Journal Performance Considerations that Everyone Should Know

Session ID: 170441
Agenda Key: 43AF
Don’t Forget Your Session Surveys

• Sign in to the Online Session Guide (www.common.org/sessions)
• Go to your personal schedule
• Click on the session that you attended
• Click on the Feedback Survey button located above the abstract

Completing session surveys helps us plan future programming and provides feedback used in speaker awards. Thank you for your participation.

This is a New Session... I need your feedback
Journal Questions?

Bruce Hansel would love to hear from you…

bhansel@us.ibm.com

And so would I … 😊
Bruce-isms

“The journal tells the story”

“If it hits the journal, we’re going to complete that operation to the database”

“A journaled object should never, ever, never become damaged”

“A Flash is a Crash”

“If async remote journal is falling behind, it's the network!”

“Disk write performance improvements will help journal”

“If you want faster IPLs and IASP vary ONs, get current”
Session Takeaways… (why wait till the end?)

- Journaling will typically perform very well
- Slow Local Journal performance? Look at the I/O Subsystem
- Slow Asynchronous Remote Journal performance? It’s the Network
- Sudden Journal performance spike? Check on SMAPP
- Journal Caching can help, but using it involves some risk
- If you want faster IPLs, Get current and stay current
- If you want faster iASP Vary ONs, Get current and stay current
Session Takeaways...

• Journaling will typically perform very well
• Slow Local Journal performance? Look at the I/O Subsystem
• Slow Asynchronous Remote Journal performance? It’s the Network
• Sudden Journal performance spike? Check on SMAPP
• Journal Caching can help, but using it involves some risk
• If you want faster IPLs, Get current and stay current
• If you want faster iASP Vary ONs, Get current and stay current

• Did I mention?
  We advise that you: Get current and stay current
Why should you Journal your objects?

- Disaster Recovery
- High Availability
- Auditing
- Preventing damage
Why should you Journal your objects?

- Disaster Recovery
  - Ability to recover from unexpected events

- High Availability

- Auditing

- Preventing damage
Why should you Journal your objects?

- Disaster Recovery
  - Ability to recover from unexpected events

- High Availability
  - Cloned systems reduce downtime

- Auditing

- Preventing damage
Why should you Journal your objects?

- **Disaster Recovery**
  - ✓ Ability to recover from unexpected events

- **High Availability**
  - ✓ Cloned systems reduce downtime

- **Auditing**
  - ✓ Ability to answer important questions

- **Preventing damage**
Why should you Journal your objects?

- **Disaster Recovery**
  - ✓ Ability to recover from unexpected events

- **High Availability**
  - ✓ Cloned systems reduce downtime

- **Auditing**
  - ✓ Ability to answer important questions

- **Preventing damage**
  - ✓ Insurance policy for IBM i objects
## Protection against Data Loss & Damage

<table>
<thead>
<tr>
<th>Risk</th>
<th>Risk prevention/protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power loss</td>
<td>Uninterruptable power supply (UPS), cache batteries for storage controller.</td>
</tr>
<tr>
<td>Disk drive failure</td>
<td>RAID protection</td>
</tr>
<tr>
<td>Storage subsystem hardware failure</td>
<td>Redundancy, hardware level replication at storage subsystem level.</td>
</tr>
<tr>
<td>System hardware failure</td>
<td>Redundancy, hardware or software level replication at system level.</td>
</tr>
<tr>
<td>Communication failure or abnormal process termination</td>
<td>Rollback of uncommitted transactions.</td>
</tr>
<tr>
<td>Abnormal system termination</td>
<td>Attempt to force all main storage pages to disk; inspect and repair all tables and indexes that were in use at time of system termination; rollback all open transactions, database recovery objects, system-managed access path protection (SMAPP).</td>
</tr>
<tr>
<td>Data corruption by software or handling error</td>
<td>Backup, point-in-time recovery by restoring last backup and applying journaled changes.</td>
</tr>
</tbody>
</table>
Database Journaling

- DML Statement (INSERT, UPDATE, DELETE)
- DDL Statement (CREATE, ALTER, DROP TABLE)
- Journal receiver
- Remote journal (optional)

"Before image"
"After image"
Forcing Data to External Storage

Asynchronous writes for best performance – but:

- What happens if system crashes?
- Data in main storage may be lost.

Three different cases:

1. **File is not journaled („legacy“):**
   File parameter FRCRATIO defines number of record changes after which data is forced to external storage

2. **Journaled, but no commitment control:**
   Each journal entry is forced to external storage (slow!)

3. **Commitment control (requires journal):**
   Journal entries are forced to external storage at commit time

Special case “Data Definition Language” (DDL):

- Database recovery objects to ensure database integrity
Commitment Control

Successful transaction:

<table>
<thead>
<tr>
<th>Start of transaction</th>
<th>DELETE FROM T1</th>
<th>INSERT INTO T2</th>
<th>INSERT INTO T3</th>
<th>UPDATE T4</th>
<th>COMMIT</th>
</tr>
</thead>
</table>

Cancelled or interrupted transaction:

| Start of transaction | DELETE FROM T1 | INSERT INTO T2 | Rollback | → Changes to T1 and T2 are reset to state before start of transaction. |

- Without commitment control, the database would remain in an inconsistent state when a transaction gets cancelled or interrupted before completion.
- Commitment control requires journaling (before and after images).
- The before image is implicitly journaled even if the journal specified only *AFTER.
- “Optimistic processing”: Changes are applied to tables immediately, rollback requires time and creates additional journal entries.
- Isolation levels from low to high: No Commit (NC), Uncommitted Read (UR), Cursor Stability (CS), Read Stability (RS), Repeatable Read (RR).
Local Journals – Receiver size options (RCVSIZOPT)

- *SYSDFT - Equivalent to specifying *MAXOPT2 and *RMVINTENT
- *NONE – Ancient, not taking advantage of receiver enhancements
- *MAXOPT1 – Small maximum size of a journal entry
- *MAXOPT2 – Large maximum size of a journal entry
- *MAXOPT3 – Large maximum size of a journal entry, much larger max sequence number
  + No automatic CHGJRN done at IPL time!
  - Concern of breaking 3rd party software
- *RMVINTENT – Space savings
- *MINFIXLEN – Space savings if your only need is recovery

Why should everyone be using (at least) *MAXOPT2 & RMVINTENT?
1. *MAXOPT2 allows for a larger receiver size (1TB vs. 2GB) and a larger max sequence number (10B vs. 2B)
2. *RMVINTENT allows internal IX entries to be recycled which prevents bloating due to large IX entries
3. Both 1 & 2 are risk free
Local Journals – Receiver size options (RCVSIZOPT)

- SQL to the rescue

```sql
-- Search for journals not using the *RMVINTENT optimization
SELECT * FROM QSYS2.JOURNAL_INFO WHERE REMOVE_INTERNAL_ENTRIES = 'NO' AND RECEIVER_MAXIMUM_SIZE LIKE '*MAX%';

-- Search for journals using the ancient RCVSIZOPT
SELECT * FROM QSYS2.JOURNAL_INFO WHERE RECEIVER_MAXIMUM_SIZE = '*MAXOPT1';
```
Local Journals – Receiver size options (RCVSIZOPT)

- Lets automate this...

```sql
-- Find any *MAXOPT1 journals and change them to *MAXOPT2 *RMVINTENT
BEGIN
DECLARE V_Journal_NAME, V_Journal_Library VARCHAR(10);
DECLARE V_EOF INTEGER DEFAULT 0;
DECLARE Journal_query_text varchar(1000) DEFAULT 'SELECT rtrim(Journal_Library), rtrim(Journal_Name) FROM QSYS2.JOURNAL_INFO WHERE RECEIVER_MAXIMUM_SIZE = ''*MAXOPT1'' ';
DECLARE Older_Journals CURSOR FOR Older_Journals_query;
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET V_EOF = 1;
PREPARE Older_Journals_query FROM Journal_query_text;
OPEN Older_Journals;
L1 : LOOP
  FETCH Older_Journals INTO V_Journal_Library, V_Journal_NAME ;
  IF (V_EOF = 1) THEN LEAVE L1; END IF;
  EXECUTE IMMEDIATE 'CALL QSYS2.QCMDEXC(''CHGJRNJRN('' CONCAT V_Journal_Library CONCAT ''/'' CONCAT V_Journal_NAME CONCAT '' ) JRNRCV(*GEN) RCVSIZOPT(*MAXOPT2 *RMVINTENT)' ');
END LOOP;
CLOSE Older_Journals;
END;
```
Local Journals – Receiver size options (RCVSIZOPT)

- IPL and Vary ON consideration...

*MAXOPT2* System-managed receivers are processed on IPL:
- CHGJRN to attach a new receiver and to reset sequence numbers

*MAXOPT3* don't require this because the max sequence number is essentially infinite vs. 10B for MAXOPT2

- Ask yourself whether you can move to *MAXOPT3* receivers?
  ➔ If the answer is YES, make the change and your IPLs and iASP Vary Ons will benefit
5 Steps to improving iASP Vary ON performance

1) Get current on PTFs
   - Recent cumulative PTF package
   - Current PTFs for IPL / vary on IASP steps

2) Increase hardware resources
   - Cores, memory, I/O subsystem

3) Consider application/environment changes
   - Eliminate inefficient practices to gain Vary ON/IPL efficiencies

   Examples of inefficient practices:
   Large, long commit cycles that need to be rolled back, RECOVER(*IPL) makes vary on wait, SMAPP set to *NONE

4) Identify problem areas for IBM to investigate

5) *MAXOPT3 topic from the previous slide
Remote Journals

A choice needs to be made... do you favor performance or reliability?

- ChgRmtJrn command activates sending entries
  - **DELIVERY(*ASYNC)** – Journal entries are replicated asynchronously
    + Faster... control returns after entry has been queued on remote journal
    - Possible data loss... no guarantee that the entry made it over
  - **DELIVERY(*SYNC)** – Journal entries are replicated synchronously
    + No data loss... control returns after remote system writes the entry
    - Slower... production side is held up for communication and processing
Remote Journals

- “To Sync or not to Sync... that is the question”

-- Review the DELIVERY choice for remote journals
-- Note that this would also show if the journals were in catch-up mode

```
SELECT JOURNAL_DELIVERY_MODE, count(*) as mode_count FROM QSYS2.JOURNAL_INFO J
WHERE JOURNAL_TYPE = '*REMOTE' AND JOURNAL_STATE = '*ACTIVE'
Group by JOURNAL_DELIVERY_MODE
order by mode_count desc;
```
Remote Journals

- Who knew that SQL was so much fun?

```sql
-- Which remote journals fell the farthest behind this week?
SELECT MAXIMUM_TIME_BEHIND, A.* FROM QSYS2.JOURNAL_INFO A
WHERE
    MAXIMUM_BEHIND_TIMESTAMP > CURRENT TIMESTAMP - 7 DAYS AND
    MAXIMUM_TIME_BEHIND > 0 AND
    MAXIMUM_TIME_BEHIND IS NOT NULL
ORDER BY MAXIMUM_TIME_BEHIND DESC;
```
Remote Journals – Validity Checking

- If you’re not confident in the network, consider using `VLDCHK(*ENABLED)`
- The default is `VLDCHK(*DISABLED)`
- **Cost...** CPU, probably negligible
- **Benefit...** Automatic detection & protection of remote journals
- If the data does not match, the data will not be written to the target system, the remote journal environment will be inactivated, and messages indicating the communications failure will be issued to the journal message queue and QHST.

```
Change Remote Journal (CHGRMTJRN)

**Synchronous sending time-out:** *SAME* 1-3600 *SAME* *SYSDEF
**Validity checking:** . . . . . . *enabled* *SAME, *DISABLED, *ENABLED
**Automatic restart:**
```

© 2016, 2017 IBM Corporation
SMAPP – A great feature & a great acronym?

- **System-managed access-path protection (SMAPP)** reduces the amount of time it takes to restart the system or vary on an independent disk pool, after an abnormal end.
- The admin sets the **AP recovery time**.
- The default AP recovery time is 50 minutes.
- Recovering from journal takes **seconds/minutes vs. hours/days** to rebuild.
- Application or Environmental changes can cause additional APs to be journaled, can surface as perf issue.
- Use the DSPRCYAP and EDTRCYAP commands to review and change SMAPP.
SMAPP – Ineligibles

- **Formula:** $\text{SUM(<Rebuild Time>)} / \text{Active Processor Count}$
- Typical reason for ineligible AP is that it is built over multiple files that are journaled to different journals
- Eliminate this or specify only eligible APs should be journaled
- Large SMAPP-ineligible APs can cause O/S to overcompensate by journaling other smaller APs
- AP journal entries are written to same journal as file over which they are built
- Specify RCVSIZOPT(*RMVINTENT) or RCVSIZOPT(*SYSDFT) to minimize impact of additional AP entries
Journal Caching

- Found in (priced) HA Journal Performance – Option 42
- Journal is a write-ahead and secure log
  - The entry is written to disk BEFORE the change is made to the object and before control is returned
- Journal Caching
  - Journal entries are cached in main storage, before writing them to disk
  - Originally designed as a Batch processing accelerator
  - Useful when journaling is being used to enable replication to a second system
- Only helps when Commitment Control is set to *NONE or NC
- Journal cache documentation:
Journal Caching

- One more time...

```sql
-- Do I have Option 42 - HA Journal Performance installed?
SELECT license_expiration, l.*
FROM qsys2.license_info l
WHERE installed = 'YES' AND
  feature_id = '5117';

-- Review whether Journals are configuration to use Journal Caching
SELECT JOURNAL_CACHE, COUNT(*) AS COUNT FROM QSYS2.JOURNAL_INFO
GROUP BY JOURNAL_CACHE;
```
Submit a Requirement

- Be part of the voice to IBM i & DB2 for i
- Explain the business problem you’re encountering
- Submit feature requests and requirements to IBM

Submit COMMON Requirement for IBM i on IBM RFE site:

Submit online at https://ibm.co/2oLNf3v or scan this QR code
Session Takeaways…

• Journaling will typically perform very well
• Slow Local Journal performance? Look at the I/O Subsystem
• Slow Asynchronous Remote Journal performance? It’s the Network
• Sudden Journal performance spike? Check on SMAPP
• Journal Caching can help, but using it involves some risk
• If you want faster IPLs, Get current and stay current
• If you want faster iASP Vary ONs, Get current and stay current

• Did I mention?
  We advise that you: Get current and stay current
References
Journal resources

- Journal management and system performance

- Remote Journal enhancements in IBM i 7.1

- Journal Technote:

- Journal Caching: Understanding the Risk of Data Loss:

- iASP Vary ON improvements:

- Remote Journal FAQ

- Soft Commit:
IBM i Performance FAQ - a MUST read!