Oracle Database Corruption

Database Block Checking- The Unknown Truth

High performance. Delivered.

accenture

Strategy | Consulting | Digital | Technology | Operations

Who We Are

- Gabriel Alonso
  - gaalonso@gmail.com
  - @gaalons0
  - https://gaalonso.wordpress.com/
- David Loinaz
  - david.loinaz.dba@gmail.com
  - @DavidLoinaz
  - https://davidloinaz.wordpress.com/

Corruption Detection/Prevention vs Performance

- In many cases, initialization parameter default settings should be understood as recommendations

- From the documentation you may find that you are suggested to set db_block_checking to “Full” while you may also find the impact in performance from doing so.

So, an obvious question arises ...

If I have my disks mirrored and I have a good database backup strategy and I've never faced a corruption, why do I need to activate any of these parameters which will impact my performance?

Block Checking – Parameter Settings

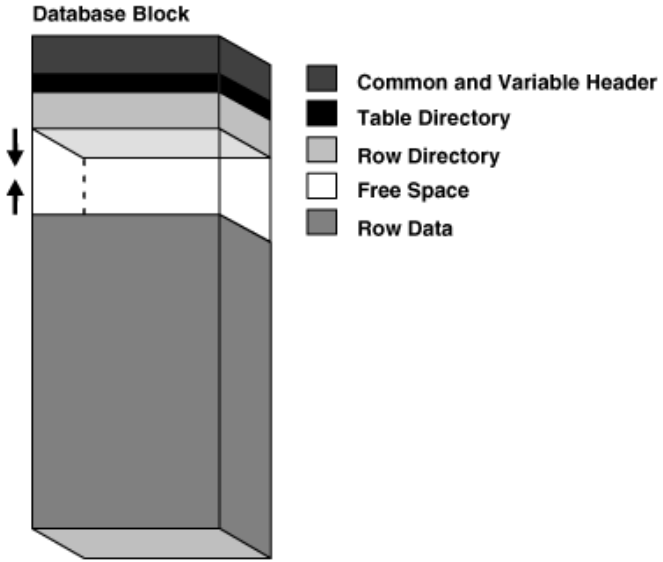
Parameter	Default	Recommended	Overhead
DB_BLOCK_CHECKING	NONE	FULL	1%-10%
DB_BLOCK_CHECKSUM	TYPICAL	TYPICAL	1%-2%
DB_LOST_WRITE_PROTECT	NONE	FULL	Negligible
DB_ULTRA_SAFE	OFF	DATA_AND_INDEX	5%-15%
		?	???

Oracle Database Block

- Oracle structures files in blocks
- Different block types and structures for each database block:
  - Undo segment header
  - KTB-managed data blocks (w/ itl)
  - Control file blocks
  - First level bitmap block
  - LOB block
  - … etc …

Oracle Database Block - layers

- KTB-managed data block (w/ itl)
  - Cache layer
  - Transaction layer
  - Data layer



Oracle Database Block – Hex Dump

```
buffer tsn: 4 rdba: 0x01c00083 (7/131)
scn: 0x1a6ca3 seq: 0x01 flg: 0x06 tail: 0x6ca30601
frmt: 0x02 chkval: 0x76c8 type: 0x06=trans data
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F516F742000 to 0x00007F516F744000
7F516F742000 0000A206 01C00083 001A6CA3 06010000  [.........l......]
7F516F742010 000076C8 00000001 00011F4F 001A6C99  [.v......O....l..]
7F516F742020 00008000 00320002 01C00080 00090009  [......2.........]
7F516F742030 000002D0 01000103 000700B3 00002005  [.............. ..]
7F516F742040 001A6CA3 00000000 00000000 00000000  [.l..............]
7F516F742050 00000000 00000000 00000000 00000000  [................]
7F516F742060 00000000 00050100 001CFFFF 1F461F62  [............b.F.]
7F516F742070 00001F46 1F8E0005 1F781F84 1F621F6D  [F.........x.m.b.]
7F516F742080 00000000 00000000 00000000 00000000  [................]
        Repeat 499 times
7F516F743FC0 00000000 012C0000 06C10202 76696604  [......,......fiv]
7F516F743FD0 02012C65 0405C102 72756F66 0202012C  [e,......four,...]
7F516F743FE0 740504C1 65657268 0202012C 740303C1  [...three,......t]
7F516F743FF0 012C6F77 02C10202 656E6F03 6CA30601  [wo,......one...l]
```

Oracle Database Block – Block Map (BBED)

KTB Data Block (Table/Cluster)

```
struct kcbh           , 20 bytes   @0
struct ktbbh          , 72 bytes   @20
struct kdbh           , 14 bytes   @100
struct kdbt[1]        , 4 bytes    @114
sb2    kdbr[5]                     @118
ub1    freespace[8006]            @128
ub1    rowdata[54]                @8134
ub4    tailchk                    @8188
```

Oracle Database Block – Hex Dump (kc, kt, kd)

```
buffer tsn: 4 rdba: 0x01c00083 (7/131)
scn: 0x1a6ca3 seq: 0x01 flg: 0x06 tail: 0x6ca30601
frmt: 0x02 chkval: 0x76c8 type: 0x06=trans data
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F516F742000 to 0x00007F516F744000
7F516F742000 0000A206 01C00083 001A6CA3 06010000  [.........l......]
7F516F742010 000076C8 00000001 00011F4F 001A6C99  [.v......O....l..]
7F516F742020 00008000 00320002 01C00080 00090009  [......2.........]
7F516F742030 000002D0 01000103 000700B3 00002005  [.............. ..]
7F516F742040 001A6CA3 00000000 00000000 00000000  [.l..............]
7F516F742050 00000000 00000000 00000000 00000000  [................]
7F516F742060 00000000 00050100 001CFFFF 1F461F62  [............b.F.]
7F516F742070 00001F46 1F8E0005 1F781F84 1F621F6D  [F.........x.m.b.]
7F516F742080 00000000 00000000 00000000 00000000  [................]
        Repeat 499 times
7F516F743FC0 00000000 012C0000 06C10202 76696604  [......,......fiv]
7F516F743FD0 02012C65 0405C102 72756F66 0202012C  [e,......four,...]
7F516F743FE0 740504C1 65657268 0202012C 740303C1  [...three,......t]
7F516F743FF0 012C6F77 02C10202 656E6F03 6CA30601  [wo,......one...l]
```

OK, but...What is a Block Corruption?

- Validate contents of block (based on database settings)
- Two different kind of corruptions:
  - Physical block corruption (media corruption)
    - fractured, misplaced, overwritten
    - checksum error, zeroed out
  - Logical block corruption (software corruption)
    - bad obj#, bad # itl's, row locked by non-existent txn
    - free list not ordered, row ends in the middle of another, total space count bad

Dealing with Corrupt Blocks (Physical)

- Not possible to detect – introduced externally
- Faulty HW
- Do not Traverse to Data Guard Environments
- Simple to address
  - Restore/recover
  - RMAN blockrecover

Dealing with Corrupt Blocks (Logical)

- Will only show up when you try to access/modify a block
  - May be long time after it was introduced
- Traverse to Data Guard environments
- Are hard to address once introduced (understatement)
  - Restore/recover will introduce the same problem.
  - Data is compromized thereafter
  - Possibilities
    - Full DB PITR – loss of data
    - Flashback – 1) not guaranteed and 2) again: loss of data
- But: It's possible to both detect and prevent them (always Oracle Bug)

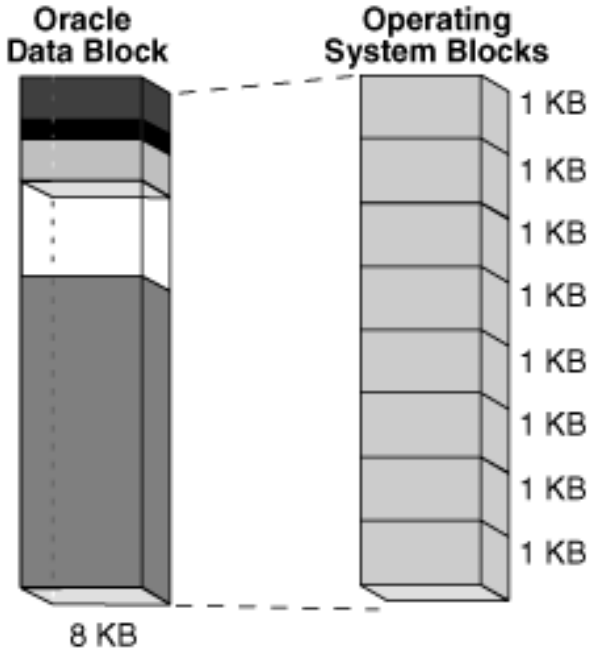
How to Detect and/or Prevent Block Corruptions?

- 3 different areas within Oracle control block contents.
  - Block Checksumming : for physical corruptions
  - Block Checking : for logical corruptions
  - Lost Write Detection : for lost write prevention
- All of them may be combined to enforce highest security and protection against block corruptions at Database level.

Oracle Database Block (O.S.) – Map

- Oracle Block Size != OS Block Size

  Which is why Checksum and Tail check is needed on each block



Oracle Database Block (O.S.) – Checksum and Tailchk

```
buffer tsn: 4 rdba: 0x01c00083 (7/131)
scn: 0x1a6ca3 seq: 0x01 flg: 0x06 tail: 0x6ca30601
frmt: 0x02 chkval: 0x76c8 type: 0x06=trans data
Hex dump of block: st=0, typ_found=1
Dump of memory from 0x00007F516F742000 to 0x00007F516F744000
7F516F742000 0000A206 01C00083 001A6CA3 06010000  [.........l......]
7F516F742010 000076C8 00000001 00011F4F 001A6C99  [.v......O....l..]
7F516F742020 00008000 00320002 01C00080 00090009  [......2.........]
7F516F742030 000002D0 01000103 000700B3 00002005  [.............. ..]
7F516F742040 001A6CA3 00000000 00000000 00000000  [.l..............]
7F516F742050 00000000 00000000 00000000 00000000  [................]
7F516F742060 00000000 00050100 001CFFFF 1F461F62  [............b.F.]
7F516F742070 00001F46 1F8E0005 1F781F84 1F621F6D  [F.........x.m.b.]
7F516F742080 00000000 00000000 00000000 00000000  [................]
        Repeat 499 times
7F516F743FC0 00000000 012C0000 06C10202 76696604  [......,......fiv]
7F516F743FD0 02012C65 0405C102 72756F66 0202012C  [e,......four,...]
7F516F743FE0 740504C1 65657268 0202012C 740303C1  [...three,......t]
7F516F743FF0 012C6F77 02C10202 656E6F03 6CA30601  [wo,......one...l]
```

Database Checksumming

- Enabled by default (TYPICAL) –
  - SYSTEM tablespace is always checksummed, even when disabled
- Helps detecting physical (media) corruptions
- Computes database block checksum at write time
- Re-calculates checksum and checks against block value upon read to see if any change occurred

Database Checksumming – parameter settings

DB_BLOCK_CHECKSUM = OFF|TYPICAL|FULL

Off	: Only SYSTEM tablespace is checksummed
Typical	: Processes all tablespaces and writes checksum values for redo log blocks as well
Full	: Oracle also verifies the checksum before a change application from update/delete statements and recomputes it after the change is applied

Database Checking

- Block Checking is ***Not*** enabled by Default (*)
- Detects logical corruptions after each modification of a block

(*) It always ***Is*** for SYSTEM tablespace

Database Checking – parameter settings

DB_BLOCK_CHECKING = OFF | LOW | MEDIUM | FULL

Off	: Only SYSTEM tablespace is checksummed
Low	: Basic block header checks are performed after block contents change in memory (for ex: insert/update/delete or after instance block transfers in RAC)
Medium	: Low + full semantic checks are performed for all objects except indexes
Full	: Low + Medium + full semantic checks are performed for all objects

*** Demo 1: block checksum vs block checking ***

Database Checking – trap

Logical corruption are trapped by Oracle at block change time:

- Oracle calls redo application call back to perform the change
- Oracle calls check function in order to validate contents after change

Ora-00600 is reported in the alert log file if block fails validation

Block never makes it to disk in an invalid/corrupt way

- No write allowed on a block which does not pass validation

Block is then re-read from disk and changes are applied again

- Transparent to client – performed by Oracle automatically

Database Checking – report

ORA-00600 for block checking started to be reported in release 10g

1st argument changed in release 11g+

10gR1: ORA-00600 [kddummy_blkchk][f][b][c]

11gR1: ORA-00600 [kdBlkCheckError][f][b][c]

f: file_id

b: block#

c: code

Database Lost Write Protection

- Not enabled by default
- Needs a standby database to actually detect the lost write.
  - Lost writes are not detected in primary database
- Performance impact is negligible

Database Lost Write Protection – Implementation

Primary DB writes SCN of each block at read time into redo stream.

Standby DB compares block SCN on the standby to block SCN in the redo stream.

If the block SCN on the primary database is lower than on the standby database, then it detects a lost write on the primary database and throws an external error (ORA-752).

Database Lost Write Protection – parameter settings

DB_LOST_WRITE_PROTECTION = NONE|TYPICAL|FULL

None	: No lost write functionality enabled
Typical	: the instance logs buffer cache reads for read/write tablespaces in the redo log
Full	: the instance logs buffer cache reads for read/only tablespaces as well

*** Demo 2: data block checking ***

Database Ultra Safe

- Appeared in 11gR1
- Implements increased level of protection.
- Controls settings of the following parameters (amongst others):
  - Db_block_checksum (DBS)
  - Db_block_checking (DBI)
  - Db_lost_write_protect (DBL)
- Parameter is Static – database needs to be bounced to change its value

Database Block Checking Parameter Settings

DB_ULTRA_SAFE = OFF|DATA_ONLY|DATA_AND_INDEX

db_ultra_safe	DBS	DBI	DBL
OFF	Untouched	Untouched	Untouched
DATA_ONLY	FULL	MEDIUM	TYPICAL
DATA_AND_INDEX	FULL	FULL	TYPICAL

Database Ultra Safe – Static Parameter

- Why is *“db_ultra_safe”* static ?
- It is static ‘cause additionally to block checking parameters it controls other parameters which are static – amongst which are:

1. _db_cache_advice_sample_factor
2. _db_lost_write_checking
3. _validate_readmem_redo

- None of these are dynamic.

Current Database Status

- RMAN may be used to check both backups and Database files using the *“check logical”* clause of the *“validate”* command

```
RMAN> validate check logical …
```

- Information stored in database views
  - v$database_block_corruption
  - v$nonlogged_block
  - v$backup_corruption
  - v$copy_corruption

Summary

Block checksumming helps detecting physical corruptions

- Physical corruptions are ”easy” to address

Block checking prevents logical corruptions from being introduced

- Logical corruptions are ***Real Show Stoppers***
- Data is unreliable
- Invalid data may spread w/out control

Lost writes are not detected by either settings

- DataGuard needed to detect lost writes
- Cost in performance terms is negligible

Summary

Default TYPICAL block checksums only writes it's value at write time and checks later at read time, It does not care about block contents.

Block checking validates block contents after each change

- ***Prevents*** writing of blocks witch do not pass validation.
- Impact depends on multiple factors – individual analysis needed

All of db_block_checksum, db_block_checking & db_lost_write_protect are dynamic parameters and may be set/unset at will via "alter system".

Now, the question again….

If I have my disks mirrored and I have a good database backup strategy and I've never faced a corruption, why do I need to activate any of these parameters which will impact my performance?

Hopefully we now know the answer.

Questions?

Q/A