iSeries

IBM

The Peformance Adjuster

Experience Report

iSeries

The Peformance Adjuster

Contents

The Performance Adjuster (QPFRADJ)

The iSeries(TM) server has the ability to automatically manage the shared memory pools without any user interaction. This function is controlled by the performance adjustment system value, QPFRADJ. When this system value is set to '2' or '3,' the system periodically checks the performance of all the active shared pools and adjusts or rearranges the storage and activity levels as needed. This function is active by default (the shipped value of QPFRADJ is '2' which means 'Adjustment at IPL and automatic adjustment'). This experience report explains how the user-defined settings on the Work with Shared Pools (WRKSHRPOOL) display affect the performance adjuster algorithm, and gives examples of how to tailor them for your environment.

The following sections contain additional information about system tuning with the QPFRADJ performance adjuster:

WRKSHRPOOL tuning data

The Work with Shared Pools (WRKSHRPOOL) display, or the Change Shared Pools (CHGSHRPOOL) command is used to influence how the tuner makes decisions about moving storage from pool to pool. The WRKSHRPOOL command lists all the shared pools on the system. There are 64 shared pools defined, but only the shared pools that have a size listed under the ″Allocated Size (M)″ column are currently allocated to active subsystems. The system may also have private pools that are active, but they are not shown on this display. The Work with System Status (WRKSYSSTS) command displays all the pools (private and shared) that are currently allocated. The performance adjuster only makes changes to shared pools, not private pools.

```
                          Work with Shared Pools
                                                                System:
Main storage size (M)  . :        4000.00

Type changes (if allowed), press Enter.

              Defined    Max   Allocated   Pool  -Paging Option--
Pool         Size (M)  Active  Size (M)     ID   Defined  Current
*MACHINE       372.64   +++++      372.64    1   *FIXED   *FIXED
*BASE         3089.26      77     3089.26    2   *FIXED   *FIXED
*INTERACT      537.83     180      537.83    3   *FIXED   *FIXED
*SPOOL            .25       1         .25    4   *FIXED   *FIXED
*SHRPOOL1         .00       0                  *FIXED
*SHRPOOL2         .00       0                  *FIXED
*SHRPOOL3         .00       0                  *FIXED
*SHRPOOL4         .00       0                  *FIXED
*SHRPOOL5         .00       0                  *FIXED
*SHRPOOL6         .00       0                  *FIXED
                                                                More...
Command
===>
F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F11=Display tuning data
F12=Cancel
```

Figure 1. Work with Shared Pools (WRKSHRPOOL) display

Use the F11 function key on the Work with Shared Pools display to go to the ″Display tuning data″ view. The performance adjuster uses the fields on this display to determine how to manage the shared memory pools.

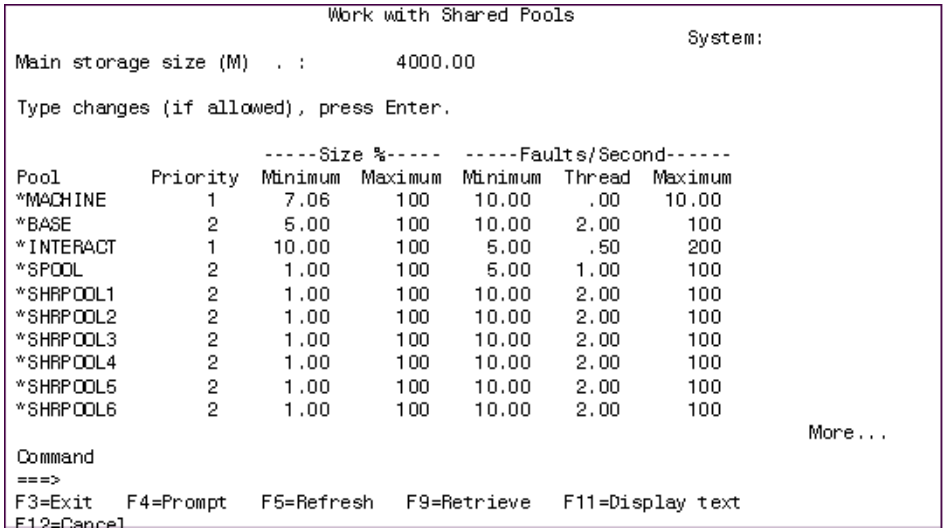

Figure 2. Work with Shared Pools, F11=Display tuning data

The “Priority,” “Size % (Minimum, Maximum)” on page 3, and “Faults/Second (Minimum, Thread, Maximum)” on page 3 can be adjusted for each shared pool, except the priority for the *MACHINE pool, which is always set to 1. The following sections explain what each of these columns are for, and how they affect the performance adjuster algorithm.

Note: The tuning data can also be displayed and changed using APIs as well as through iSeries (TM) Navigator. However, this report refers only to the WRKSHRPOOL command display, as shown in Figures 1 and 2. The Retrieve System Status (QWCRSSTS) API can be used to retrieve the tuning data fields, and the Change Pool Attributes (QUSCHGPA) API can be used to change them. If you prefer to use iSeries Navigator, follow these steps to get to the shared pool tuning values:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Memory Pools**, and then click **Shared Pools**.
5. Right-click the pool you want to use (for example, Base) and select **Properties**.
6. Click the **Tuning** tab to see the list of **Tuning values** for the selected pool.

Priority

The Priority column on WRKSHRPOOL refers to the priority of the pool relative to the priority of the other storage pools. The priority can be set to any value between 1 and 14, with 1 being the best priority and 14 being the worst. The priority of pools is generally easy to determine if you know what is running in each pool. To display a list of jobs currently active in a shared pool, see Determine the number of jobs in a memory pool. Keep in mind that the number and types of jobs and threads in a given pool may vary at different times. Assign lower (better) priorities to pools which have more important jobs and threads running in them, and assign higher (worse) priorities to pools that have work which should not be allowed to interrupt more important work.

Priority is only one part of the performance adjuster algorithm, and it does not necessarily take precedence over the other factors which determine how storage is distributed among the pools. The acceptable faulting rate determined by the Minimum, Thread, and Maximum Faults/Second columns, as well as the Size % columns on the WRKSHRPOOL display also help determine which pools receive or give up storage. Since priority may be easiest to determine, start by adjusting it first, and then adjust the other values if needed.

Size % (Minimum, Maximum)

The minimum and maximum size percent fields are used as guidelines by the performance adjuster to set boundaries for the size of the pool. The minimum size percent is the percentage of main storage that should be kept in the pool, even when there is no paging. This field is useful when you have a pool that you want to keep at or above a certain size, even when it is not being used, so that it is ready when new work arrives.

The maximum size percent can be set to any value higher than the minimum size percent. This can be used to keep a pool from growing too big. Workloads often benefit from additional storage only up to a certain point, and anything added beyond that point does not improve the performance because it is just excess storage that is not being used effectively. You may be able to determine this threshold by observing the performance of threads running in the pool. Start by constraining the size of the pool, and note the paging rate in the pool and response times of the threads running in the pool. As you add storage to the pool, watch the paging and response times to determine if they are improving. When they no longer improve, you know that you have just crossed the threshold, or working set size, for this workload. Divide the pool size by the total main storage and enter this number under the Maximum Size % field. If the workload is too variable to determine a single value for Maximum Size %, keep this field set to the default of 100%.

A change to the minimum or maximum size percent does not automatically change the pool size to be within the new boundaries if it is currently set to a size outside those limits. Faulting is the trigger for all storage changes, and the minimum and maximum size percents are only guidelines when a change is made as the result of paging in one or more pools. For example, if the current pool size of QSPL is set to 256K, and you change the minimum size percent to 2% of a 1024M main storage, the pool size will not automatically bump up to 20M. If there is no activity in the pool, it will remain at the current size of 256K.

You should also be aware that as the sum of all the minimum size percents increases, especially as it approaches or exceeds 70% total, the ability of the performance adjuster to effectively make pool size changes is greatly reduced. For example, if there are 5 shared pools active, and the minimum size percents of the pools are 20%, 10%, 28%, 7%, and 10%, the sum is equal to 75%. This leaves only 25% of the main storage that can be moved between pools by the performance adjuster. If there are private pools also allocated, there may be even less. This may not leave enough free storage for the performance adjuster to use when it needs to respond to a large increase in paging in one or more pools. At that point, it may be better to just turn off the performance adjuster and set the pool sizes to the desired fixed sizes.

If you are running in a logical partition (LPAR) that allows dynamic movement of memory, setting these fields can be very tricky, because percentages are not fixed sizes. One percent of the total main storage may be an acceptable minimum size when the system's main storage size is large. However, if you take away half the storage from the partition, one percent may no longer be an acceptable level. You need to make sure that the minimum size percent is large enough for each pool, even when the total main storage size is at its smallest setting. Similarly, the maximum size percent may become too small for a pool if storage is moved out of the partition. The default of 100% may be the best choice for the maximum size percent in this type of LPAR environment.

Faults/Second (Minimum, Thread, Maximum)

The three columns listed under Faults/Second in Figure 2 are used by the performance adjuster to calculate an acceptable level of faulting in each shared storage pool. The formula used to determine the acceptable faulting rate for a pool is:

```
MIN ((minflt  +  ( thdflt  x  threads)), maxflt)  =  acceptable fault rate
```

In the above formula, MIN is the minimum of the two values, minflt is the minimum page faults per second, thdflt is the number of page faults per second per thread, threads is the number of active threads in the pool, and maxflt is the maximum number of page faults per second. Applying this formula to shared pool *SHRPOOL1 from Figure 2, and assuming there are 25 threads currently active, the acceptable faulting rate would be calculated as:

```
MIN(( 10   +  ( 2   x  25  )),  100 )  =   60 faults per second
```

If *SHRPOOL1 has fewer than 60 faults per second, it is considered to have good performance by the performance adjuster. If it has more than 60 faults per second, *SHRPOOL1 may be a candidate to receive storage from other pools that have better current faulting rates. As more threads enter the pool, the acceptable faulting rate increases, but it can never exceed the Maximum Faults/Second, as defined in the last column in Figure 2. For example, if the number of threads in *SHRPOOL1 increases from 25 to 50, the Minimum Faults of 10, plus 2 faults per second times 50 active threads equals 110 faults per second. However, 110 is greater than the Maximum Faults/Second which is set to 100 for *SHRPOOL1, so 100 will be used as the acceptable rate instead of 110. Plugging these numbers into our formula, the acceptable faulting rate is:

```
MIN(( 10  + ( 2  x  50  )),  100  )  =  100 faults per second
```

Changes to the Faults/Second formula can be made either on the WRKSHRPOOL display, or with the MINFAULT, JOBFAULT, and MAXFAULT parameters on the Change Shared Pool (CHGSHRPOOL) command. When determining the number of active threads, only the threads that have used CPU time during the past tuning interval are considered to be active threads. Signed on interactive users which are not currently executing any instructions are not counted, for example.

Be aware that choosing a value of zero for the faults per thread will make the faulting rate act more as a fixed rate, independent of the number of active threads. In this case, the minimum and maximum faults per second should be set to the same value, because the formula will always produce the same result no matter how many threads are active. The *MACHINE pool is an example of this:

```
10 faults +  (0 faults/thread * number of threads)  = 10
```

The faulting guidelines are based on an average disk response time of 0.01 seconds. The faults/thread can be adjusted in proportion to the actual disk response time. For example, if the actual disk response time is 0.005 seconds, then the faults/thread can be twice the default value, because it can handle twice the number of faults. The formula for determining the faults/thread for a pool with interactive work is:

```
(faulting response time target/disk response time)
-------------------------------------------------  = faults per thread
        transaction cycle time
```

The faulting response time target is the amount of time, in seconds, that a thread spends page faulting during a transaction. The disk response time is the average time, in seconds, that an I/O operation can be handled. The transaction cycle time is the average time, in seconds, between transactions. This is also commonly known as the ″key/think″ time, or time the user spends typing or thinking between transactions. If we set a transaction response time goal of 1 second and assume only 10% of the time will be spent faulting (0.10 seconds/transaction), and we know the disk response time is 0.01 seconds, and assume there is 1 transaction every 20 seconds per job, we can use the above formula to calculate the faults/thread:

```
(0.10  /  0.01)
---------------  =  0.5 faults per thread
      20
```

When setting the maximum faults per second for each pool, consider the total number of faults that the system can handle. A rough rule of thumb to determine total system faults is 100 faults/second per processor times the average processor utilization. For example, a 12-way system with an average of 60% CPU per processor would have a total system guideline of:

```
100  X  12  X  .60  =  720 faults/second
```

Divide the calculated number of total faults accordingly among the shared pools' Maximum Faults fields, making adjustments for private pools as necessary.

Tuning interval

The performance adjuster program analyzes all the shared pools, makes any necessary adjustments, and then goes to sleep for a period of time before waking up and doing it all over again. The period of time that it waits, or the tuning interval, is 60 seconds. If a large amount of storage has been moved in the past interval, the wait time will be extended to 120 seconds. Because the act of moving storage causes paging to occur, this extended settling period keeps the performance adjuster from responding to the memory movement itself. 60 seconds is the recommended tuning interval, but it may not be the best interval for all workloads. The interval can be changed to any value between 20 and 120 seconds. To change the tuning interval, you will need to create the following data area:

```
CRTDTAARA DTAARA(QUSRSYS/QPFRADJWT) TYPE(*DEC) LEN(3 0) VALUE(60)
```

In the above example, the value of '60' is the number of seconds to wait between tuning intervals. The value can be replaced by any whole number between 20 and 120 (including 20 and 120). The value can either be entered on the CRTDTAARA command, or by using the Change Data Area (CHGDTAARA) command. The data area name, library, and type must match the above command exactly. The QPFRADJ system value must be changed to '0' and then back to '2' or '3' after the data area is created in order for the change to take effect immediately. To change the interval back to the default, the data area must be changed back to a value of '60'.

Case studies

1. Large shift in workload from one pool to another.

Problem:

Overnight backups are run in the batch pool each night. At 8:00AM, a large number of interactive users start to sign on in the interactive pool. The interactive users have very poor performance while trying to sign on. After several minutes, interactive response times return to normal.

The performance adjuster shifts most of the storage from the interactive pool to the batch pool overnight, while the backups are running in the base pool, and all the interactive users have signed off. When the backups complete, there is no activity in either the batch pool or the interactive pool, so the pool sizes remain the same. When the first users start signing on to the system in the interactive pool, this causes paging in the interactive pool, and the performance adjuster begins moving storage back from the batch pool into the interactive pool. But the performance adjuster only runs every minute, and during this time, hundreds of users may all be trying to sign on at the same time. The performance adjuster eventually moves enough storage to handle the shift in workload, but it is painful for the interactive users during this transition.

Solution:

We could decrease the tuning interval, but even at 20 second intervals, there will still be a delay, and 20 seconds may be too small for the normal processing during the day. Another choice is to set the minimum percent for the interactive pool higher so that it doesn't get drained so low at night. But that may not be feasible if the batch pool really does need the extra storage at night. The best solution in this case may be to help the performance adjuster by moving the storage back to the interactive pool using the CHGSHRPOOL command. This can be automated by calling the command from the backup program after all the other processing has completed. Another option is to use the job scheduler to execute the CHGSHRPOOL command(s) at a specific time. For example:

```
ADDJOBSCDE JOB(POOLS) CMD(CHGSHRPOOL POOL(*INTERACT) SIZE(300000))
  FRQ(*WEEKLY) SCDDATE(*NONE) SCDDAY(*ALL) SCDTIME('07:45:00')
```

The Add Job Schedule Entry command above will submit a job to call the CHGSHRPOOL command at 7:45AM every morning, which will move memory from the base pool to the interactive pool. Assuming there is enough free storage available, the new interactive pool size will be 300,000K.

2. Pool with higher priority not getting enough storage.

Problem:

*SHRPOOL5 has a high faulting rate of 100+ faults per second, but the performance adjuster is not giving it any storage. *SHRPOOL1 has a lower faulting rate of about 50 faults per second, but is not giving up storage, and in some cases it is even being increased in size, even though *SHRPOOL5 has a better priority.

Solution:

Priority is only one factor in determining how to manage the storage pool sizes. We also need to examine the acceptable faulting rate for the pools and the minimum and maximum size percents.

- First, make sure that the maximum size percent is not preventing *SHRPOOL5 from getting any bigger. If it is already at or near its maximum size, that is why the pool is not being increased.
- Check the sum of all of the minimum size percents to make sure that there is excess storage available for the performance adjuster to distribute. If not, decrease some of the minimum sizes.
- Next, look at the number of active threads in the pools, and calculate the acceptable faulting rate for each pool. Even if *SHRPOOL5 has the same values in the Faults/Second fields as the other shared pools it is competing with, if *SHRPOOL5 has 50 threads, and the *SHRPOOL1 only has 1 thread, then the acceptable faulting rate is going to be much higher for *SHRPOOL5 (100 faults) than is it for *SHRPOOL1 (12 faults). If this is a fairly normal workload for the pools (and not just a spike in the number of threads for *SHRPOOL5), then it may help to either increase the faults per thread for *SHRPOOL1, and/or decrease the faults per thread for *SHRPOOL5.

  It is best to make small incremental changes, observing the effects of each change over a period of time when making adjustments to the Faults/Second.

3. Interactive pool sluggish after upgrade.

Problem:

Interactive jobs are performing poorly after an upgrade. The faulting rate is high in the interactive pool, but the performance adjuster is not responding to it, even though the base pool has a comparatively low faulting rate.

Solution:

The default tuning options on WRKSHRPOOL - F11 differ depending on the model type. iSeries (TM) server models (those models which end in an 'S' on the QMODEL system value) swap the tuning defaults for the *BASE and *INTERACT pool. The *BASE pool is given a priority of 1, and the *INTERACT pool is given a priority of 2 on server models. And the Faults per thread, Minimum and Maximum faults are also switched between these two pools. This is done to keep the interactive work from impacting the server jobs. However, if the interactive work is important on your system, you may want to make adjustments to the priority, and faults per second for the *BASE and *INTERACT pools. These values can be swapped between the two pools, set equal to each other, or set to some other value as needed.

In V5R3 and later releases, the default values for the *BASE and *INTERACT pools are swapped for server models with Standard Edition, but are set equal to each other (with both pools having a priority of 1) on servers with Enterprise Edition. For example, server models with Standard Edition give priority to the *BASE pool over the *INTERACT pool, the same as server models in previous releases. However, servers with Enterprise Edition give equal priority to both the *BASE and *INTERACT pool. If there is a large disparity in the number of active threads in these two pools, the Faults per thread, Minimum faults, and Maximum faults will likely need to be adjusted to take the difference in threads into account. Otherwise, the pool with fewer threads will appear to be favored over the other pool, even though it may be faulting at a much higher rate.

References and resources

iSeries(TM) Information Center
- Work Management
- System Values
- Work with Job Schedule Entries

Disclaimer

Information is provided ″AS IS″ without warranty of any kind. Mention or reference to non-IBM products is for informational purposes only and does not constitute an endorsement of such products by IBM.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Printed in USA