Exadata Sparse Disk Snapshots
Safe Harbor

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle’s future plans, expectations, beliefs, intentions and prospects are “forward-looking statements” and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle’s Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading “Risk Factors.” These filings are available on the SEC’s website or on Oracle’s website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.
Agenda

1. Features
2. Hierarchical Snapshots
3. Sparse Test Masters
4. Monitoring and Statistics
5. Miscellaneous
Agenda

1 Features
2 Hierarchical Snapshots
3 Sparse Test Masters
4 Monitoring and Statistics
5 Miscellaneous
Recap Features

Exadata Sparse Snapshot Benefits

• Ability to quickly create space efficient test/dev databases on Exadata
• Sparse snapshot test/dev databases can use all Exadata Smart features including smart offload capabilities so applications can evaluate using Exadata features
• Sparse snapshot test/dev databases are NOT full copies resulting in storage and cost savings
• HCC storage compression works transparently providing additional storage savings
Features

Terminology

• Sparse Griddisk
  • Virtual size
  • Physical size
• Sparse Diskgroup
• Sparse Data Files
• Exadata Snapshots
• Exadata Test Master
• Exadata Sparse Test Master
Features

Test/Dev Lifecycle using Exadata Snapshots

• Production Database runs on Exadata
• Full Cloned Test Master database created from Standby or Production on Exadata
  • Optionally mask sensitive data in Test Master
  • Space efficient Test/Dev databases created from here with one-command for PDBs
  • Exadata Smart features (query offload; storage index; smart log; smart flash cache; HCC; etc) available on snapshots
  • Challenge – Refreshing the Test Master invalidates existing snapshots; must create new full Test Master to create new refreshed Exadata snapshots

Exadata Hardware

Primary

Redo Apply

Standby

Full Clone

Test Master (Read Only)

Snapshot #1

Snapshot #2

Snapshot #3

SIZE

Full

Full

Sparse

Sized
Features

Sparse Database/File/GridDisk

- **Sparse Database**
  - Only the datafiles in a sparse database are sparse
  - controlfile/online redo logs/tempfiles; etc are not sparse

- **Sparse File**
  - Sparse datafile points back to Test Master database datafile
  - Only allocates blocks on-demand during writes

- **Sparse Griddisk**
  - Exposes a virtual size in addition to a physical size
  - Max. allowed virtual size/disk = 100TB
  - Max. Allowed aggregate physical size/disk = 4TB

Cellcli> create griddisk all harddisk prefix=SPARSE size=200G, virtualsize=5TB
Features

Sparse Diskgroup

- Sparse Files can only be created in a sparse diskgroup
  - `cell.sparse_dg` is a new attribute that must be set to `allsparse` for a sparse diskgroup
  - Must have `compatible.asm` and `compatible.rdbms` set to 12.1.0.2 or higher
  - Uses 16X extent size for 4M AU; each extent is 64M
  - Sparse diskgroups use Virtually Allocated Metadata

SQL> create diskgroup SPARSEDG
   normal redundancy
   disk 'o/*/SPARSE_*'
   attribute
      'compatible.asm' = '19.0.0.0',
      'compatible.rdbms' = '12.2.0.2',
      'cell.smart_scan_capable' = 'true',
      'cell.sparse_dg' = 'allsparse',
      'au_size' = '4M';
Features
Create Exadata Snapshot Database

- Create sparse griddisks on the storage cells
- Create sparse diskgroup on the ASM instance
- Setup Test Master Database
  - Enable ASM ACCESS_CONTROL on the Test Master Diskgroup
  - Create full clone and mask; OR
  - Convert existing full clone on Exadata to a Test Master; OR
  - Convert standby database to a Test Master
- Create Sparse/Snapshot Database
  - `create_file_dest` must be set to a SPARSE diskgroup
  - Pluggable database.. Using SNAPSHOT COPY
  - Non-container database... using `DBMS_DNFS.CLONEDB_RENAMEFILE`

```sql
SQL> alter pluggable database TestMaster open read only;

SQL> create pluggable database JohnTest from TestMaster create_file_dest='+SPARSEDG' SNAPSHOT COPY;
```
Features

Exadata Snapshot Databases
Features
Exadata Snapshot PDBs

- **Production Database**
  - PDB1
  - PDB2
  - PDB3

- **Production System**
  - Clone PDBs
    - PDB1C1
    - PDB2C1
    - PDB3C1

- **Test Master PDBs**
  - Read only or mounted
  - Full copy of cloned PDBs

- **Snapshot PDBs**
  - Read only or read write
  - Sparse files contain only changed blocks

- **Test/Dev System**
  - Test Masters
  - Snapshots

Copyright © 2019 Oracle and/or its affiliates.
Features

Use Standby Database as Test Master

- Standby database cannot be running redo apply while serving as Test Master
- Must have ASM ACCESS_CONTROL enabled + ownership set
- Periodically refresh
  - Drop all snapshots including datafiles
  - Make all Test Master datafiles read write
  - Refresh Test Master from production via
    - DATAGUARD REDO APPLY; OR
    - RMAN RECOVER ... FROM SERVICE
  - Close Test Master and make all TM datafiles RO
  - Create new test snapshots for next week’s testing
Features

Refresh Standby Database

Step 1: Convert to test master database
- Defer redo transport
- Convert to Data Guard snapshot standby
- Prepare to be test master database
- Close database and open read-only

Step 2: Create Snapshots
- Create snapshots and use for dev/test

Step 3: Refresh test master database
- Drop snapshots
- Convert Data Guard snapshot standby back to original state
- Apply RMAN incremental from production to refresh the Data Guard replica
- Enable redo transport to complete the resynchronization process
- Repeat step 1
Features
Efficient Sparse Database Backup & Recovery

Option to create a L0 Sparse backups as Backup Set or Image Copy

```
BACKUP AS [NON]SPARSE {BACKUPSET | COPY} ...
```

Choose Sparse backup option per device

```
CONFIGURE DEVICE TYPE {SBT|DISK}..SPARSE ON|OFF;
```

Restore a Sparse Database as Sparse instead of full

```
RESTORE .. FROM {SPARSE|NONSPARSE} ..
```

Support for regular RMAN operations
- TSPITR, DUPLICATE, DELETE, LIST, SHOW, CATALOG etc.
- Sparse Backup Database, Tablespace, Datafiles, CDB, PDB
- Duplicate as Sparse Database instead of complete database

```
DUPLICATE DATABASE DB1 AS DB2 FROM SPARSE ...
```

COMPATIBLE parameter to be set to 12.2 or higher

Recommend 18c or later for sparse backup usage

Copyright © 2019 Oracle and/or its affiliates.
Agenda

1. Features
2. Hierarchical Snapshots
3. Sparse Test Masters
4. Monitoring and Statistics
5. Miscellaneous
Hierarchical Snapshots

Architecture

• Create Snapshots of databases from previously created Snapshots
  • CREATE PLUGGABLE DATABASE JOHNMon from JOHNTESTDB create_file_dest='+SPARSE' SNAPSHOT COPY;
• Syntax and technology remain unchanged
• Works with pluggable and non-pluggable databases
• Use case example
  • Development releases nightly build of the database
  • Tester creates a snapshot for himself and finds a bug
  • Tester creates a snapshot of his snapshot
  • Tester provides the new copy back to development for analysis
• Recommend snapshot tree depth <10 for performance
Agenda

1. Features
2. Hierarchical Snapshots
3. Sparse Test Masters
4. Monitoring and Statistics
5. Miscellaneous
Sparse Test Masters

Architecture

• Use cases
  • Ability to create snapshots at different points of time; without using up full space for each point of time.
  • Test Master can be a writable Data Guard Target

• Steps
  • At any point on the Test Master; stop data guard redo apply; and create a ‘Mon Master’.
  • Create a new Test Master which is space efficient snapshot of ‘Mon Master’
  • ‘Mon Master’ is now read only and can be treated as a parent to create other test/dev snapshots.
  • Repeat the step on Test Master to create ‘Tue Master’ (it is space efficient).
  • ‘Tue Master’ is a sparse test master
Sparse Test Masters

Life Cycle Start (Create Test Master)

- Create new Data Guard or RMAN full sized Test Master (TM) – can be in sparse or non-sparse diskgroup
  - Sparse Test Master files must be in sparse diskgroup

Starting Configuration

- Stop Data Guard Redo Apply to original Test Master
  - Make it a Read Only Test Master
- Create new Sparse Test Master
  - Start Data Guard Redo Apply to Refresh
Sparse Test Masters

Create Snapshots

- TM original is a Read Only Test Master
- Create Exadata snapshots from it
  - Joe Snap
  - Mary Snap
Sparse Test Masters

Refresh Test Master + Create New Snapshots

• Repeat process to create new Exadata Snapshots while keeping prior Exadata snapshots
• All Sparse Test Masters and Snapshots are sparse sized
Sparse Test Masters

Additional Test Master Refreshes

- We recommend a maximum of 10 levels – 9 Sparse Test Masters
  - For performance reasons we like to limit the hierarchical tree depth
  - Inclusive of levels of Test Masters and levels beneath a Test Master
- Beyond 10 levels
  - Drop all snapshots and sync TM original current to start over – OR-
  - Space permitting create new full Test Master
Agenda

1. Features
2. Hierarchical Snapshots
3. Sparse Test Masters
4. Monitoring and Statistics
5. Miscellaneous
Monitoring and Statistics

- Sparse IO stats in RDBMS
- Wait events in RDBMS
- v$ views
  - v$asm_disk_sparse
  - v$asm_diskgroup_sparse
  - v$clonedfile
RDBMS stats (v$sysstat; v$mystat)

New stats

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical read snap IO request no data</td>
<td>No physical IO done for these (i.e. wasted roundtrips)</td>
</tr>
<tr>
<td>Physical read snap IO request base</td>
<td>Number of physical IOs on base level</td>
</tr>
<tr>
<td>Physical read snap IO request copy</td>
<td>Number of physical IOs on any snap hierarchy</td>
</tr>
<tr>
<td>Physical read snap bytes base</td>
<td>Number of bytes read from the base</td>
</tr>
<tr>
<td>Physical read snap bytes copy</td>
<td>Number of bytes read from the snap</td>
</tr>
</tbody>
</table>
RDBMS stats (v$sysstat; v$mystat) contd...

Updated stats

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical read total IO requests</td>
<td>Number of physical IOs submitted by user</td>
</tr>
<tr>
<td>Physical read total multi block requests</td>
<td>Number of multi block IOs submitted by user</td>
</tr>
</tbody>
</table>

- Only the IOs that lead to a real physical IO will be counted here, using the same logic as described in wait events to omit completely sparse IOs.
RDBMS wait events

• Following wait events are monitored for 0 byte reads returned; i.e. sparse buffers
  • cell single block physical read
  • cell multi block physical read
  • cell list of blocks physical read

• List of blocks wait events are also tracked

• Then, we change the wait event to “cell sparse block physical read”
  • this wait event is significantly faster since there is no IO involved and if the request is large in size, then even network transfer is significantly faster because of packing
## ASM Sparse Disk (v$asm_disk_sparse)

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP_NUMBER</td>
<td>Number of the diskgroup containing the disk</td>
</tr>
<tr>
<td>DISK_NUMBER</td>
<td>Number assigned to the disk within this diskgroup</td>
</tr>
<tr>
<td>INCARNATION</td>
<td>Incarnation number for the disk</td>
</tr>
<tr>
<td>ALLOCATED_MAT_MB</td>
<td>Total used physical capacity on this disk</td>
</tr>
<tr>
<td>TOTAL_MAT_MB</td>
<td>Total physical capacity on this disk</td>
</tr>
<tr>
<td>SPARSE_READS</td>
<td>Number of read requests on sparse regions of this disk</td>
</tr>
<tr>
<td>SPARSE_BYTES_READ</td>
<td>Bytes read from sparse regions of this disk</td>
</tr>
<tr>
<td>SPARSE_READ_TIME</td>
<td>Time taken by sparse read IOs</td>
</tr>
</tbody>
</table>
v$asm_disk_sparse

SQL> select
  2       DISK_NUMBER          dsk_num,
  2      ALLOCATED_MAT_MB     alloc,
  2      TOTAL_MAT_MB         total
  3  from V$ASM_DISK_SPARSE
  4  where GROUP_NUMBER = 5;

<table>
<thead>
<tr>
<th>DSK_NUM</th>
<th>ALLOC</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5536</td>
<td>204774</td>
</tr>
<tr>
<td>1</td>
<td>5424</td>
<td>204774</td>
</tr>
<tr>
<td>2</td>
<td>5532</td>
<td>204774</td>
</tr>
<tr>
<td>3</td>
<td>5424</td>
<td>204774</td>
</tr>
<tr>
<td>4</td>
<td>5424</td>
<td>204774</td>
</tr>
</tbody>
</table>
## ASM Sparse Diskgroup (v$asm_diskgroup_sparse)

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP_NUMBER</td>
<td>Cluster wide number assigned to the diskgroup</td>
</tr>
<tr>
<td>ALLOCATED_MAT_MB</td>
<td>Total used physical capacity of the diskgroup</td>
</tr>
<tr>
<td>TOTAL_MAT_MB</td>
<td>Total physical capacity of the diskgroup</td>
</tr>
</tbody>
</table>

```sql
SQL> select
    2    ALLOCATED_MAT_MB alloc,
    3    TOTAL_MAT_MB total
    4 from V$ASM_DISKGROUP_SPARSE
    5 where GROUP_NUMBER = 5;

ALLOC      TOTAL
---------- ----------
197208    7371864
```

Name Meaning

GROUP_NUMBER Cluster wide number assigned to the diskgroup
ALLOCATED_MAT_MB Total used physical capacity of the diskgroup
TOTAL_MAT_MB Total physical capacity of the diskgroup
## v$clonedfile

<table>
<thead>
<tr>
<th>FILENUMBER</th>
<th>SNAPSHOTFILENAME</th>
<th>CLONEFILENAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>+DATA/TESTMASTER/09D05108AB70216BE053D6CBF00AA040/DATAFILE/system.257.865863315</td>
<td>+SPARSEDG/JOHNTEST/09D05108AB70216BE053D6CBF00AA041/DATAFILE/system.257.865863315</td>
</tr>
<tr>
<td>17</td>
<td>+DATA/TESTMASTER/09D05108AB70216BE053D6CBF00AA041/DATAFILE/sysaux.258.865863317</td>
<td>+SPARSEDG/JOHNTEST/09D05108AB70216BE053D6CBF00AA041/DATAFILE/sysaux.258.865863317</td>
</tr>
</tbody>
</table>

- Only works on mounted databases/files
- Can be run in either database instance or in ASM
  - In snapshot instance will display parent files for that snapshot
  - In ASM instance, possible to see parent/child relationships for all open/mounted snapshots
- Currently no way to view this parent/child relationship via ASMCMD
Agenda

1. Features
2. Hierarchical Snapshots
3. Sparse Test Masters
4. Monitoring and Statistics
5. Miscellaneous
Resize operations

Resizing

• Virtual or physical space of sparse can be changed
  • Remember the per disk virtual size limit of 100Tb and physical size limit of 4Tb

• To Modify Virtual Space
  • To increase
    • First alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,...,SPARSE_CD_11_CELL01 virtualSize=newBiggerSize;
    • Then alter the disk group in an ASM instance
      SQL> alter diskgroup SPARSE resize all size newBiggerSize;
  • To decrease
    • Ensure you have the free space to reduce virtual size
    • First alter the diskgroup in ASM
      SQL> alter diskgroup SPARSE resize all size newSmallerSize;
    • Then alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,...,SPARSE_CD_11_CELL01 virtualSize=newSmallerSize;
Resize operations

Resizing

• Physical space increases may require resizing of other disk groups/grid disks
  • See 3.3.3 Resizing Grid Disks in Oracle Exadata System Software
• Once physical space is available
  • To increase
    • First alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,....,SPARSE_CD_11_CELL01 Size=newBiggerSize;
    • No changes need to be done in ASM
  • To decrease
    • Ensure you have the free space to reduce physical size by checking physical usage in ASM
      SQL> SELECT sum(allocated_mat_mb) FROM v$asm_disk_sparse
        WHERE group_number = group_number_of_disksgrp_to_shrink;
    • Alter the cell disks
      CellCLI> alter griddisk SPARSE_CD_00_CELL01,SPARSE_CD_01_CELL01,....,SPARSE_CD_11_CELL01 Size=newSmallerSize;
    • No changes need to be done in ASM
ASMCMD sparse operations

Sparse File COPY

- Datafiles sometimes need to be copied
  - from one diskgroup to another; OR
  - one hardware to another

- Copy a sparse file in a space efficient manner to a new destination
  - `asmcmd> cp --sparse <src_sparse_file_list> <tgt_file_or_dir>
  - Need sparse copy to prevent exploding the file size at the destination
  - Destination file or directory must be on a sparse diskgroup
  - We also copy the OSD header of the source file to the destination file
  - If a sparse copy is done on a local ASM instance; the parent file is set during the copy
  - If a sparse copy is done on a remote ASM instance; parent file must exist in same state and should be explicitly set by the user

- The command accepts a set of input source files and copies it to a destination directory
ASMCMD sparse operations

Sparse File `setSparseParent`

- When the parent file is moved; copied to a different diskgroup/hardware
  - Must update child file’s parent info
- Sets the parent of a sparse file
  - `asmcmd> setSparseParent <sparse_child_file> <parent_file>`
  - Child file must be a sparse file
  - Parent file may be a sparse or non-sparse file
- Parent and child must have a valid relationship
  - child’s block 0 information must match with parent’s block 1 information
  - Indicates parent is at a precise point in time when the child was created (SCN; timestamp; etc)
- Parent and child files must be on the same ASM instance
- Most common usage after an RMAN restore with SET NEWNAME
Misc Improvements

• Sparse File in block 0 stores information about parent file's block 1 → SCN, timestamp
  • On an open of a sparse file we also open the parent, and at this point we validate the child and parent are still at a valid point
  • Prevents `setspareparent` from assigning an incorrect parent
  • Protects against opening the child if someone incorrectly wrote to the parent file

• Better errors when attempting a write on a read-only parent file
  • ORA-17528: A read only file or a file opened read only cannot be written to: +DATAFILE/dbs/cdb1_pdb1_ax.f
Release

All Exadata Snapshot features

• Available
  • Database software → 12.2.0.1
  • Grid software → 12.2.0.1
  • Cell software → 12.2.1.1.0
  • RMAN sparse backups → 18.1.0.0

• Recommended 19c or later
Documentation

- Exadata Storage Server Software User's Guide
  - Chapter 9 ➔ Setting up Oracle Exadata Storage Snapshots
Integrated Cloud
Applications & Platform Services