International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 5 Issue 1, November-December 2020 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Oracle in Memory Configuration

Vijay Kumar Tiwari

IT Consultant, HCL America Inc, Texas, United States

How to cite this paper: Vijay Kumar Tiwari "Oracle in Memory Configuration" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-5 | Issue-1, December 2020, pp.1465, URL: www.ijtsrd.com/papers/ijtsrd38275.pdf

IJTSRD38275

Copyright © 2020 by author (s) and International Journal of Trend in Scientific Research and Development Journal. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0) (http://creativecommons.org/licenses/by/4.0)

INTRODUCTION:

Oracle Database In-Memory provides a unique dual-format design that enables tables to be instantaneously represented in memory using traditional row format and a new in-memory column format. The Oracle SQL Optimizer by design routes analytic queries to the column format and OLTP queries to the row format, transparently delivering best-of-both-worlds performance. Oracle Database automatically maintains full transactional uniformity between the row and the column formats, just as it maintains uniformity between tables and indexes today. The new column format is a pure in-memory format and is not persistent on disk, so there are no additional storage costs or storage synchronization issues.

How to configure

show parameter in memory;

- The size of the IM column store is controlled by the INMEMORY_SIZE parameter.

If INMEMORY_SIZE parameter is set to 0 it means Database In-Memory is not enabled, as there is no IM column store allocated.

We can also check the In-Memory Area by querying v$SGA.

- Enabling Database In-Memory is actually a multi-step process.

A. First we must allocate memory for the IM column store by setting the INMEMORY_SIZE parameter to a non-zero value that is greater than 100MB.

i.e.

ALTER SYSTEM SET inmemory_size = 20G scope=spfile sid=*;

B. bounce the database so these parameter changes can take effect.

C. confirm:

select name, value from v$sga;

show parameter inmemory;

D. We now should have an IM column store. But Database In-Memory is still not in use because no objects have been populated into the IM column store.

To confirm this you can look at two new v$ views, v$IM_SEGMENTS and v$IM_USER_SEGMENTS that indicate what objects are in the In-Memory Column Store.

The IM column store we should be populated with the most performance-critical data in the database.

Less performance-critical data can reside on lower cost flash or disk. Of course, if your database is small enough, you can populate all of your tables into the IM column store.

Only objects with the INMEMORY attribute are populated into the IM column store.

The INMEMORY attribute can be specified on a tablespace, table, (sub)partition, or materialized view.

Queries to find the inmemory usage

```
select * from gv$sga;
select * from gv$im_segments;
select * from gv$im_user_segments;
select sum(inmemory_size)/1024/1024 INMEMORY_SIZE_MB from gV$IM_SEGMENTS;
select sum(alloc_bytes)/1024/1024 ALLOC_MB, sum(used_bytes)/1024/1024 USED_MB from gv$inmemory_area;

select INST_ID, POOL, ALLOC_BYTES/1024/1024 ALLOC_MB, USED_BYTES/1024/1024 USED_MB, POPULATE_STATUS from GV$INMEMORY_AREA;
```

Conclusion In short:

Analytics queries/workload runs in column store while transaction runs against the Buffer cache.

Optimizer decide by his own when to use what. In memory Advisor can be installed to get more detailed Picture.