iSeries

Spool Performance Considerations

Experience Report

iSeries

Spool Performance Considerations

Contents

Introduction

In the past several iSeries(TM) releases, OS/400(R) software improvements have been made to allow more and more jobs on the system. In addition, hardware improvements and server consolidation have resulted in larger and more complex workloads running on the iSeries(TM). This has had the effect of increasing the number of spool files created on the system which is reaching the limit of the spool infrastructure's ability to perform at an acceptable level. Many customers are experiencing problems with long delays in completing spool operations, creating customer satisfaction issues.

This experience report will:

- Identify spool objects that are most likely to encounter lock contention
- Provide information about messages that can be expected if performance bottlenecks are encountered
- Identify several scenarios that can cause spool performance problems
- Provide recommendations to reduce specific spool performance issues

Spool objects that can encounter lock contention

Output queue (OUTQ)

This object is a repository for spooled files. Internally, this object is implemented as an independent index (type 0E subtype 02). Each spooled file on an output queue is represented as an entry on the output queue index. This object has been notorious for bottlenecking (even back as far as Release 2 of S/38(TM)). Our current internal design does not permit shared access to the output queue object without compromising data integrity. For each action taken on a spooled file, an exclusive lock is obtained on the output queue where the spooled file resides. These actions include adding (creating a spooled file), removing (DLTSPLF), holding (HLDSPLF), releasing (RLSPLF), changing (CHGSPLFA) or listing (WRKOUTQ) spooled files on the output queue. What this really means, is that a Work with Output Queue (WRKOUTQ) command will conflict with a create of a spooled file which will conflict with a delete of a spooled file, etc. See the "Spool lock contention scenarios" on page 4 section for more information.

Internal print queues (PRTQ) (QSPUSRQ, QSPALLQ, QSPDEVQ, QSPFRMQ, QSPUDTQ, QSPASPQ, QSPQJBQ)

These internal objects are used to efficiently list spooled files using the Work with Spooled Files (WRKSPLF) command. Internally, these objects are implemented as independent indexes (type 0E subtype C7). Each spooled file on the system is represented as an entry on each index. Each index has a different key that is used to subset a list of spooled files. For example, if an interactive user does a WRKSPLF USER(QSYS) we lock the QSPUSRQ index, which is keyed off of user name. Only those spooled files owned by user QSYS are listed. Similarly, if a user does a WRKSPLF DEV(PRT01), we lock the QSPDEVQ index, which is keyed off of device name. Only those spooled files on device output queue PRT01 are listed. This prevents the need to check all spooled files on the system to determine if it matches the filter criteria on the WRKSPLF command. Like the output queue, the major shortcoming of the internal print queue design is that exclusive locks are required when adding (creating a spooled file), holding (HLDSPLF), releasing (RLSPLF), changing (CHGSPLFA), removing (DLTSPLF) or reading (WRKSPLF) a spooled file on the system. What this really means, is that a WRKSPLF USER(QSYS) command will conflict with a create of a spooled file which will conflict with a delete of a spooled file, etc. Our current implementation does not allow for concurrent access to an internal print queue index. See the "Spool lock contention scenarios" on page 4 section for more information.

Additional overhead is required to ensure that these indexes are synchronized. This is a major reason why the Spool portion of an abnormal IPL can be long running.

Output file control block (OFCB)

The OFCB, or spool database member, is where the data and attributes are stored for a spooled file. The spooled file attributes are stored in a space associated with the database member. We obtain an exclusive space location lock on the database member to synchronize access to the spooled file attributes. Lock contention on this object is not typically a problem because multiple jobs are not usually attempting to access the same spooled file at the same time.

Spool control block (SCB)

One spool control block (type 19 subtype C2) exists for each job on the system. This object hangs off of the Work control block table entry and is primarily used to hold the counters for the number of spooled files for a job and certain job attributes. An exclusive lock is required on the SCB when adding (creating a spooled file) or removing (DLTSPLF) spooled files for a particular job or if an attribute of the job is changing that affects spooled files. Since moving the attributes of the spooled file out of the SCB into the database member in V5R1M0, this object has not been a major source of contention. However, contention problems have been experienced by customers that create many spooled files on behalf of other users because of swapping or network spooled file activity (SNDNETSPLF or SNDTCPSPLF). Typically we see this on SCBs that are attached to QPRTJOB jobs. See the "Spool lock contention scenarios" on page 4 section for more information.

Reader writer control block (RWCB)

One RWCB object exists on each system (type 19 subtype C5). This object contains one entry for each active writer on the system. An exclusive lock is required on the RWCB when listing (WRKWTR), starting (STRPRTWTR), ending (ENDWTR) or changing (CHGWTR) a writer. Contention on this object is more likely to occur when starting or ending all or many writers.

Messages that can indicate spool lock contention

MCH5802 - *Lock operation for object &1 not satisfied.*

This message is issued to several spool programs and is normally externalized. Issuing this message to the job log allows IBM support to determine what object failed the lock instruction. In addition, the second level text for this message was enhanced in V5R3M0 to include the qualified job name of the job that is holding the lock on the object. In deadlock conditions, the customer is then able to end the job holding the lock and operations on that object can continue without IBM support involvement. This is especially useful for internal objects, when the Work with Object Locks (WRKOBJLCK) command cannot determine the lock holder. Note: It may be difficult for IBM support to debug a problem if the job holding the lock on the object is ended without collecting documentation first.

This message also indicates the amount of time the job waited to lock the object before giving up. This value is generally the default wait time specified in the subsystem class description where the job was initiated. The shipped default is 30 seconds. Increasing this value may reduce the frequency of this message. However, if the class default wait time is increased, the bottleneck for a particular object may expand as jobs wait longer for the lock on the object.

MCH5802 can be issued for the following objects: Output Queue (OUTQ), Internal Print Queue (PRTQ), Spool Control Block (SCB), Reader Writer Control Block (RWCB)

MCH5804 - *Lock space location operation not satisfied in specified time interval.*

This message is issued to several spool programs and is normally externalized. Issuing this message to the job log allows IBM support to determine what program and instruction number attempted the space location lock. In addition, the second level text for this message was enhanced in V5R3M0 to include the qualified job name of the job that is holding the lock on the space. In deadlock conditions, the customer is then able to end the job holding the lock and operations on that object can continue without IBM

support involvement. This is especially useful for space location locks, when the WRKOBJLCK command cannot determine the lock holder. Note: It may be difficult for IBM support to debug a problem if the job holding the lock on the space location is ended without collecting documentation first.

This message also indicates the amount of time the job waited to lock the space location before giving up. This value is generally the default wait time specified in the subsystem class description where the job was initiated. The shipped default is 30 seconds. Increasing this value may reduce the frequency of this message. However, if the class default wait time is increased, the bottleneck for a particular object may expand as jobs wait longer for the space location lock.

MCH5804 can be issued for the following objects: Output File Control Block (OFCB), Output Queue (OUTQ)

CPF3330 - *Necessary resource not available.*

This message is issued from several spool programs to interactive and batch jobs that encounter lock time out on spool objects. This message is usually preceded by MCH5802 or MCH5804 which identifies the real object that failed the lock. For applications that are using spool related CL commands or APIs, this message is expected and should be monitored for. If the error is encountered, the failing function should be retried.

CPF4218 - *Output queue &6 in &7 not available.*

This message is issued by spooled file open (QSPOPEN) if the job creating the spooled file cannot obtain a lock on the output queue in 10 minutes. The spooled file will fail to create. The WRKOBJLCK command can be used to determine the job that is holding the lock on the output queue.

CPF4218 can be issued for the following objects: Output Queue (OUTQ)

CPF2528 - *Job log not written to output queue because of &1.*

This message is issued by message handler if a job creating a job log cannot obtain a lock on the output queue or internal print queues in 10 minutes. This message is issued to the QHST history log and QSYSOPR message queue.

CPF2528 can be issued for the following objects: Output Queue (OUTQ), Internal Print Queue (PRTQ)

CPF4235 - *Not able to open spooled file. Reason code is &6.*

For Reason code 1, the message is issued by spooled file open (QSPOPEN) if the job creating the spooled file cannot obtain a lock on an internal print queue object in 10 minutes. The spooled file will fail to create. Since this is an internal object, the WRKOBJLCK command cannot be used to determine the job holding the lock. The Display Lock Status (DSPLCKSTS) command, shipped in the QSPTLIB library (available via PTF), can be used to determine the lock holder.

For Reason code 2, the message is issued by spooled file open (QSPOPEN) if the system has reached the maximum number of spooled files. For V5R2M0 and V5R3M0, the maximum number of spooled files is 2.61 million in SYSBAS (QSYS). Reduce the number of spooled files on the system.

CPF4235 RC1 can be issued for the following objects: Internal Print Queue (PRTQ)

General spool performance recommendations

The following recommendations can be used as guidelines to streamline spool performance:

- Reduce the number of spooled files on the system.
- Spread the spooled files out to as many output queues, users and jobs as is practical.
- Ensure that the system is tuned correctly. For operations such as WRKOUTQ and WRKSPLF, paging throughput is a significant gating factor. Increase resources such as memory that is allocated to jobs processing lists of spooled files.
- Dedicate at least 1 megabyte of main storage to the *SPOOL pool for each active writer on the system.
- Ensure that the QRCLSPLSTG system value is not set to *NONE. Using *NONE can adversely affect the performance of creating a spooled file.
- Avoid the following long running operations during peak activity in a highly active spooling environment:
  - WRKSPLF USER(*ALL)
  - Many basic assistance level spool and print operations
  - QUSLSPL API calls using format SPLF0100 or SPLF0200 to list all spooled files on the system
  - QGYOLSPL API calls using format OSPL0100 or OSPL0200 to list all spooled files on the system
  - CHGJOB OUTPTY(X) against a job with hundreds or thousands of spooled files
  - CHGJOB SPLFACN(*DETACH) against a job with hundreds or thousands of spooled files
  - HLDJOB SPLFILE(*YES) against a job with hundreds or thousands of spooled files
  - RLSJOB against a job with hundreds or thousands of spooled files that were previously held with HLDJOB SPLFILE(*YES)
  - CLROUTQ
  - CALL PGM(QSYS/QSPFIXUP)

Spool lock contention scenarios

Output Queue Lock Contention

Since the days of the S/38(TM), output queue lock contention has been a problem. Most recently the main culprit has been output queue QEZJOBLOG. When many jobs end at the same time and attempt to cut job logs, bottlenecking can occur. If the bottlenecking is severe enough, message CPF2528, CPF4218 and MCH5802/CPF3330 can occur. Output queue contention is not limited to QEZJOBLOG. Some customers funnel all or the majority of spooled output to one or only a few output queues. This can result in bottlenecking that can be much worse than QEZJOBLOG.

Scenario 1

Thousands of jobs end at the same time and attempt to cut job logs to output queue QEZJOBLOG.

Result:

In this type of environment, severe bottlenecking on QEZJOBLOG can result. Add to this users doing a WRKOUTQ OUTQ(QEZJOBLOG) and you could have severe contention on the output queue. A WRKOUTQ OUTQ(QEZJOBLOG) will take a snapshot of the output queue while holding an exclusive lock on it. This could take several seconds or minutes depending on what resources are dedicated to the job.

Symptoms:

Jobs attempting to create job logs or access spooled files on QEZJOBLOG will go into LCKW or hang in END status until the job log for the job can be cut or the user doing the WRKOUTQ has obtained the snapshot. Messages CPF2528, MCH5802/CPF3330 and CPF4218 can result. The WRKOBJLCK OBJ(QEZJOBLOG) OBJTYPE(*OUTQ) command may show jobs waiting for *EXCL locks. The holder of the lock will move from one job to the next as the bottleneck subsides.

Recommendations:

- Change the logging level of job descriptions to not cut job logs or cut job logs only where needed. Note: If a job ends abnormally, the job log will be cut regardless of the job's logging level. Decreasing the number of spooled files on QEZJOBLOG will result in the WRKOUTQ OUTQ(QEZJOBLOG) command holding the exclusive lock for a shorter duration.
- When calling the ENDSBS command, use the ENDSBSOPT(*NOJOBLOG) parameter to reduce the amount of job logs created.
- Spread the job log spooled files out to several output queues. An example of how to route job logs created by different subsystems to a different job log output queue is as follows:
  1. Create a library for each subsystem defined on the system that requires a separate job log output queue.
  2. Create a duplicate of the QUSRSYS/QEZJOBLOG output queue and put a copy in each library created.
  3. Create a duplicate of the QSYS/QPJOBLOG printer file and put a copy in each library created.
  4. Change the OUTQ attribute of each duplicated QPJOBLOG printer file from QUSRSYS/QEZJOBLOG to *LIBL/QEZJOBLOG.
  5. Create a routing entry for each subsystem to call separate programs that do a CHGSYSLIBL LIB(x) so that the system portion of the library list is modified to have the new libraries created in step 1 above QSYS.

     Note: When using this technique, be aware that automatic cleanup of system output will not be done on these duplicated output queues.
- Ensure that the system is tuned correctly. For operation such as WRKOUTQ, paging throughput is a significant gating factor. Increase resources such as memory that is allocated to jobs accessing job logs.
- At off peak times when the output queue is not in use, move spooled files to alternate output queues. An application like DLTOLDSPLF could be modified to move spooled files older than X number of days using the CHGSPLFA command (instead of DLTSPLF).
- Decrease the retention period for job logs using OA cleanup functions (GO CLEANUP).
- To reduce contention on QEZJOBLOG, use the WRKSPLF or WRKJOB OPTION(*SPLF) instead of WRKOUTQ to access job log spooled files.
- Ensure that QEZJOBLOG is only being used to hold job log spooled files.
- Ensure that the QRCLSPLSTG system value is not set to *NONE. Using *NONE can adversely affect the performance of creating a spooled file.

Scenario 2

A customer creates 30,000 spooled files to an output queue (OUTQA) per day. The customer's spooled file retention policy is to keep the spooled files in OUTQA for 7 days. The output queue has an average of 200,000 - 210,000 spooled files. During peak operations, the customer has 10 applications simultaneously creating and changing spooled files on the output queue. The customer also has 25 unique users on the system attempting to access the same spooled files to display, change, hold or release them.

Result:

In this type of environment, severe bottlenecking on the output queue can result. For each operation performed on a spooled file in OUTQA, an exclusive lock is obtained on the output queue. Add to this users doing a WRKOUTQ OUTQ(OUTQA) and you could have severe contention on the output queue. This could take several seconds or minutes depending on what resources are dedicated to the job. As the number of spooled files, applications creating spooled files or users accessing spooled files on the output queues increase, so will the contention.

Symptoms:

Jobs attempting to create or access spooled files on the output queue will go into LCKW resulting in decreased throughput for the jobs. Messages MCH5802/CPF3330 and CPF4218 can result. The

WRKOBJLCK OBJ(OUTQA) OBJTYPE(*OUTQ) command may show jobs waiting for *EXCL locks. The holder of the lock will move from one job to the next as the bottleneck subsides.

Recommendations:

- Spread the spooled files out to as many output queues as is practical. Decreasing the number of spooled files on the output queue will result in the WRKOUTQ OUTQ(OUTQA) command holding the exclusive lock for a shorter duration.
- Ensure that the system is tuned correctly. For operation such as WRKOUTQ, paging throughput is a significant gating factor. Increase resources such as memory that is allocated to jobs accessing job logs.
- Verify that all of the spooled files on the output queue are actually needed. We have had numerous cases where customer and business partner applications create temporary spooled files and immediately delete them. We recommend avoiding the use of spooled files as a vehicle to store temporary data whenever practical. Spooled file auditing can help to determine if this is a problem, as most customers are not aware this is happening.
- At off peak times when the output queue is not in use, move spooled files to alternate output queues. An application like DLTOLDSPLF could be modified to move spooled files older than X number of days using the CHGSPLFA command (instead of DLTSPLF).
- To reduce contention on an output queue with thousands of spooled files, use the WRKSPLF or WRKJOB OPTION(*SPLF) instead of WRKOUTQ to access the spooled files.
- Ensure that job logs and system dumps are generated to dedicated output queues.
- Ensure that the QRCLSPLSTG system value is not set to *NONE. Using *NONE can adversely affect the performance of creating a spooled file.

Internal Print Queue Lock Contention

The main culprit for contention on the internal print queue objects is WRKSPLF commands that list all or the majority of spooled files on the system. A WRKSPLF USER(*ALL) will take a snapshot of the QSPALLQ internal print queue while holding an exclusive lock on the index. This index contains an entry for each spooled file on the system so generating the snapshot can take several seconds or minutes. While the index is locked, no spooled files on the system can be created, held, released, deleted, or changed. If the bottlenecking is severe enough, message CPF4235 RC1 and MCH5802/CPF3330 can occur.

Scenario 1

At peak operating times, on the system with 300,000 spooled files, a user does a WRKSPLF USER(*ALL).

Result:

A WRKSPLF USER(*ALL) will take a snapshot of the QSPALLQ index while holding an exclusive lock on it. This could take several seconds or minutes depending on what resources are dedicated to the job. Until the snapshot is complete, no spooled files on the system can be created, held, released, deleted, or changed.

Symptoms:

Jobs attempting to create, hold, release, delete, change or list spooled files will hang in LCKW status. Messages MCH5802/CPF3330 and CPF4235 RC1 can result. Since this is an internal object, the WRKOBJLCK command cannot be used to determine the job holding the lock. The DSPLCKSTS command, shipped in the QSPTLIB library (available via PTF), can be used to determine the lock holder. Once the user of the WRKSPLF USER(*ALL) completes the snapshot of the index, the bottleneck will work its way out.

Recommendations:

- Reduce the number of spooled files on the system.
- Ensure that the system is tuned correctly. For operations such as WRKSPLF, paging throughput is a significant gating factor. Increase resources such as memory that is allocated to jobs accessing spooled files.

- Use something other than WRKSPLF USER(*ALL) to subset the list of spooled files. Perhaps filtering on user, form type or user data may be more appropriate. Use WRKJOB OPTION(*SPLF) or WRKOUTQ to get a list of spooled files.
- Use the support added in V5R3M0 to store spooled files in an Independent auxiliary storage pool (IASP). This new design replaces the internal print queue objects with keyed database logical files. This approach allows for shared access to the database logical files.

Scenario 2

At peak operating times, on the system with 300,000 spooled files, a user does a WRKSPLF USER(USERX). However, USERX owns 290,000 spooled files.

Result:

A WRKSPLF USER(USERX) will take a snapshot of the QSPUSRQ index while holding an exclusive lock on it. The snapshot will only include those spooled file owned by user USERX which is almost as inefficient as doing a WRKSPLF USER(*ALL). This could take several seconds or minutes depending on what resources are dedicated to the job. Until the snapshot is complete, no spooled files on the system can be created, held, released, deleted, or changed.

Symptoms:

Jobs attempting to create, hold, release, delete, change or list spooled files will hang in LCKW status. Messages MCH5802/CPF3330 and CPF4235 RC1 can result. Since this is an internal object, the WRKOBJLCK command cannot be used to determine the job holding the lock. The DSPLCKSTS command, shipped in the QSPTLIB library (available via PTF), can be used to determine the lock holder. Once the user of the WRKSPLF USER(*ALL) completes the snapshot of the index, the bottleneck will work its way out.

Recommendations:

- Spread the spooled files across a greater number of users.
- Reduce the number of spooled files on the system.
- Ensure that the system is tuned correctly. For operation such as WRKSPLF, paging throughput is a significant gating factor. Increase resources such as memory that is allocated to jobs accessing spooled files.
- Use something other than WRKSPLF USER(USERX) to subset the list of spooled files. Perhaps filtering on device, form type or user data may be more appropriate. Use WRKJOB OPTION(*SPLF) or WRKOUTQ to get a list of spooled files.
- Use the support added in V5R3M0 to store spooled files in an IASP.

Scenario 3

At peak operating times, on the system with 300,000 spooled files, a user does a WRKSPLF SELECT(*ALL *ALL *STD). However, 290,000 spooled files on the system have a form type value of *STD.

Result:

A WRKSPLF SELECT(*ALL *ALL *STD) will take a snapshot of the QSPFRMQ index while holding an exclusive lock on it. The snapshot will only include those spooled files with form type *STD, which is almost as inefficient as doing a WRKSPLF USER(*ALL). This could take several seconds depending on what resources are dedicated to the job. Until the snapshot is complete, no spooled files on the system can be created, held, released, deleted, or changed.

Symptoms:

Jobs attempting to create, hold, release, delete, change or list spooled files will hang in LCKW status. Messages MCH5802/CPF3330 and CPF4235 RC1 can result. Since this is an internal object, the WRKOBJLCK command cannot be used to determine the job holding the lock. The DSPLCKSTS command, shipped in the QSPTLIB library (available via PTF), can be used to determine the lock holder. Once the user of the WRKSPLF SELECT(*ALL *ALL *STD) completes the snapshot of the index, the bottleneck will work its way out.

Recommendations:

- Reduce the number of spooled files on the system.
- Ensure that the system is tuned correctly. For operation such as WRKSPLF, paging throughput is a significant gating factor. Increase resources such as memory that is allocated to jobs accessing spooled files.
- Use something other than WRKSPLF SELECT(*ALL *ALL *STD) to subset the list of spooled files. Perhaps filtering on device, user or user data may be more appropriate. Use WRKJOB OPTION(*SPLF) or WRKOUTQ to get a list of spooled files.
- Use the support added in V5R3M0 to store spooled files in an IASP.

Spool Control Block Lock Contention

The main culprit for contention on spool control block (SCB) objects are server jobs creating spooled files while swapped to a single application user. This can cause contention on the SCB of QPRTJOB jobs. If the bottlenecking is severe enough, message MCH5802/CPF3330 can occur.

Scenario 1

At peak operating times, hundreds of server jobs are swapped to the same user creating, changing and deleting spooled files.

Result:

In this scenario, all of the spooled files will be created under the same QPRTJOB job associated with the user that the server jobs are swapped to. This can cause contention on the SCB object attached to the QPRTJOB.

Symptoms:

Jobs attempting to create, delete or change spooled files for that QPRTJOB could go into LCKW. Messages MCH5802/CPF3330 can result. Since this is an internal object, the WRKOBJLCK command cannot be used to determine the job holding the lock. The DSPLCKSTS command, shipped in the QSPTLIB library (available via PTF), can be used to determine the lock holder.

Recommendations:

- Reduce the number of server jobs that are swapped to that particular user.
- If possible, increase the number of users that the server job is swapping to.
- Ensure that the system is tuned correctly. Increase resources such as memory that is allocated to jobs accessing the QPRTJOB job's spooled files.

How IBM is dealing with this problem

- Changes in V5R3M0 were made to increase the number of spooled files created to the same output queue at any one time. This performance improvement was made to reduce contention on the output queue object when many jobs are creating spooled files to the same output queue at the same time. However, this change does not allow for shared access to the output queue. A spooled file create, delete or change will still conflict with a WRKOUTQ operation.
- Support was added in V5R3M0 to allow output queues and spooled files in an Independent auxiliary storage pool (IASP). This new support replaces the Internal print queue design with keyed database logical files. Each IASP can hold approximately 5 million spooled files. Another benefit of storing spooled files in an IASP is reduced spool abnormal IPL times. When the IASP is varied on, a background server job does the spooled file validation normally done during IPL for spooled files in SYSBAS.
- Design work is being pursued to provide enhanced spooled file management support.
- Design work is being pursued to allow for shared access and retained data integrity when accessing the internal print queue and output queue objects.

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