

Managing Sequences in a RAC Environment

Joel Goodman
Oracle University UK
May 8th 2008

Agenda

- **Introduction**
- **Application Sequences with Non-RAC Databases**
- **Oracle Sequences with Non-RAC Databases**
- **Application Sequence with RAC Databases**
- **Oracle Sequences with RAC Databases**
- **Sequences and Index Contention**
- **Migration Issues for Existing Applications**
- **Sequences and Services**
- **Demonstration**
- **Questions & Answers**



About Me



- **Joel.Goodman@oracle.com**
- **Application Development and Support 1976 to 1982**
- **Software Development and Consultancy 1983 to 1986**
- **IT Training Mainframe Technology 1986 to 1993**
- **Oracle Support 1994 to 1996**
- **Oracle University 1997 to Present**
- **OCP DBA 1997 to Present - Releases 7.3 to 10g**
- **Oracle 9i OCM DBA from 2002 10g from 2006**
- **DBA Curriculum Delivery: Basic, Advanced, DSI**
- **Technical Head of DBA OCP/OCM Development Team**
- **Member DBA Curriculum Quality Review Team**
- **Member Oak Table Network**

About This Topic

- **There is more to Sequences than meets the eye**
 - **Comprehension of Sequences**
 - **Architecture of Sequences**
 - **Administration Issues**
 - **Application Migration Issues**
 - **Performance Issues**
- **The good news is that Sequences are NOT a cost option** 😊

Sequence Number Generation in Oracle

- **What is a Sequence**
- **How may Sequence Numbers be handled in Oracle**
 - **With an Application Defined Number Generator**
 - **With an Oracle Sequence Number Generator**

Note: The term “Sequence numbers” is used here to describe allocating numbers for application or system use that may have gaps and may wrap around after reaching a defined maximum value. They need not start with 1 nor be incremented by 1;

Creating Application Sequences

- Define an Application Sequence Table
- Insert the Initial value into the Sequence column

```
SQL> CREATE TABLE SEQTAB (SEQNUM NUMBER);  
Table created.
```

```
SQL> INSERT INTO SEQTAB VALUES (1);  
1 row created.
```

```
SQL> COMMIT;  
Commit complete.
```

Using Application Sequences

- **Application gets current Sequence number from table**
- **Then the number is incremented and committed**

```
SQL> VARIABLE SEQNUM NUMBER
SQL> SELECT SEQNUM INTO :SEQNUM FROM SEQTAB FOR UPDATE;
SEQNUM
-----
          1
1 row selected.
```

```
SQL> UPDATE SEQTAB SET SEQNUM = SEQNUM + 1;
1 row updated.
```

```
SQL> COMMIT;
Commit complete.
```

Issues with Application Defined Sequences in Single Instance Databases

- **Problems with this approach**
 - Row level locking will cause contention
 - Update of sequence is not a recursive transaction
 - This may be handled by using Autonomous Transactions for the update of the sequence
 - At some level of concurrency contention will be a serious issue
- **Advantages with this approach**
 - Gaps may be eliminated by deferring the commit at the risk of additional contention but this can not occur with an Autonomous Transactions

Issues with Application Sequences in RAC Databases

- **Problems with this approach**
 - Same issues as with a Single Instance **plus:**
 - Extra Global Enqueue Services traffic due to Global TX and TM Enqueues for both Transaction and Table
 - Extra Global Cache Services traffic for shipping the dirty table block amongst involved instances
- **Advantages with this approach**
 - Same advantages as with a Single Instance

Oracle Sequences with Non-RAC Databases

- Created with the `CREATE SEQUENCE` SQL statement
- `NOCACHE` is preferred for low volume use of sequences as this will not expose the sequences to the risk of potentially large gaps.
- `CACHE` is the default option and is used to improve performance for high volume use of sequences
- Sequence definition and current attributes may be viewed in `DBA_SEQUENCES`

CACHE Option Details Non-RAC

- **The CACHE option (protected by SQ Enqueue):**
 - **Caches sequence numbers in the Row Cache**
 - **CACHE value defaults to 20**
 - **This is visible in V\$_SEQUENCES.NEXTVALUE**
 - **Improves performance by reducing DD access**
 - **Gaps may occur due to:**
 - **ROLLBACK SQL statement issued**
 - **Process cleaned up by PMON**
 - **Metadata ages out of the row cache**
 - **Shared Pool is flushed**
 - **Instance shutdown**
 - **DG Switchover or Failover**
- **Use DBMS_SHARED_POOL.KEEP to keep sequences and prevent gaps due to aging**
- **Use V\$DB_OBJECT_CACHE to check for kept objects**

Creating and Keeping a Cached Sequence

```
SQL> CREATE SEQUENCE KEEPSEQ START WITH 1 INCREMENT BY 1 CACHE 1000;  
Sequence created.
```

```
SQL> EXEC DBMS_SHARED_POOL.KEEP('KEEPSEQ','Q');  
PL/SQL procedure successfully completed.
```

```
SQL> SELECT OWNER,NAME,KEPT FROM V$DB_OBJECT_CACHE WHERE  
TYPE='SEQUENCE';
```

OWNER	NAME	KEPT
-----	-----	----
SYS	KEEPSEQ	YES

```
1 row selected.
```

Using DBA_SEQUENCES in a Single Instance Database

- Use DBA_SEQUENCES to see the database wide properties of Sequences
- LAST_NUMBER column shows value last updated in the dictionary for each sequence

```
SQL> SELECT MIN_VALUE,MAX_VALUE,INCREMENT_BY,
2  CYCLE_FLAG,CACHE_SIZE,LAST_NUMBER
3  FROM DBA_SEQUENCES
4  WHERE SEQUENCE_NAME = 'KEEPSEQ';
```

MIN_VALUE	MAX_VALUE	INCREMENT_BY	C	CACHE_SIZE	LAST_NUMBER
1	1.0000E+27	1	Y	1000	5435

1 row selected.

Using V\$_SEQUENCES

- Use V\$_SEQUENCES to see next value to be assigned for a sequence from the cache

```
SQL> SELECT SEQUENCE_OWNER, SEQUENCE_NAME, NEXTVALUE
        FROM V$_SEQUENCES WHERE SEQUENCE_NAME = 'KEEPSEQ' ORDER BY 1;
```

SEQUENCE_O	SEQUENCE_N	NEXTVALUE
-----	-----	-----
SYS	KEEPSEQ	2

```
1 row selected.
```

Oracle Sequences with RAC Databases

- **Created and administered as in a Single Instance**
- **Real Application Clusters allows multiple instances to obtain unique numbers from the same sequence**
- **Sequences in Real Application Clusters:**
 - **Have the same options as single-instance databases plus the ORDER and NOORDER options**
 - **ORDER and NOORDER options control ordering of generated numbers**
 - **The Global Cache Service may need to coordinate sequences across instances**
 - **SQ Enqueue will be Global**

CACHE Option Details RAC

- **Mostly the same as in non RAC Databases**
- **Use `GV$SEQUENCES.NEXTVALUE`**
- **`DBMS_SHARED_POOL.KEEP` must be used in all Instances**
- **Use `GV$DB_OBJECT_CACHE` to check for kept objects**
- **Enqueue Waits will appear on `DC_SEQUENCES`**
- **If cache size is too small, you will see contention for `SQ enqueue` in the `V$ENQUEUE_STATISTICS` data dictionary view**

ORDER Option Details in RAC

- **ORDER or NOORDER options are used to control behaviour of Sequences in all cases.**
- **Default options for Sequences are CACHE with NOORDER**
- **Performance of combined options worst to best:**
 - **NOCACHE with ORDER**
 - **NOCACHE with NOORDER**
 - **CACHE with ORDER**
 - **CACHE with NOORDER**

Sequences with NOCACHE and ORDER

- Use these settings when ordered sequences are required and gaps need to be minimised
- Note that gaps may always occur due to failures or rollbacks
- This has the biggest performance impact on RAC
- May cause high wait times for 'row cache lock' waits for the DC_SEQUENCES row cache entry
- GET requests, GES requests and GES conflicts may be correlated with the number of executions for a SQL statement obtaining a new sequence number.

Sequences with NOCACHE and NOORDER

- **Use these settings when ordered sequences are NOT required and gaps need to be minimised**
- **Note that gaps may always occur due to failures or rollbacks**
- **This combination performs better than NOCACHE with ORDER**
- **Each instance allocates numbers by access to the database but cache fusion may delay sending current seq\$ block to a requesting instance if the block is busy owing to many sequence number allocations from the instance owning the current block image.**
- **This means that ordering is not guaranteed despite the absence of caching**

Sequences with `CACHE` and `ORDER`

- Use these settings when ordered sequences are required and gaps are tolerated
- This combination performs better than `NOCACHE` type sequences
- Each instances caches the same set of numbers in the row cache
- Server processes wait on “DFS Lock Handle”
- The synchronization of the sequence next value is done using the lock value in an `SV` instance lock on the sequence resource managed by `LCK0`
- When the requesting `LCK0` process receives the lock grant, it can forward the lock value (i.e. new sequence number) to the foreground process (requestor).

Sequences with `CACHE` and `NOORDER`

- **Use these settings when ordered sequences are NOT required and gaps are tolerated**
- **This combination performs better than `CACHE` with `ORDER` type sequences and provides best performance in RAC**
- **Each instances caches its own range of numbers in the row cache**
- **Caches are not Synchronised**

Using DBA_SEQUENCES in a RAC Database

- Use DBA_SEQUENCES as in a single instance
- Use ORDER_FLAG Column to check for ordering

```
SQL> CREATE SEQUENCE KEEPSEQ START WITH 1 INCREMENT BY 1 CACHE 1000
      ORDER;
Sequence created.
```

```
SQL> SELECT MIN_VALUE,MAX_VALUE,INCREMENT_BY,
2  CYCLE_FLAG,ORDER_FLAG,CACHE_SIZE,LAST_NUMBER
3  FROM DBA_SEQUENCES
4  WHERE SEQUENCE_NAME = 'KEEPSEQ';
```

MIN_VALUE	MAX_VALUE	INCREMENT_BY	C	O	CACHE_SIZE	LAST_NUMBER
1	1.0000E+27	1	Y	Y	1000	1

Using GV\$_SEQUENCES

- Use GV\$_SEQUENCES to see next value to be assigned for a sequence in each instance
- Use ORDER_FLAG column to check for ordering
- Note that without the ORDER option each instance caches its own range of numbers

```
SQL>SELECT INST_ID,SEQUENCE_OWNER,SEQUENCE_NAME,NEXTVALUE,ORDER_FLAG  
FROM GV$_SEQUENCES WHERE SEQUENCE_NAME = 'KEEPSEQ' ORDER BY 1;
```

INST_ID	SEQUENCE_O	SEQUENCE_N	NEXTVALUE	O
1	SYS	KEEPSEQ	13	Y
2	SYS	KEEPSEQ	1027	Y

Sequence Contention: Symptoms

- **Wait events**
 - enq: SQ – contention
 - Row cache lock
 - DFS lock handle
- **System statistics**
 - global enqueue CPU used by this session
 - global enqueue get time
 - global enqueue gets async
 - global enqueue gets sync
 - global enqueue releases

Checking For SQ Contention

- Use the following to check for “SQ” waits

```
SQL>SELECT SID,P1,P2,P3,SECONDS_IN_WAIT  
FROM V$SESSION  
WHERE EVENT = 'enq: SQ - contention';
```

SID	P1	P2	P3	SECONDS_IN_WAIT
---	-----	-----	-----	-----
37	31937	0		0
56	31924	0		1

Checking For DFS lock Handle

- Use the following to find DFS lock handle locks

```
SQL> SELECT SID,TYPE, ID1, ID2
       FROM V$LOCK
       WHERE TYPE='SV' ;
```

SID	EVENT	P1	P2	P3
21	DFS lock handle	1398145029	276750	
33	DFS lock handle	1398145029	276750	
36	DFS lock handle	1398145029	276750	
52	DFS lock handle	1398145029	276750	
59	DFS lock handle	1398145029	276750	

Checking For SV Contention

- Use the following to check for “SV” waits

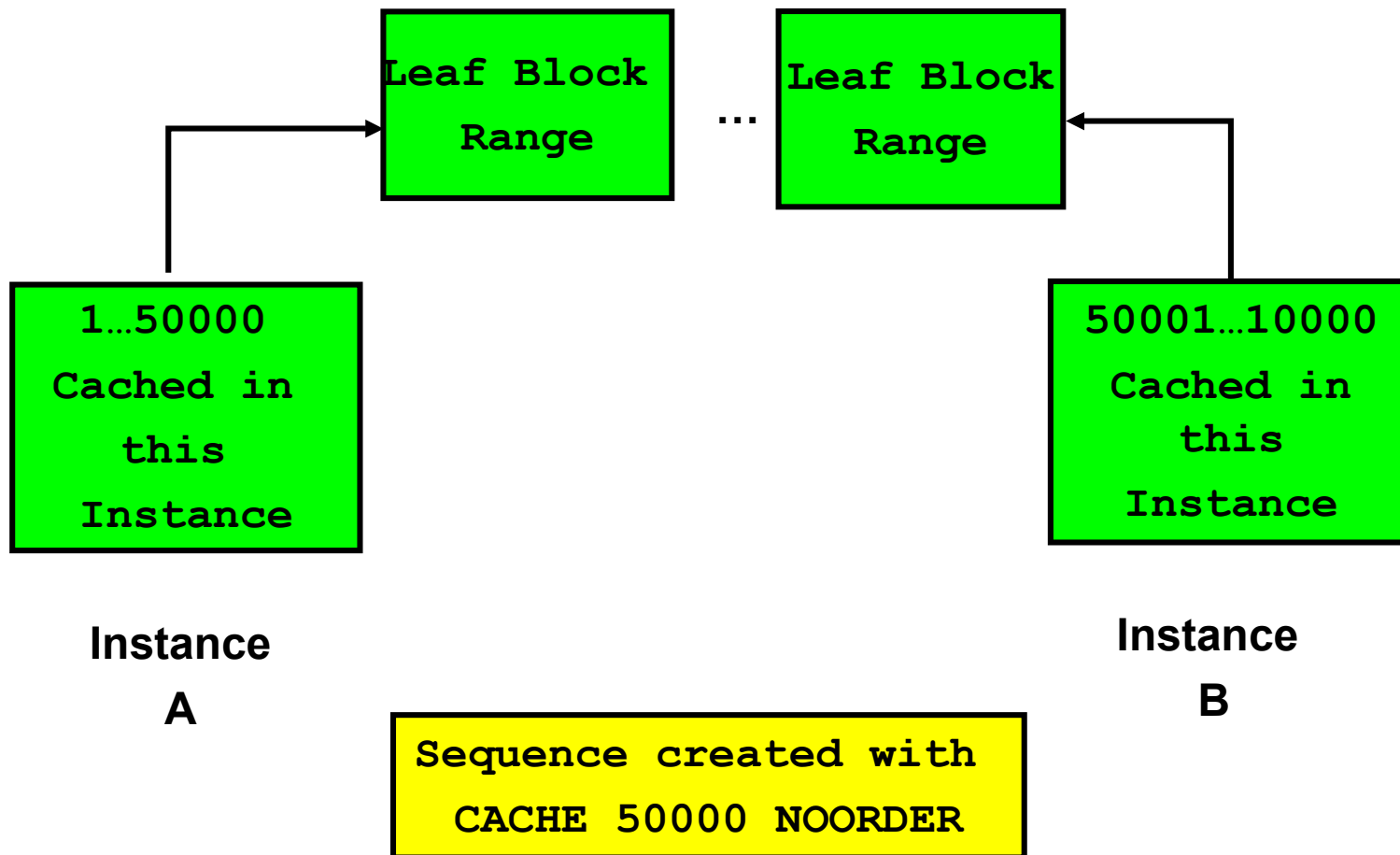
```
SQL> SELECT SID, CHR(BITAND(P1,-16777216)/16777215) ||  
        CHR(BITAND(P1, 16711680)/65535) "LOCK",  
        TO_CHAR(BITAND(P1, 65536)) "MODE",  
        P2, P3,SECONDS_IN_WAIT  
FROM V$SESSION  
WHERE EVENT = 'DFS lock handle';
```

SID	LOCK	MODE	P2	P3	SECONDS_IN_WAIT
25	SV	0	27675	0	0
35	SV	0	27675	0	0
41	SV	0	27675	0	0
45	SV	0	27675	0	0
76	SV	0	27675	0	1

Oracle Sequences and Index Contention

- **Insert intensive applications using Sequence generated index keys may suffer from index leaf block contention**
- **In RAC, this may lead to a high rate of Global Cache Service traffic**
- **By using larger sequence caches with the NOORDER option successive index block splits tend to create instance affinity to index leaf block ranges**
- **After a certain number splits, each instance will write to different ranges of leaves in the index tree.**
- **The higher the insert rate the larger the CACHE value should be.**

CACHE, NOORDER and Index Leaf Ranges



Index Block Contention: Symptoms

- **Wait events**
 - enq: TX - index contention
 - gc buffer busy on Index Branch or Leaf Blocks
 - gc current block busy on Remote Undo Headers
 - gc current split
- **System statistics**
 - Leaf node splits
 - Branch node splits
 - Exchange deadlocks
 - gcs refuse xid
 - gcs ast xid
 - Service ITL waits

Migration Issues for Existing Applications Converting to RAC

- **Functionality**
 - Is application sensitive to ordering for business or legal reasons
 - If yes then Sequence may require redefinition as NOORDER option is the default
- **Performance**
 - Performance may degrade in RAC due to Global Cache Service traffic overheads especially with Indexes based on sequence generated keys.
 - Solution to this is bigger sequence cache but NOORDER is needed to make this work.
 - This may not be possible if functionality demands ordered numbers due to application requirements

Sequences and Services

- **High volume insert intensive applications using Sequence generated keys add extra overhead**
- **NOCACHE is the worst for performance**
- **CACHE NOORDER is the best for performance**
- **Services may help improve performance even for CACHE NOORDER sequences if the insert application uses a service with only one Preferred Instance.**
- **This eliminates the Interconnect traffic and synchronisation overhead required to coordinate the next value from amongst all the instances when using CACHE ORDER.**

Services Administration

- **SRVCTL may be used to create Services for RAC Databases**
- **EM Database or Grid Control may be used as well**
- **Application must connect with the correct TNSNAMES entry so that the Preferred Instance for the service is the only instance used by the application**
- **This assures that the Sequence is only incremented by one instance and eliminates the extra block shipment overhead**

Using SRVCTL to Add a Service

```
Oracle$ srvctl add service -h
```

```
Usage: srvctl add service -d <name> -s <service name> -r  
"<preferred_list>" [-a "<available_list>"] [-P <TAF_policy>]
```

```
-d <name>           Unique name for the database  
-s <service>       Service name  
-r "<pref_list>"   List of preferred instances  
-a "<avail_list>"  List of available instances  
-P <TAF_policy>    TAF policy (NONE, BASIC, or PRECONNECT)
```

```
Usage: srvctl add service -d <name> -s <service name> -u {-r  
"<new_pref_inst>" | -a "<new_avail_inst>"} -
```

```
-d <name>           Unique name for the database  
-s <service>       Service name  
-u                 Add a new instance to service configuration  
-r <new_pref_inst> Name of new preferred instance  
-a <new_avail_inst> Name of new available instance  
-h                 Print usage
```

```
Oracle$ srvctl add service -d ORCL -s SEQAPP -r ORCL1 -a ORCL2
```

Specifying the Service in TNSNAMES .ORA

```
SEQAPP =  
  (DESCRIPTION =  
    (ADDRESS = (PROTOCOL = TCP) (HOST = MYHOST1VIP) (PORT = 1521))  
    (CONNECT_DATA =  
      (SERVER = DEDICATED)  
      (SERVICE_NAME = SEQAPP)  
    )  
  )  
)
```

Demonstration Environment

The demo environment consists of:

- **Oracle 11g RAC on Linux**
- **Oracle Database 11g – 11.1.0.6.0**
- **To request this demo script please Email me at:**
 - **Joel.Goodman@oracle.com**

Demonstration Script

The script will demonstrate the use of sequences in a RAC environment and focus on the all possible possible combinations using CACHE and ORDER. It will also track the sequence using the Data Dictionary and the undocumented dynamic performance view GV\$_SEQUENCES

Q U E S T I O N S
&
A N S W E R S

TGIO

Thank You

Please Email any questions to:

- Joel.Goodman@Oracle.com