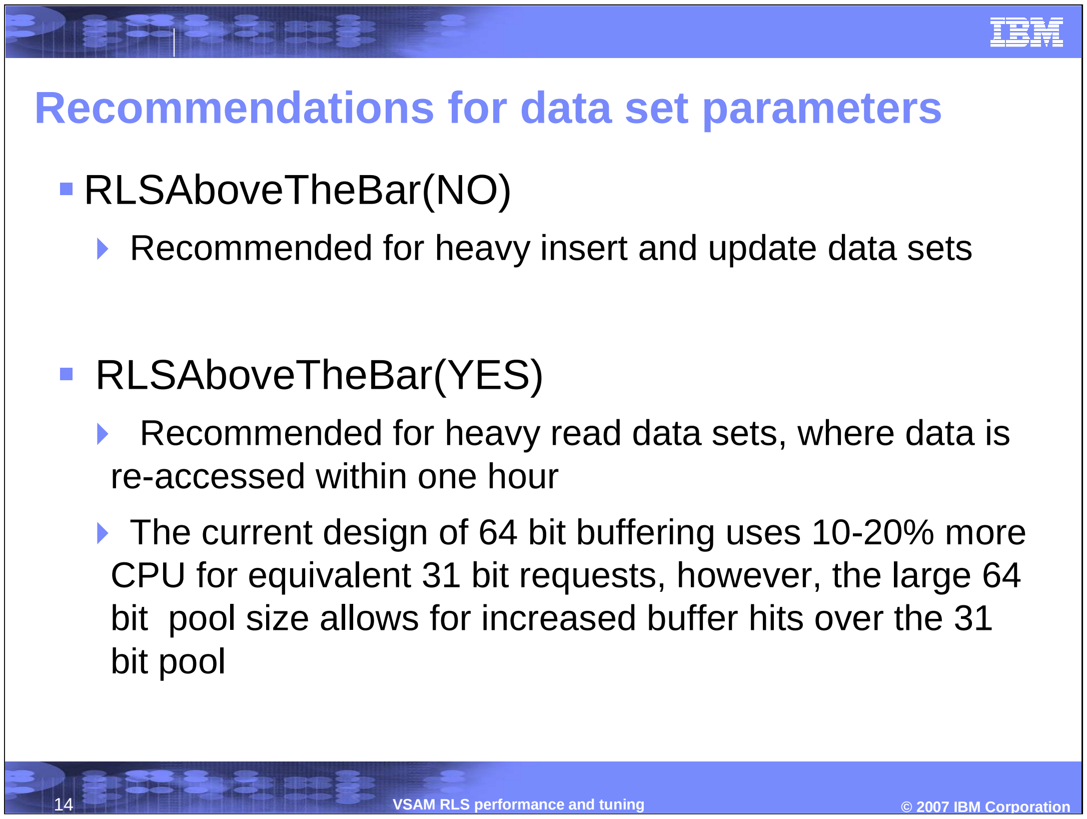
- Lock_Structure_Size = 10M * number_of_Systems_in_sysplex * Lock_entry_Size
  - Lock_entry_size (depends on the CFRM MAXSYSTEM value):

MAXSYSTEM <= 7	Lock_entry_size = 2
MAXSYSTEM >= 8 & <24	Lock_entry_size = 4
MAXSYSTEM >=24 & <=32	Lock_entry_size = 8

- Example: MAXSYSTEM = 24 and 8 systems in sysplex
  - IGWLOCK00 = 10M * 4 * 8 = 320M
- Too small lock structure results in increased false contention rates. Contention (true or false) result in asynchronous lock requests
- Refer to CF Activity Report for IGWLOCK00 contention rates
- Recommended false contention rate is <.5%

13 VSAM RLS performance and tuning © 2007 IBM Corporation

Sizing a lock structure is a lot simpler. Lock structure is not related to the size of the buffer and the cache. It's a completely different item. This formula came right out of the storage and administration reference book. So you can read that as well. Basically, the lock structure size depends on the number of systems on the sysplex, and on the actual lock entry sizes. Those are determined by XCF and by snack system parameters that you specified in your CFRM policy. There would be a problem when the lock structure is too small. The thing that you might be looking at is your false contention rate. If you have a false contention rate greater than 0.5 %, that's an indication of a lock structure that is too small. You might want to make it bigger.

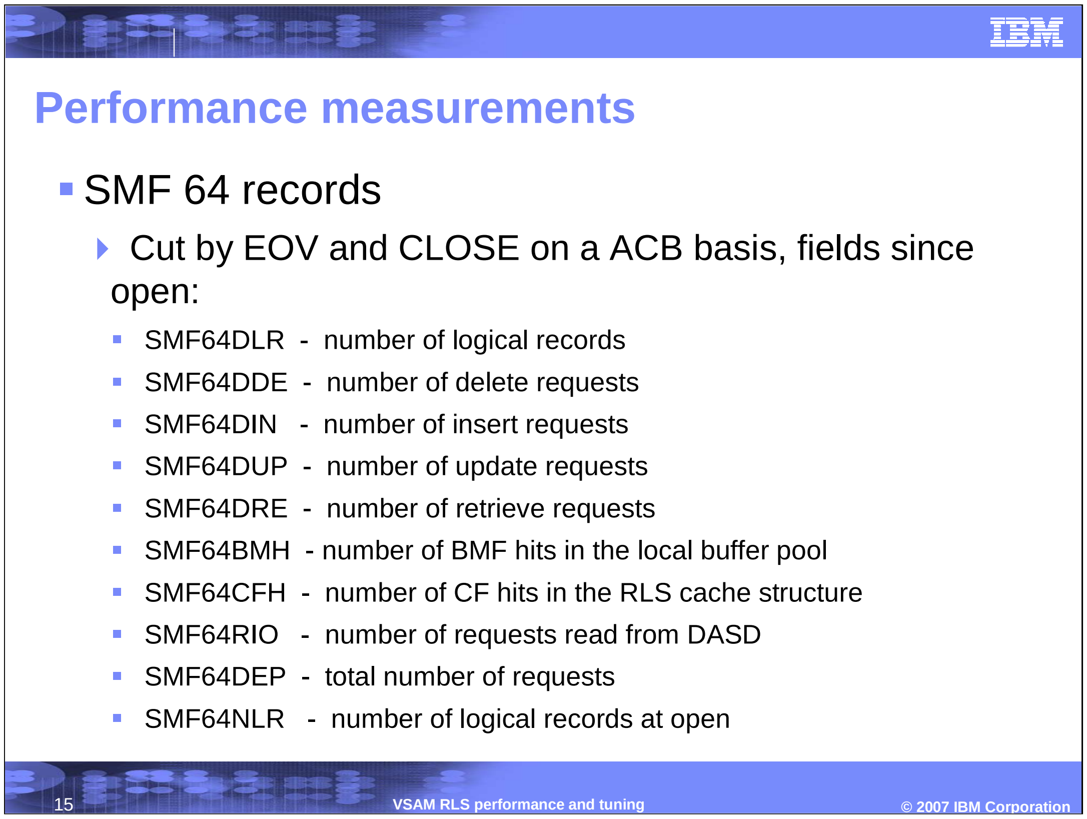


Recommendations for data set parameters

- RLSAboveTheBar(NO)
  - Recommended for heavy insert and update data sets
- RLSAboveTheBar(YES)
  - Recommended for heavy read data sets, where data is re-accessed within one hour
  - The current design of 64 bit buffering uses 10-20% more CPU for equivalent 31 bit requests, however, the large 64 bit pool size allows for increased buffer hits over the 31 bit pool

14 VSAM RLS performance and tuning © 2007 IBM Corporation

As far as recommendations for the data set parameters, the RLSAboveTheBar(NO) should be used for data sets that got updated all the time. So in this case only the 31 bit pool will be used. The reason for that is that the 64 bit path uses a little more CPU than the 31 bit path. If you will not be getting buffer hit and just updating your data, then it's best to avoid the 64 bit path due to the CPU cost. However if you will read a lot of data and you will be re-reading it within an hour work time frame, then the bigger the pool is the better. Using the 64 bit pool is recommended for a large data set when you are trying to re-access the data within one hour. We will be planning enhancements to reduce the little more CPU cost in a 64 bit path.

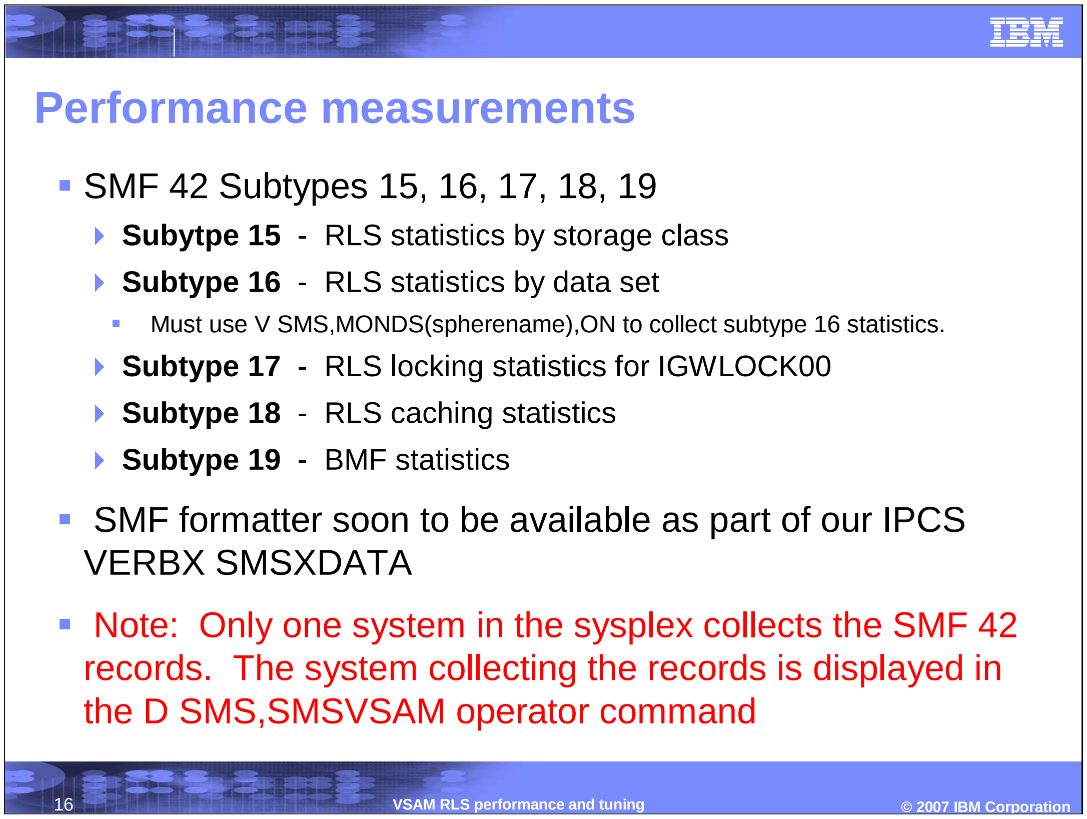


Performance measurements

- SMF 64 records
  - Cut by EOV and CLOSE on a ACB basis, fields since open:
    - SMF64DLR - number of logical records
    - SMF64DDE - number of delete requests
    - SMF64DIN - number of insert requests
    - SMF64DUP - number of update requests
    - SMF64DRE - number of retrieve requests
    - SMF64BMH - number of BMF hits in the local buffer pool
    - SMF64CFH - number of CF hits in the RLS cache structure
    - SMF64RIO - number of requests read from DASD
    - SMF64DEP - total number of requests
    - SMF64NLR - number of logical records at open

15 VSAM RLS performance and tuning © 2007 IBM Corporation

Once you have gone through all your sizing and your system all setup great. How do you prove that you are doing this? Are you actually getting buffer hit? How many much data are you reading within an hour? You can figure all that out by looking at the SMF data. You can use the SMF 64 records. The SMF 64 records are cut by end of volume and close. For each time you extend a dataset, you get a 64 record cut. Each time you close an ACB, you get one. So ideally you want to look at the close SMF 64 records. There are these fields that on this slide that show you the number logical records in a data set, the number of deleted records, inserted, updated, retrieved, and so forth. There are some fields specifically for RLS. There is a number of BMF hits in the local buffer pool, the number of cache hits in the cache structure, and the number of read requests that went to DASD. Then, there is the total number of requests. There is the number of logical records open. These are help information. When you open it? How many are there close it? These are all cut on the time frame. By looking at the timestamps, you can see how much data you have read during this particular time frame. You can see what kind of buffer hits you are getting and so forth by looking at these fields.

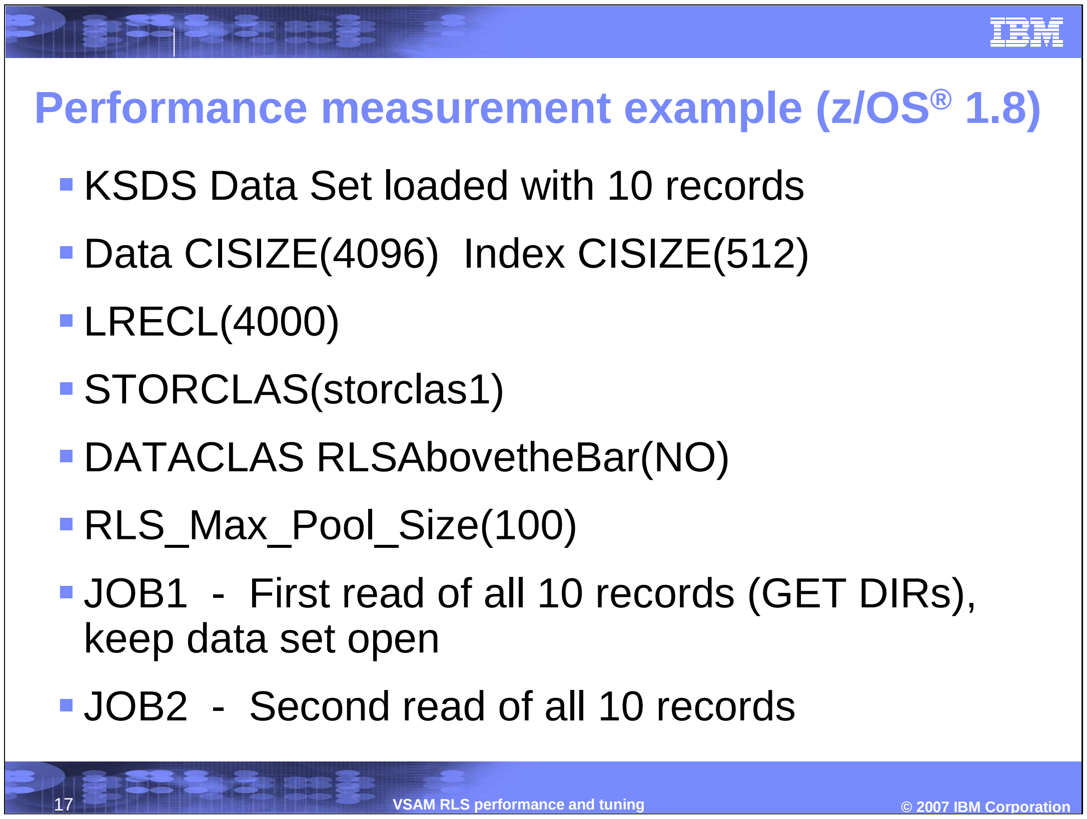


Performance measurements

- SMF 42 Subtypes 15, 16, 17, 18, 19
  - **Subtype 15** - RLS statistics by storage class
  - **Subtype 16** - RLS statistics by data set
    - Must use V SMS,MONDS(spherename),ON to collect subtype 16 statistics.
  - **Subtype 17** - RLS locking statistics for IGWLOCK00
  - **Subtype 18** - RLS caching statistics
  - **Subtype 19** - BMF statistics
- SMF formatter soon to be available as part of our IPCS VERBX SMSXDATA
- Note: Only one system in the sysplex collects the SMF 42 records. The system collecting the records is displayed in the D SMS,SMSVSAM operator command

16 VSAM RLS performance and tuning © 2007 IBM Corporation

In addition to the SMF 64 records, there are the SMF 42 records, which has five subtypes, 15 to 19. Each of those keeps some different data about RLS statistics. Subtype 15 keeps statistics by storage class. Subtype 16 statistics by each individual dataset. However, in order to get to this subtype, you have to actually issue a command to turn it on first. Subtype 17 is all statistics related to locking SMLS. Subtype 18 is caching statistics, which involves with the SCM component. Subtype 19 is BMF statistics, which involves the LRUs. So you can find the information there. There is a SMF formatter soon to be available. Another thing to know about SMF 42 data is that this data is sysplex wide. These records also provide tracking across the sysplex. One system is designated to collect the data. When you look at the SMF 42 data, you'll have to go to the system which is in charge of collecting the data. So you have to define that with the D SMS,SMSVSAM,ALL command. It will point out which system is collecting the data. Therefore, that is the system you want to go to get the data from.

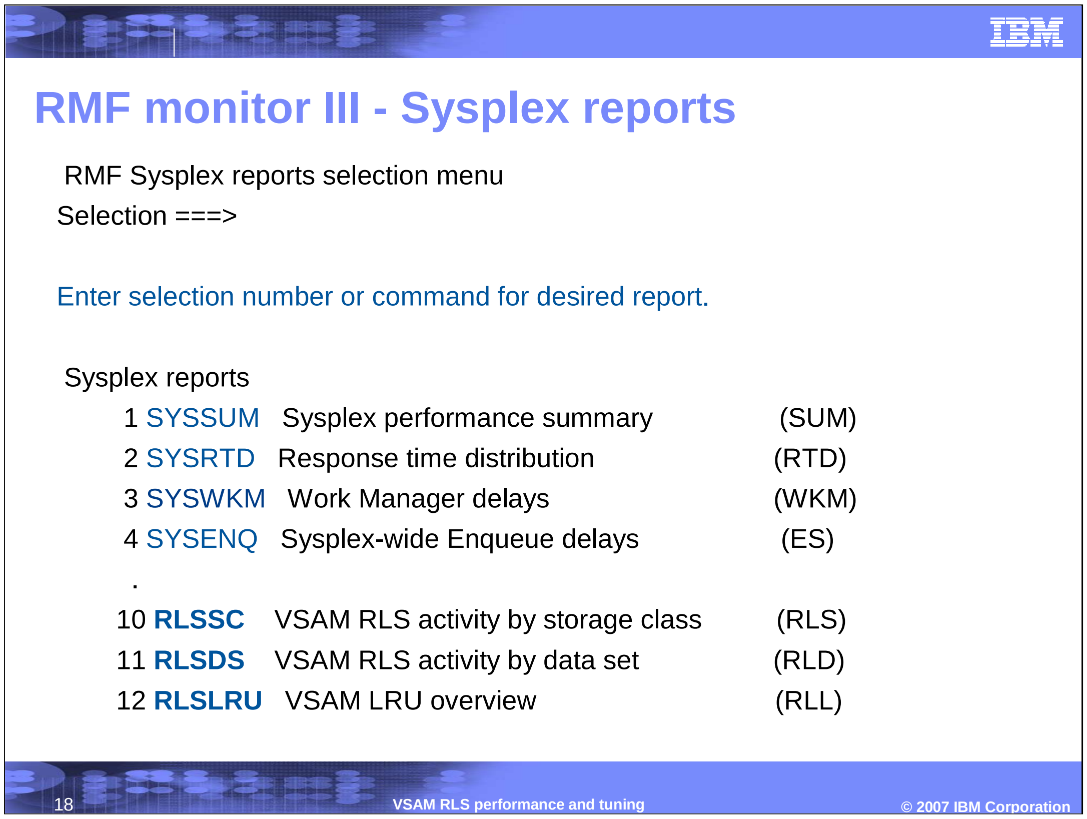


Performance measurement example (z/OS® 1.8)

- KSDS Data Set loaded with 10 records
- Data CISIZE(4096) Index CISIZE(512)
- LRECL(4000)
- STORCLAS(storclas1)
- DATACLAS RLSAbovetheBar(NO)
- RLS_Max_Pool_Size(100)
- JOB1 - First read of all 10 records (GET DIRs), keep data set open
- JOB2 - Second read of all 10 records

17 VSAM RLS performance and tuning © 2007 IBM Corporation

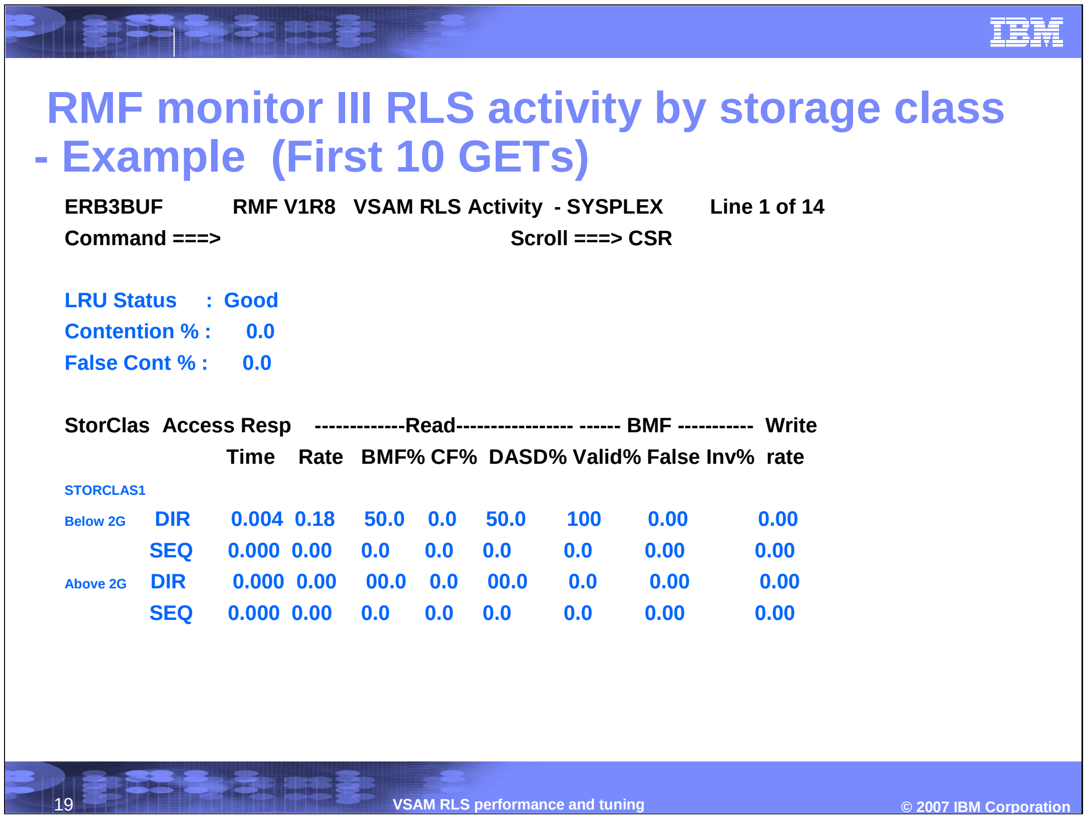
Here is an example so that you can see how this works. This is a really little simple example. In this example, a KSDS data set is used, and there are only 10 records in it, with a data CISIZE of 4096 bytes or 4 KB and index size of 512 bytes. Each of the records is 4000 bytes. That means there is one data record per CI. It only fits a record with CISIZE of 4096 bytes. The dataset got the storage class of STORCLAS1. Since this is a very small data set, the Above the Bar option will not be used, instead the 31 bit pool will be used. The RLS Max Pool Size is set to 100 MB. If you add up your 10 records with 4096, it will come up less than 100 MB. Then, a job will be sent to read all 10 records, do a Get direct, read through all 10 records, and leave the data set open. Now, the job could close it, and re-open it within 10 minutes and still see all those 10 CI in the buffer pool. Just for safety sake, the dataset is left open to keep those buffers there for one hour. Those buffers need to be re-read within one hour to get a buffer hit. Then, there is the job number 2 to re-read all those 10 records. So the first job had to go in and read them all in into the buffer pool. The second job is going to take advantage of the fact that they are all in the buffer pool and get hopefully 100% buffer hit. The data set statistics can be seen by using the SMF data.



RMF monitor III - Sysplex reports

```
 RMF Sysplex reports selection menu
Selection ===>

Enter selection number or command for desired report.

 Sysplex reports
        1 SYSSUM   Sysplex performance summary          (SUM)
        2 SYSRTD   Response time distribution           (RTD)
        3 SYSWKM   Work Manager delays                  (WKM)
        4 SYSENQ   Sysplex-wide Enqueue delays          (ES)
         .
       10 RLSSC    VSAM RLS activity by storage class   (RLS)
       11 RLSDS    VSAM RLS activity by data set        (RLD)
       12 RLSLRU   VSAM LRU overview                    (RLL)
```

You can use the RMF online reports to see what happened. If you go to RMF monitor 3, and then you go to the sysplex report, you will get to this panel. On the bottom, three are the RLS related reports. Option 10 shows the RLS activities by storage class. Option 11 is by data set. Option 12 is an overview of the LRU. You have these 3 reports that are very helpful at monitoring performance. You would have to turn on the collection of SMF 42 by data set before you could use the report. If you did not, you could always go to the storage class, which is going to show all data sets for this storage class. There is only one data set in the storage class.

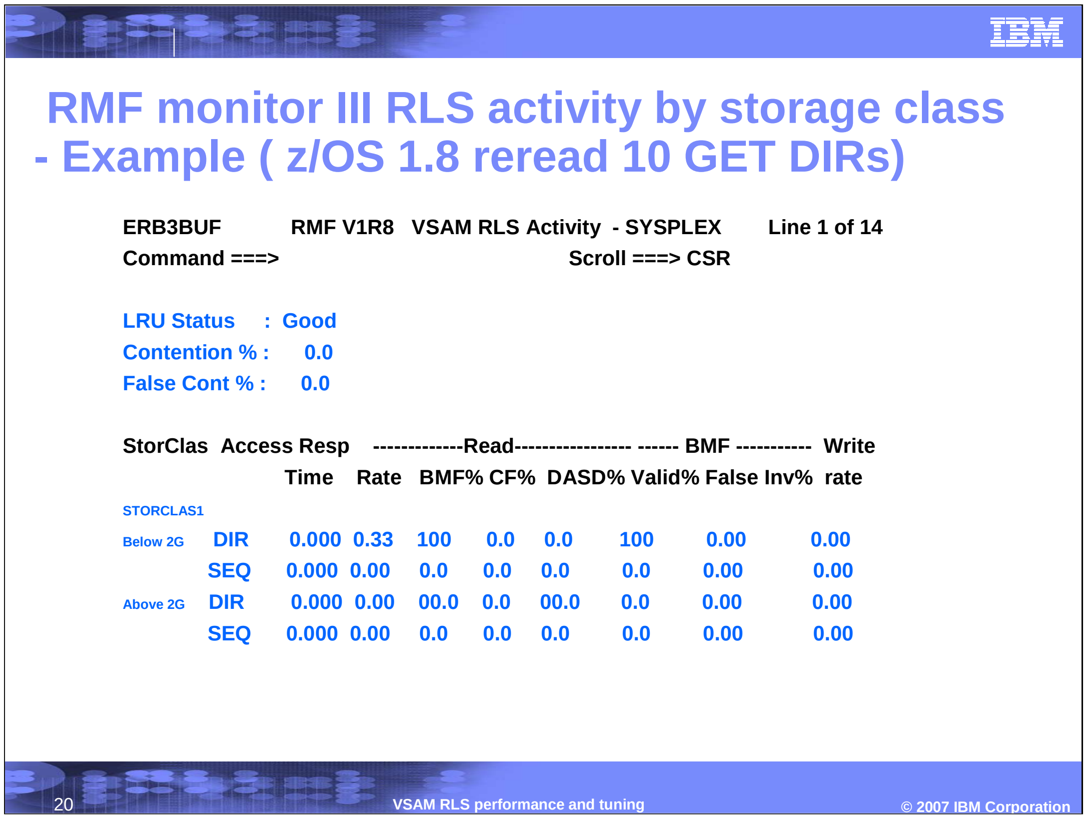


RMF monitor III RLS activity by storage class - Example (First 10 GETs)

```
ERB3BUF          RMF V1R8   VSAM RLS Activity  - SYSPLEX       Line 1 of 14
Command ===>                                         Scroll ===> CSR

LRU Status     :  Good
Contention % :       0.0
False Cont % :      0.0

StorClas  Access Resp    -------------Read----------------- ------ BMF ----------  Write
                       Time    Rate   BMF% CF%  DASD% Valid% False Inv%  rate
STORCLAS1
Below 2G     DIR      0.004  0.18    50.0   0.0    50.0      100       0.00         0.00
             SEQ     0.000  0.00    0.0    0.0    0.0        0.0       0.00         0.00
Above 2G     DIR      0.000  0.00    00.0   0.0    00.0      0.0       0.00         0.00
             SEQ     0.000  0.00    0.0    0.0    0.0        0.0       0.00         0.00
```

19 VSAM RLS performance and tuning © 2007 IBM Corporation

This is a report by storage class. This is the interval for the first job ran. Job 1 ran from this particular time frame. It is the interval that SMF data was cut during that time. This is for those first 10 gets. On the bottom, the storage class name is shown. It will show you whether it was a 31 bit pool or a 64 bit pool was in use. In this case, it is using the 31 bit pool with a direct Get request. There are some activities in that first line. If you look under Read, BMF percentage is 50%. If you look under DASD, it says 50%. And if you look under valid percentage, it says 100%. These data indicates 50% of the buffering requests submitted and found in the buffer pool and 50% required access to DASD. Of the 50% that found in the buffer pool, they were 100% valid. Why is there any buffer hit when it was the first read? The reason is because the index records are read in and re-read. For each Get request, the index was searched, and at that point, the job keep getting buffer hit. That’s why it ended up at 50%. In the SMF 64 records, the index was getting 100% hit. The data was getting zero hit. That explains how it added up 50%. Under the BMF, there is this false invalid percentage. This is a really important data to watch on, because what that tells you is whether your cache structure was too small. If you see anything over there, you have to think about that your cache structure was too small. The way that field works is that to look for these buffers in the buffer pool, it will go to ask the cache structure for a valid buffer. If the cache structure is too small, and the directory entry is assigned in another buffer because directory entry was ran out, then the buffer becomes invalid because the directory entry was stolen. Therefore, that value indicates the time a buffer went invalid due to a field in a cache structure. This example has showed you that you do not want to have XCF stealing directory entries due to not enough space to keep track of all the valid buffers. That is an indication of when XCF is stealing one of the valid buffer away.

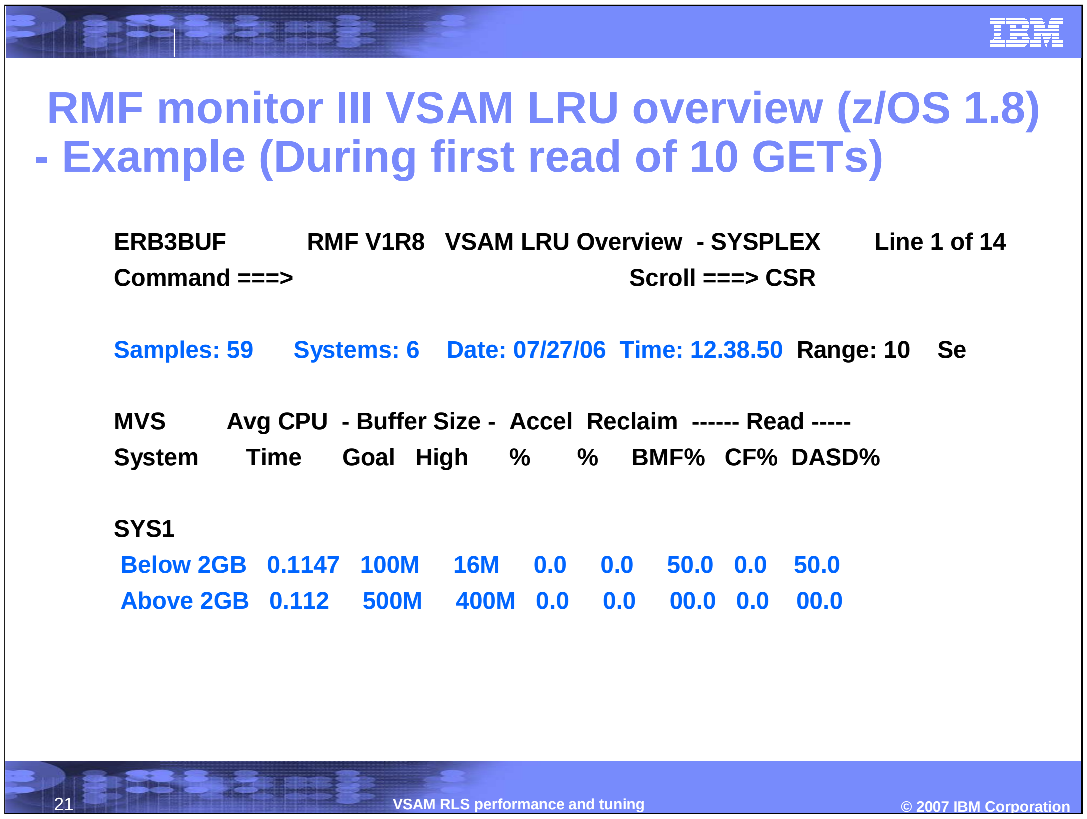


RMF monitor III RLS activity by storage class - Example (z/OS 1.8 reread 10 GET DIRs)

```
ERB3BUF        RMF V1R8   VSAM RLS Activity  - SYSPLEX       Line 1 of 14
Command ===>                                   Scroll ===> CSR

LRU Status     :  Good
Contention % :      0.0
False Cont % :      0.0

StorClas  Access Resp    -------------Read----------------- ------ BMF -----------  Write
                 Time    Rate   BMF% CF%  DASD% Valid% False Inv%  rate
STORCLAS1
Below 2G     DIR     0.000  0.33    100      0.0     0.0       100        0.00          0.00
             SEQ     0.000  0.00    0.0      0.0     0.0       0.0        0.00          0.00
Above 2G     DIR     0.000  0.00    00.0     0.0     00.0      0.0        0.00          0.00
             SEQ     0.000  0.00    0.0      0.0     0.0       0.0        0.00          0.00
```

20 VSAM RLS performance and tuning © 2007 IBM Corporation

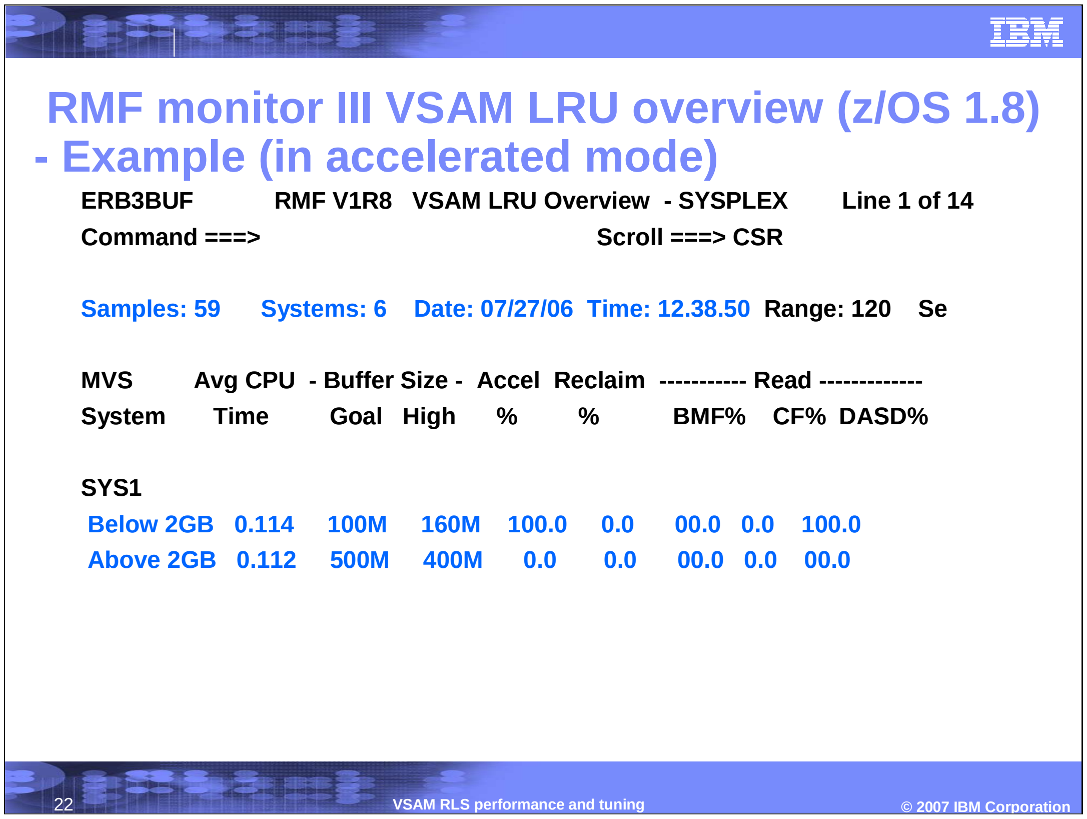
The second job re-reads the 10 Get's. All the 10 records should be in the buffer pool, and the second job should be able to get them all. As shown in the RFM report, the second job got 100% buffer reads and 100% valid. This is the best case. That means every request ran in the 10 thousandths of a second. That is really fast.



RMF monitor III VSAM LRU overview (z/OS 1.8) - Example (During first read of 10 GETs)

```
ERB3BUF       RMF V1R8   VSAM LRU Overview  - SYSPLEX      Line 1 of 14
Command ===>                                  Scroll ===> CSR

Samples: 59    Systems: 6    Date: 07/27/06  Time: 12.38.50  Range: 10    Se

MVS        Avg CPU  - Buffer Size -  Accel  Reclaim  ------ Read -----
System      Time     Goal   High      %       %      BMF%   CF%  DASD%

SYS1
 Below 2GB  0.1147  100M    16M     0.0     0.0     50.0   0.0    50.0
 Above 2GB  0.112   500M    400M    0.0     0.0     00.0   0.0    00.0
```

This LRU report shows the activities during the interval where the job one have just read the first 10 Get's. This report shows each system in the sysplex. In this case, there is only system one. It shows you what the LRU was doing for the 31 bit pool and for the 64 bit pool. It shows that 100 MB was specified as the goal for the 31 bit pool. You want to stay below 80% of 100 MB. Obviously 16 MB is way below 80%. So the LRU will always be in the normal mode. That means those 10 buffers can only be kept for one hour. This data set has a 4K data CI and a 512 K index CI size. There are two little mini-pools created within the 31 bit pool. One is the 4 K buffer pool, and the other is a 2 K buffer pool, which is a minimum size. It comes out to be 16 MB even though only a small portion of the pool is used. The reason is that pre-allocation of buffer has occurred, which may come out a bit larger than you really use. In this case, it came out to be about 16 MB. An extent for this pool is pre-allocated.

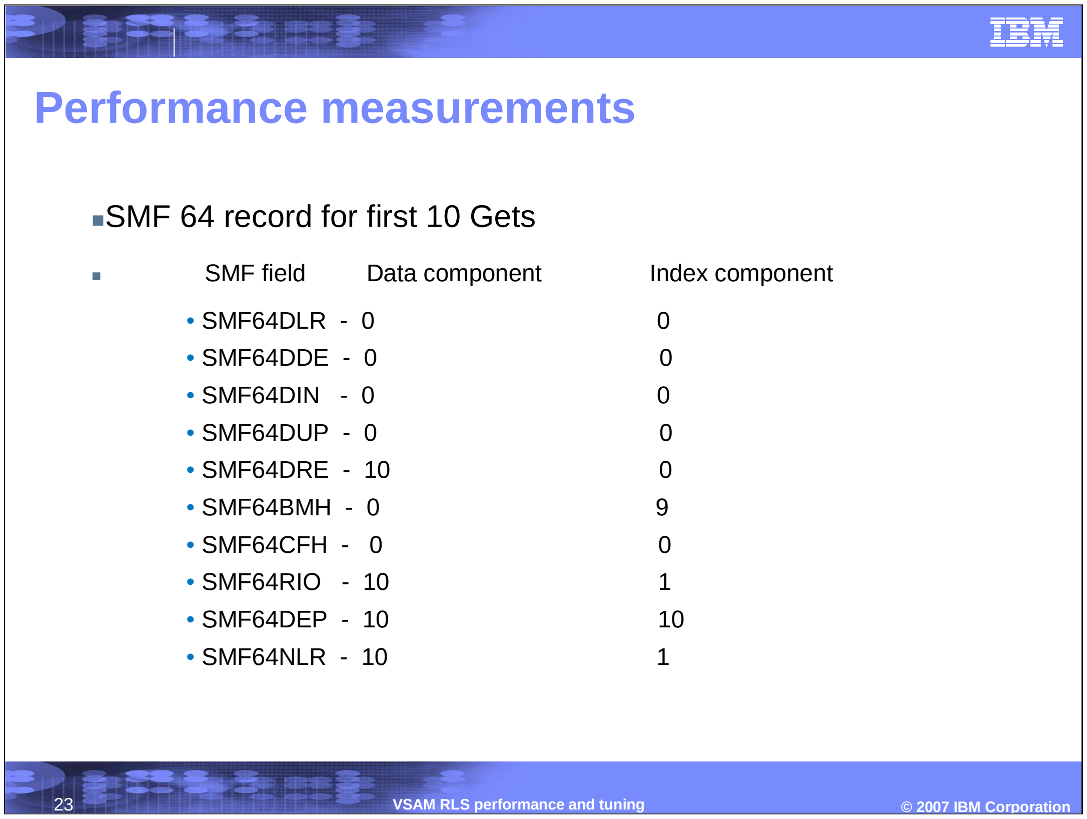


RMF monitor III VSAM LRU overview (z/OS 1.8) - Example (in accelerated mode)

```
ERB3BUF        RMF V1R8   VSAM LRU Overview  - SYSPLEX       Line 1 of 14
Command ===>                                       Scroll ===> CSR

Samples: 59     Systems: 6    Date: 07/27/06  Time: 12.38.50  Range: 120    Se

MVS        Avg CPU  - Buffer Size -  Accel  Reclaim  ----------- Read -------------
System      Time      Goal   High     %       %        BMF%    CF%  DASD%

SYS1
 Below 2GB   0.114    100M    160M   100.0    0.0     00.0   0.0    100.0
 Above 2GB   0.112    500M    400M    0.0     0.0     00.0   0.0    00.0
```

22   VSAM RLS performance and tuning   © 2007 IBM Corporation

This example shows the activities when the buffers are read after one hour, which pushes the LRU out of the normal mode. In this example, there is the 2 GB 31 bit pool with a goal of 100 MB. During this interval, the buffer pool got up to 160 MB. That is well over the 80% of normal mode, which pushes the LRU into the accelerated mode. As a result, the report shows that during this interval, for 100% of the time, the LRU was in accelerated mode. That means the LRU is starting to toss out buffers that are older than one hour.



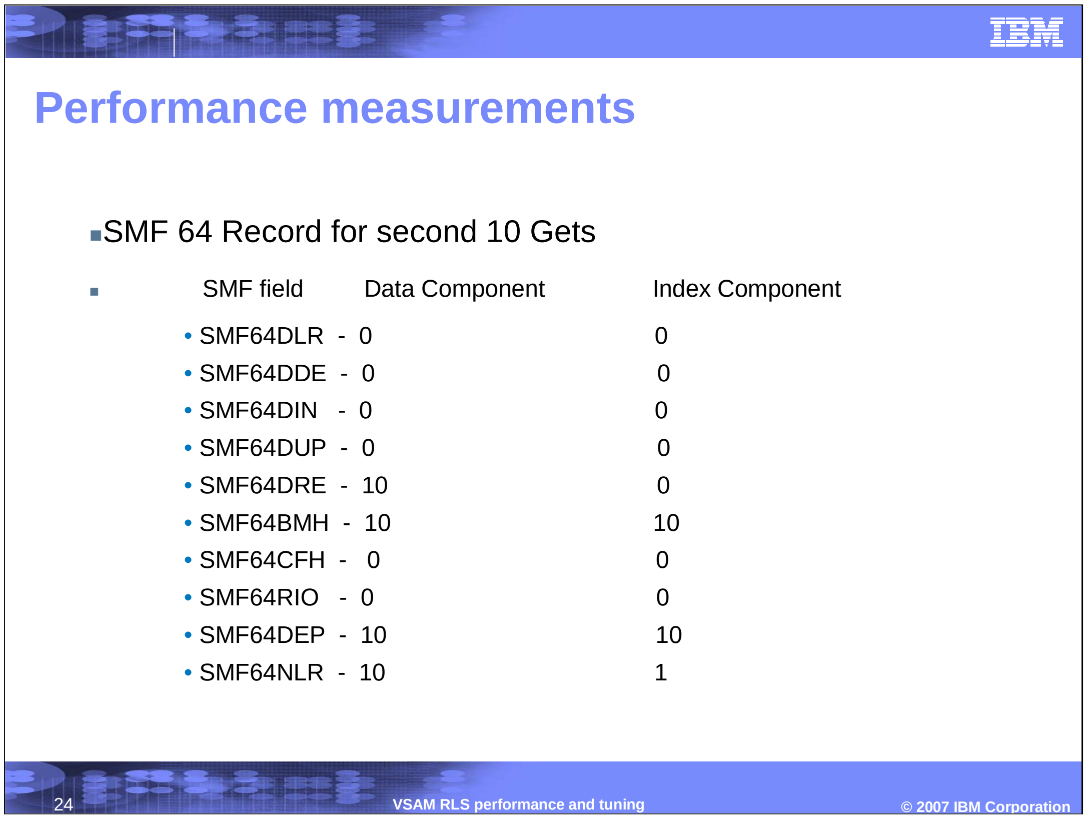
Performance measurements

- SMF 64 record for first 10 Gets

SMF field	Data component	Index component
SMF64DLR	0	0
SMF64DDE	0	0
SMF64DIN	0	0
SMF64DUP	0	0
SMF64DRE	10	0
SMF64BMH	0	9
SMF64CFH	0	0
SMF64RIO	10	1
SMF64DEP	10	10
SMF64NLR	10	1

23 VSAM RLS performance and tuning © 2007 IBM Corporation

This example continues on with the previous two jobs. The first job is doing the first 10 Gets. It was 50% BMF read due to the index hit. With the report by data set, you would see a broken down by data and index component, and you would have seen why you would have gotten the 50% over all value. The number of records read was 10, and the number of DASD read, the RIO, was 10. The job 1 had to go to the DASD 10 times to read the records. Where as if you look at the index component, and you got 0 buffer hit and 0 cache hit. In the index component, the job 1 had to go once to the DASD, because it was such a tiny data set that there is only one level index structure. Therefore, it just had to go get that level one index record and to the DASD onetime. However, nine other times got buffer hits. The averaged value turns out to be 50%. For the second 10 reads, the SMF 64 records shows 10 buffer hits for all 10 read requests. There are 10 logical records in the data set so forth.



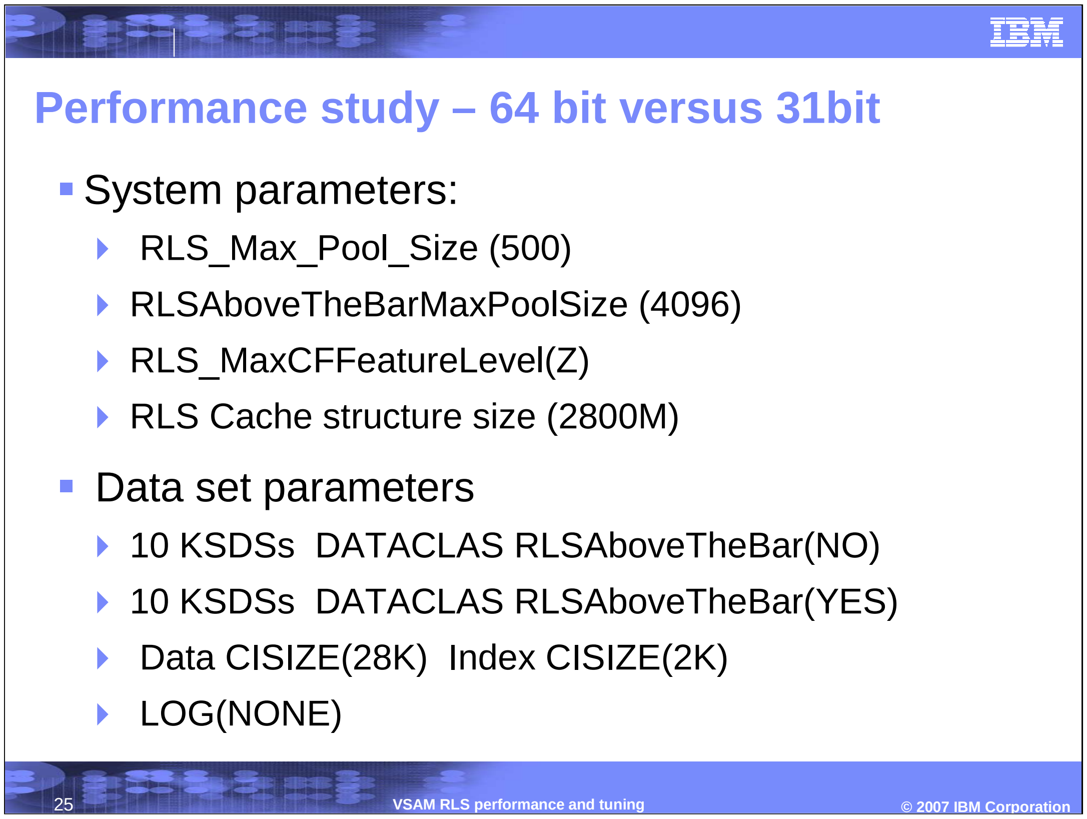
Performance measurements

- SMF 64 Record for second 10 Gets

SMF field	Data Component	Index Component
SMF64DLR	0	0
SMF64DDE	0	0
SMF64DIN	0	0
SMF64DUP	0	0
SMF64DRE	10	0
SMF64BMH	10	10
SMF64CFH	0	0
SMF64RIO	0	0
SMF64DEP	10	10
SMF64NLR	10	1

24 VSAM RLS performance and tuning © 2007 IBM Corporation

This is an example of the SMF 64 record for the second 10 Get's. This example tries compare 64 bit with 31 bit so see how much with a bigger buffer pool would have helped you. The system was setup with RLS Max Pool Size of 500 MB for the 64 bit pool and with a 4 GB 64 bit pool. The system was set to feature level Z, which only caches CI less than 4 K. There was a 2.8 GB cache structure. The data set was defined to be KSDS. As a result, there are 10 records to be in the 31 bit pool, and there are 10 more to be in the 64 bit pool.

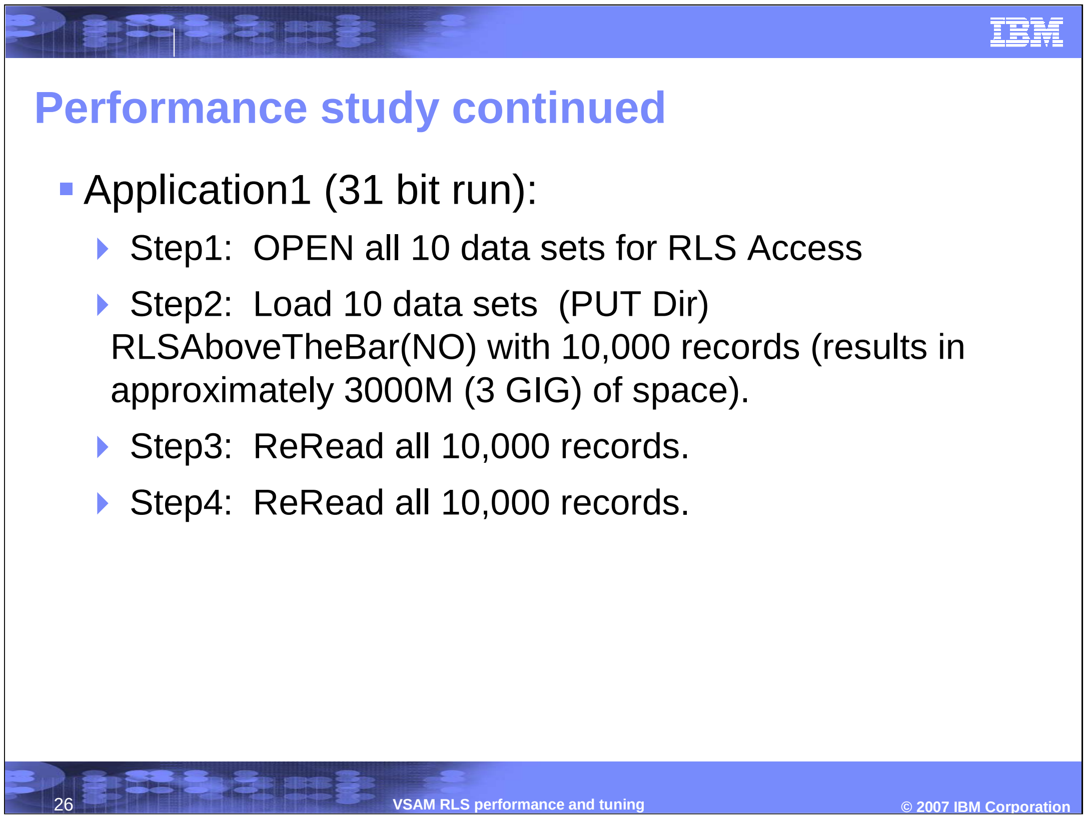


Performance study – 64 bit versus 31bit

- System parameters:
  - RLS_Max_Pool_Size (500)
  - RLSAboveTheBarMaxPoolSize (4096)
  - RLS_MaxCFFeatureLevel(Z)
  - RLS Cache structure size (2800M)
- Data set parameters
  - 10 KSDSs DATACLAS RLSAboveTheBar(NO)
  - 10 KSDSs DATACLAS RLSAboveTheBar(YES)
  - Data CISIZE(28K) Index CISIZE(2K)
  - LOG(NONE)

25 VSAM RLS performance and tuning © 2007 IBM Corporation

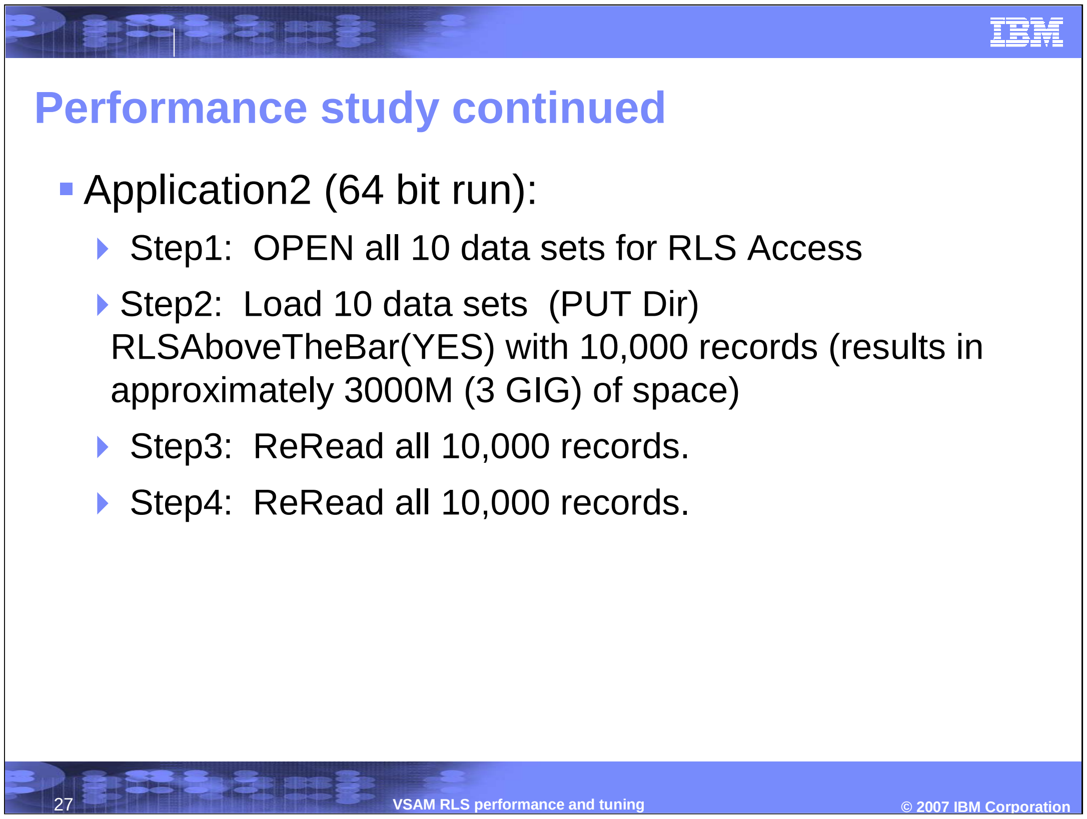
Now the system is setup to compare the performance of accessing 10 data sets to see if the 64 bit pool really help. The CI sizes are all the same for the 20 data sets. So, there are 28 K data CI and 2 K index CI. These are non-recoverable data sets.



Performance study continued

- Application1 (31 bit run):
  - Step1: OPEN all 10 data sets for RLS Access
  - Step2: Load 10 data sets (PUT Dir) RLSAboveTheBar(NO) with 10,000 records (results in approximately 3000M (3 GIG) of space).
  - Step3: ReRead all 10,000 records.
  - Step4: ReRead all 10,000 records.

A job 1 or application 1 will be running with a 31 bit pool. The job will open all those 10 datasets with RLSAboveTheBar(NO) in step 1. In step 2, all 10 data sets are loaded with Put direct requests. There are 10 thousands records in each data set. This sums up to 10 thousands of 28 K records and CI's for 10 data sets. The total space will be approximately 3 GB. Since the 64 bit pool is defined with size of 4 GB. The important point is that the 3 GB of records is certainly bigger than the 31 bit pool, which only goes up to 1.7 GB, and it is smaller than the 64 bit pool. In the next step all 10 data sets are loaded. Then, all 10,000 records will be re-read, and then all 10,000 records will be re-read again.

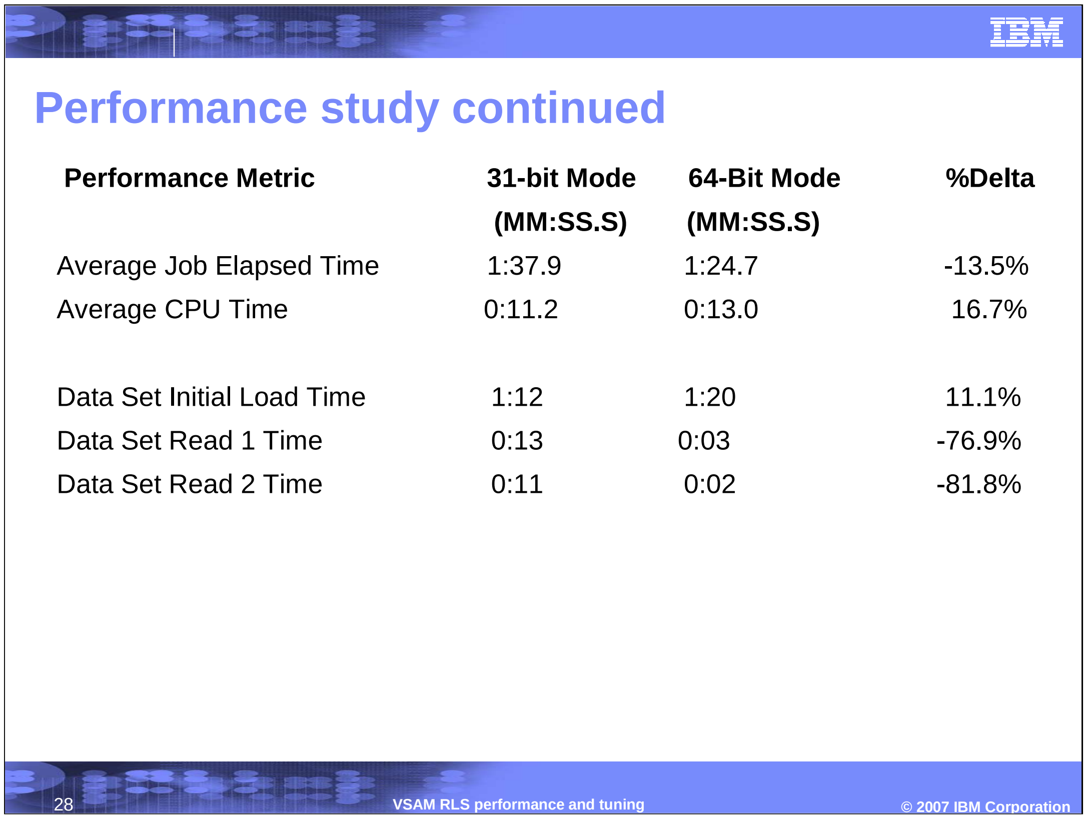


Performance study continued

- Application2 (64 bit run):
  - Step1: OPEN all 10 data sets for RLS Access
  - Step2: Load 10 data sets (PUT Dir) RLSAboveTheBar(YES) with 10,000 records (results in approximately 3000M (3 GIG) of space)
  - Step3: ReRead all 10,000 records.
  - Step4: ReRead all 10,000 records.

27 VSAM RLS performance and tuning © 2007 IBM Corporation

The job 2 or application 2 is going to do exactly the same thing except the 64 bit pool will be utilized instead.

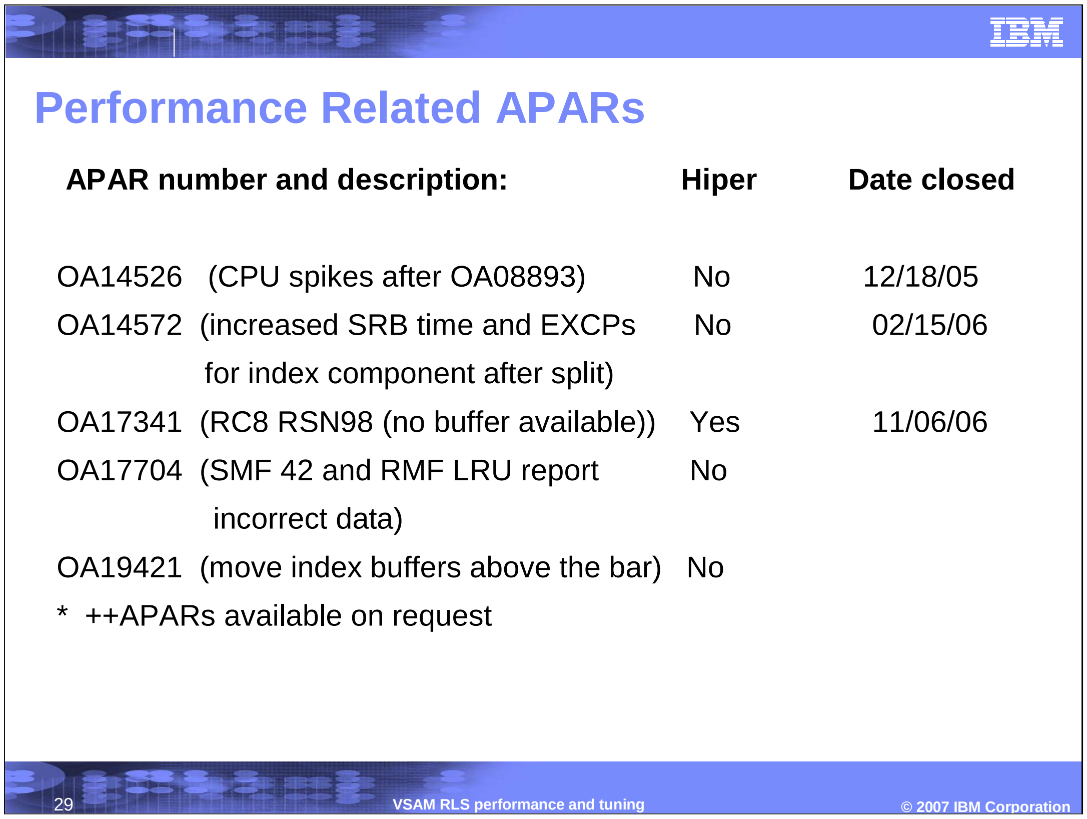


Performance study continued

Performance Metric	31-bit Mode (MM:SS.S)	64-Bit Mode (MM:SS.S)	%Delta
Average Job Elapsed Time	1:37.9	1:24.7	-13.5%
Average CPU Time	0:11.2	0:13.0	16.7%
Data Set Initial Load Time	1:12	1:20	11.1%
Data Set Read 1 Time	0:13	0:03	-76.9%
Data Set Read 2 Time	0:11	0:02	-81.8%

28 VSAM RLS performance and tuning © 2007 IBM Corporation

Here's the result of the measured elapsed time and CPU time. The report is broken down in those three steps: the load step, the read of those 10,000 records, and the re-read of 10,000 records. If you noticed, the delta on the size there. At the bottom right corner, it had almost 82% improvement in performance. This experiment can be expanded to use base VSAM, because you can consider base VSAM to sort of use 31 bit pool, because base VSAM can only run in 31 bit mode. It can only have so big of an LSR pool. Even though base VSAM path length is a lot shorter, but I/O performance will end up with a huge cost. If you got buffering as you are getting of 64 bit buffering, then you can gain a better performance than base VSAM applications. With this experiment, you have to look at it how is it designed. It was perfectly designed for the available size pool. When there is 3 GB of total data, it won't fit in the 31 bit pool. You will have to do a lot of LRU activities and going back out to DASD to re-read to fit more of data in, and re-reading it. However, the 4 GB 64 bit pool perfectly holds all the data. You will get basically 100% buffer hit when you're using the 64 bit pool. That is how you can see such a huge improvement.



Performance Related APARs

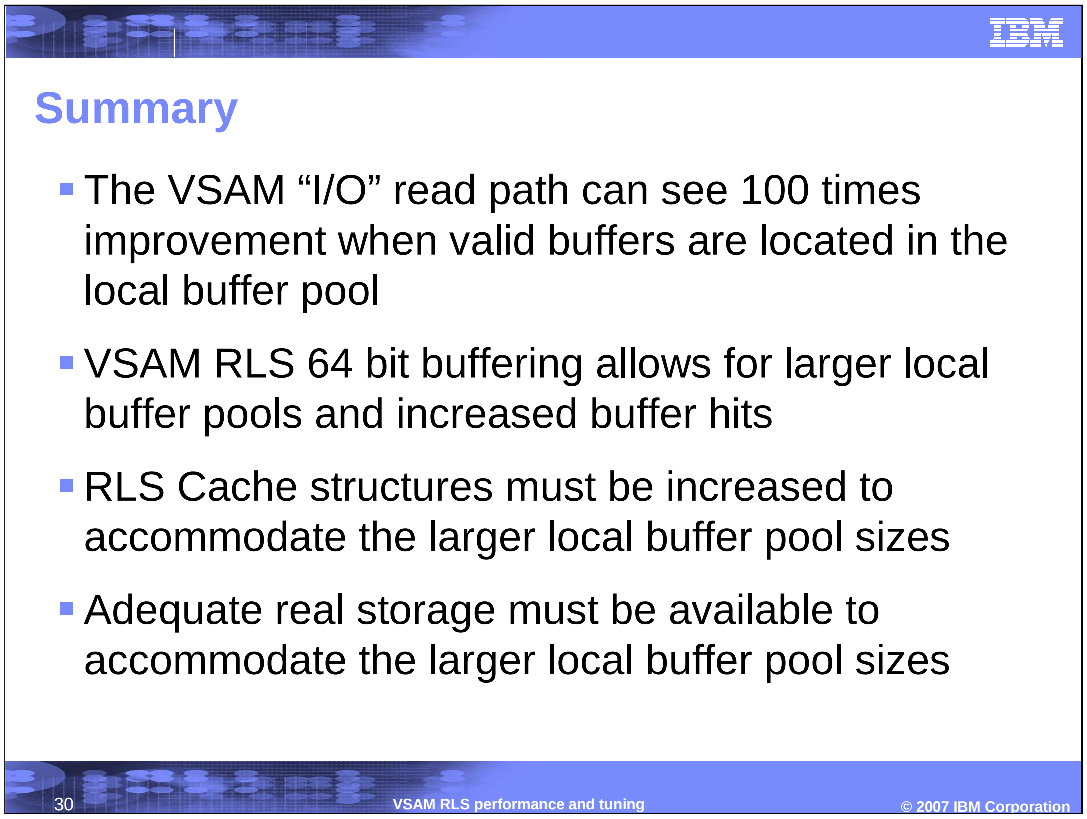
APAR number and description:	Hiper	Date closed
OA14526 (CPU spikes after OA08893)	No	12/18/05
OA14572 (increased SRB time and EXCPs for index component after split)	No	02/15/06
OA17341 (RC8 RSN98 (no buffer available))	Yes	11/06/06
OA17704 (SMF 42 and RMF LRU report incorrect data)	No	
OA19421 (move index buffers above the bar)	No	

* ++APARs available on request

29 VSAM RLS performance and tuning © 2007 IBM Corporation

There are some available APARs related to performance. Some of them fix bug related issues and some of them are enhancements related. The APARs shown here are recommended to be on your systems before you are serious about getting into the 64 bit buffering or performance in general. The first four are bugs related. The last one is enhancement, OA19421. The 64 bit design only has data buffers in the 64 bit pool. So even if you said RLSAboveTheBar(YES) for your data set, only the data buffers will be bufferred in the 64 bit pool. The index buffers are still staying in the 31 bit pool. With this APAR OA19421, the index buffers are also moved into the 64 bit pool.

Check with the support center on the status of APAR closing.

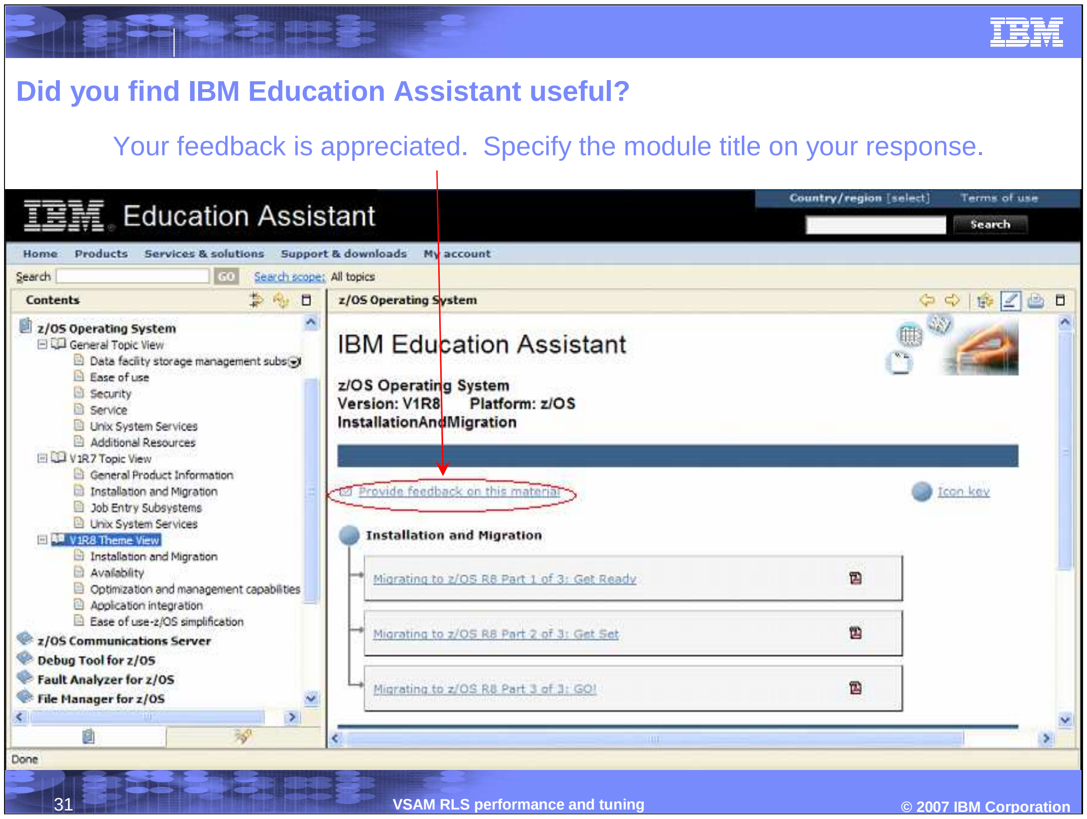


Summary

- The VSAM "I/O" read path can see 100 times improvement when valid buffers are located in the local buffer pool
- VSAM RLS 64 bit buffering allows for larger local buffer pools and increased buffer hits
- RLS Cache structures must be increased to accommodate the larger local buffer pool sizes
- Adequate real storage must be available to accommodate the larger local buffer pool sizes

30 VSAM RLS performance and tuning © 2007 IBM Corporation

In summary, with the VSAM "I/O" read path, you can see 100 times improvement if you can get complete valid buffers in the buffer pool. The 64 bit provides a larger buffer pool. Whether the 31 bit pool or 64 bit pool will be used, you need to make sure your cache structures matched with the pool sizes. With a matched cache size, you will not limit the buffering due to a small cache structure. The last important point is that you will have to consider how much real storage you have before you start defining these larger pool sizes.



Did you find IBM Education Assistant useful?

Your feedback is appreciated. Specify the module title on your response.

IBM Education Assistant

Country/region [select] Terms of use

Search

Home Products Services & solutions Support & downloads My account

Search GO Search scope: All topics

Contents

- z/OS Operating System
  - General Topic View
    - Data facility storage management subs
    - Ease of use
    - Security
    - Service
    - Unix System Services
    - Additional Resources
  - V1R7 Topic View
    - General Product Information
    - Installation and Migration
    - Job Entry Subsystems
    - Unix System Services
  - V1R8 Theme View
    - Installation and Migration
    - Availability
    - Optimization and management capabilities
    - Application integration
    - Ease of use-z/OS simplification
- z/OS Communications Server
- Debug Tool for z/OS
- Fault Analyzer for z/OS
- File Manager for z/OS

Done

z/OS Operating System

IBM Education Assistant

z/OS Operating System
Version: V1R8 Platform: z/OS
InstallationAndMigration

Provide feedback on this material

Icon key

Installation and Migration

- Migrating to z/OS R8 Part 1 of 3: Get Ready
- Migrating to z/OS R8 Part 2 of 3: Get Set
- Migrating to z/OS R8 Part 3 of 3: GO!

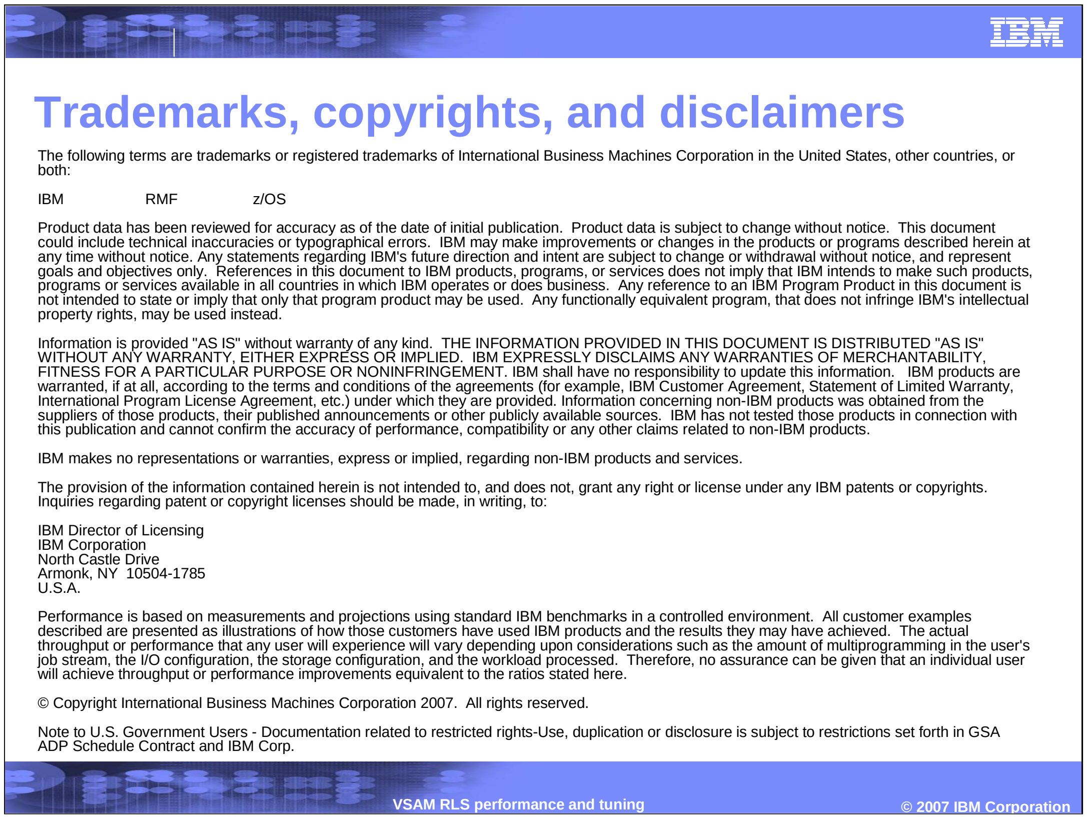
31 VSAM RLS performance and tuning © 2007 IBM Corporation

In order to supply you with pertinent and timely information in IBM Education modules, your opinions are important. To help IBM in creating these modules, take the time to help us out. In your feedback to IBM please answer the following three questions:

1. How helpful was this IEA presentation? Give a rating from 1 to 5 where 1 = very helpful and 5 = not at all helpful.

2. Did this presentation save you a service call to IBM? Yes or No.

3. If there are any other topics you would like to see covered in IEA, what are they? ________________________________



Trademarks, copyrights, and disclaimers