Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

ORACLE

ORACLE SUPPORT

Oracle 12c: In-Memory Database Tips with Linux on IBM z Systems™

Rhoda Pereira, Principal Support Engineer, Oracle
Chris Crocker Principal Support Engineer, Oracle
David Simpson, Oracle Technical Specialist, IBM

April - 2016

ORACLE

Agenda

1. What is In-Memory Option?
2. When to Use In-Memory option
3. How to use In-Memory option
4. Experiences with In-Memory option with IBM z Systems
5. Testing, Results and Issues with In-Memory option

ORACLE

Oracle Database In Memory 12.1.0.2

- Overview
- When to use it?
  - Benefits
  - When NOT to use it?
- Getting Started in Using In-Memory option
- Q&A

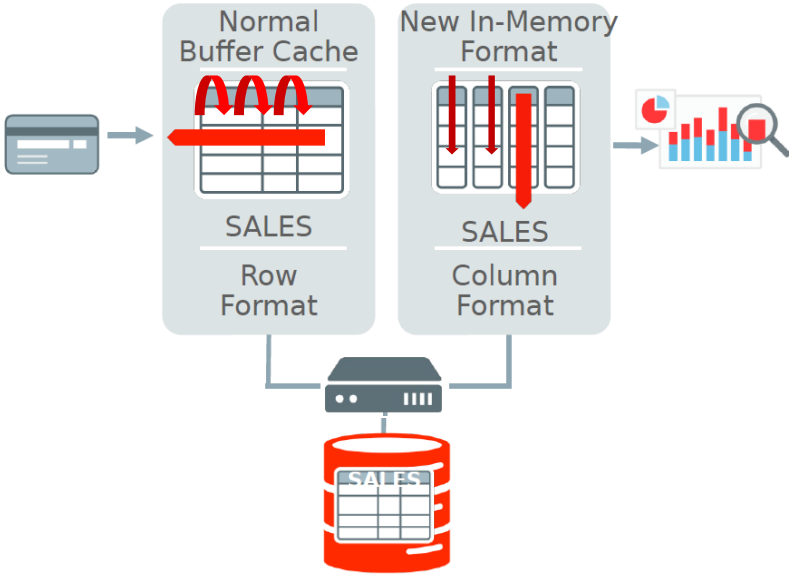
ORACLE

Overview

- Highly optimized In-Memory Column store (aka IM column store)
  - Introduced in version 12.1.0.2
  - It is optional, static SGA pool
  - stores copies of tables and partitions in a columnar format
    - Accelerate column-oriented data access
  - The columnar format exists only ‘in memory’
  - Only for objects created with `INMEMORY` are candidates

ORACLE

Row-Format vs Column Format Data Access



Buffer Cache

- Row format

- Each column is being read

In-Memory

- Stores and access the data in columns

Oracle Database In-Memory Goals

- Real Time Analytics
- Accelerate Mixed Workload OLTP
- No Changes to Applications
- Trivial to Implement

ORACLE

Database objects that can use In-Memory Column Store:

- Tables*
- Materialized Views*
- Partitions*
  - * can store all or subset of its columns

- Tablespaces
  - Enables all tables and materialized views

ORACLE

Database objects CANNOT use In-Memory Column Store:

- Objects owned by SYS user → System & SYSAUX
- Index Organized Tables (IOTs)
- Clustered tables

- Datatypes
  - LONGS
  - Out of Line LOBS

ORACLE

CANNOT use In-Memory Column Store:

- Active Data Guard standby instance

- BUT CAN BE USED
  - Logical Standby Instance
  - Instance maintained by Oracle Golden Gate

ORACLE

In-Memory Column Store is good for..

- Speeds up **analytic data access**
- Scanning rows and applying filters that use the following operators

  `=, <, >,` and `IN`
  - Querying a subset in a table
  - Contain complex calculations and aggregations
  - Accelerating joins

ORACLE

It doesn't help with…

- Data Loading
- Logging on and off the database
- Parsing
- PL/SQL

ORACLE

It does not improve performance:

- Queries with complex predicates
- Queries that select a large number of columns
- Queries that return a large number of rows
- Queries with multiple large table joins

ORACLE

What will benefit In-Memory column store?

BENEFIT MORE:

- Business Applications
- Ad-hoc analytic queries
- Data Warehouse workloads

BENEFIT LESS:

- Pure OLTP databases perform short transactions using index

ORACLE

Populating the In-Memory Column Store

- Database In-Memory adds a new **INMEMORY** attributes for tables and materialized views

- **NO INMEMORY** clause to disable in-memory attribute

- Populated by background process – *worker process* (*ora_w001_orcl*)

ORACLE

`INMEMORY` can be used:

- CREATE/ALTER TABLE
- CREATE/ALTER TABLESPACE
- CREATE/ALTER MATERIALIZE VIEWS

ORACLE

Population...

- Streaming mechanism → In-Memory Compression Units
- Simultaneously columnizing → PRIORITY LEVEL
- Compressing the data → In-Memory Compression AKA In-Memory Columnar Expression

IM Column Store Population Options

- Instance Startup

  reconstruct entire memory everytime the database instance starts

- Specify a priority level using `INMEMORY PRIORITY` clause

ORACLE

How population is controlled?

PRIORITY	DESCRIPTION
CRITICAL	Object is populated immediately after the database is opened
HIGH	Object is populated after all `CRITICAL` objects have been populated, if space remains available in the IM column store
MEDIUM	Object is populated after all `CRITICAL` and `HIGH` objects have been populated, and space remains available in the IM column store
LOW	Object is populated after all `CRITICAL`, `HIGH`, and `MEDIUM` objects have been populated, if space remains available in the IM column store
NONE	Objects only populated after they are scanned for the first time (Default), if space is available in the IM column store

ORACLE

In-Memory Compression Unit (IMCU)

- objects populated in the column store is made up of IMCU
- Each IMCU contains the column entries for a subset of rows in the object
- Size depends on datatypes

ORACLE

In-Memory Compression Unit (IMCU)

- Contains the column entries for subset of rows
- Associated with a transaction journal → Used to keep the column store transactionally consistent
- Read the same order it appears in the row format

ORACLE

In-Memory Columnar Compression

- Special compression formats optimized for access speed rather than storage reduction.
  - Decrease the amount of memory processed for each column
  - Database used SIMD vector (array) in a single CPU clock cycle

ORACLE

Single Instruction processing Multiple Data values (SIMD)

- When data does not need to be scanned in IM column store
- Allows a set of column values to be evaluated in single CPU instruction
- Enables Oracle db In-Memory to scan billions of rows per second
- Memcompress for Query High
- Memcompress for Capacity Low
- Memcompress for Capacity High

For full description refer:
http://docs.oracle.com/database/121/ADMIN/memory.htm#BABGFGBD

ORACLE

Compression Methods

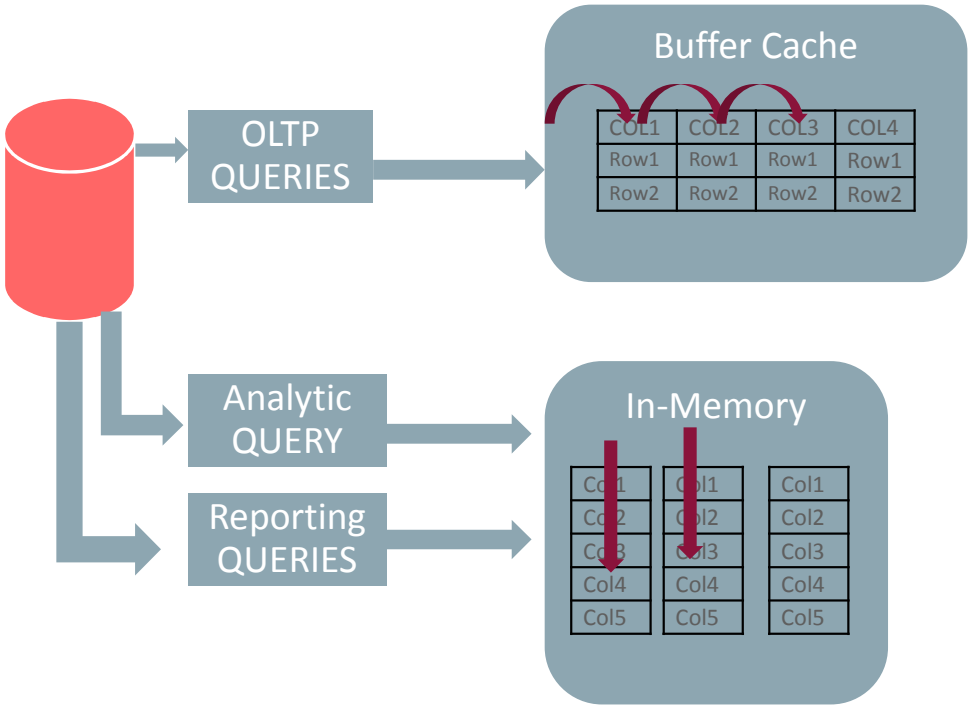
- No Memcompress
- Memcompress for DML
- Memcompress for Query Low - Default – Best for query performance
- Memcompress for Query High
- **Memcompress for Capacity Low**
- Memcompress for Capacity High

For full description refer:
http://docs.oracle.com/database/121/ADMIN/memory.htm#BABGFGBD

ORACLE

Dual Memory Formats: Columnar & Row

- Oracle database can read
  - Buffer cache - or-
  - In-Memory -or-
  - BOTH



Population In-Memory Column using RAC

- Distributed among the database instances
- `NO DUPLICATE` specifies to maintain only one copy
- `DUPLICATE ALL` each in-memory available in all database instances
- `DUPLICATE` maintain two copies of an object in different instances

ORACLE

Putting into action

```
oracle@strkf32:~> sqlplus / as sysdba
   SQL*Plus: Release 12.1.0.2.0 Production on Tue Aug 5
   06:19:08 2014
   Copyright (c) 1982, 2014, Oracle.  All rights
   reserved.
   Connected to:
   Oracle Database 12c Enterprise Edition Release
   12.1.0.2.0 - 64bit Production
   With the Partitioning, Automatic Storage Management,
   OLAP, Advanced Analytics
   and Real Application Testing options
```

Oracle Database In Memory 12.1.0.2
How to check:

- By default In Memory Column Store is disabled:

```
SQL> show parameter inmemory

NAME                                                 TYPE          VALUE
------------------------------------
inmemory_clause_default                              string
inmemory_force                                       string         DEFAULT
inmemory_max_populate_servers                        integer        0
inmemory_query                                       string         ENABLE
inmemory_size                                        big integer    0
inmemory_trickle_repopulate_servers_percent          integer        1
optimizer_inmemory_aware                             boolean        TRUE
```

ORACLE

Oracle Database In Memory 12.1.0.2
How to enable:

- Increase `sga_target` / `sga_max_size` to accommodate IMC within SGA
- `compatible=12.1.0.2.0`
- Inmemory_size must be => 100M

```
SQL> alter system set
       inmemory_size=200M|G scope=spfile;
```

ORACLE

In-Memory area is subdivided into two pools:

- 64 K pool used for metadata
- 1 MB pool for actual columns
- The amount of available memory in each pool is visible in the V$INMEMORY_AREA

ORACLE

Oracle Database In Memory 12.1.0.2

- In Memory Column Store now allocated

```
SQL> show sga
Total System Global Area 1476395008 bytes
Fixed Size                  3006872 bytes
Variable Size             352325224 bytes
Database Buffers          889192448 bytes
Redo Buffers               13766656 bytes
In-Memory Area            218103808 bytes
```

ORACLE

Oracle Database In Memory 12.1.0.2

- You can enable the IM column store table, materialized view, tablespace and partition:

```
sql> alter table emp inmemory;
sql>alter table emp inmemory no inmemory (column name);
sql>alter tablespace tbs1 default inmemory
sql> create materialized view oe.prod_info_mv inmemory as
select * from oe.product_information;
```

ORACLE

Column Store options

Priority:

- None
- Low
- Medium
- High
- Critical

Compression Methods

- No Memcompress
- Memcompress for DML
- Memcompress for Query Low - Default – Best for query performance
- Memcompress for Query High
- Memcompress for Capacity Low
- Memcompress for Capacity High

```
SQL>alter table emp inmemory
Priority Critical  Memcompress
for Query High;
```

ORACLE

Oracle Database In Memory 12.1.0.2

- Check status of table load into Column Store

```
SELECT v.owner, v.segment_name name, v.populate_status status,
       v.bytes_not_populated
FROM   v$im_segments v;

 OWNER            NAME              STATUS           BYTES_NOT_POPULATED
 HR               EMP               STARTED           183992320
```

- At this point you can see Worker Processes loading at OS level:

```
          ps -ef | grep _w00  (top)
          oracle   18516     1  0 09:04 ?        00:00:00 ora_w004_orcl
          oracle   18725     1  0 09:05 ?        00:00:00 ora_w005_orcl
          oracle   18733     1  0 09:05 ?        00:00:00 ora_w006_orcl
```

ORACLE

Oracle Database In Memory 12.1.0.2

- Load of Table completed:

```
SQL> OWNER    NAME              STATUS          BYTES_NOT_POPULATED
     HR       EMP               COMPLETED              0
```

```
SQL> Select  table_name, inmemory from user_tables;

TABLE_NAME                     INMEMORY
-------------------            ------------
DEPARTMENTS                    DISABLED
JOBS                           DISABLED
EMPLOYEES                      DISABLED
JOB_HISTORY                    DISABLED
:
```

Additional Resources

- InMemory column store on RAC:

https://blogs.oracle.com/In-Memory/entry/oracle_database_in_memory_on

https://blogs.oracle.com/In-Memory/

- How to configure in-memory column store:

https://docs.oracle.com/database/121/TGDBA/tune_sga.htm#TGDBA95378

ORACLE

Oracle 12c: In-Memory Database with Linux on IBM z Systems™

David Simpson

Oracle Technical Specialist and Certified Professional

IBM z Systems Technical Team

simpson.dave@us.ibm.com

Agenda

1. What is In-Memory Option?
2. When to Use In-Memory option
3. How to use In-Memory option
4. Experiences with In-Memory option with IBM z Systems
5. Testing, Results and Issues with In-Memory option

Test Environment Used:

Analytics Reporting Workload

- **1TB** Database (8 concurrent users)
- 26 Queries total
  - 2 very complex (1a and 3b)
  - 3 moderate in complexity (9, 10 & 11)
  - 21 simple in complexity
- Star schema, bitmap foreign key indexes were removed (more efficient execution).
- Large sales fact table is range-hash partitioned,
  - Fact table has 9B rows, 16 numeric columns and is 726G
  - In-Memory requires minimum 250G SGA and 200G In-memory size
  - Prior to in-memory runs, the sales_fact table is fully populated into memory (priority high) with the default compression
  - Tests ran under z/VM and without z/VM virtualization.

Key Oracle In-Memory Views:

V$INMEMORY_AREA – shows the overall allocated inmemory size
V$IM_SEGMENTS– shows the relevant In Memory detail for each segment (size, population status, priority, compression, etc)
V$IM_COLUMN_LEVEL – displays inmemory settings per column

```
SQL> select pool,alloc_bytes,used_bytes,populate_status from v$inmemory_area;
POOL                      ALLOC_BYTES       USED_BYTES        POPULATE_STATUS
--------------------------  ----------------  ----------------  --------------------------
1MB POOL                  170,708,172,800  170,370,531,328     DONE
64KB POOL                  42,932,895,744    1,258,684,416     DONE
```

```
SQL> select segment_name,sum(inmemory_size),sum(bytes), sum(bytes_not_populated) from   v$im_segments group by
segment_name;
SEGMENT_NAME   SUM(INMEMORY_SIZE)        SUM(BYTES)          SUM(BYTES_NOT_POPULATED)
---------------  ------------------        ----------------    ------------------------
SALES_FACT       171,437,391,872           725,941,747,712                0
```

```
SQL>   select segment_name,segment_type,inmemory_priority, inmemory_distribute, inmemory_compression from
v$im_segments;
SEGMENT_NAME   SEGMENT_TYPE   INMEMORY_PRIOR   INMEMORY_DISTR   INMEMORY_COMPRESS
-------------------------------------------------------------------------------------------------------------
SALES_FACT       TABLE          HIGH                AUTO              FOR QUERY LOW
```

Expected performance benefits of Oracle In-Memory

Benefits offered by Oracle in Memory*:

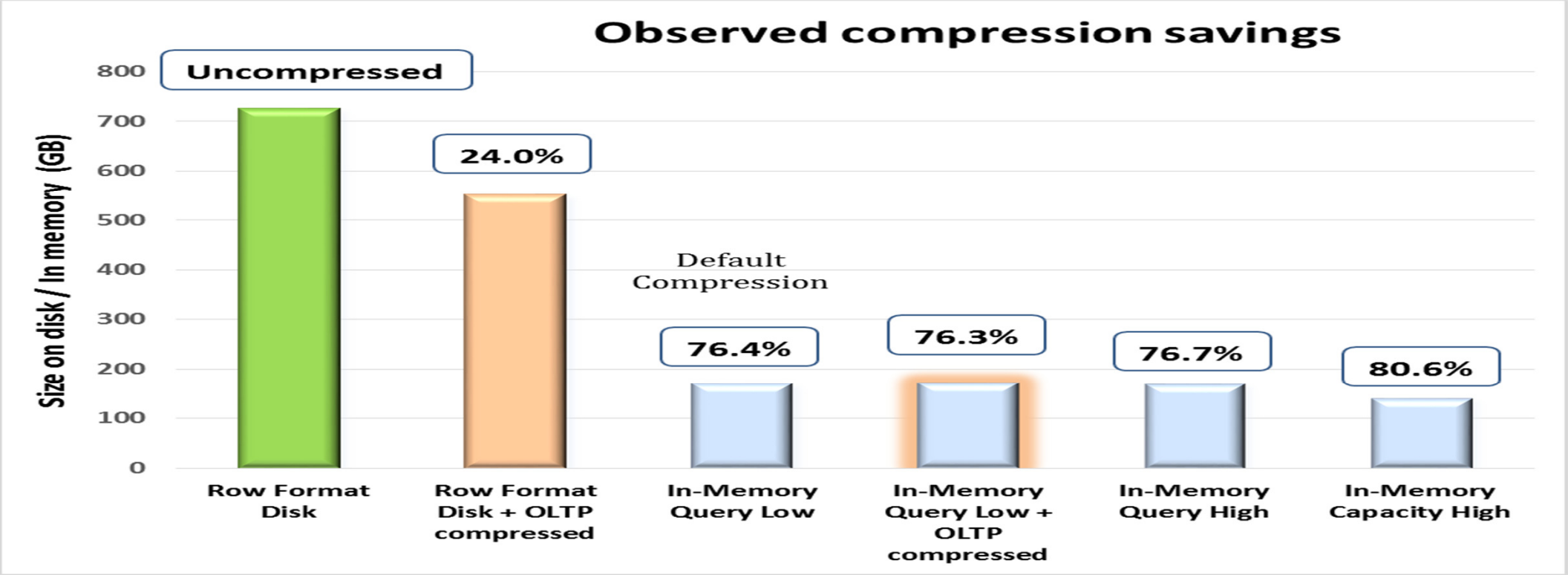
- Faster scanning of large number of rows / applying filters that use operators such as =,<,>, and IN. Faster querying of a subset of columns in a table, for example, selecting 5 of 100 columns.
- Performance for joins converting predicates on small dimension tables to filters on large fact table.
- Efficient aggregation by using VECTOR GROUP BY transformation and vector array processing
- Reduced storage space and less processing overhead because fewer indexes, materialized views, and OLAP cubes are required when IM column store is used.

Where we expect to see reduced or no benefit with In-Memory:

- Queries that return many rows
- Queries that select many columns, for example, 99 of 100 columns
- Queries that do not benefit from vector group by aggregation such as queries that join multiple large tables

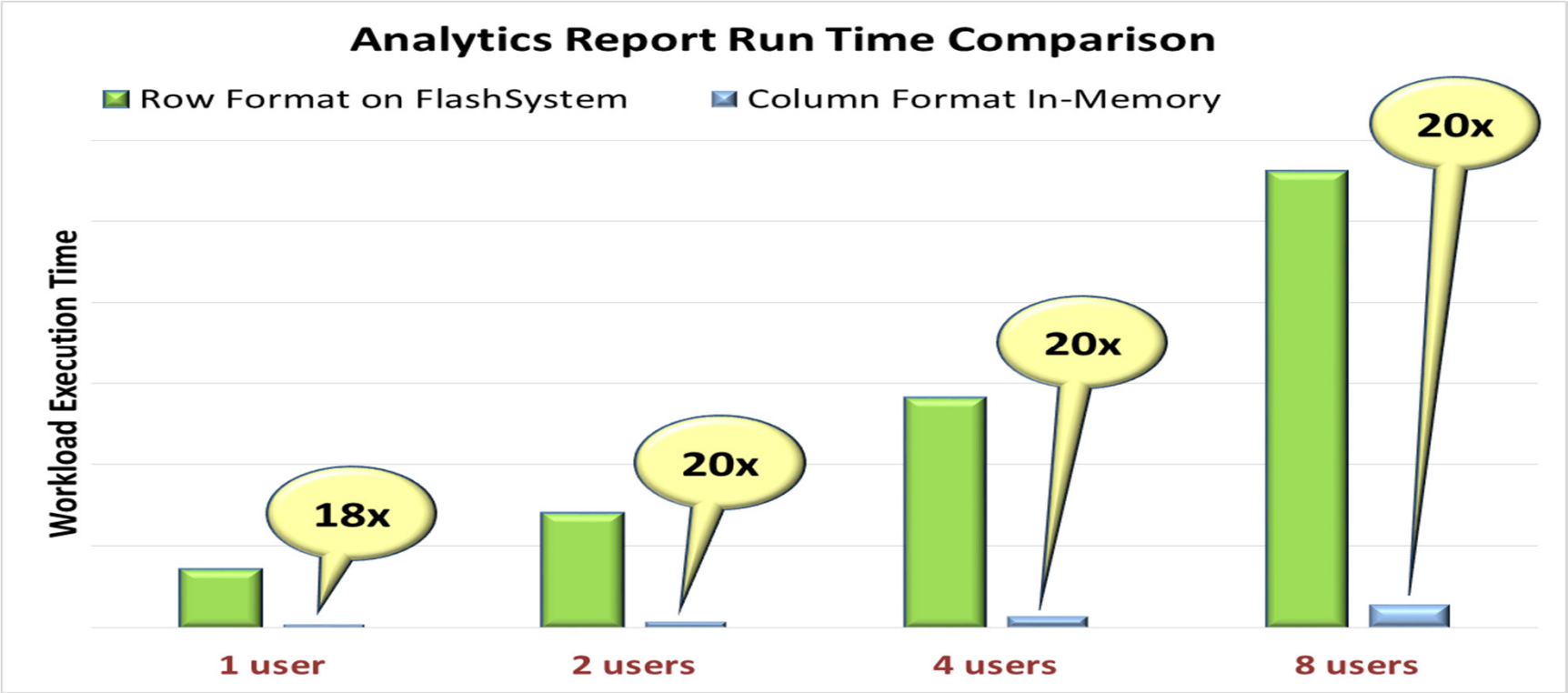
* According to the Oracle 12c Release 1 Database Data Warehousing Guide, https://docs.oracle.com/database/121/

In-Memory Compression effectiveness – a comparison



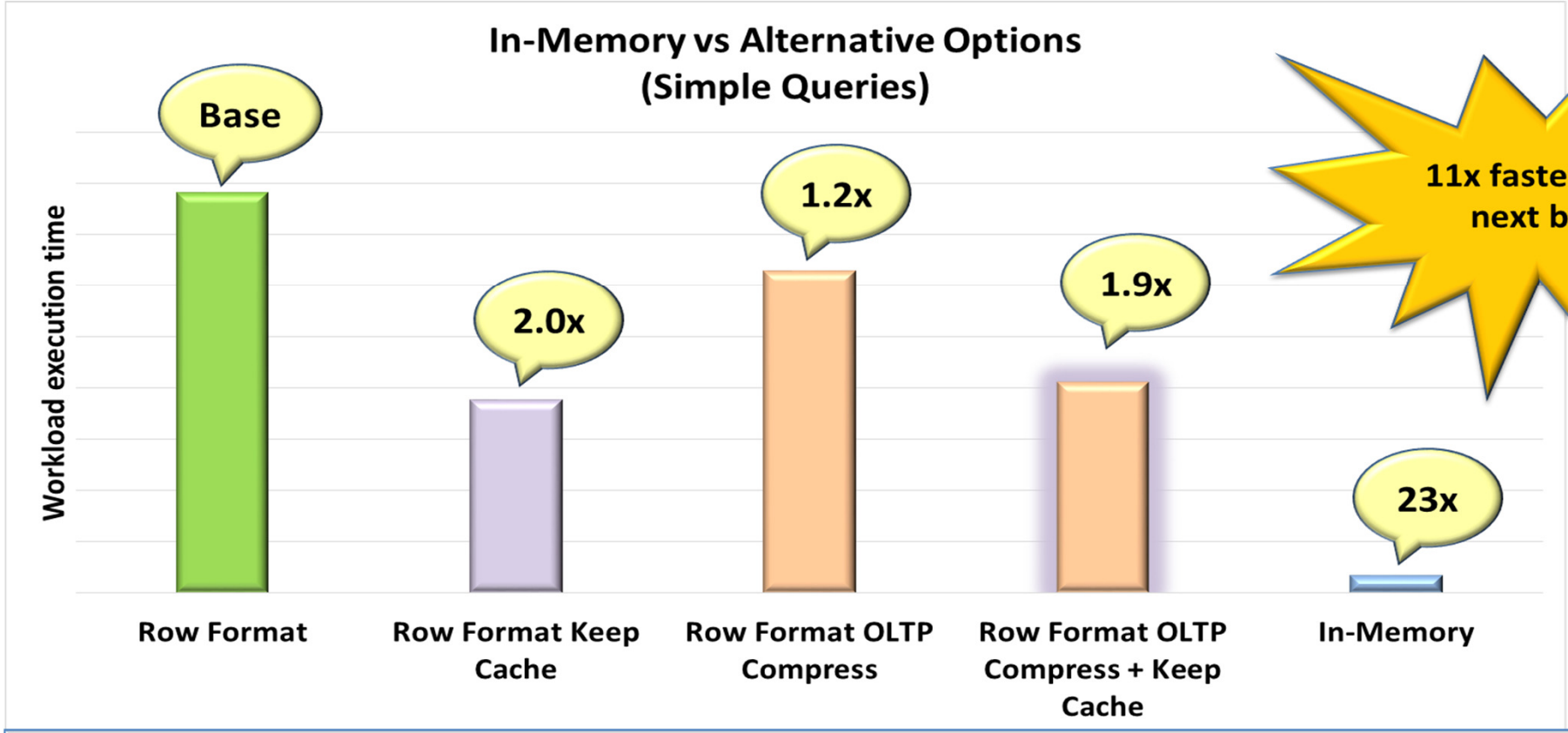
Note: Tests required to set inmemory_size to 200GB to completely store the 171.4GB of compressed actual data in memory. The remaining ~28.6GB were used for the additional metadata required.

“Real World” Analytics Reporting with Oracle Database In-Memory



The table queried had 9 Billion rows with 16 numeric columns and was loaded into the In-Memory area with compression set to “Query Low”. The execution time was measured from the start of the first report until the last report had completed for all users..

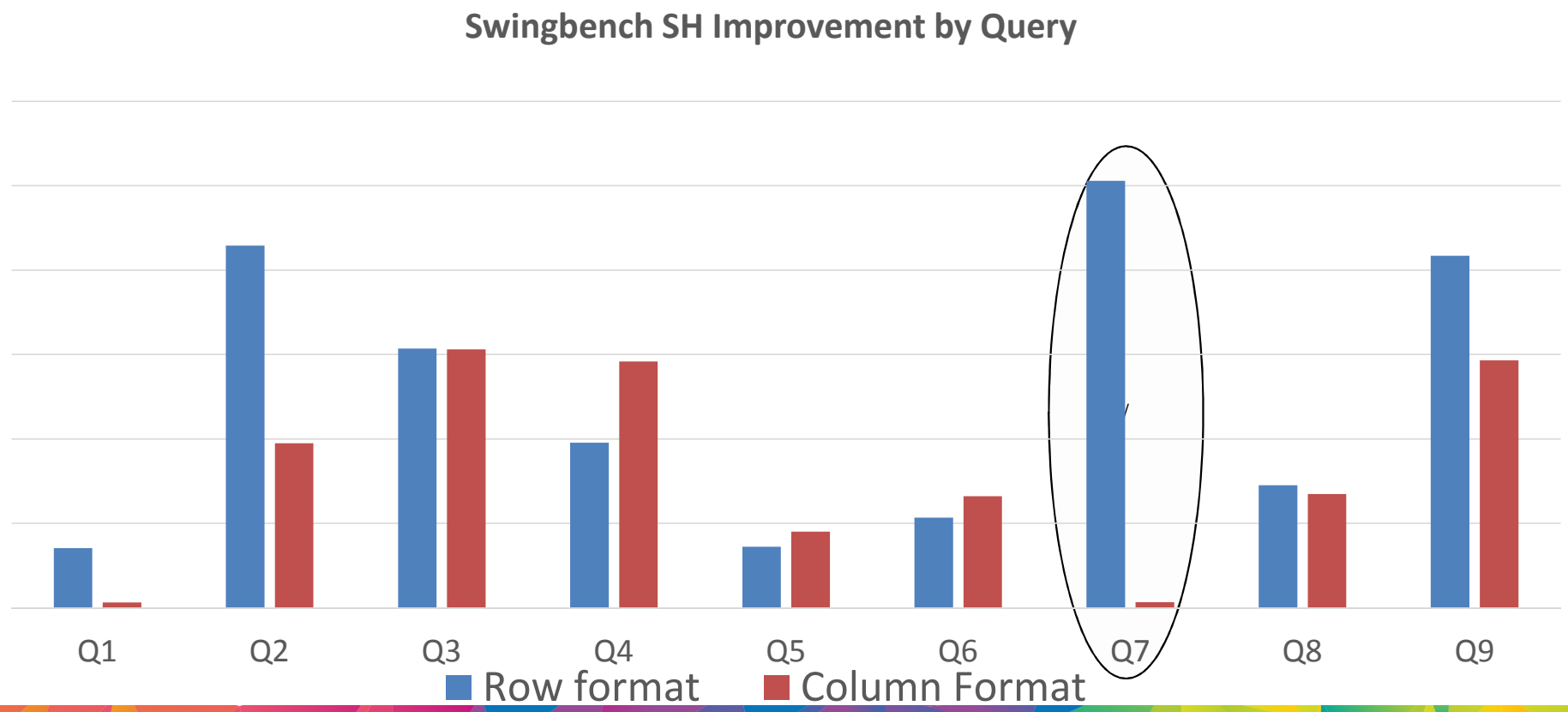
Comparing Oracle Database In-Memory alternatives using “simple queries”



Results with Oracle DB In-Memory feature are **11x** faster than the next best alternative (Row Format Keep Cache), while requiring ~ 3x (550G) less memory!

Swingbench Sales History

Some Swingbench SH queries had minor performance benefits with In-Memory Option



Where's the difference ?

Query 6 – Better with Row Format:

```
SELECT channel_desc, calendar_month_desc, countries.country_iso_code,
            TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
  FROM sales, customers, times, channels, countries
  WHERE sales.time_id=times.time_id
  AND sales.cust_id=customers.cust_id
  AND sales.channel_id= channels.channel_id
  AND customers.country_id = countries.country_id
  AND channels.channel_desc IN ('Partners','Tele Sales' )
  AND times.calendar_month_desc IN  ('2005-07','2005-08')
  AND countries.country_iso_code IN ('NZ','PL')
 GROUP BY CUBE(channel_desc, calendar_month_desc, countries.country_iso_code);
```

- Joining two large tables
- Vector transformation not used for cube functions

Where's the difference ?

Query 7 – Better with Column Format In Memory:

```
SELECT t.time_id, to_char(SUM(amount_sold), '9,999,999,999') AS sales,  to_char(AVG(SUM(amount_sold))
over(ORDER BY t.time_id range BETWEEN INTERVAL '2' DAY preceding AND INTERVAL '2' DAY following),
'9,999,999,999') AS entered_5_day_avg
 FROM sales s,   times t
 WHERE t.calendar_month_desc IN('2008-08','2008-10','2008-11','2008-12')
 AND s.time_id = t.time_id
 GROUP BY t.time_id
 ORDER BY t.time_id;
```

- Joining one large fact table with a small dimension table
- 'Group by' effectively uses vector transformation

Observations During Testing

Dynamic Sampling: *When optimizer_dynamic_sampling=2, optimizer does extra work parsing query the first time – generally taking longer to find the plan than to execute the query.*

Workarounds –1) **ALTER SESSION SET "_fix_control"='7452863:0';**

2) Optimizer_dynamic_sampling=0

Statistics level: *Some In Memory queries take substantially longer when statistics_level=all than statistics_level=typical. This was most frequently seen in statements with mathematical operations such as sum, avg, max, etc.* ***Recommendation: validate statistics_level=typical***

SQL execution keeps the same plan hash value from IM and non-IM plans *– making it difficult to accurately pull execution plans from dba_hist views.*

Vector Transformation (faster) plans infrequently selected without parameter changes *(complex query set – simple queries ok) used* ***"_always_vector_transformation"=TRUE*** *for some testing, using the hint is better.*

Use Cases for In-Memory

- *Extreme run time improvements are possible for Data Warehouse or Analytics workloads with star schemas.*
- *No other alternatives (additional CPU, keep cache, etc) can compensate for the performance improvements with this feature.*
- *Workloads do not benefit uniformly from Oracle In-Memory.*
- *IO subsystems overburdened by workload throughput can offload that IO to memory, freeing up more capacity for other workloads.*
- *Reduce indexes created for just analytic reports that are maintained by OLTP transactions.*

Learn More about Oracle In Memory

My Oracle Support Notes

- Oracle Database In-Memory Option (DBIM) Basics and Interaction with Data Warehousing Features
  - Doc ID 1903683.1
- In-Memory Aggregation a New Feature in 12.1.0.2
  - Doc ID 1935305.1

Oracle Database In Memory Whitepaper

- http://www.oracle.com/us/products/database/options/database-in-memory/overview/index.html

Oracle Database Online Documentation 12c Release 1

- https://docs.oracle.com/database/121/

Oracle In-Memory Blog

- https://blogs.oracle.com/In-Memory/

Summary

What we covered today

- Introduction to In-Memory Option
- In-Memory Option When and How to use it
- Our Experiences with In-Memory Option with IBM z Systems

ORACLE

Q & A

- To ask a question on the phone line, select *1 on your phone.
- To ask a question online, use the Q&A area at the top.
- Your question will be read aloud in the order received.
- Question can also be asked on the My Oracle Support Communities