SecureFile LOBs
internals for a better understanding

Martin Berger
DOAG 2018

Martin Berger

Oracle DBA since 2000

http://berxblog.blogspot.com
@martinberx
martin.a.berger@gmail.com

Why?


used bytes SYS_LOB0000021993C00003$$
10^9 bytes
160,0
150,0
140,0
130,0
120,0
110,0
100,0
90,0
80,0
12.02.2018 00:00:00
13.02.2018 00:00:00
14.02.2018 00:00:00
15.02.2018 00:00:00
16.02.2018 00:00:00
17.02.2018 00:00:00
18.02.2018 00:00:00
19.02.2018 00:00:00
20.02.2018 00:00:00
21.02.2018 00:00:00
22.02.2018 00:00:00
23.02.2018 00:00:00
24.02.2018 00:00:00
25.02.2018 00:00:00
26.02.2018 00:00:00
27.02.2018 00:00:00
28.02.2018 00:00:00
01.03.2018 00:00:00
02.03.2018 00:00:00
03.03.2018 00:00:00
04.03.2018 00:00:00
05.03.2018 00:00:00
06.03.2018 00:00:00
07.03.2018 00:00:00
08.03.2018 00:00:00
09.03.2018 00:00:00
10.03.2018 00:00:00
11.03.2018 00:00:00
12.03.2018 00:00:00
13.03.2018 00:00:00
14.03.2018 00:00:00
USED_BYTES

What is a SecureFile LOB

Glossary:

SECUREFILE

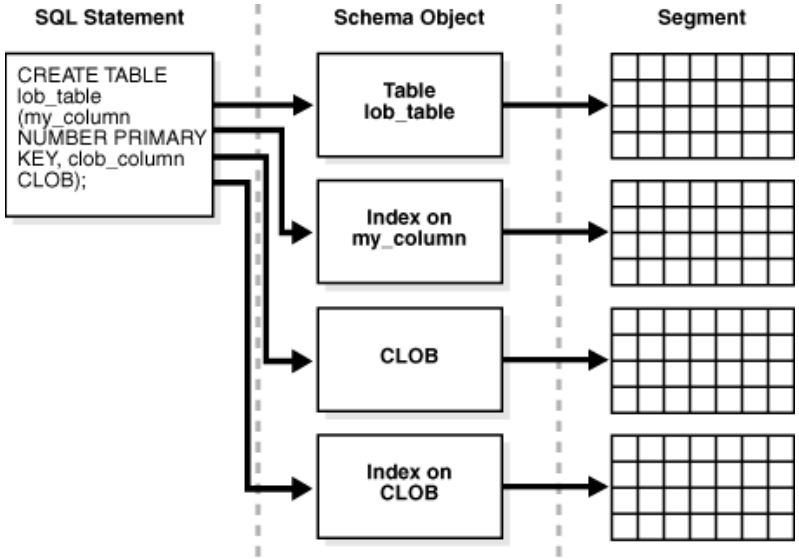
LOB storage parameter that allows deduplication, encryption, and compression. The opposite parameter, that does not allow these features, is `BASICFILE`.

https://docs.oracle.com/en/database/oracle/oracle-database/18/adlob/glossary.html#GUID-57B2D7FA-1461-43D3-936F-655009768759

Database Concepts - Logical Storage Structures:

Figure 12-20 shows that the data for `lob_table` is stored in one segment, while the implicitly created index is in a different segment. Also, the CLOB data is stored in its own segment, as is its associated CLOB index. Thus, the `CREATE TABLE` statement results in the creation of *four* different segments.

https://docs.oracle.com/en/database/oracle/oracle-database/18/cncpt/logical-storage-structures.html#GUID-BFC69020-DECA-4F2D-8BED-57A612A4E7C8



What is a SecureFile LOB

Database Concepts - Concepts for Database Developers:

Overview of LOBs

Large object (LOB) data types enable you to store and manipulate large blocks of unstructured data in binary or character format.

The **large object (LOB)** data type provides efficient, random, piece-wise access to the data.

Internal LOBs

The database stores LOBs differently from other data types. Creating a LOB column implicitly creates a LOB segment and a LOB index. The tablespace containing the LOB segment and LOB index, which are always stored together, may be different from the tablespace containing the table.

…

The LOB segment stores data in pieces called *chunks*. A chunk is a logically contiguous set of data blocks and is the smallest unit of allocation for a LOB. A row in the table stores a pointer called a *LOB locator*, which points to the LOB index. When the table is queried, the database uses the LOB index to quickly locate the LOB chunks.

The database manages read consistency for LOB segments differently from other data. Instead of using undo data to record changes, the database stores the before images in the segment itself. When a transaction changes a LOB, the database allocates a new chunk and leaves the old data in place. If the transaction rolls back, then the database rolls back the changes to the index, which points to the old chunk.

https://docs.oracle.com/en/database/oracle/oracle-database/18/cncpt/concepts-for-database-developers.html#GUID-3C50EAB8-FC39-4BB3-B680-4EACCE49E866

What is a SecureFile LOB

Database SecureFiles and Large Objects Developer's Guide

Audience

Oracle Database SecureFiles and Large Objects Developer's Guide is intended for programmers who develop new applications using LOBs and DBFS, and those who have previously implemented this technology and now want to take advantage of new features.

Efficient and secure storage of multimedia and unstructured data is increasingly important, and this guide is a key resource for this topic within the Oracle Application Developers documentation set.

https://docs.oracle.com/en/database/oracle/oracle-database/18/adlob/preface.html#GUID-1B1902CC-DDA6-499B-92FE-C88CCF69C686

More Informations - PL/SQL

DBMS_SPACE.SPACE_USAGE

The second form of the procedure returns information about SECUREFILE LOB space usage. It will return the amount of space in blocks being used by all the SECUREFILE LOBs in the LOB segment. The procedure displays the space actively used by the LOB column, freed space that has retention expired, and freed space that has retention unexpired. Note that this overload can be used only on SECUREFILE LOBs.

https://docs.oracle.com/en/database/oracle/oracle-database/18/arpls/DBMS_SPACE.html#GUID-1115B610-8956-426F-B615-9118225F911F

```
DBMS_SPACE.SPACE_USAGE(
   segment_owner        IN   VARCHAR2,
   segment_name         IN   VARCHAR2,
   segment_type         IN   VARCHAR2,
   segment_size_blocks  OUT  NUMBER,
   segment_size_bytes   OUT  NUMBER,
   used_blocks          OUT  NUMBER,
   used_bytes           OUT  NUMBER,
   expired_blocks       OUT  NUMBER,
   expired_bytes        OUT  NUMBER,
   unexpired_blocks     OUT  NUMBER,
   unexpired_bytes      OUT  NUMBER,
   partition_name       IN   VARCHAR2 DEFAULT NULL);
```

More Informations - PL/SQL

DBMS_SPACE.SPACE_USAGE

```
set serveroutput on
declare
v_segment_size_blocks number;    v_segment_size_bytes number;
v_used_blocks number;            v_used_bytes number;
v_expired_blocks number;         v_expired_bytes number;
v_unexpired_blocks number;       v_unexpired_bytes number;
begin
dbms_space.space_usage ('&owner', '&LOB_SEGMENT', 'LOB',
  v_segment_size_blocks, v_segment_size_bytes, v_used_blocks,
  v_used_bytes, v_expired_blocks, v_expired_bytes,
  v_unexpired_blocks, v_unexpired_bytes);
dbms_output.put_line('Segment size in blocks = '||v_segment_size_blocks);
dbms_output.put_line('Used Blocks            = '||v_used_blocks);
dbms_output.put_line('Expired Blocks         = '||v_expired_blocks);
dbms_output.put_line('Unxpired Blocks        = '||v_unexpired_blocks);
end;
/
```

```
Segment size in blocks = 191892224
Used Blocks            =  74424867
Expired Blocks         = 117195491
Unxpired Blocks        = 0
```

More Informations - PL/SQL

DBMS_LOBUTIL (undoc.)

dbmslobu.sql

```
Rem    NOTES
Rem The new package DBMS_LOBUTIL is a container for diagnostic and
Rem utility functions and procedures specific to 11globs.
Rem
Rem Since diagnostic operations are not part of the standard programmatic
Rem APIs in DBMS_LOB, they are provided in a separate namespace to avoid
Rem clutter.  The diagnostic API is also not quite as critical to document
Rem for end-users; its main use is for internal developer, QA, and DDR use
Rem (especially since it peeks into the internal structure of 11glob
Rem inodes and lobmaps).
```

prvtlobu.plb

kkxlnu_% c functions

http://psoug.org/reference/dbms_lobutil.html

More Informations - PL/SQL

DBMS_LOBUTIL (undoc.)

SQL> @**LOBMAP** "*select rowid, LOB_COL from TABLE ... "*
old 9: CURSOR CRS IS &1 ;
new 9: CURSOR CRS IS *select rowid, LOB_COL from TABLE ...* ;

```
ROWID   = AAAibXAAEAAAACjAAA
ROWNUM  = 1
LOBID   = 0000000100000CC53472
EXTENT# = 0
HOLE?   = n
Superchunk cont? = n
Overallocation  = n
rdba    = 16777396 - 0x  10000B4
File    = 4
Block   = 180
nblks   = 2
offset = 0
length = 11389
---
```

```
...
---
ROWID   = AAAibXAAEAAAACnAAA
ROWNUM  = 2
LOBID   = 0000000100000CC5F81E
EXTENT# = 0
HOLE?   = n
Superchunk cont? = n
Overallocation  = n
rdba    = 16777398 - 0x  10000B6
File    = 4
Block   = 182
nblks   = 2
offset = 0
length = 16120
---
ROWID   = AAAibXAAEAAAACnAAA
ROWNUM  = 2
LOBID   = 0000000100000CC5F81E
EXTENT# = 1
HOLE?   = n
Superchunk cont? = n
Overallocation  = n
rdba    = 16777413 - 0x  10000C5
File    = 4
Block   = 197
nblks   = 3
offset = 16120
length = 24180
---
...
```

More Informations - PL/SQL

block dumps ;-)

Block types

Description	Block Type
NGLOB: Extent Map	0x3c
NGLOB: Hash Bucket	0x3d
NGLOB: Committed Free Space	0x3e
NGLOB: Segment Header	0x3f
NGLOB: Lob Extent Header	0x40
NGLOB: Persistent Undo	0x41
NGLOB: Lob Extent Header	0x45
trans_data	0x06

Block types

NGLOB: Extent Header

Block dump from disk:
buffer tsn: 4 rdba: 0x0103e580 (4/255360)
scn: 0xa9c.1cf87e98 seq: 0x02 flg: 0x04 tail: 0x7e984502
frmt: 0x02 chkval: 0xfec6 type: 0x45=NGLOB: Lob Extent Header
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F6247D60800 to 0x00007F6247D62800
7F6247D60800 0000A245 0103E580 1CF87E98 04020A9C [E........~......]
7F6247D60810 0000FEC6 0103E58B 00000005 00000000 [................]
7F6247D60820 59A53020 59A53020 00000000 00000000 [0.Y 0.Y........]
7F6247D60830 00000000 0006EF63 00000000 00000001 [....c...........]
7F6247D60840 0103E58B 00000005 00000000 00000000 [................]
7F6247D60850 00000000 00000000 00000000 00000000 [................]
Repeat 505 times
7F6247D627F0 00000000 00000000 00000000 7E984502 [.............E.~]
Dump of Extent Header Block
---
sdba: 0x0103e58b **len**:5 flag:0x0 **synctime**:1503997984 **updtime**:1503997984
objd:454499 inc:0 total:1 opcode:0 xid: 0x0000.000.00000000
entry 0: sdba: 0x0103e58b len:5 fdba: 0x00000000
---

sdba - points to 1st trans data block
synctime & **updtime** - unix epoc timestamps
len - number of “*trans data*” blocks following

Block types

NGLOB: Segment Header

```
Block dump from disk:
buffer tsn: 4 rdba: 0x0103e581 (4/255361)
scn: 0xa9c.1cf87e98 seq: 0x04 flg: 0x04 tail: 0x7e983f04
frmt: 0x02 chkval: 0x12dd type: 0x3f=NGLOB: Segment Header
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F6247D60800 to 0x00007F6247D62800
...
NGLOB Segment Header
  Extent Control Header
  ----------------------------------------------------------------
  Extent Header:: spare1: 0      spare2: 0      #extents: 1      #blocks: 16
                  last map  0x00000000  #maps: 0      offset: 8168
      Highwater::  0x0103e590  ext#: 0      blk#: 16     ext size: 16
  #blocks in seg. hdr's freelists: 0
  #blocks below: 16
  mapblk  0x00000000  offset: 0
      Disk Lock:: Locked by xid:  0x01b3.015.00002ca0
     Map Header:: next  0x00000000  #extents: 1    obj#: 454499 flag: 0x40000000
  Extent Map
  ----------------------------------------------------------------
   0x0103e580  length: 16
```

Block types

NGLOB: Segment Header (cont.)

CFS hbbcnt:7 hbbmx:7 Mxrng:7 UFS hbbcnt:2 PUA cnt:0 Flag:2
Segment Stats

Retention: -1
Free Space: 5
PUA Batchsize: 50
UFS Array

DBA: 0x0103e582 Inst:1
DBA: 0x0103e583 Inst:1
Ufs off:152 sz:512
CFS Array

Range: 0 DBA: 0x0103e584
Range: 1 DBA: 0x0103e585
Range: 2 DBA: 0x0103e586
Range: 3 DBA: 0x0103e587
Range: 4 DBA: 0x0103e588
Range: 5 DBA: 0x0103e589
Range: 6 DBA: 0x0103e58a
Cfs off:664 sz:576
PUA Array
---
pua off:1240 sz:8

`CFS` `Committed Free Space (blocks)` -
`UFS` `Uncommitted Free Space (blocks)`
`PUA` `Persistent Undo ??`

In `Segment Stats`, `Free Space:` is `5` which matches the 5 `trans data` blocks available.
`UFS Array` contains 2 blocks, so it matches the count above.
`CFS Array` contains 7 blocks, which matches the count above also.
`PUA Array` is empty - there are no `NGLOB: Persistent Undo` blocks

Block types

NGLOB: Hash Bucket (UFS)

```
Block dump from disk:
buffer tsn: 4 rdba: 0x0103e582 (4/255362)
scn: 0xa9c.1cf87e98 seq: 0x02 flg: 0x04 tail: 0x7e983d02
frmt: 0x02 chkval: 0x86ba type: 0x3d=NGLOB: Hash Bucket
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F6247D60800 to 0x00007F6247D62800
...
  Dump of Hash Bucket Block
  ------------------------------------------------------------
Hash Bucket Header Dump
Inst:1 Flag:6 Total:0 FSG_COUNT:0 OBJD:454499 Inc:0
Self Descriptive
 fsg_count:0
Head:0x 0x00000000  Tail:0x 0x00000000
 Opcdode:0 Locker Xid: 0x0000.000.00000000
Fsbdba: 0x0    Fbrdba: 0x0
Head Cache Entries
-------------------
-------------------
Tail Cache Entries
-------------------
-------------------
Free Space Chunk Summary
Inst:0 Minc:0 Maxc:0 Count:0 Tot:0 MAXC:0
UFS List
---------------
```

Block types

NGLOB: Hash Bucket (CFS)

```
Block dump from disk:
buffer tsn: 4 rdba: 0x0103e584 (4/255364)
scn: 0xa9c.1cf87e98 seq: 0x02 flg: 0x04 tail: 0x7e983d02
frmt: 0x02 chkval: 0x86be type: 0x3d=NGLOB: Hash Bucket
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F6247D60800 to 0x00007F6247D62800
...
  Dump of Hash Bucket Block
  ------------------------------------------------------------
Hash Bucket Header Dump
Range: 2k to 32k
Inst:1 Flag:5 Total:0 FSG_COUNT:0 OBJD:454499 Inc:0
Self Descriptive
 fsg_count:0
Head:0x 0x00000000  Tail:0x 0x00000000
 Opcdode:0 Locker Xid: 0x0000.000.00000000
Fsbdba: 0x0    Fbrdba: 0x0
Head Cache Entries
-------------------
-------------------
Tail Cache Entries
-------------------
-------------------
Free Space Chunk Summary
Inst:1 Minc:0 Maxc:0 Count:0 Tot:0 MAXC:0
CFS Chunk List
--------------
```

Range: 2k to 32k
Range: 32k to 64k
Range: 64k to 128k
Range: 128k to 256k
Range: 256k to 512k
Range: 512k to 1m
Range: 1m to 64m

Block types

NGLOB: Committed Free Space

```
Block dump from disk:
buffer tsn: 4 rdba: 0x0103ea01 (4/256513)
scn: 0xa9c.1f0271f4 seq: 0x02 flg: 0x04 tail: 0x71f43e02
frmt: 0x02 chkval: 0x6d43 type: 0x3e=NGLOB: Committed Free Space
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007FDC79DC4800 to 0x00007FDC79DC6800
...
  Dump of NGLOB: Uncommitted Free Space Block
  ------------------------------------------------------------
FSB Header
--------------
objd:454499 inc:0 hdba:  0x0103e582 Rng:-1
 prev:  0x0103ea01  next:  0x0103ea01  edba:  0x0103ea00  head:  0x0103ea01
 Xid: 0x0000.000.00000000 Op:0
--------------
Free Space Chunk Summary
Inst:1 Minc:0 Maxc:0 Count:0 Tot:0 MAXC:0
UFS List
--------------
```

Block types

NGLOB: Persistent UNDO

```
Block dump from disk:
buffer tsn: 4 rdba: 0x0103ea06 (4/256518)
scn: 0xa9c.1f027209 seq: 0x01 flg: 0x04 tail: 0x72094001
frmt: 0x02 chkval: 0x641e type: 0x40=NGLOB: Persistent Undo
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007FDC79DC4800 to 0x00007FDC79DC6800
...
  Dump of Persistent Undo Block
  ------------------------------------------------------------
 Inst: 1 Objd:454499 Inc:3472328295419215872 SyncTime:4120848872352594377 Flag:0x1
 Total: 2 LoadTime:1504000457 Opcode:1 Xid: 0x01b3.01d.00002ca6
  ------------------------------------------------------------
Chunk: dba: 0x103e58b: len: 1: Xid:  0x01b3.003.00002c9b: Scn: 2716.520253961: Flag: IN USE: FBR: False
Chunk: dba: 0x103e58c: len: 4: Xid:  0x0000.000.00000000: Scn: 2708.1852084450: Flag: FREE: FBR: False
```

Block types

trans data

```
Block header dump:  0x0103e58b
 Object id on Block? Y
 seg/obj: 0x6ef63  csc: 0xa9c.1f027209  itc: 1  flg: E  typ: 5 - LOCAL LOBS
     fsl: 0  fnx: 0xffffffff ver: 0x01

 Itl           Xid                  Uba         Flag  Lck        Scn/Fsc
0x01   0x01b3.003.00002c9b  0x00000000.0000.00  -B--    0  fsc 0x0000.00000000
========
bdba    [0x0103e58b]
kdlich  [0x7f381badb84c 56]
  flg0  0x20 [ver=0 typ=data lock=n]
  flg1  0x00
  scn   0x0a9c.1f027209
  lid   00000001000099bf1091
  rid   0x00000000.0000
kdlidh  [0x7f381badb864 24]
  flg2  0x00 [ver=0 lid=short-rowid hash=n cmap=n pfill=n]
  flg3  0x00
  pskip 0
  sskip 0
  hash  00000000000000000000000000000000000000000
  hwm   1
  spr   0
  data  [0x7f381badb880 52 8060]
41 8c ed 85 f8 d5 98 1a 68 52 c2 34 4d 23 84 ee f4 09 5e 21 a7 94 aa 85 da ee
51 5b 49 a1 d0 d1 38 e1 62 9d a1 8b 29 31 c7 c5 d6 d8 48 c3 2f 66 09 15 3a 8c
...
```

8k block

128	byte	header
8060	byte	LOB Data
4	byte	trailer

Extent structure

1st Extent (16 blocks)

NGLOB: Lob Extent Header

NGLOB: Segment Header

NGLOB: Hash Bucket (UFS)

NGLOB: Hash Bucket (UFS)

NGLOB: Hash Bucket (CFS: 2k - 32k)

NGLOB: Hash Bucket (CFS: 64k - 128k)

NGLOB: Hash Bucket (CFS: 128k - 256k)

NGLOB: Hash Bucket (CFS: 256k - 512k)

NGLOB: Hash Bucket (CFS: 512k - 1m)

NGLOB: Hash Bucket (CFS: 1m - 64m)

trans data
(5x)
...

Extent structure

(other) Extent (512 blocks)

NGLOB: Lob Extent Header

NGLOB: Committed Free Space

NGLOB: Committed Free Space

NGLOB: Committed Free Space

NGLOB: Committed Free Space

NGLOB: Persistent Undo
(50 x <= PUA Batchsize: 50 from NGLOB: Segment Header)

trans data
(73x)
.

NGLOB: Lob Extent Header

trans data
(127x)

NGLOB: Lob Extent Header

trans data
(127x)

NGLOB: Lob Extent Header

trans data
(127x)

Parameters

Parameter	Value	Description
_dbfs_modify_implicit_fetch	TRUE	DBFS Link allows implicit fetch on modify - only on SecureFiles
_enable_securefile_flashback_opt	FALSE	Enable securefile flashback optimization
_kdli_STOP_bsz	0	undocumented parameter for internal use only
_kdli_STOP_dba	0	undocumented parameter for internal use only
_kdli_STOP_fsz	0	undocumented parameter for internal use only
_kdli_STOP_nio	0	undocumented parameter for internal use only
_kdli_STOP_tsn	0	undocumented parameter for internal use only
_kdli_allow_corrupt	TRUE	allow corrupt filesystem_logging data blocks during read/write
_kdli_buffer_inject	TRUE	use buffer injection for CACHE [NO]LOGGING lobs
_kdli_cache_inode	TRUE	cache inode state across calls
_kdli_cache_read_threshold	0	minimum lob size for cache->nocache read (0 disables heuristic)
_kdli_cache_size	8	maximum #entries in inode cache
_kdli_cache_verify	FALSE	verify cached inode via deserialization
_kdli_cache_write_threshold	0	minimum lob size for cache->nocache write (0 disables heuristic)

Parameters

_kdli_cacheable_length	0	minimum lob length for inode cacheability
_kdli_checkpoint_flush	FALSE	do not invalidate cache buffers after write
_kdli_dbc	none	override db_block_checking setting for securefiles
_kdli_delay_flushes	TRUE	delay flushing cache writes to direct-write lobs
_kdli_descn_adj	FALSE	coalesce extents with deallocation scn adjustment
_kdli_flush_cache_reads	FALSE	flush cache-reads data blocks after load
_kdli_flush_injections	TRUE	flush injected buffers of CACHE NOLOGGING lobs before commit
_kdli_force_cr	TRUE	force CR when reading data blocks of direct-write lobs
_kdli_force_cr_meta	TRUE	force CR when reading metadata blocks of direct-write lobs
_kdli_force_storage	none	force storage settings for all lobs
_kdli_full_readahead_threshold	0	maximum lob size for full readahead
_kdli_inject_assert	0	inject asserts into the inode
_kdli_inject_batch	0	buffer injection batch size [1, KCBNEWMAX]
_kdli_inject_crash	0	inject crashes into the inode
_kdli_inline_xfm	TRUE	allow inline transformed lobs

Parameters

_kdli_inode_preference	data	inline inode evolution preference (data, headless, lhb)
_kdli_inplace_overwrite	0	maximum inplace overwrite size (> chunksize)
_kdli_itree_entries	0	#entries in lhb/itree blocks (for testing only)
_kdli_memory_protect	FALSE	trace accesses to inode memory outside kdli API functions
_kdli_mts_so	TRUE	use state objects in shared server for asyncIO pipelines
_kdli_oneblk	FALSE	allocate chunks as single blocks
_kdli_preallocation_mode	length	preallocation mode for lob growth
_kdli_preallocation_pct	0	percentage preallocation [0 .. inf) for lob growth
_kdli_ralc_length	10485760	lob length threshold to trigger rounded allocations
_kdli_ralc_rounding	1048576	rounding granularity for rounded allocations
_kdli_rci_lobmap_entries	255	#entries in RCI lobmap before migration to lhb
_kdli_readahead_limit	0	shared/cached IO readahead limit
_kdli_readahead_strategy	contig	shared/cached IO readahead strategy
_kdli_recent_scn	FALSE	use recent (not dependent) scns for block format/allocation
_kdli_reshape	FALSE	reshape an inode to inline or headless on length truncation

Parameters

_kdli_safe_callbacks	TRUE	invoke inode read/write callbacks safely
_kdli_sio_async	TRUE	asynchronous shared IO
_kdli_sio_backoff	FALSE	use exponential backoff when attempting SIOP allocations
_kdli_sio_bps	0	maximum blocks per IO slot
_kdli_sio_dop	2	degree-of-parallelism in the SIO keep pool
_kdli_sio_fbwrite_pct	35	percentage of buffer used for direct writes in flashback-db
_kdli_sio_fgio	TRUE	reap asynchronous IO in the foreground
_kdli_sio_fileopen	none	shared IO fileopen mode: datasync vs nodatasync vs async
_kdli_sio_flush	FALSE	enable shared IO pool operations
_kdli_sio_free	TRUE	free IO buffers when not in active use
_kdli_sio_min_read	0	shared IO pool read threshold
_kdli_sio_min_write	0	shared IO pool write threshold
_kdli_sio_nbufs	8	maximum #IO buffers to allocate per session
_kdli_sio_niods	8	maximum #IO descriptors to allocate per session
_kdli_sio_on	TRUE	enable shared IO pool operations

Parameters

_kdli_sio_pga	FALSE	use PGA allocations for direct IO
_kdli_sio_pga_top	FALSE	PGA allocations come from toplevel PGA heap
_kdli_sio_strategy	extent	shared IO strategy: block vs. extent
_kdli_sio_write_pct	100	percentage of buffer used for direct writes
_kdli_small_cache_limit	32	size limit of small inode cache
_kdli_sort_dbas	FALSE	sort dbas during chunkification
_kdli_space_cache_limit	2048	maximum #blocks in per-segment space cache
_kdli_space_cache_segments	16	#segments in space cache
_kdli_squeeze	TRUE	compact lobmap extents with contiguous dbas
_kdli_timer_dmp	FALSE	dump inode timers on session termination
_kdli_timer_trc	FALSE	trace inode timers to uts/tracefile
_kdli_trace	0	inode trace level
_kdli_vll_direct	TRUE	use skip-navigation and direct-positioning in vll-domain
_kdlxp_dedup_hash_algo	SHA1	secure hash algorithm for deduplication - only on SecureFiles
_kdlxp_lobcmpadp	FALSE	enable adaptive compression - only on SecureFiles

Parameters

_kdlxp_lobcmplevel	2	Default securefile compression
_kdlxp_lobcmprciver	1	Default securefile compression map version
_kdlxp_lobcompress	FALSE	enable lob compression - only on SecureFiles
_kdlxp_lobdeduplicate	FALSE	enable lob deduplication - only on SecureFiles
_kdlxp_lobdedupvalidate	TRUE	enable deduplicate validate - only on SecureFiles
_kdlxp_lobencrypt	FALSE	enable lob encryption - only on SecureFiles
_kdlxp_mincmp	20	minimum comp ratio in pct - only on SecureFiles
_kdlxp_mincmplen	200	minimum loblen to compress - only on SecureFiles
_kdlxp_uncmp	FALSE	lob data uncompressed - only on SecureFiles
_kdlxp_xfmcache	TRUE	enable xfm cache - only on SecureFiles
_securefile_log_num_latches	0	Maximum number of open descriptors for securefile log
_securefile_log_shared_pool_size	0	Size of securefile log buffer pool from SGA
_securefile_timers	FALSE	collect kdlu timers and accumulate per layers
_securefiles_breakreten_retry	5	segment retry before dishonoring retention
_securefiles_bulkinsert	FALSE	securefiles segment insert only optization

Parameters

_securefiles_concurrency_estimate	12	securefiles concurrency estimate
_securefiles_fg_retry	100	segment retry before foreground waits
_securefiles_forceflush	FALSE	securefiles force flush before allocation
_securefiles_memory_percentofSGA	8	securefiles memory as percent of SGA
_securefiles_spcutl	FALSE	securefiles segment utl optimization
_sf_default_enabled	TRUE	enable 12g securefile default

_kdli_trace

Seems to be a bitmap of flags. (mentioned in some BUGs in MOS) -e.g. Bug 17283676 : MOVE LOB EXTREMELY SLOW

_securefiles_concurrency_estimate

[1..50] - e.g. Securefiles DMLs cause high 'buffer busy waits' & 'enq: TX - contention' wait events leading to whole database performance degradation (Doc ID 1532311.1)

SMCO

SMCO (Space Management Coordinator) For Autoextend On Datafiles And How To Disable/Enable (Doc ID 743773.1)

Tablespace-level space (Extent) pre-allocation.
…
- ecurefile lob segment pre-extension.
- Securefile lob segment in-memory dispenser space pre-allocation.
- Securefile lob segment space reclamation (moving free chunks from uncommitted free space area to committed free space area).

…

_ENABLE_SPACE_PREALLOCATION = 3 (default)

Bitmap:
* 0 to turn off the tbs pre-extension feature.
* 1 To enable tablespace extension.
* 2 To enable segment growth.
* 4 To enable chunk allocation.

Bugs

26007010	DATABASE GREW EVEN THOUGH RESOURCES DELETED	80 - Development to QA/Fix Delivered Internal	14-Nov-2018 01:49
26513067	ASSIGNMENT OF INSTANCE AFFINITY FLAWED IN KTSLA_HBB_UPDATE_CACHE	80 - Development to QA/Fix Delivered Internal	16-Nov-2018 15:18
27801337	EXPIRED BYTES NOT REUSED ON LOB INSERT	80 - Development to QA/Fix Delivered Internal	12-Nov-2018 13:35
26696955	LOB SEGMENT SIZE IS SIGNIFICANTLY INCREASING DESPITE OF SMALL ACTUAL DATA IN IT	36 - Duplicate Bug. To Filer	11-Oct-2018 10:22
27636345	ORA-00600 [KTEUPROPAGATETIME:CLSVIOL_KCBGCUR_9]	36 - Duplicate Bug. To Filer	15-Aug-2018 10:30
27391136	ORA-600 [KCBZIB_LOBDS_1] WHILE DOING INSERT THROUGH DBLINK IN SECUREFILE LOB	30 - More Information Requested.	09-Nov-2018 14:35

Links

Oracle SecureFile System http://www.vldb.org/pvldb/1/1454170.pdf
Oracle SecureFiles: Prepared for the Digital Deluge http://www.oracle.com/technetwork/database/securefile-blobs-like-a-high-perfor-131490.pdf
Using SecureFiles on a Sun Storage 6780 Array
http://www.oracle.com/technetwork/articles/systems-hardware-architecture/using-securefiles-ss6780-array-163564.pdf
Oracle Database 11g: SecureFiles http://www.oracle.com/technetwork/database/options/compression/overview/securefiles-131281.pdf
NoCOUG-2014_11_LOB_Internals http://www.nocoug.org/download/2014-11/NoCOUG-2014_11_LOB_Internals.pdf
SecureFiles Performance http://www.oracle.com/technetwork/database/database-technologies/performance/securefilesperformancepaper-130360.pdf
LOB versus SecureFiles http://www.database-consult.de/docs/LOBversusSF1.pdf
Implementing SecureFile lobs with WebCenter Content, Lessons learned while pushing the limits of SecureFile lobs
https://oraclemva.files.wordpress.com/2013/04/pushing-the-limits-with-secure-files.pdf
Oracle LOB Internals and Performance Tuning https://www.slideshare.net/tanelp/oracle-lob-internals-and-performance-tuning
LOB Internals www.juliandyke.com/Presentations/LOBInternals.ppt
Oracle DBMS_LOBUTIL http://psoug.org/reference/dbms_lobutil.html
https://jonathanlewis.wordpress.com/2015/01/21/lob-space/

securefile patent http://www.freepatentsonline.com/5999943.html

LOBMAP.sql

```
set serveroutput on
declare
       RID     ROWID;
       ALOC    SYS.ANYDATA;set serveroutput on
declare
       RID     ROWID;
       ALOC    SYS.ANYDATA;
       BLOC    BLOB;
       CLOC    CLOB;
       INODE   DBMS_LOBUTIL_INODE_T;
       LMAP    DBMS_LOBUTIL_LOBMAP_T;
       ROWN    NUMBER;
                                              CURSOR  CRS IS  &1 ;

   BEGIN
      ROWN := 0;
      OPEN CRS;

      LOOP
         FETCH CRS INTO RID, BLOC;
         ROWN := ROWN+1 ;
         EXIT WHEN CRS%NOTFOUND;

         IF BLOC IS NOT NULL THEN
            INODE := DBMS_LOBUTIL.GETINODE(BLOC);
         ELSE
            CONTINUE;
         END IF;

         IF ((INODE.LENGTH = 0) OR (INODE.EXTENTS = 0)) THEN
            CONTINUE;
         END IF;

         FOR N IN 0 .. INODE.EXTENTS-1 LOOP
               LMAP := DBMS_LOBUTIL.GETLOBMAP(BLOC, N);

DBMS_OUTPUT.PUT_LINE(' ROWID   = ' || RID);
DBMS_OUTPUT.PUT_LINE(' ROWNUM  = ' || ROWN );
DBMS_OUTPUT.PUT_LINE(' LOBID   = ' || LMAP.LOBID );
DBMS_OUTPUT.PUT_LINE(' EXTENT# = ' || N );
DBMS_OUTPUT.PUT_LINE(' HOLE?   = ' ||
                  CASE
                     WHEN BITAND(LMAP.EFLAG,1) = 0
                        THEN 'n' ELSE 'y'
                  END);
DBMS_OUTPUT.PUT_LINE(' Superchunk cont? = ' ||
                  CASE
                     WHEN BITAND(LMAP.EFLAG,2) = 0
                        THEN 'n' ELSE 'y'
                  END);
DBMS_OUTPUT.PUT_LINE(' Overallocation  = ' ||
                  CASE
                     WHEN BITAND(LMAP.EFLAG,4) = 0
                        THEN 'n' ELSE 'y'
                  END);
DBMS_OUTPUT.PUT_LINE(' rdba    = ' || LMAP.RDBA || ' - 0x' || to_char(LMAP.RDBA, 'XXXXXXXX'));
DBMS_OUTPUT.PUT_LINE(' File    = ' || DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(LMAP.RDBA) );
DBMS_OUTPUT.PUT_LINE(' Block   = ' || DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(LMAP.RDBA) );
DBMS_OUTPUT.PUT_LINE(' nblks   = ' ||  LMAP.NBLKS );
DBMS_OUTPUT.PUT_LINE(' offset = ' ||
                  CASE
                     WHEN BITAND(LMAP.EFLAG,2) = 0
                        THEN LMAP.OFFSET
                        ELSE NULL
                  END);
DBMS_OUTPUT.PUT_LINE(' length = ' ||
                  CASE
                     WHEN BITAND(LMAP.EFLAG,2) = 0
                        THEN LMAP.LENGTH
                        ELSE NULL
                  END);
DBMS_OUTPUT.PUT_LINE(' --- ');

         END LOOP;
      END LOOP;

      CLOSE CRS;
   END;
/
```

Martin Berger

Oracle DBA since 2000

http://berxblog.blogspot.com
@martinberx
martin.a.berger@gmail.com