Keep Buffer 활용방안

Techdata OTS Team

2015.11

Contents

1. 개요

Oracle은 유저가 요청한 작업을 빠르게 처리하기 위해 Buffer Cache라는것을 사용한다.
Buffer Cache는 SGA에 위치하고 있으며, 오라클 인스턴스에 접속하는 모든 프로세스에 의해 공유된다.
이 Buffer Cache는 오라클 I/O 관리의 핵심으로 자주 사용하는 데이터 파일의 블록들을 메모리에 상주시킴으로써 물리적인 I/O Operation을 줄이는 역할을 한다. Buffer Cache를 효과적으로 사용하면 물리적 I/O가 줄어들고 자연스럽게 I/O 성능문제를 해결할 수 있다.

오라클의 버전이 올라감에 따라 Buffer Cache를 처리하는 알고리즘은 끊임없이 개선되었고, 더불어 새로운 관리 방법들이 제공되었다. Oracle 7까지의 Buffer Cache는 하나의 틀로서 운영되었고, 각 Object의 특성에 따른 차별적인 Buffer Cache 이용이 어려웠다. 이런 면을 해결하기 위해 Oracle 8부터 Multiple buffer pool이라는 기능을 지원하게 되었는데, 이로 인해 각 Object의 특성이나 엑세스 빈도 등 차별성을 고려하여 Buffer Cache를 보다 세밀하게 관리 할 수 있게 되었다.

Keep Buffer는 이 Multiple Buffer Pool을 구성하는 여러 영역중의 하나이다.

2. Keep Buffer 사용 목적 및 특성

Keep Buffer의 사용 목적은 본래의 Buffer Cache의 목적과 마찬가지로, Object를 메모리에 상주시킴으로써 물리적인 I/O를 피하는데 있다.

Keep Buffer Pool과 Default Buffer Pool이 데이터 블록을 Cache하거나 Flush 할 때 서로 다른 알고리즘을 사용하지는 않는다. 그럼에도 불구하고 하나였던 Buffer Cache영역을 Multiple Buffer Pool로 나누게 된 이유는 Object의 특성을 고려한 Buffer Cache 이용을 위함이다.
KEEP Buffer의 메모리 공간은 Sequential하게 관리된다. 한번 KEEP된 세그먼트는 KEEP Buffer의 메모리 공간을 모두 할당하기 전까지 메모리에 유지되다가KEEP Buffer 공간을 모두 할당하게 되면, 가장 오래된 블록부터 default pool로 밀려난다. 때문에 KEEP 대상이 되는 세그먼트들의 사이즈를 정확히 계산하여 KEEP Buffer 크기를 적절히 할당해야 한다.

3. Keep Buffer 사용 절차 (1/13)

Keep Buffer의 사용은 다음과 같은 순서로 진행한다.

① KEEP 대상 선정하기
② OS Memory & SGA 공간 확인
③ KEEP Buffer 설정
④ 테이블/인덱스 속성 변경
⑤ 테이블/인덱스 Keeping
⑥ KEEP 효율성 체크
⑦ KEEP 대상 선정 기준

3. Keep Buffer 사용 절차 (2/13)

① KEEP 대상 선정 하기

KEEP 대상 선정에 있어서 명확한 기준점은 없다. 실제 업무를 고려하여 각 DB의 운영환경에 맞는 대상을 선정해야 한다. 만약 KEEP하는 대상이 아주 빈번하게 사용되는 블록 이라면, 기본적인 Default Buffer를 사용해도 Cache돼있을 가능성이 높은데, 이런 경우에는 Keep Buffer 사용이 성능상의 이점을 가져오지 못한다. 반대로 자주 사용되지 않는 블록을 Keep Buffer에 상주시킨다면, 사용하지 않는 메모리를 가지고 있는 것이기 때문에 전반적인 성능을 저하시키는 요인이 될 수도 있다.

때문에 Keep 대상을 선정하는데 있어서는 실제 업무의 고려가 필수적이다. 예를 들어 하루에 1번만 수행되는 프로그램인데 어떻게 해서든 수행시간을 단축시켜야 하는 경우나, 많은 업무를 처리하는 시간대에 꼭 수행 되야 하는데 많은 I/O때문에 병목현상을 일으켜서 시스템에 전반적으로 악영향을 끼치는 프로그램 등이 있을 수 있다. 이런 프로그램들이 사용하는 Object 들이 Keep 대상이 될 수 있다. 그러나 위와 같은 업무 프로그램에 사용되는 모든 세그먼트를 KEEP
Buffer에 상주 시킬 수 는 없다. 그러므로 KEEP Buffer에 상주시킬 대상은 각 DB 운영환경을 고려하여 선정해야 한다. 이외에도 크기, DML빈도, 데이터 엑세스 빈도에 따라 Keep하기에 적절한 세그먼트들이 존재한다.

3. Keep Buffer 사용 절차 (3/13)

선정 기준[1]. 프로그램 중요도

Keep 대상 선정에 있어서 가장 중요한 부분이 바로 해당 세그먼트를 조회하는 업무 프로그램의 중요도 이다. 해당 프로그램이 중요하지 않다면 굳이 Keep Buffer를 사용해야 할 필요가 없다. 반대로 해당 세그먼트를 조회하는 프로그램이 중요도가 아주 높고 어떻게든 수행시간을 단축해야 한다면 프로그램 수행 빈도와 상관없이KEEP 대상으로의 선정을 고려할 수 있다.

선정 기준[2]. 세그먼트 크기

세그먼트 크기가 일정하지 않고, 과다하게 커지는 세그먼트는 Keep Buffer의 효율성을 떨어뜨릴 수 있다. Keep된 세그먼트는 Keep Buffer의 용량이 부족하면 오래된 블록부터 Default Buffer로 밀려나게 되는데, 크기가 계속 커지는 세그먼트가 Keep Buffer에 존재한다면 타 세그먼트를 조회하는 프로그램의 성능 저하를 가져올 수 있기 때문이다. 따라서 일정한 사이즈 또는 변동 량이 심하지 않으면서 최대 크기가 일정 수준 이하인 경우의 세그먼트를 선정하는 것이 바람직하다. 예를 들면 '최대 크기가 10만 블록 이하인 세그먼트' 같은 기준을 정할 수 있다.

선정 기준[3]. Full Table Scan & Index Full Scan & Index Fast Full Scan

KEEP Buffer에 KEEP된 세그먼트를 조회할 때 효율성을 극대화 하기 위해서는 다소 많은 량을 처리해야 하는 경우이다. Scan 범위가 넓은 비효율 Index Scan이나 Full Table Scan, Index Fast Full Scan으로 처리되는 세그먼트가 대상이 될 수 있다.

3. Keep Buffer 사용 절차 (4/13)

KEEP 대상 선정 SQLScript (1/2)

```
SELECT owner ,
       table_name ,
       index_name ,
       partition_name ,
       SUM( blocks ) AS t_blocks
FROM   (
        SELECT sg.owner ,
               decode( SUBSTR( s.ob_type , 1 , 5 ) , 'TABLE' , s.ob_name , 'INDEX' , (
                                 SELECT table_name
                                 FROM   dba_indexes
                                 WHERE  index_name = s.ob_name
                               ) ) AS table_name ,
               decode( SUBSTR( s.ob_type , 1 , 5 ) , 'INDEX' , s.ob_name ) AS index_name ,
               sg.partition_name ,
               sg.blocks
        FROM   (
                SELECT DISTINCT object_name AS ob_name ,
                       object_type AS ob_type
                FROM   v$sql_plan
                WHERE ( operation = 'TABLE ACCESS'
                        AND options = 'FULL' )
                OR ( operation = 'INDEX'
                        AND options = 'FULL SCAN' )
                OR ( operation = 'INDEX'
                        AND options = 'FAST FULL SCAN' ) --> 선정 기준[3]
               ) s ,
               dba_segments sg
```

3. Keep Buffer 사용 절차 (5/13)

KEEP 대상 선정 SQLScript (2/2)

```
WHERE   s.ob_name = sg.segment_name
        )
GROUP   BY owner ,
        table_name ,
        index_name ,
        partition_name
HAVING SUM( blocks ) > 100000 --> 선정 기준[2]SELECT * FROM DUAL
```

3. Keep Buffer 사용 절차 (6/13)

② OS Memory & SGA 공간 확인

OS Memory SGA공간 확인 SQLScript
$ cat /proc/meminfo
MemTotal: 4055152 kB
MemFree: 1390308 kB
Buffers: 166768 kB
Cached: 2019992 kB
SwapCached: 0 kB
Active: 1118484 kB
Inactive: 1277864 kB
.....

3. Keep Buffer 사용 절차 (7/13)

SGA공간 확인 SQLScript

● SGA전체 size확인

```
SELECT name , ROUND( bytes/1024/1024 ) "size(MB)"
FROM    V$SGAINFO;
```

NAME	size(MB)
Fixed SGA Size	2
Redo Buffers	5
Buffer Cache Size	48
Shared Pool Size	128
Large Pool Size	0
Java Pool Size	24
Streams Pool Size	0
Shared IO Pool Size	0
Granule Size	4
Maximum SGA Size	207
Startup overhead in Shared Pool	72
Free SGA Memory Available	0

● Data Buffer size확인

```
 SELECT name , current_size
FROM    v$buffer_pool;
```

NAME	CURRENT_SIZE
DEFAULT	48

3. Keep Buffer 사용 절차 (8/13)

③ KEEP BUFFER 설정

KEEP Buffer 설정은 KEEP Buffer 크기와 SGA 여유공간에 따라, Online 작업 또는 Offline 작업으로 수행한다. 이 문서의 스크립트는 SGA 영역의 메모리 관리를 수동으로 하는 경우를 바탕으로 작성 하였다.

KEEP Buffer 설정 Script

```
-- 현재 상태 Check

@sga

NAME                             size(MB)
-------------------------------- ----------
Buffer Cache Size                     500
Maximum SGA Size                     1019
Free SGA Memory Available             228

@bc

NAME             CURRENT_SIZE
---------------- ------------
DEFAULT                   500
```

3. Keep Buffer 사용 절차 (9/13)

● KEEP Buffer의 크기가 SGA의 Free 공간보다 작은 경우 Online (KEEP Buffer 100M)

```
SQL> alter system set db_keep_cache_size = 100M scope = both;

System altered.

-- Keep 적용후 Memory 현황 Check

SQL> @sga

NAME                            size(MB)
------------------------------- ----------
Buffer Cache Size                      600
Maximum SGA Size                      1019
Free SGA Memory Available              128

SQL> @bc

NAME             CURRENT_SIZE
---------------- ------------
KEEP                      100
DEFAULT                   500
```

3. Keep Buffer 사용 절차 (10/13)

● KEEP Buffer의 크기가 SGA의 Free 공간보다 큰 경우 (KEEP Buffer 300M)

1) SGA 전체 크기 늘린 후 KEEP Buffer 할당 Offline 작업 필요

```
SQL> alter system set sga_max_size = 1100M scope = spfile;
System altered.

SQL> alter system set db_keep_cache_size = 300M scope = spfile;
System altered.

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup
ORACLE instance started.

Total System Global Area     1169227776 bytes
Fixed Size                      2212696 bytes
Variable Size                 301993128 bytes
Database Buffers              855638016 bytes
Redo Buffers                    9383936 bytes
Database mounted.
Database opened.
```

3. Keep Buffer 사용 절차 (11/13)

```
SQL> @sga

NAME                             size(MB)
-------------------------------- ----------
Buffer Cache Size                       816
Maximum SGA Size                       1115
Free SGA Memory Available                 0

SQL> @bc

NAME             CURRENT_SIZE
---------------------- ------------
KEEP                        304
DEFAULT                     512
```

=> SGA 전체 크기가 1G를 초과하면서 Granule 크기가 16Mb 로 늘어났다
이 때 지정한 값 보다 큰 16의 배수 중 가장 작은 크기의 값이 할당된다.

3. Keep Buffer 사용 절차 (12/13)

2) SGA의 다른 영역의 크기를 줄인 후 KEEP Buffer 할당 할 경우 Online 작업 가능

Online 상태에서 변경 가능 Parameter 추출 Script

```
SELECT name , issys_modifiable
FROM   v$parameter
WHERE  name LIKE '%size%'
AND    issys_modifiable = 'IMMEDIATE'
```

NAME	ISSYS_MOD
shared_pool_size	IMMEDIATE
large_pool_size	IMMEDIATE
java_pool_size	IMMEDIATE
streams_pool_size	IMMEDIATE
db_cache_size	IMMEDIATE
db_2k_cache_size	IMMEDIATE
db_4k_cache_size	IMMEDIATE
db_8k_cache_size	IMMEDIATE
db_16k_cache_size	IMMEDIATE
db_32k_cache_size	IMMEDIATE
db_keep_cache_size	IMMEDIATE
db_recycle_cache_size	IMMEDIATE
_shared_io_pool_size	IMMEDIATE
db_flash_cache_size	IMMEDIATE
db_recovery_file_dest_size	IMMEDIATE
result_cache_max_size	IMMEDIATE
workarea_size_policy	IMMEDIATE
max_dump_file_size	IMMEDIATE

=> 해당 Parameter 값을 적절히 조절하여 Free Memory 확보 후 KEEP설정

3. Keep Buffer 사용 절차 (13/13)

④ 테이블/인덱스 속성 변경

테이블/인덱스 속성 변경 Script

```
ALTER TABLE T1 STORAGE (BUFFER_POOL KEEP);          --테이블 속성 변경
ALTER INDEX T1_PK STORAGE (BUFFER_POOL KEEP);       --인덱스 속성 변경
ALTER TABLE P1 MODIFY PARTITION P1_1 STORAGE (BUFFER_POOL KEEP); --파티션 테이블 속성 변경
ALTER INDEX P1_ID1 MODIFY PARTITION P1_ID1_1 STORAGE (BUFFER_POOL KEEP);
--파티션 인덱스 속성 변경
```

⑤ 테이블/인덱스 Keeping

Segment의 Buffer Pool 이 KEEP으로 설정된 테이블과 인덱스는 Query 시 KEEP Buffer에 해당 세그먼트의 블록을 로딩하게 된다. 그러므로 최초 세그먼트를 Loading할 때에는 Disk I/O가 발생하게 된다. 만일 처음 실행하는 때를 포함하여 모든 Application의 조회에서 Disk I/O를 제거하고 싶다면, 업무가 진행되기 전에 해당 세그먼트들을 Full Table Scan이나 Index Fast Full Scan으로 KEEP Buffer에 로딩시키면 된다.

4. Keep Buffer 효율성 판단 (1/4)

KEEP Buffer의 사용에 명확한 기준이 정해져 있는 것이 아니라 운영 환경에 따라 차이가 존재 한다. 때문에 모든 운영 환경에서 같은 방법으로 효율성을 판단하기에는 무리가 있다. 하지만 다음과 같은 자료들이 KEEP Buffer의 효율성을 판단하는데 근거가 될 수 있다.

KEEP Buffer Size & Hit Ratio SQLScript

```
SELECT current_size keep_size , seg_size , ROUND( seg_size/current_size*100 , 1 ) "Ratio(%)"
FROM   v$buffer_pool ,
       (
        SELECT SUM( bytes ) /1024 /1024 seg_size
        FROM   dba_segments
        WHERE  buffer_pool = 'KEEP'
       )
WHERE  name = 'KEEP'

KEEP_SIZE   SEG_SIZE    Ratio(%)
----------  ----------  ----------
       304         118        38.8

SELECT db_block_gets , consistent_gets , physical_reads ,
       CASE
           WHEN db_block_gets+consistent_gets <> 0
           THEN ROUND(( 1-( physical_reads/( db_block_gets+consistent_gets ) ) ) *100 , 2 )
       END "Keep_Hit(%)"
FROM   v$buffer_pool_statistics
WHERE  name = 'KEEP'

DB_BLOCK_GETS  CONSISTENT_GETS   PHYSICAL_READS    Keep_Hit(%)
-------------- ----------------- ----------------- -------------
            0             44474              4435         90.03
```

4. Keep Buffer 효율성 판단 (2/4)

만약 KEEP Buffer를 사용하는 Segment 크기의 총 합이 KEEP Buffer 크기보다 작은 경우, 해당 Segment들의 크기가 더 이상 커지지 않는다면, 한 번 KEEP 영역으로 올라간 Segment의 Cache Hit Ratio가 100%에 가깝게 될 것이다.

만일 KEEP Buffer의 크기가 해당 영역을 사용하는 Segment들의 크기보다 작다면 KEEP Buffer 영역에서 경합이 발생하고, 그로 인해 Physical I/O가 발생하여 Cache Hit Ratio가 떨어질 수 있다. 이때 KEEP Buffer 의 크기를 늘려주거나 중요도가 떨어지는 Segment의 KEEP Buffer 사용을 막는 방안을 고려 해 볼 수 있다. 반대로 KEEP Buffer의 크기가 Segment들의 크기보다 많이 크다면, 사용하지 않는 메모리 공간을 차지하고 있는 것이므로 KEEP Buffer의 크기를 줄이는 것을 고려 해 볼 수 있다. 따라서 시스템 성능과 Segment들의 중요도에 따라 효율적인 KEEP Buffer 의 크기 조절이 필요하다.

다음 스크립트로dba_hist_seg_stat 뷰를 조회하여 Segment 조회시 발생하는 I/O 발생량에 대한 AWR 정보를 확인 하여 Segment별 효율성을 판단 할 수 있다.

4. Keep Buffer 효율성 판단 (3/4)

Segment I/O SQLScript (1/2)

```
accept i_begin_time prompt 'Enter begin time[YYYYMMDDHH24]: '
accept i_end_time   prompt 'Enter end   time[YYYYMMDDHH24]: '

variable v_begin_time char(10)
variable v_end_time   char(10)

exec :v_begin_time:=&i_begin_time
exec :v_end_time  :=&i_end_time

SELECT /*+ leading(k) */
       s.dbid ,
       decode( SUBSTR( o.object_type , 1 , 5 ) , 'TABLE' , o.object_name , 'INDEX' , (
                                                  SELECT table_name
                                                  FROM   dba_indexes
                                                  WHERE  index_name = o.object_name
                                                  AND    owner = k.owner
                                                 ) ) AS table_name ,
       decode( SUBSTR( o.object_type , 1 , 5 ) , 'INDEX' , o.object_name ) AS index_name ,
       s.snap_id ,
       TO_CHAR( w.begin_interval_time , 'yyyymmdd.hh24' ) AS begin_time ,
       s.physical_reads_delta ,
       s.physical_reads_direct_delta ,
       s.physical_reads_delta + s.physical_reads_direct_delta AS total_diskio
FROM   sys.wrm$_snapshot w ,
       dba_hist_seg_stat s ,
       dba_objects o ,
       (
        SELECT owner ,
               segment_name
        FROM   dba_segments
```

4. Keep Buffer 효율성 판단 (4/4)

Segment I/O SQLScript (2/2)

```
WHERE  buffer_pool = 'KEEP'
       ) k
WHERE  w.begin_interval_time >= to_timestamp( '2013062510' , 'yyyymmddhh24' )
AND    w.end_interval_time <= to_timestamp( '2013062518' , 'yyyymmddhh24' )
AND    w.snap_id = s.snap_id
AND    w.dbid = s.dbid
AND    w.instance_number = s.instance_number
AND    s.obj# = o.object_id
AND    k.segment_name = o.object_name
AND    k.owner = o.owner
ORDER  BY  2 , 3 , 5
```

5. 결론

KEEP Buffer를 사용하는데 있어 가장 중요한 것이 업무의 반영일 것이다.
자주 사용하는 Object 들만 상주하는 Buffer Cache 하나만 사용하는 것이 시스템 전체의 관점에서 보면 효율적일 수도 있다. 하지만 업무의 중요성이나 특성을 고려한다면 다른 결과가 나올 수 있다. 적게 실행 되더라도 중요도가 높은 업무가 있을 수 있고, 수행시간 단축이 매우 중요한 업무가 있을 수 있다. 이러한 업무에 대한 특성을 반영한 운영계획을 세우는데 있어서, KEEP Buffer를 효율적으로 사용 할 수 있다면, 시스템 성능 향상에 큰 도움이 될 것이다.