ORACLE TECHNOLOGY NETWORK

Virtual Technology Summit

Hands-On Learning With Oracle and Community Experts

Where Technology and Community Meet

InMemory: A User Success Story

Arup Nanda
Nov 10, 2014

Please Stand By. This session will begin promptly at the time indicated on the agenda. Thank You.

ORACLE

Thank You for Joining Us Today

ORACLE

The Conundrum of Index Access

Why performance tuning often is a self defeating proposition

Not current!

Space consumption

COL1	COL2	COL3	COL4
123	ABC	X	Y
234	DEF	Y	Z
345	GHI	Z	A

Index

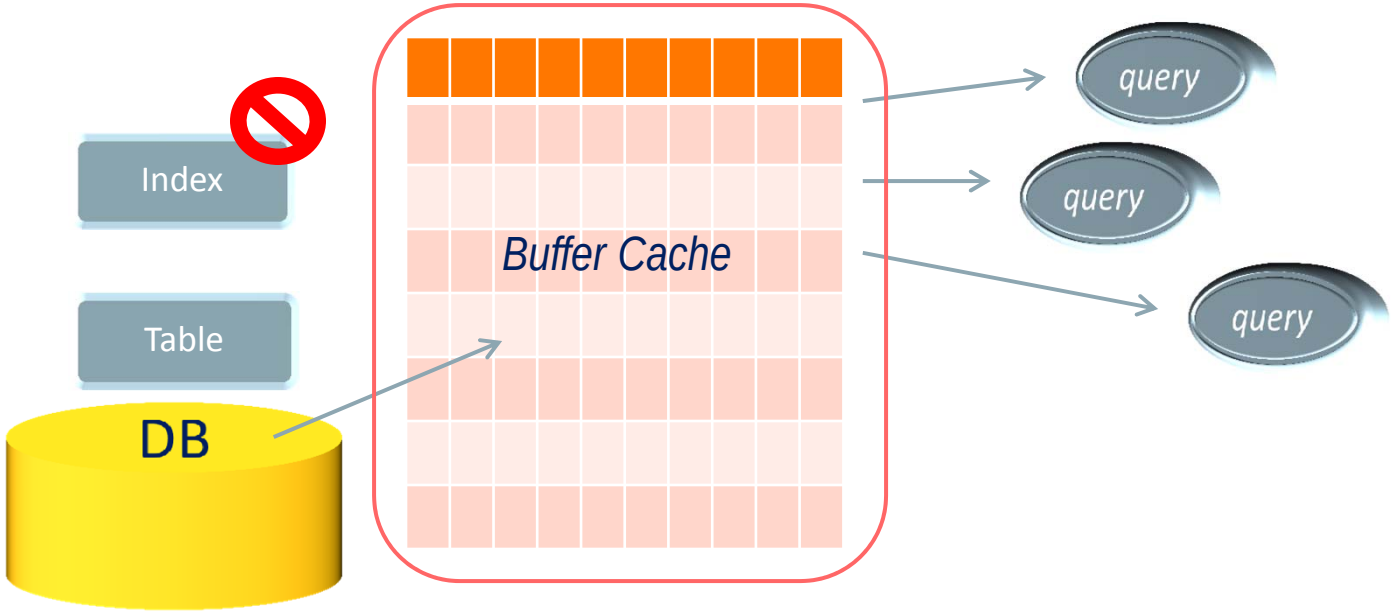
Index

Index

Slows down DML processing!

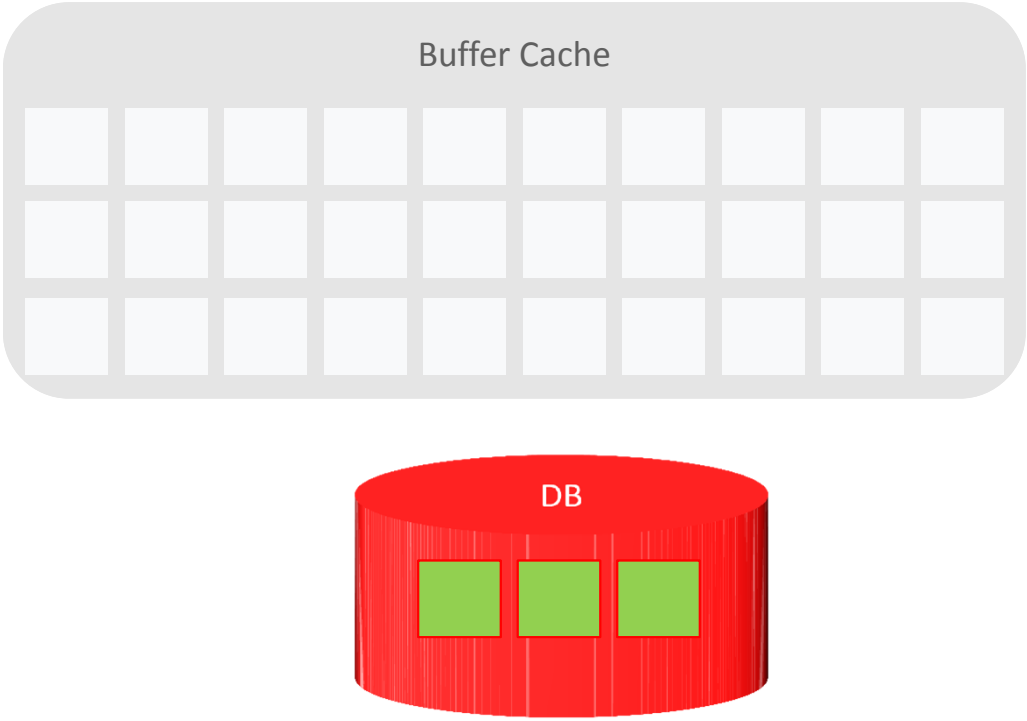
Space consumption

Accessing Without Indexes

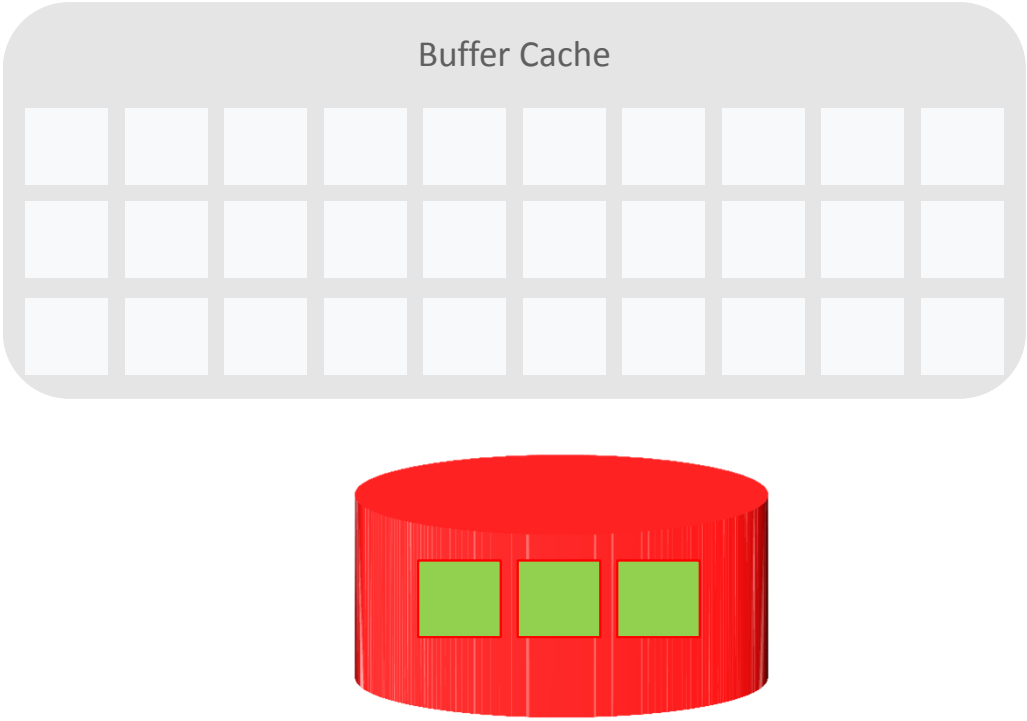


Need Memory *Where*?

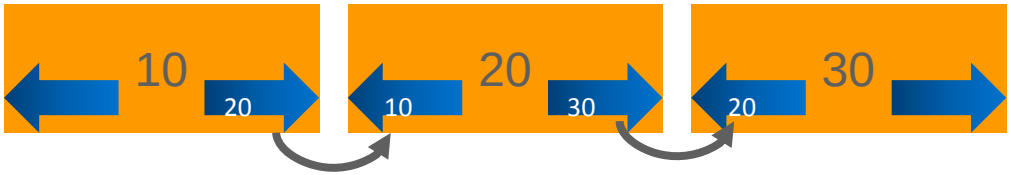
Why a bigger buffer cache is *not* the answer



Buffer Transfer

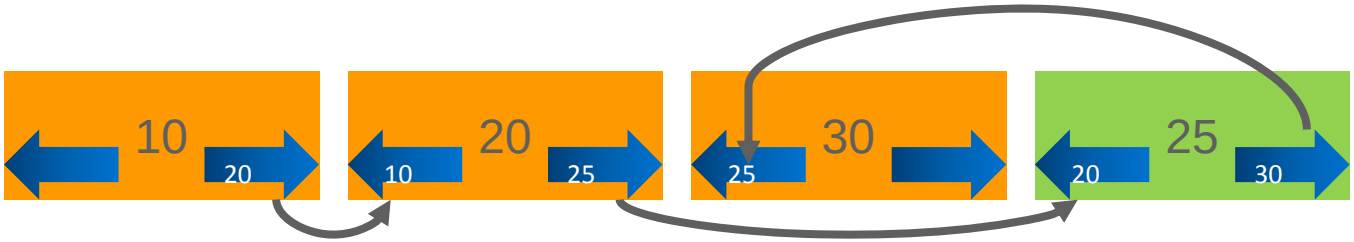


Buffer Linked Lists



Each buffer has a prior and a next pointer that shows the next buffer in this list. This is known as a linked list.

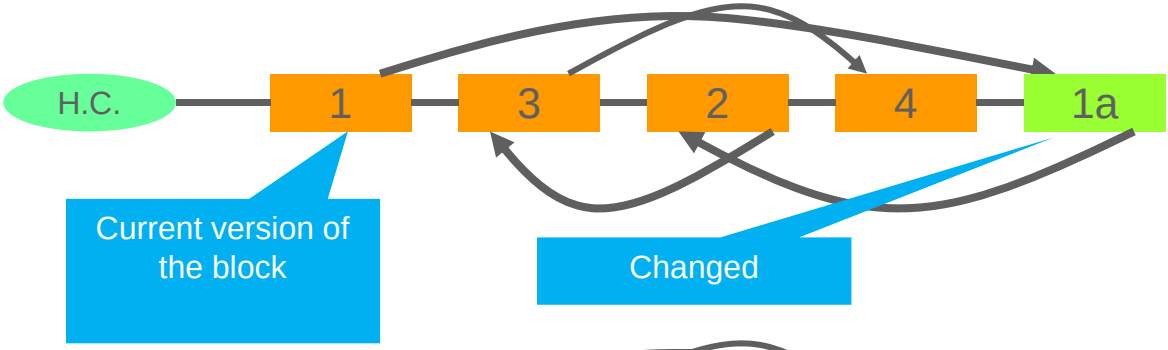
Hash Chains



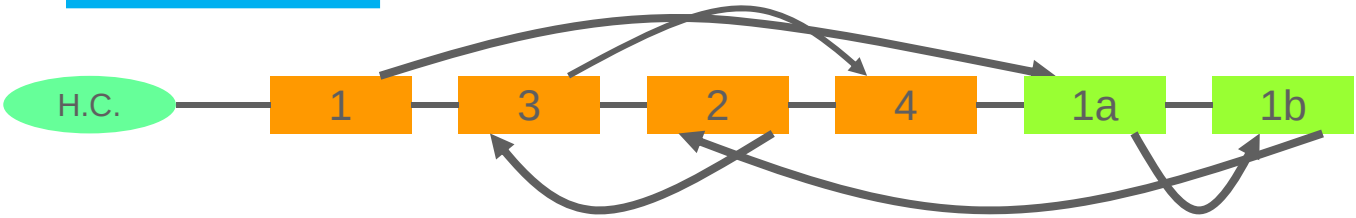
When a new buffer comes in, only the pointers are updated

Multi-versioning

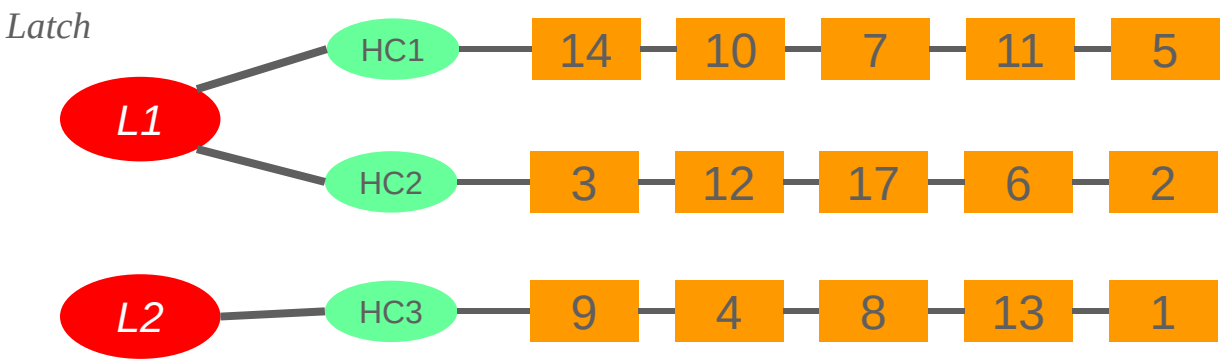
Buffer 1 was updated



Buffer 1 was updated once more



CBC Latch

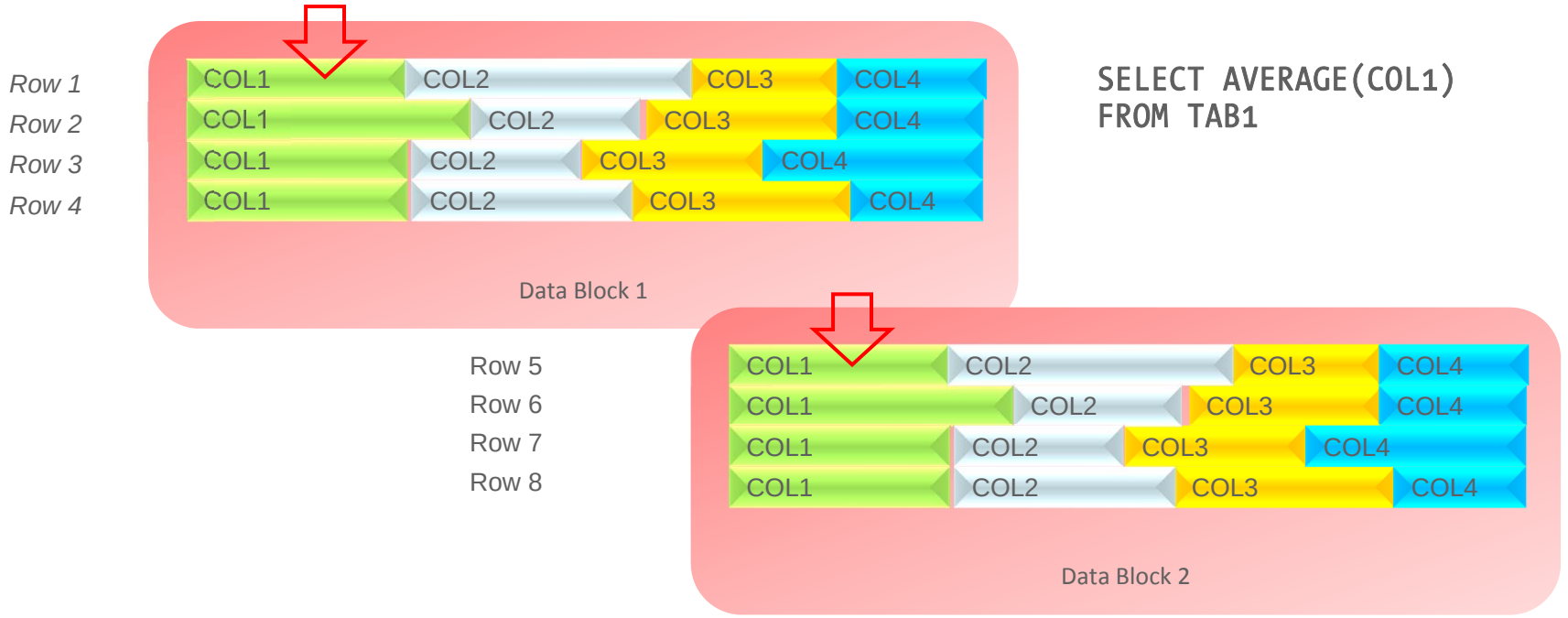


No. of hash buckets = init.ora parameter `_db_block_hash_buckets`
No. of latches = `_db_block_hash_latches`

Alternatives to Buffer Cache

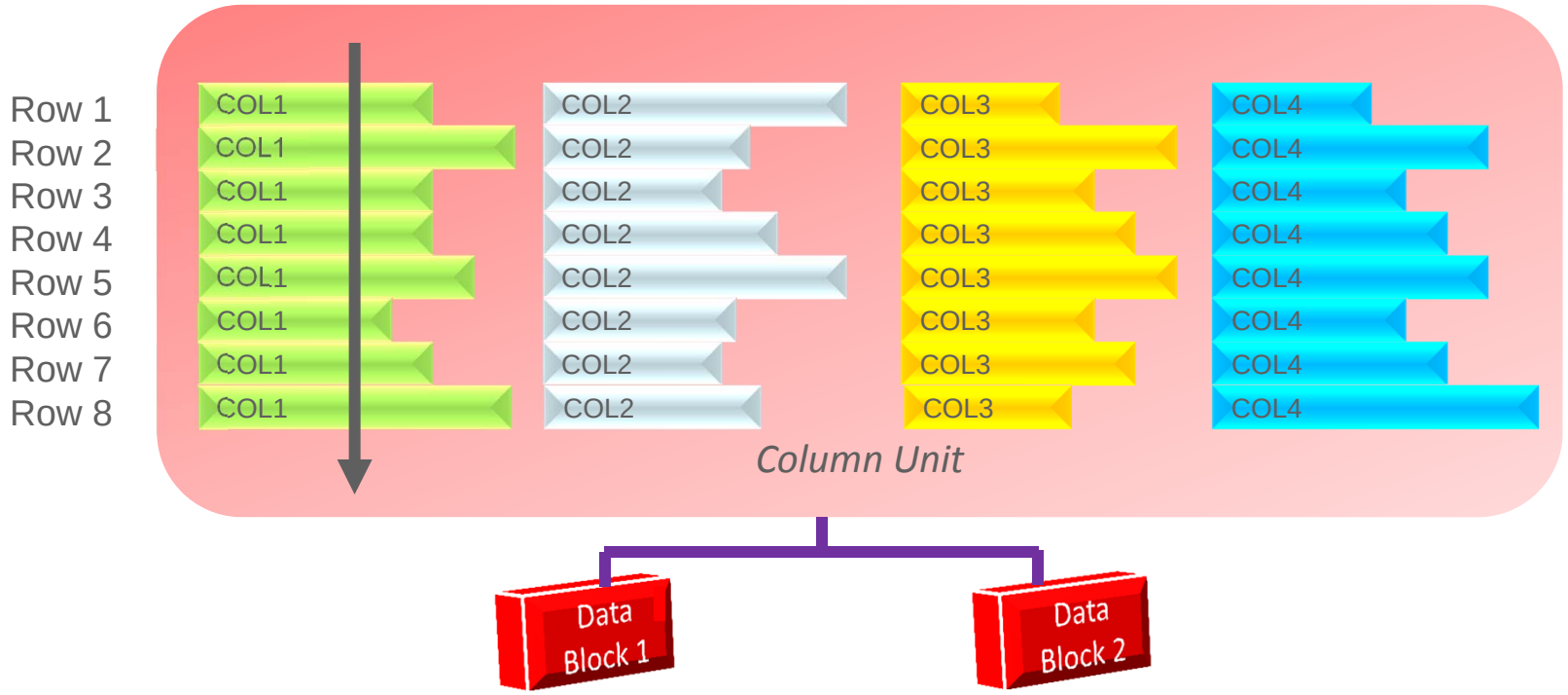
- **Result Cache**
  - Results of queries cached in shared pool
  - Gives results directly instead of running the query again
  - Re-executes the query if the underlying data changes
- **Function Result Cache**
  - Stores the output of PL/SQL functions
  - With the same predicates, gives the results without re-execution
- **Smart Flash**
  - Stores data blocks in intermediate storage
  - Specific platforms: Exadata, ZFS, etc.
  - Provides results for SELECT queries
  - Not for writes

Traditional Database Store

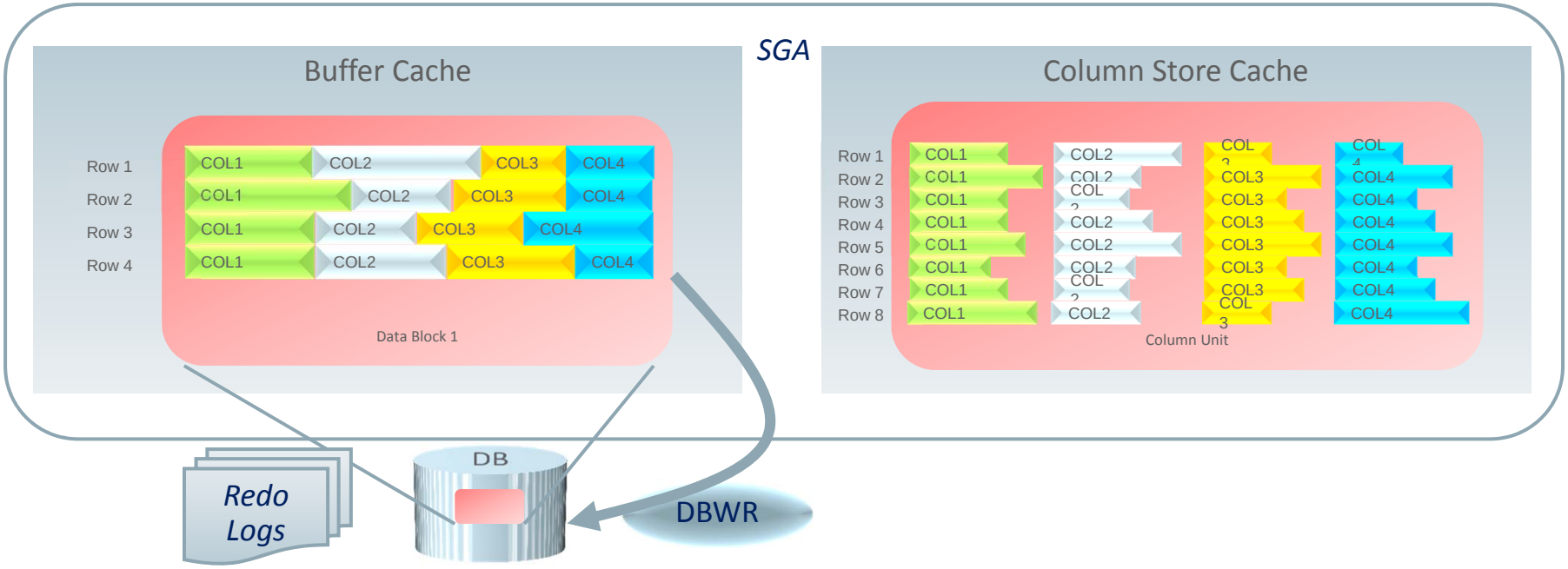



```
SELECT AVERAGE(COL1)
FROM TAB1
```

Column Store Database



Column Store Cache



Column Store Cache

- What is it?
  - Another memory area in SGA
  - Defined by a parameter INMEMORY_SIZE
  - Separate from buffer cache
- How is it populated?
  - Only selected segments go there

```
SQL> alter table TableName inmemory;
```

- Population is done by new background processes.
  - Coordinator: IMCO
  - The actual data population is done by two new background processes—SMCO and Wnnn.
- Manual deletion from this store
  - No automatic LRU algorithm

Population Options

ALTER T1 INMEMORY [PRIORITY p] [MEMCOMPRESS FOR c]

Subclause PRIORITY

- CRITICAL, HIGH, MEDIUM: priority of loading into InMemory store
- NONE: not automatically populated

Compression

- Save on memory space
- Options
  - FOR DML
  - FOR QUERY LOW (the default)
  - FOR QUERY HIGH
  - FOR CAPACITY LOW
  - FOR CAPACITY HIGH
- NOMEMCOMPRESS

Demo

1. Download the example schemas for your platform from OTN
   http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html
2. Install the example schemas, especially the SH schema, in any Oracle Database 12.1.0.2
3. Download the demo scripts for this session from XXX
4. Log on to the database as SH user
5. Execute the scripts shown in the slides

Putting a Partitioned Table into InMemory

Each partition can have different property

```
alter table costs modify partition COSTS_Q4_2003 inmemory no memcompress priority high
/
alter table costs modify partition COSTS_Q1_2003 inmemory memcompress for query priority high
/
alter table costs modify partition COSTS_Q1_2002 inmemory memcompress for capacity low priority high
/
alter table costs modify partition COSTS_Q4_2001 inmemory memcompress for capacity high priority high
/
alter table costs modify partition COSTS_Q1_2000 inmemory memcompress for capacity high priority medium
/
```

alt1.sql

Checking for InMem Attrib of Segments

```
select partition_name, inmemory_priority, inmemory_distribute, inmemory_compression
from user_tab_partitions
where table_name = 'COSTS'
order by 1;

PARTITION_NAME  INMEMORY INMEMORY_DISTRI INMEMORY_COMPRESS
--------------- -------- --------------- -----------------
COSTS_1995      NONE     AUTO            FOR CAPACITY HIGH
COSTS_1996      NONE     AUTO            FOR CAPACITY HIGH
COSTS_H1_1997   NONE     AUTO            FOR CAPACITY HIGH
COSTS_H2_1997
COSTS_Q1_1998   NONE     AUTO            FOR CAPACITY HIGH
COSTS_Q1_1999   NONE     AUTO            FOR CAPACITY HIGH
… truncated for brevity …
```

part1.sql

Checking for Segments in Memory

```
select partition_name, populate_status, bytes_not_populated "Not_Pop",
      bytes, inmemory_size, bytes/inmemory_size comp_ratio
FROM   v$im_segments;

PARTITION_NAME  POPULATE_S Not_Pop      BYTES INMEMORY_SIZE COMP_RATIO
--------------- ---------- ------- ---------- ------------- ----------
COSTS_Q3_2000   COMPLETED        0    8388608       1179648 7.11111111
COSTS_Q4_2001   COMPLETED        0    8388608       1179648 7.11111111
COSTS_Q1_2001   COMPLETED        0    8388608       1179648 7.11111111
```

vim1.sql

Checking SGA

During Startup

```
SQL> startup

ORACLE instance started.

Total System Global Area 2600468480 bytes
Fixed Size                 2927768 bytes
Variable Size            402654056 bytes
Database Buffers          33554432 bytes
Redo Buffers              13848576 bytes
In-Memory Area          2147483648 bytes
```

Running Instance

```
SQL> select * from v$sgainfo;

NAME                          BYTES RES     CON_ID
---------------------- ----------- --- ----------
Fixed SGA Size               2929552 No           0
Redo Buffers                30621696 No           0
Buffer Cache Size         1509949440 Yes          0
In-Memory Area Size       1073741824 No           0
Shared Pool Size           436207616 Yes          0
Large Pool Size            150994944 Yes          0
…
```

Test for Plan in InMemory

```
Elapsed: 00:00:06.25
Plan hash value: 4155192936
-----------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name     | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |          |       |       |    97 (100)|          |       |       |
|*  1 |  FILTER                            |          |       |       |            |          |       |       |
|   2 |   SORT GROUP BY                    |          |  3665 |   168K|    97   (5)| 00:00:01 |       |       |
|*  3 |    HASH JOIN                       |          | 82112 |  3768K|    94   (2)| 00:00:01 |       |       |
|   4 |     TABLE ACCESS FULL              | PRODUCTS |    72 |  2160 |     3   (0)| 00:00:01 |       |       |
|   5 |     PARTITION RANGE ALL            |          | 82112 |  1363K|    91   (2)| 00:00:01 |     1 |    28 |
|   6 |      TABLE ACCESS INMEMORY FULL    | COSTS    | 82112 |  1363K|    91   (2)| 00:00:01 |     1 |    28 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("UNIT_PRICE")/COUNT("UNIT_PRICE")>1000)
   3 - access("P"."PROD_ID"="C"."PROD_ID")
```

Comparison of Execution

```
alter session set inmemory_query = disable;

select /*+ gather_plan_statistics */ prod_name, time_id, avg(unit_price) avg_cost
from costs c, products p
where p.prod_id = c.prod_id
group by  prod_name, time_id
having avg(unit_price) > 1000
order by  prod_name, time_id
/
select * from
table(dbms_xplan.display_cursor(format=>'TYPICAL +allstats'))
/
```

q1.sql
dis.sql
q1.sql

Plan without InMem

```
Plan hash value: 4155192936

-------------------------------------------------------------------------------------------------------
| Id  | Operation              | Name     | E-Rows |E-Bytes| Cost (%CPU)| E-Time   | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |          |   3665 |   168K|   141   (3)| 00:00:01 |       |       |
|*  1 |  FILTER                |          |        |       |            |          |       |       |
|   2 |   SORT GROUP BY        |          |   3665 |   168K|   141   (3)| 00:00:01 |       |       |
|*  3 |    HASH JOIN           |          |  82112 |  3768K|   139   (1)| 00:00:01 |       |       |
|   4 |     TABLE ACCESS FULL  | PRODUCTS |     72 |  2160 |     3   (0)| 00:00:01 |       |       |
|   5 |     PARTITION RANGE ALL|          |  82112 |  1363K|   136   (1)| 00:00:01 |     1 |    28 |
|   6 |      TABLE ACCESS FULL | COSTS    |  82112 |  1363K|   136   (1)| 00:00:01 |     1 |    28 |
-------------------------------------------------------------------------------------------------------
```

Actual Rows in Plans

With InMemory

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	Pstart	Pstop	A-Rows	A-Time	Buffers	OMem
0	SELECT STATEMENT		2			97 (100)				4452	00:00:00.30	220	
* 1	FILTER		2							4452	00:00:00.30	220	
2	SORT GROUP BY		2	1802	84694	97 (5)	00:00:01			72066	00:00:00.36	220	3809K
* 3	HASH JOIN		2	82112	3768K	94 (2)	00:00:01			182K	00:00:00.05	220	1185K
4	TABLE ACCESS FULL	PRODUCTS	2	72	2160	3 (0)	00:00:01			144	00:00:00.01	8	
5	PARTITION RANGE ALL		2	82112	1363K	91 (2)	00:00:01	1	28	182K	00:00:00.01	212	
6	TABLE ACCESS INMEMORY FULL	COSTS	56	82112	1363K	91 (2)	00:00:01	1	28	182K	00:00:00.02	212	

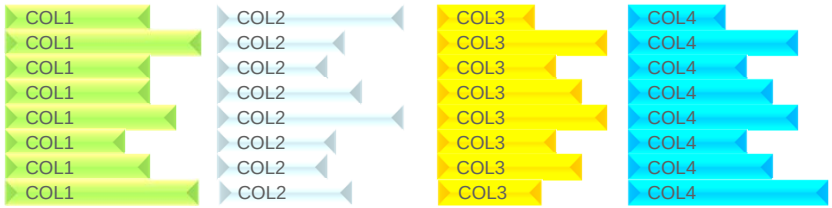
Without InMemory

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	E-Time	Pstart	Pstop	A-Rows	A-Time	Buffers	OMem
0	SELECT STATEMENT		2			141 (100)				4452	00:00:00.32	758	
* 1	FILTER		2							4452	00:00:00.32	758	
2	SORT GROUP BY		2	1802	84694	141 (3)	00:00:01			72066	00:00:00.38	758	3809K
* 3	HASH JOIN		2	82112	3768K	139 (1)	00:00:01			182K	00:00:00.07	758	1185K
4	TABLE ACCESS FULL	PRODUCTS	2	72	2160	3 (0)	00:00:01			144	00:00:00.01	8	
5	PARTITION RANGE ALL		2	82112	1363K	136 (1)	00:00:01	1	28	182K	00:00:00.03	750	
6	TABLE ACCESS FULL	COSTS	56	82112	1363K	136 (1)	00:00:01	1	28	182K	00:00:00.02	750	

Individual Columns

No need to place the entire table in InMem. Only a handful of columns can go too.

```
alter table t1 inmemory
memcompress for capacity low (c2,c3)
no inmemory (c4,c5);
```



InMemory in Multitenant

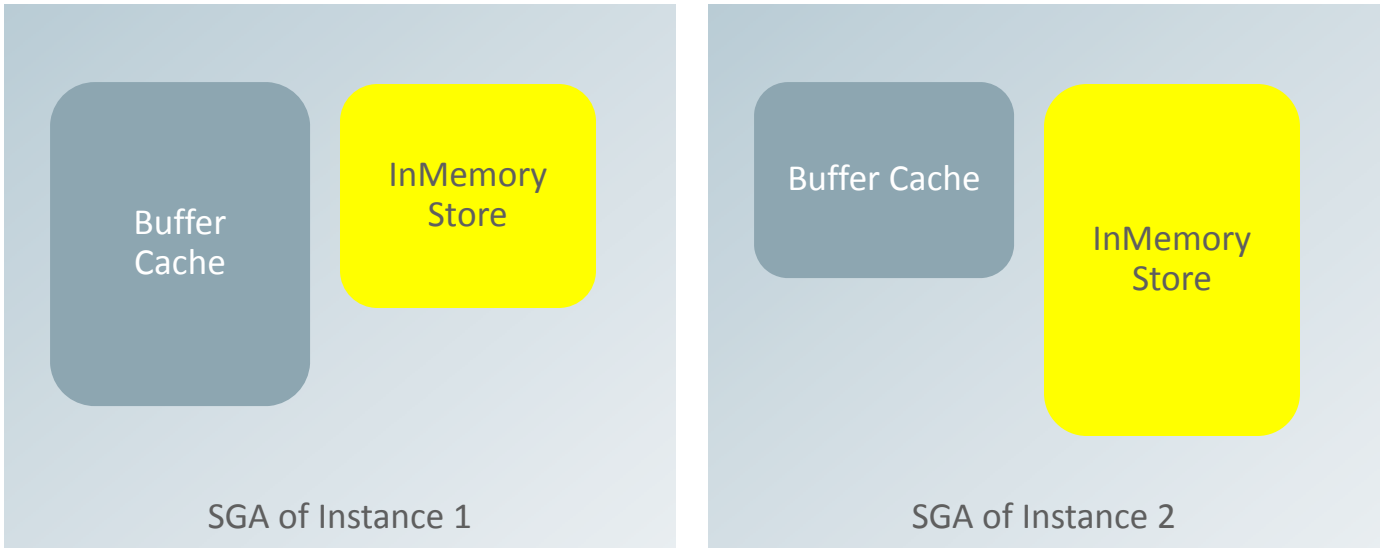
Each PDB can have a different INMEMORY_SIZE

InMemory Store Divisions
- 64KB Pool = metadata
- 1MB Pool = actual data

```
select con_id, pool, alloc_bytes, used_bytes, populate_status
from  v$inmemory_area
/

CON_ID POOL                      ALLOC_BYTES USED_BYTES POPULATE_STATUS
---------- ------------------------- ----------- ---------- -------------------------
         1 1MB POOL                    854589440   17825792 DONE
         1 64KB POOL                   201326592    2228224 DONE
         2 1MB POOL                    854589440   17825792 DONE
         2 64KB POOL                   201326592    2228224 DONE
         3 1MB POOL                    854589440   17825792 DONE
         3 64KB POOL                   201326592    2228224 DONE
```

InMem in RAC

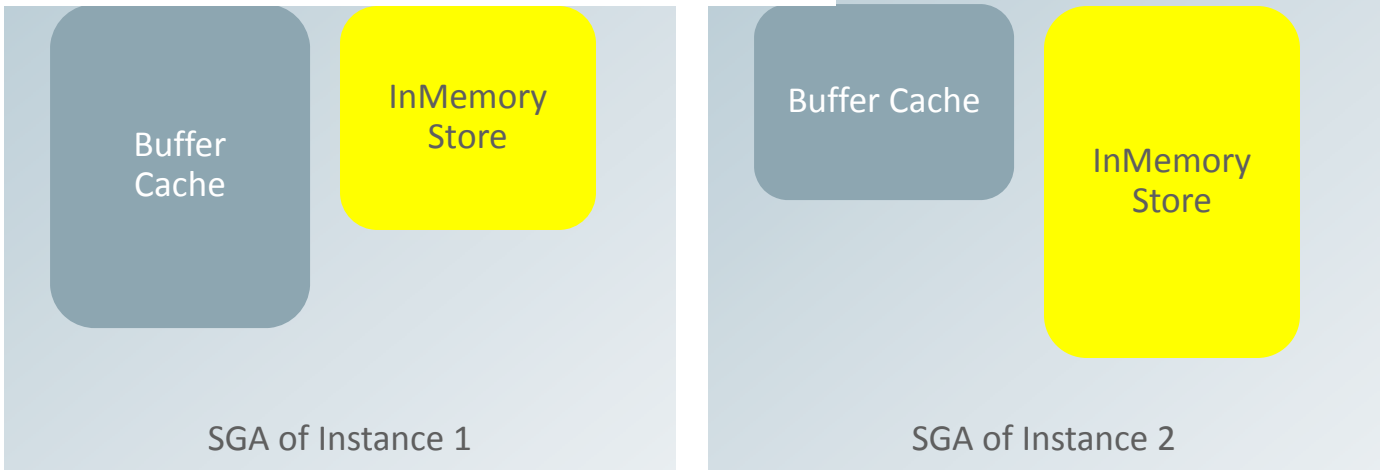


InMem in RAC: Distribution

```
alter table …
inmemory priority none
memcompress for capacity high
distribute auto no duplicate
```

- AUTO
- BY ROWID RANGE
- BY ROWID PARTITION
- SUBPARTITION

- DUPLICATE
- NO DUPLICATE



InMem –vs- Other Caching Technologies

How is InMem Store different from Result Cache and Smart Flash

Result Cache

- Computed results of a query
- Resident in Shared Pool
- Query specific. A different query on the same table may not use RC

InMem Store

- Stored data from a table or partition
- Not query specific
- Any query on that table/partition will use it

Smart Flash

- Blocks as found in database

InMem Store

- Columnar store
- Can be compressed

How to Use InMem

Check Segment Use

```
select owner, object_name, subobject_name, value
from v$segment_statistics
where statistic_name = 'logical reads'
and value > 0
and owner not in ('SYS','SYSTEM')
order by 4 desc,1,2,3;

OWNER           OBJECT_NAME          SUBOBJECT_NAME                VALUE
--------------- -------------------- -------------------- ------------
SH              COSTS                COSTS_Q4_2001                208
SH              COSTS                COSTS_Q1_2001                176
SH              COSTS                COSTS_Q2_2000                160
SH              COSTS                COSTS_Q3_2001                128
…
```

seg1.sql

How Much Compression?

Use Compression Advisor

```
set serveroutput on size 99999
declare
        l_blkcnt_cmp PLS_INTEGER;
        l_blkcnt_uncmp PLS_INTEGER;
        l_row_cmp PLS_INTEGER;
        l_row_uncmp PLS_INTEGER;
        l_cmp_ratio PLS_INTEGER;
        l_comptype_str VARCHAR2(100);
        comp_ratio_allrows NUMBER := -1;
BEGIN
        dbms_compression.get_compression_ratio (
                ownname         => 'SH',
                objname         => 'CUSTOMERS',
                subobjname      => null,
                scratchtbsname  => 'USERS',
                comptype        =>
dbms_compression.comp_inmemory_query_high,
                blkcnt_cmp      => l_blkcnt_cmp,
                blkcnt_uncmp    => l_blkcnt_uncmp,
                row_cmp         => l_row_cmp,
                row_uncmp       => l_row_uncmp,
                cmp_ratio       => l_cmp_ratio,
                comptype_str    => l_comptype_str,
                subset_numrows  =>
dbms_compression.comp_ratio_allrows
);
dbms_output.put_line('Comp Ratio='|| l_cmp_ratio);
END;
/
```

comp1.sql

Takeaways

1. Identify segments candidates for InMemory Cache
2. Identify possible compression for each segment
3. Use DBMS_COMPRESSION.GET_COMPRESSION_RATIO to predict compressed size
4. Determine the total compressed size
5. Set the InMemory_Size parameter in the instance (recycle required)
6. Set the InMemory properties for each segment
7. Monitor the usage in V$IM_SEGMENTS
8. Make the indexes invisible on those tables, check the query plan
9. Adjust the compression properties
10. Adjust the InMemory Store size, if needed

Lessons Learned

1. Segments < 64KB do not go to InMem store
2. Use DISTRIBUTE BY PARTITION for Hash Partitioned Tables in RAC
3. Do not enable DUPLICATE. Instead run analytic queries in individual nodes using a special service
4. DUPLICATE option is only for engineered systems
5. If CPU consumption goes high, you can reduce the maximum amount of worker processes by changing the parameter `inmemory_max_populate_servers`
6. Resource Manager

```
dbms_resource_manager.set_consumer_group_mapping (
  attribute      => 'ORACLE_FUNCTION',
  value          => 'INMEMORY',
  consumer_group => 'DEFAULT_CONSUMER_GROUP'
);
```

res1.sql

Why Do It?

1. No need to decide between row and column store. Choose both.
2. Ridiculously simple to implement
3. Makes queries faster without building new indexes
4. No need for additional structures such as Materialized Views
5. Allows running analytical queries in OLTP environment

Q+A