ABOUT MULTIBLOCK READS
Who am I?

• Frits Hoogland
  – Working with Oracle products since 1996

• Blog: http://fritshoogland.wordpress.com
• Twitter: @fritshoogland
• Email: fhoogland@vxcompany.com
• Oracle ACE Director
• OakTable Member
Agenda

• Full scan implementation
  – Version 10 and earlier versus version 11 and later

• Direct path read slots

• ‘autotune’ / adaptive direct path reads
What is this presentation about?

• Multiblock reads can behave different after 10.2
• This could lead to different behavior of applications using the database.

• I assume the audience to have basic understanding about:
  – Oracle execution plans
  – Oracle SQL/10046 extended traces
  – General execution behavior of the RDBMS engine
  – C language in general
Row source operations

• Multiblock reads are an optimised method to read database blocks from disk for a database process.
  
  – Mainly used for the:
    › ‘TABLE ACCESS FULL’
    › ‘FAST FULL INDEX SCAN’
    › ‘BITMAP FULL SCAN’
  – rowsource operations.
Row source operations

• For much of other segment access rowsource actions, like:
  – ‘INDEX UNIQUE SCAN’
  – ‘INDEX RANGE SCAN’
  – ‘INDEX FULL SCAN’
  – ‘TABLE ACCESS BY INDEX ROWID’
• single block reads are mostly used.

• The order in which individual blocks are read is important.
db file multiblock read count

• Multiblock reads are done up to DB_FILE_MULTIBLOCK_READ_COUNT blocks.
  – If MBRC is unset, default is ‘maximum IO size that can be efficiently performed’.
  – Most operating systems allow a single IO operation up to 1 MB.
  – “Autotuned” (set to 0) seems to calculate its value by using the parameters ‘sessions’ and ‘db_cache_size’.
  – I prefer to set it manually.
My test environment

- Mac OSX Mountain Lion, VM Ware fusion
  - VM: OL6u3 x64
    - Database version 10.2.0.1 and 11.2.0.3
    - ASM GI 11.2.0.3
  - Sample tables:
    - T1 - 21504 blocks - 176M - 1’000’000 rows
      - PK index - 2304 blocks / 19M
    - T2 - 21504 blocks - 176M - 1’000’000 rows
First test

• 10.2.0.1 instance:
  – sga_target = 600M
  – Effective buffercache size = 450M
  – Freshly started
First test

```
TS@v10201 > select /*+ index(t t1_pk_ix) */ count(id), sum(scattered) from t1 t;

<table>
<thead>
<tr>
<th>COUNT(ID)</th>
<th>SUM(SCATTERED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>9999500000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>5</td>
<td>23234</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>T1</td>
<td>1000K</td>
<td>4884K</td>
<td>23234 (1)</td>
</tr>
<tr>
<td>3</td>
<td>INDEX FULL SCAN</td>
<td>T1_PK_IX</td>
<td>1000K</td>
<td></td>
<td>2253 (2)</td>
</tr>
</tbody>
</table>
```
First test

• How would you expect Oracle 10.2.0.1 to execute this?

  – In other words:
  – What would be the result of a SQL trace with waits? *

* If all blocks need to be read from disk (i.e. not cached)
First test

• My guess would be:
  – Index root bock (1 block)
  – None, one or more branch blocks (1 block)
  – Index leaf block, fetch values (1 block)
  – Table block via index rowid, fetch value(s) (1/1+ block)
  – Index values, block value(s), etc.
First test

• That should look like something like this:

```
WAIT #8: nam='db file sequential read'  ela= 326  file#=5  block#=43028  blocks=1
WAIT #8: nam='db file sequential read'  ela= 197  file#=5  block#=43719  blocks=1
WAIT #8: nam='db file sequential read'  ela= 227  file#=5  block#=43029  blocks=1
WAIT #8: nam='db file sequential read'  ela= 125  file#=5  block#=20   blocks=1
WAIT #8: nam='db file sequential read'  ela= 109  file#=5  block#=21   blocks=1
WAIT #8: nam='db file sequential read'  ela= 242  file#=5  block#=22   blocks=1
WAIT #8: nam='db file sequential read'  ela= 98   file#=5  block#=23   blocks=1
WAIT #8: nam='db file sequential read'  ela= 76   file#=5  block#=24   blocks=1
WAIT #8: nam='db file sequential read'  ela= 77   file#=5  block#=25   blocks=1
WAIT #8: nam='db file sequential read'  ela= 77   file#=5  block#=26   blocks=1
WAIT #8: nam='db file sequential read'  ela= 105  file#=5  block#=27   blocks=1
WAIT #8: nam='db file sequential read'  ela= 82   file#=5  block#=28   blocks=1
WAIT #8: nam='db file sequential read'  ela= 71   file#=5  block#=29   blocks=1
WAIT #8: nam='db file sequential read'  ela= 93   file#=5  block#=43030 blocks=1
...```
First test

• Instead, I get:

WAIT #4: nam='db file scattered read' ela= 361 file#=5 block#=43025 blocks=8
WAIT #4: nam='db file scattered read' ela= 220 file#=5 block#=43713 blocks=8
WAIT #4: nam='db file scattered read' ela= 205 file#=5 block#=17 blocks=8
WAIT #4: nam='db file scattered read' ela= 219 file#=5 block#=25 blocks=8
WAIT #4: nam='db file scattered read' ela= 192 file#=5 block#=33 blocks=8
WAIT #4: nam='db file scattered read' ela= 141 file#=5 block#=41 blocks=8
WAIT #4: nam='db file scattered read' ela= 123 file#=5 block#=49 blocks=8
WAIT #4: nam='db file scattered read' ela= 190 file#=5 block#=57 blocks=8
WAIT #4: nam='db file scattered read' ela= 231 file#=5 block#=43033 blocks=8
WAIT #4: nam='db file scattered read' ela= 113 file#=5 block#=65 blocks=8
...

Friday, February 8, 13
First test

• Sets of 8 blocks are read for rowsources which really need a single block.
• Reason:
  – This is an empty cache.
  – Oracle reads multiple blocks to get the cache filled.
  – ‘cache warming’
    ‣ Statistic (‘physical reads cache prefetch’)
• Needed to tune the BC down to 50M and pre-warm it with another table to get single block reads again (!!)
db_file_multiblock_read_count

- MBRC is the **maximum** amount of blocks read in one IO.

- Buffered MBRC cannot cross extent borders.

- Concepts guide on full table scans: (11.2 version)
  
  - A scan of table data in which the database sequentially reads all rows from a table and filters out those that do not meet the selection criteria. All data blocks under the high water mark are scanned.
Full scan - Oracle 10.2

• Let’s look at an Oracle 10.2.0.1 database

  – SGA_TARGET 600M

  – Table TS.T2 size 21504 blks / 176M
TS@v10201 > set autot on exp stat
TS@v10201 > select count(*) from t2;

    COUNT(*)
-------------
       1000000

Execution Plan
---------------------
Plan hash value: 3724264953

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>3674 (1)</td>
<td>00:00:45</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td>3674 (1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
<td>1007K</td>
<td>3674 (1)</td>
<td>00:00:45</td>
</tr>
</tbody>
</table>
Full scan - Oracle 10.2

Statistics

- 212 recursive calls
- 0 db block gets
- 20976 consistent gets
- 20942 physical reads
- 0 redo size
- 515 bytes sent via SQL*Net to client
- 469 bytes received via SQL*Net from client
- 2 SQL*Net roundtrips to/from client
- 4 sorts (memory)
- 0 sorts (disk)
- 1 rows processed
SYS@v10201 AS SYSDBA>
select object_id, object_name, owner from dba_objects where object_name = 'T2';

<table>
<thead>
<tr>
<th>OBJECT_ID</th>
<th>OBJECT_NAME</th>
<th>OWNER</th>
</tr>
</thead>
<tbody>
<tr>
<td>10237</td>
<td>T2</td>
<td>TS</td>
</tr>
</tbody>
</table>

SYS@v10201 AS SYSDBA> select * from x$kcboqh where obj# = 10237;

<table>
<thead>
<tr>
<th>ADDR</th>
<th>INDX</th>
<th>INST_ID</th>
<th>TS#</th>
<th>OBJ#</th>
<th>NUM_BUF</th>
<th>HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFFFFD7FFD5C6FA8</td>
<td>335</td>
<td>1</td>
<td>5</td>
<td>10237</td>
<td>20942</td>
<td>00000000038FBCF840</td>
</tr>
</tbody>
</table>
Full scan - Oracle 10.2

TS@v10201 > select count(*) from t2;

Statistics

0  recursive calls
0  db block gets
20953  consistent gets
0  physical reads
0  redo size
515  bytes sent via SQL*Net to client
469  bytes received via SQL*Net from client
2  SQL*Net roundtrips to/from client
0  sorts (memory)
0  sorts (disk)
1  rows processed
Full scan - Oracle 11.2

• Now look at an Oracle 11.2.0.3 database
  – SGA_TARGET 600M
  – Table TS.T2 size 21504 blks / 176M
TS@v11203 > select count(*) from t2;

    COUNT(*)
    -------
       1000000

Execution Plan

Plan hash value: 3724264953

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>3672 (1)</td>
<td>00:00:45</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS FULL</td>
<td>T2</td>
<td>1000K</td>
<td>3672 (1)</td>
<td>00:00:45</td>
</tr>
</tbody>
</table>
Statistics

- 217 recursive calls
- 0 db block gets
- 20970 consistent gets
- 20942 physical reads
- 0 redo size
- 526 bytes sent via SQL*Net to client
- 523 bytes received via SQL*Net from client
- 2 SQL*Net roundtrips to/from client
- 4 sorts (memory)
- 0 sorts (disk)
- 1 rows processed
SYS@v11203 AS SYSDBA>
select object_id, object_name, owner from dba_objects where object_name = 'T2';

<table>
<thead>
<tr>
<th>OBJECT_ID</th>
<th>OBJECT_NAME</th>
<th>OWNER</th>
</tr>
</thead>
<tbody>
<tr>
<td>66614</td>
<td>T2</td>
<td>TS</td>
</tr>
</tbody>
</table>

SYS@v11203 AS SYSDBA> select * from x$kcboqh where obj# = 66614;

<table>
<thead>
<tr>
<th>ADDR</th>
<th>INDX</th>
<th>INST_ID</th>
<th>TS#</th>
<th>OBJ#</th>
<th>NUM_BUF</th>
<th>HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFFFFD7FFC541B18</td>
<td>43</td>
<td>1</td>
<td>5</td>
<td>66614</td>
<td>1</td>
<td>0000000039043E470</td>
</tr>
</tbody>
</table>
Full scan - Oracle 11.2

TS@v11203 > select count(*) from t2;

Statistics
----------------------------------------------------------
   0  recursive calls
   0  db block gets
20945  consistent gets
20941  physical reads
   0  redo size
   526  bytes sent via SQL*Net to client
   523  bytes received via SQL*Net from client
    2  SQL*Net roundtrips to/from client
   0  sorts (memory)
   0  sorts (disk)
   1  rows processed
Full scan 10.2 vs. 11.2

- Why does version 10 caches all the blocks read,
- And version 11 only 1 of them??

- Let’s do an extended SQL trace
  – AKA 10046 level 8 trace.
Full scan 10.2 vs. 11.2

Relevant part of 10046/8 trace file of version 10.2.0.1:

| WAIT #1: nam='db file sequential read' ela= 32941 file#=5 block#=19 blocks=1 |
| WAIT #1: nam='db file scattered read' ela= 4003 file#=5 block#=20 blocks=5 |
| WAIT #1: nam='db file scattered read' ela= 6048 file#=5 block#=25 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 1155 file#=5 block#=34 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 860 file#=5 block#=41 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 837 file#=5 block#=50 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 1009 file#=5 block#=57 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 890 file#=5 block#=66 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 837 file#=5 block#=73 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 10461 file#=5 block#=82 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 623 file#=5 block#=89 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 1077 file#=5 block#=98 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 49146 file#=5 block#=105 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 719 file#=5 block#=114 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 1093 file#=5 block#=121 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 1293 file#=5 block#=130 blocks=7 |
| WAIT #1: nam='db file scattered read' ela= 2103 file#=5 block#=137 blocks=8 |
| WAIT #1: nam='db file scattered read' ela= 42206 file#=5 block#=147 blocks=126 |
Full scan 10.2 vs. 11.2

Relevant part of 10046/8 trace file of version 11.2.0.3:

```sql
WAIT #140120507194664: nam='db file sequential read' ela= 12607 file#=5
  block#=43394 blocks=1 obj#=14033 tim=1329685383169372
  nam='direct path read' ela= 50599 file number=5 first dba=43395 block cnt=13
  nam='direct path read' ela= 21483 file number=5 first dba=43425 block cnt=15
  nam='direct path read' ela= 10766 file number=5 first dba=43441 block cnt=15
  nam='direct path read' ela= 12915 file number=5 first dba=43457 block cnt=15
  nam='direct path read' ela= 12583 file number=5 first dba=43473 block cnt=15
  nam='direct path read' ela= 11899 file number=5 first dba=43489 block cnt=15
  nam='direct path read' ela= 10010 file number=5 first dba=43505 block cnt=15
  nam='direct path read' ela= 160237 file number=5 first dba=43522 block cnt=126
  nam='direct path read' ela= 25561 file number=5 first dba=43650 block cnt=126
  nam='direct path read' ela= 121507 file number=5 first dba=43778 block cnt=126
  nam='direct path read' ela= 25253 file number=5 first dba=43906 block cnt=126
```
First single block read

• The segment header is read separately
  – Single block, read into SGA

• The header block is listed in dba_segments

<table>
<thead>
<tr>
<th>OWNER</th>
<th>SEGMENT_NAME</th>
<th>HEADER_FILE</th>
<th>HEADER_BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>T2</td>
<td>5</td>
<td>130</td>
</tr>
</tbody>
</table>
Full scan 10.2 vs. 11.2

• A full scan uses direct path reads in the v11 case.
  – Noticeable by ‘direct path read’ event
  – Direct path reads go to PGA
  – Which means the blocks read are not cached
Full scan 10.2 vs. 11.2

• Do all full scans in version 11 always use direct path?

• Direct path reads are considered
  – if #blocks of the segment > 5*_small_table_threshold

• PS: MOS note 787373.1
• “How does Oracle load data into the buffer cache for table scans?”
• Mentions _small_table_threshold being the limit
  – Note INCORRECT!
Direct path read

Small table threshold of my Oracle 11 instance:

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>_small_table_threshold</td>
<td>245</td>
</tr>
</tbody>
</table>

This means objects up to $245 \times 5 = 1225$ blocks will be read into buffercache / SGA.

Let’s create a table with a size just below 1225 blocks:

TS@v11203 > create table t1_small as select * from t1 where id <= 47000;

TS@v11203 > exec dbms_stats.gather_table_stats(null,‘T1_SMALL’);
Direct path read

```
SYS@v11203 AS SYSDBA>

    select segment_name, blocks, bytes
    from dba_segments where segment_name = 'T1_SMALL';

    SEGMENT_NAME    BLOCKS    BYTES
    --------------- -------     -----
    T1_SMALL        1024      8388608

SQL@v11203 AS SYSDBA> alter system flush buffer_cache;
```
Direct path read

```sql
TS@v11203 > set autot trace exp stat
TS@v11203 > select count(*) from t1_small;
```

Execution Plan
```
<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>176 (1)</td>
<td>00:00:03</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS FULL</td>
<td>T1_SMALL</td>
<td>47000</td>
<td>176 (1)</td>
<td>00:00:03</td>
</tr>
</tbody>
</table>
```

Plan hash value: 1277318887
Direct path read

Statistics

0 recursive calls
0 db block gets
983 consistent gets
979 physical reads
0 redo size
527 bytes sent via SQL*Net to client
523 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
1 rows processed
Direct path read

SYS@v11203 AS SYSDBA>
select object_id, object_name, owner
from dba_objects where object_name = 'T1_SMALL';

<table>
<thead>
<tr>
<th>OBJECT_ID</th>
<th>OBJECT_NAME</th>
<th>OWNER</th>
</tr>
</thead>
<tbody>
<tr>
<td>66729</td>
<td>T1_SMALL</td>
<td>TS</td>
</tr>
</tbody>
</table>

SYS@v11203 AS SYSDBA> select * from x$kcboqh where obj# = 66729;

<table>
<thead>
<tr>
<th>ADDR</th>
<th>INDX</th>
<th>INST_ID</th>
<th>TS#</th>
<th>OBJ#</th>
<th>NUM_BUF</th>
<th>HEADER</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFFD7FFC6E1EF0</td>
<td>0</td>
<td>5</td>
<td>66729</td>
<td>979</td>
<td>000000390437840</td>
<td></td>
</tr>
</tbody>
</table>

Friday, February 8, 13
Direct path read

Ah, now the full scan is buffered!

Another scan will reuse the cached blocks now:

```
TS@v11203 > select count(*) from t1_small;
...
```

Statistics

----------------------------------------------------------
0   recursive calls
0   db block gets
983  consistent gets
0   physical reads
Direct path read

• What type of wait event will be used for a full scan:
  – Oracle version 11.2
  – If segment is *smaller* than 5 * _small_table_threshold*
Direct path read

Well, try it:

```sql
TS@v11203 > alter session set events '10046 trace name context forever, level 8';
TS@v11203 > select count(*) from t1_small;
...
TS@v11203 > alter session set events '10046 trace name context off';
```

It shows:

```
WAIT #140358956326184: nam='db file sequential read' ela= 38476 file#=5 block#=88706 blocks=1 obj#=14047 tim=1330369985672633
nam='db file scattered read' ela= 116037 file#=5 block#=88707 blocks=5
nam='db file scattered read' ela= 56675 file#=5 block#=88712 blocks=8
nam='db file scattered read' ela= 11195 file#=5 block#=88721 blocks=7
nam='db file scattered read' ela= 132928 file#=5 block#=88728 blocks=8
nam='db file scattered read' ela= 18692 file#=5 block#=88737 blocks=7
nam='db file scattered read' ela= 87817 file#=5 block#=88744 blocks=8
```
Oracle 11 multiblock IO

• In version 11 of the Oracle database
  – Multiblocks reads use both wait events:
    ‣ db file scattered read
    ‣ direct path read
  – Which are two different codepath’s
Implementation

• Buffered multiblock reads
  – Buffered multiblock reads == ‘db file scattered read’
  – Up to version 10 the ONLY option for non-PQ multiblock reads
  – Starting from version 11, a possible multiblock read option
Buffered multiblock reads

SYS@v10201 AS SYSDBA> select segment_name, extent_id, block_id, blocks, bytes from dba_extents where segment_name = 'T2' and owner = 'TS' order by extent_id;

<table>
<thead>
<tr>
<th>SEGMENT_NAME</th>
<th>EXTENT_ID</th>
<th>BLOCKS</th>
<th>BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td>0</td>
<td>8</td>
<td>65536</td>
</tr>
<tr>
<td>T2</td>
<td>15</td>
<td>8</td>
<td>65536</td>
</tr>
<tr>
<td>T2</td>
<td>16</td>
<td>128</td>
<td>1048576</td>
</tr>
<tr>
<td>T2</td>
<td>78</td>
<td>128</td>
<td>1048576</td>
</tr>
<tr>
<td>T2</td>
<td>79</td>
<td>1024</td>
<td>8388608</td>
</tr>
<tr>
<td>T2</td>
<td>91</td>
<td>1024</td>
<td>8388608</td>
</tr>
</tbody>
</table>
Buffered multiblock reads

Version 10 multiblock reads:

WAIT #2: nam='db file sequential read' ela= 12292 file#=5 block#=19 blocks=1
WAIT #2: nam='db file scattered read' ela= 179162 file#=5 block#=20 blocks=5
WAIT #2: nam='db file scattered read' ela= 47597 file#=5 block#=25 blocks=8
WAIT #2: nam='db file scattered read' ela= 5206 file#=5 block#=34 blocks=7
WAIT #2: nam='db file scattered read' ela= 94101 file#=5 block#=41 blocks=8
WAIT #2: nam='db file scattered read' ela= 512 file#=5 block#=50 blocks=7
WAIT #2: nam='db file scattered read' ela= 87657 file#=5 block#=57 blocks=8
WAIT #2: nam='db file scattered read' ela= 27488 file#=5 block#=66 blocks=7
WAIT #2: nam='db file scattered read' ela= 24316 file#=5 block#=73 blocks=8
WAIT #2: nam='db file scattered read' ela= 55251 file#=5 block#=82 blocks=7
WAIT #2: nam='db file scattered read' ela= 641 file#=5 block#=89 blocks=8
WAIT #2: nam='db file scattered read' ela= 455 file#=5 block#=98 blocks=7
WAIT #2: nam='db file scattered read' ela= 43826 file#=5 block#=105 blocks=8
WAIT #2: nam='db file scattered read' ela= 32685 file#=5 block#=114 blocks=7
WAIT #2: nam='db file scattered read' ela= 60212 file#=5 block#=121 blocks=8
WAIT #2: nam='db file scattered read' ela= 37735 file#=5 block#=130 blocks=7
WAIT #2: nam='db file scattered read' ela= 59565 file#=5 block#=137 blocks=8

(ps: edited for clarity)
nam='db file scattered read' ela= 87657 file#=5 block#=147 blocks=126
Non buffered multiblock reads

WAIT #140120507194664: nam='db file sequential read' ela= 12607 file#=5 block#=43394 blocks=1 obj#=14033 tim=1329685383169372

nam='direct path read' ela= 50599 file number=5 first dba=43395 block cnt=13
nam='direct path read' ela= 21483 file number=5 first dba=43425 block cnt=15
nam='direct path read' ela= 10766 file number=5 first dba=43441 block cnt=15
nam='direct path read' ela= 12915 file number=5 first dba=43457 block cnt=15
nam='direct path read' ela= 12583 file number=5 first dba=43473 block cnt=15
nam='direct path read' ela= 11899 file number=5 first dba=43489 block cnt=15
nam='direct path read' ela= 10010 file number=5 first dba=43505 block cnt=15
nam='direct path read' ela= 160237 file number=5 first dba=43522 block cnt=126
nam='direct path read' ela= 25561 file number=5 first dba=43650 block cnt=126
nam='direct path read' ela= 121507 file number=5 first dba=43778 block cnt=126
nam='direct path read' ela= 25253 file number=5 first dba=43906 block cnt=126
ASSM

• Automatic segment space management
  – Tablespace property
  – Default since Oracle 10.2
• Uses L 1/2/3 bitmap blocks for space management

• With extent size of
  – 8 blocks: 1 BMB as first block of every other extent
  – 128 blocks: 2 BMB as first blocks in all extents
  – 1024 blocks: 4 BMB as first blocks in all extents
Multiblock implementation

• Conclusion:

  – Buffered reads scan up to:
    ‣ Non data (space admin. bitmap) block
    ‣ Extent border
    ‣ Block already in cache (from TOP, didn’t test this)

  – Direct path/non buffered reads scan up to:
    ‣ Non data (space admin. bitmap) block
    ‣ Block already in cache (from TOP, didn’t test this)
Waits and implementation

• ‘Wait’ or wait event
  – Part of the formula:
    › Elapsed time = CPU time + Wait time

• Inside the Oracle database it is meant to record the time spent in a specific part of the oracle database code not running on CPU.

• Let’s look at the implementation of some of the wait events for multiblock reads!
strace

- Linux tool for tracing (viewing) system calls
  - Solaris/AIX: truss, HPUX: tusc.

- Very, very, useful to understand what is happening
- Much people are using it for years
- **STRACE LIES!** (at least on linux)
strace lies

• Strace doesn’t show io_getevents() if:
  – timeout struct set to {0,0} (‘zero’)
  – does not succeed in reaping min_nr IO’s

• This strace omission is not documented
strace lies

- This is best seen with system’s IO capability severely throttled (1 IOPS)
- See http://fritshoogland.wordpress.com/2012/12/15/throttling-io-with-linux/

- Cgroups
  - Control groups
  - Linux feature
  - Fully available with OL6
strace lies

• Strace output
  – Version 11.2.0.3 (reason shown later)
  – IO throttled to 1 IOPS
  – Full table scan doing count(*) on t2
  – With 10046 at level 8
    ▸ To show where waits are occurring

  – Start of FTS, up to first reap of IO
strace lies

```
io_submit(139801394388992, 1, {{0x7f260a8b3450, 0, 0, 0, 257}}) = 1
io_submit(139801394388992, 1, {{0x7f260a8b31f8, 0, 0, 0, 257}}) = 1
io_getevents(139801394388992, 1, 128, {{0x7f260a8b3450, 0x7f260a8b3450, 106496, 0}}, {600, 0}) = 1
write(8, "WAIT #139801362351208: nam='dire"..., 133) = 133

* edited for clarity
```
strace lies

• Profile the same using ‘gdb’

• Set breakpoints at functions:
  – io_submit, io_getevents_0_4
  – kslwtbctx, kslwttectx

• Let gdb continue after breakpoint

• The symbol table is preserved in the oracle binary
• Making it able to set breakpoints at functions
strace lies

#0  io_submit (ctx=0x7f46fe708000, nr=1, iocbs=0x7fff24547ce0) at io_submit.c:23
#0  io_submit (ctx=0x7f46fe708000, nr=1, iocbs=0x7fff24547ce0) at io_submit.c:23
Breakpoint 3, io_getevents_0_4 (ctx=0x7f46fe708000, min_nr=2, nr=128, events=0x7fff24550348, timeout=0x7fff24551350) at io_getevents.c:46
Breakpoint 3, io_getevents_0_4 (ctx=0x7f46fe708000, min_nr=2, nr=128, events=0x7fff24553428, timeout=0x7fff24554430) at io_getevents.c:46
Breakpoint 3, io_getevents_0_4 (ctx=0x7f46fe708000, min_nr=2, nr=128, events=0x7fff24550148, timeout=0x7fff24551150) at io_getevents.c:46
Breakpoint 3, io_getevents_0_4 (ctx=0x7f46fe708000, min_nr=2, nr=128, events=0x7fff24553228, timeout=0x7fff24554230) at io_getevents.c:46
#0  0x00000000000000008f9a652 in kslwtbctx ()
Breakpoint 3, io_getevents_0_4 (ctx=0x7f46fe708000, min_nr=2, nr=128, events=0x7fff24550148, timeout=0x7fff24551150) at io_getevents.c:46
#0  0x00000000000000008fa1334 in kslwtectx ()

* edited for clarity
db file scattered read

Basic principle

file # and # blocks are determined

ela time of ‘db file scattered read’

read ready, blocks available

read call of # bytes

time
db file scattered read

Implementation
synchronous IO
10.2.0.1/11.2.0.1/11.2.0.3

file # and # blocks are
determined

ela time of ‘db file scattered read’

read ready, blocks
available

grayed means ‘optional’

pread64(fd, buf, #bytes, offset)
db file scattered read

Implementation asynchronous IO 10.2.0.1

file # and # blocks are determined

ela time of ‘db file scattered read’

read ready, blocks available

io_submit(aio_ctx, cb, {iocb})

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)
db file scattered read

Implementation asynchronous IO 11.2.0.1

file # and # blocks are determined

ela time of ‘db file scattered read’

read ready, blocks available

time

io_submit(aio_ctx, cb, {iocb})

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)
db file scattered read

Implementation asynchronous IO
11.2.0.3

file # and # blocks are determined

ela time of ‘db file scattered read’

read ready, blocks available

io_submit(aio_ctx, cb, {iocb})

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)
direct path read - 11g

- Time spent on waiting for reading blocks for putting them into the PGA

- Reports wait time of the request that gets reaped with a timed io_getevents() call.
- Multiple IO requests can be submitted with AIO
- At start, Oracle tries to keep 2 IO’s in flight
- Wait time is only reported if ‘waiting’ occurs
  - Waiting means: not ALL IO’s can be reaped immediately after submitting
direct path read 11g

Implementation
asynchronous IO
11.2.0.1

file № and № blocks are
determined
for a number of IO’s

 ela time of ‘direct path read’*

read ready, blocks
available

time

io_submit(aio_ctx, #cb, {iocb})

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)
direct path read 11g

Implementation asynchronous IO 11.2.0.3

file # and # blocks are determined for a number of IO’s

io_submit(aio_ctx, cb, {iocb})

ela time of ‘direct path read’*

read ready, blocks available

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)
direct path read

Implementation asynchronous IO 10.2.0.1

file # and # blocks are determined for a number of IO’s

io_submit(aio_ctx, cb, {iocb})

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)

ela time of ‘direct path read’

read ready, blocks available

time
direct path read

Implementation
synchronous IO
10.2.0.1/11.2.0.1/11.2.0.3

file # and # blocks are determined for a number of IO's

ela time of 'direct path read'

read ready, blocks available

time

pread64(fd, buf, #bytes, offset)
kfk: async disk IO

• Only seen with ‘direct path read’ waits and ASM
• Always seen in version 11.2.0.1
• Gone with 11.2.0.2+

• Not normally seen in version 11.2.0.2+

• KFK = Kernel File ASM code layer
kfk: async disk io

Implementation asynchronous IO 11.2.0.1

File # and # blocks are determined for a number of IO’s

Ela time of ‘direct path read’*

Read ready, blocks available

Time

io_submit(aio_ctx, #cb, {iocb})

io_getevents(aio_ctx, min_nr, nr, io_event, timeout)

Kfk: async disk io
IO Slots

Discussion with Kerry Osborne about IO’s on Exadata
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options
[oracle@dm01db01 ~]$ sqlplus cg/cg

SQL*Plus: Release 11.2.0.1.0 Production on Tue Nov 22 23:00:47 2011

Copyright (c) 1982, 2009, Oracle.  All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options

SQL> alter session set cell_offload_processing=false;
Session altered.

(reverse-i-search)`cou`: select count(*) from cg_var;
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP, Data Mining and Real Application Testing options
[oracle@dm01db01 [vxone1] ~]$ rsqlplus cg/cg

SQL*Plus: Release 11.2.0.2.0 Production on Tue Nov 22 23:45:33 2011
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP, Data Mining and Real Application Testing options

SQL> alter session set cell_offload_processing=false;
Session altered.
(reverse-i-search)`':
IO Slots

• Jonathan Lewis pointed me to ‘total number of slots’
  – v$sysstat
  – v$sesstat

• Global or per session number of slots

• ‘Slots are a unit of I/O and this factor controls the number of outstanding I/Os’
  – Comment with event 10353
IO Slots

• ‘total number of slots’
  
  – Is **NOT** cumulative!

• So you won’t capture this statistic when taking delta’s from v$sysstat/v$sesstat!
IO Slots

• Let’s look at the throughput statistics again
  – But together with number of slots
IO physical read total bytes

<table>
<thead>
<tr>
<th>Time</th>
<th>GB per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>22:54:40</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:54:50</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:00</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:10</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:20</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:30</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:40</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:50</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:00</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:10</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:20</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:30</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:40</td>
<td>0.000000006</td>
</tr>
</tbody>
</table>

- physical read total bytes

Total number of slots

<table>
<thead>
<tr>
<th>Time</th>
<th>Number of slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>22:54:40</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:54:50</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:00</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:10</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:20</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:30</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:40</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:55:50</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:00</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:10</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:20</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:30</td>
<td>0.000000006</td>
</tr>
<tr>
<td>22:56:40</td>
<td>0.000000006</td>
</tr>
</tbody>
</table>

- total number of slots

Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options

(reverse-i-search)``: alter session set cell_offload_processing=false;
IO physical read total bytes

GB per second


- physical read total bytes

total number of slots

number of slots


- total number of slots

Copyright (c) 1982, 2010, Oracle. All rights reserved.

ERROR:
ORA-28002: the password will expire within 7 days

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.2.0 - 64bit Production
With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,
Data Mining and Real Application Testing options.

(reverse-i-search)`':  

Friday, February 8, 13
IO Slots

- IO Slots is a mechanism to take advantage of storage bandwidth using AIO
- With version 11 direct path reads can be used by both PQ slaves as well as non PQ foregrounds
  - IO Slots are not used with buffered reads

- Each outstanding asynchronous IO request is tracked using what is called a ‘slot’
- Default and minimal number of slots: 2
‘autotune’

• The direct path code changed with version 11
• Second observation:
  – The database foreground measures direct path IO effectiveness
  – It measures time, wait time and throughput
  – The oracle process has the ability to add more asynchronous IO slots
  – Only does so starting from 11.2.0.2
    ‣ Although the mechanism is there in 11.2.0.1
‘autotune’

• Introducing event 10365
  – “turn on debug information for adaptive direct reads”

• Set to 1 to get debug information
  – alter session set events ‘10365 trace name context forever, level 1’
‘autotune’

kcbldrsini: Timestamp 61180 ms
kcbldrsini: Current idx 16
kcbldrsini: Initializing kcbldrps
kcbldrsini: Slave idx 17
kcbldrsini: Number slots 2
kcbldrsini: Number of slots per session 2

kcblsinc:Timing time 1693472, wait time 1291416, ratio 76 st 248752270 cur 250445744
kcblsinc: Timing curidx 17 session idx 17
kcblsinc: Timestamp 64180 ms
kcblsinc: Current idx 17
kcblsinc: Slave idx 17
kcblsinc: Number slots 2
kcblsinc: Number of slots per session 2
kcblsinc: Previous throughput 8378 state 2
kcblsinc: adaptive direct read mode 1, adaptive direct write mode 0


‘autotune’

*** 2011-11-28 22:58:54.988
kcblsinc: Timing time 2962717, wait time 2923226, ratio 98 st 253662983 cur 256625702
kcblsinc: Timing curidx 19 session idx 19
kcblsinc: Timestamp 70270 ms
kcblsinc: Current idx 19
kcblsinc: Slave idx 19
kcblsinc: Number slots 2
kcblsinc: Number of slots per session 2
kcblsinc: Previous throughput 11210 state 1
kcblsinc: adaptive direct read mode 1, adaptive direct write mode 0
kcblsinc: Adding extra slos 1

*** 2011-11-28 22:58:58.999
kcblsinc: Timing time 4011239, wait time 3528563, ratio 87 st 256625785 cur 260637026
kcblsinc: Timing curidx 20 session idx 20
kcblsinc: Timestamp 74170 ms
kcblsinc: Current idx 20
kcblsinc: Slave idx 20
kcblsinc: Number slots 3
kcblsinc: Number of slots per session 3
kcblsinc: Previous throughput 12299 state 2
‘autotune’

• Looking at the 10365 trace, the reason 11.2.0.1 does not ‘autotune’ could be guessed....
‘autotune’

kcblsinc:Timing time 3092929, wait time 0, ratio 0 st 4271872759 cur 4274965690
kcblsinc: Timing curidx 65 session idx 65
kcblsinc: Timestamp 192430 ms
kcblsinc: Current idx 65
kcblsinc: Slave idx 65
kcblsinc: Number slots 2
kcblsinc: Number of slots per session 2
kcblsinc: Previous throughput 20655 state 2
kcblsinc: adaptive direct read mode 1, adaptive direct write mode 0

kcblsinc:Timing time 2944852, wait time 0, ratio 0 st 4274965762 cur 4277910616
kcblsinc: Timing curidx 66 session idx 66
kcblsinc: Timestamp 195430 ms
kcblsinc: Current idx 66
kcblsinc: Slave idx 66
kcblsinc: Number slots 2
kcblsinc: Number of slots per session 2
kcblsinc: Previous throughput 20746 state 1
kcblsinc: adaptive direct read mode 1, adaptive direct write mode 0
11.2.0.3 FAST IO

# slots starts with 2

io_submit(aio_ctx, 1, {iocb})
kcbgsinc ()
io_getevents(aio_ctx, 2, 128, io_event, {0, 0}) OK!
kcbgtcr ()

#### Friday, February 8, 13
IO slots

11.2.0.3 FAST IO

kcblsinc ()
+1 'slos'

io_submit(aio_ctx, 1, {iocb})

io_getevents(aio_ctx, 3, 128, io_event, {0, 0})
OK!
Time and waits

• Waits implementation
  – Most are system call instrumentation
    ‣ db file sequential read
  – ‘direct path read’ is different.
    ‣ Only shows up if not all IO can be reaped immediately
    ‣ The wait only occurs if process is truly waiting
    ‣ With AIO, a process has the ability to keep on processing without waiting on IO
    ‣ Wait time is not physical IO latency
Conclusion

• In Oracle version 10.2 and earlier non-PX reads use:
  – db file sequential read / db file scattered read events
  – Read blocks go to buffercache.

• Starting from Oracle version 11 reads could do both
  – buffered reads
  – unbuffered or direct path reads
Conclusion

• Direct path read is decision in IO codepath of full scan.
  – NOT an optimiser decision(!)

• In Oracle version 11, a read is done buffered, unless database decides to do a direct path read

• Direct path read decision is influenced by
  – Type of read (FTS or FFIS)
  – Size of segment (> 5 * _small_table_threshold)
  – Number of blocks cached (< ~ 50%)
Conclusion

• By default, (AIO) direct path read uses two slots.
  – ‘autotune’ scales up in steps.
  – I’ve witnessed it scale up to 32 slots.

• Direct path code has an ‘autotune’ function, which can add IO slots.
  – In order to be able to use more bandwidth
  – Direct path ‘autotune’ works for PX reads too!

• ‘autotune’ does not kick in with Oracle version 11.2.0.1
Thank you for attending!

Questions?
Thanks, Links, etc.

- Tanel Poder
- Jason Arneil
- Klaas-Jan Jongsma
- Doug Burns
- Cary Millsap
- [http://www.oracle.com/pls/db112/homepage](http://www.oracle.com/pls/db112/homepage)