



Oracle Database In-Memory

AskTOM Office Hours – 23c In-Memory Deep Vectorlization Enhancements

Andy Rivenes

Database In-Memory Product Management

Email: andy.rivenes@oracle.com

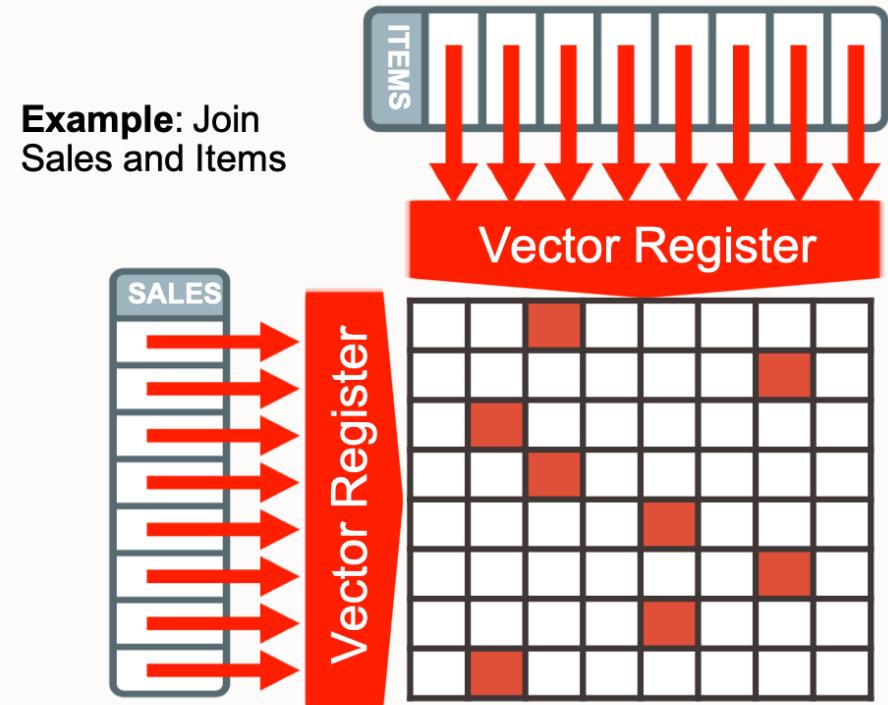
Twitter: @TheInMemoryGuy

Blog: blogs.oracle.com/in-memory

In-Memory Vectorized Joins

In-Memory Vectorized Joins – 21c Review

- [Ask TOM Office Hours – Database In-Memory Vectorized Joins in 21c](#)
- In-Memory Vectorized Joins uses a new *In-Memory Vectorization framework* to accelerate **Complex Joins**
- Optimizes Hash Joins
 - Performs join operations during in-memory scans
 - Match multiple rows between SALES and ITEMS tables in a single SIMD Vector Instruction
- AVX SIMD support provides maximum benefit
- Utilizes Join Groups for enhanced performance
- Can leverage IMDS for increased performance
- Can improve performance by up to 15x versus a non-IM hash join



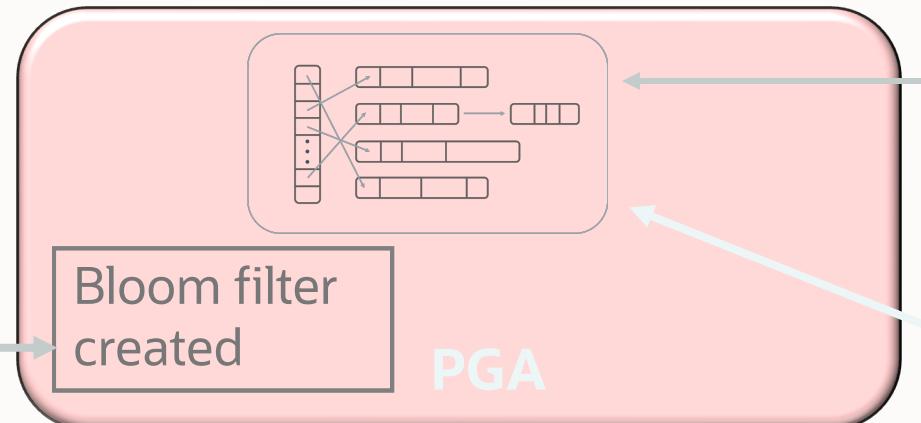
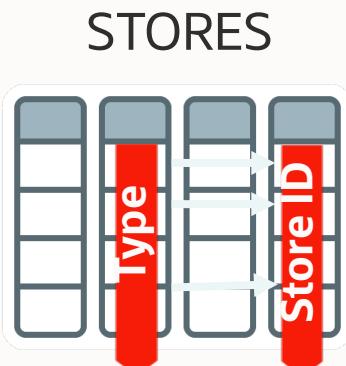
Database In-Memory Hash Join With Vectorized Join

2 Bloom Filter

Creation: A bit vector is created that has a bit set for each join column there is a value for & 0 where there is no value

1 Table Scan:

STORES table is scanned, filter predicates applied and matching rows sent to hash join



3 Bloom Filter sent: Bit vector is sent as an additional filter criteria to the scan of the SALES table

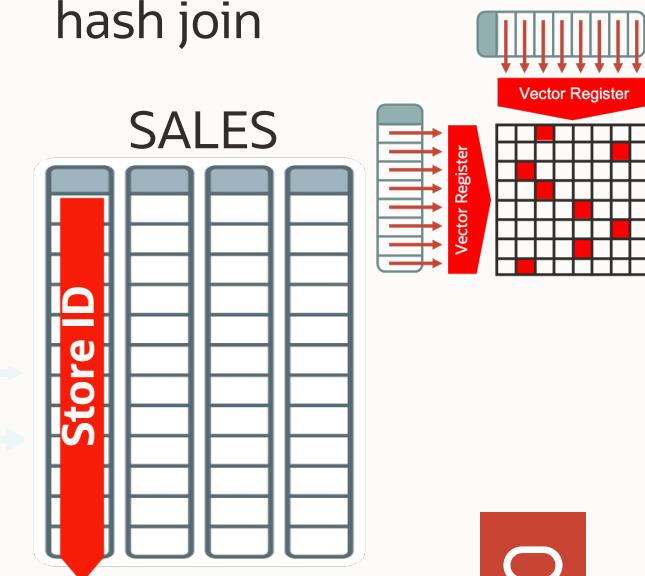


4 Table Scan: SALES table is scanned and rows are filtered based on filter predicates

6 Hash Join: Join completed by probing into the hash table from the store_id to find actual matching rows

5 Reduced rows sent:

Only rows that have a match in the bit vector get sent to the hash join



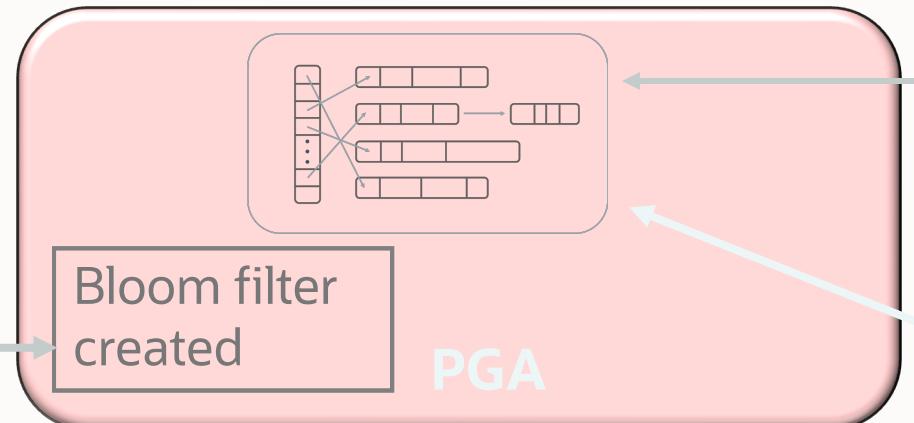
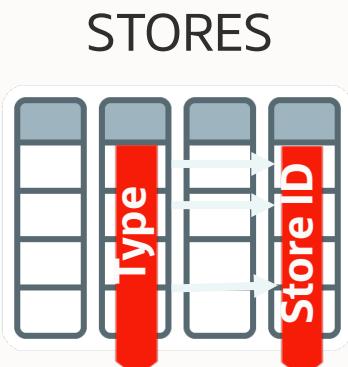
Database In-Memory Hash Join With Vectorized Join

2 Bloom Filter

Creation: A bit vector is created that has a bit set for each join column there is a value for & 0 where there is no value

1 Table Scan:

STORES table is scanned, filter predicates applied and matching rows sent to hash join



3 Bloom Filter sent: Bit vector is sent as an additional filter criteria to the scan of the SALES table

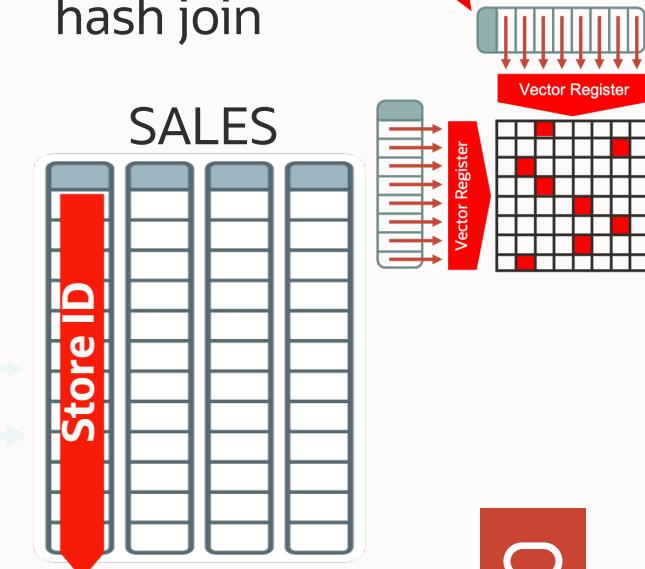


4 Table Scan: SALES table is scanned and rows are filtered based on filter predicates

6 Hash Join: Join completed by probing into the hash table from the store_id to find actual matching rows

In-Memory Vectorized Join

materialized
vector get sent to the
hash join



How Do You Enable Vectorized Joins?

- In-Memory vectorized joins can be controlled with the following initialization parameter:

`inmemory_deep_vectorization` boolean TRUE

- Default value is TRUE and can be disabled by setting it to FALSE

How Do You Tell If a Vectorized Join Was Used?

SQL Monitor Active Report

Details

Plan Hash Value 475248987

	Operation	Object	Information	Est. Rows	Timeline	Exec	Rows	Mem	Temp
0	SELECT STATEMENT			1,820K					
1	SORT GROUP BY NOSORT								
2	HASH JOIN		炬	1,820K					
3	JOIN FILTER CREATE	:BF0000	炬	365					
4	TABLE ACCESS INMEMORY FULL	DATE_DIM	表	365					
5	JOIN FILTER USE	:BF0000	炬	12M					
6	TABLE ACCESS INMEMORY FULL	LINEORDER	表	12M					

Other Information

DeepVec Hash Joins 1
DeepVec Hash Join 2,564
Flags

OK

The screenshot shows the Oracle SQL Monitor Active Report interface. The main window displays a plan trace with a plan hash value of 475248987. The trace consists of seven operations: a SELECT STATEMENT, a SORT GROUP BY NOSORT, a HASH JOIN, a JOIN FILTER CREATE, a TABLE ACCESS INMEMORY FULL for DATE_DIM, another JOIN FILTER USE, and a final TABLE ACCESS INMEMORY FULL for LINEORDER. A red arrow points from the 'Information' column of the HASH JOIN row to a tooltip titled 'Other Information'. The tooltip contains three pieces of information: 'DeepVec Hash Joins 1', 'DeepVec Hash Join 2,564', and 'Flags'. In the bottom right corner of the tooltip, there is an 'OK' button.

How Do You Tell If a Vectorized Join Was Used?

SQL*Plus way to determine Deep Vector usage (similar to Join Groups)

```
PROMPT Deep Vectorization Usage: ;
PROMPT -----
PROMPT ;

SELECT
    '    ' || deepvec.rowsource_id || ' - ' row_source_id,
CASE
    WHEN deepvec.deepvec_hj IS NOT NULL
    THEN
        'deep vector hash joins used: ' || deepvec.deepvec_hj ||
        ', deep vector hash join flags: ' || deepvec.deepvec_hj_flags
    ELSE
        'deep vector HJ was NOT leveraged'
END deep_vector_hash_join_usage_info
FROM
(SELECT EXTRACT(DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_XML,
    q'#//operation[@name='HASH JOIN' and @parent_id]#'') xmldata
  FROM DUAL) hj_operation_data,
XMLTABLE('/operation'
    PASSING hj_operation_data.xmldata
    COLUMNS
        "ROWSOURCE_ID"      VARCHAR2(5) PATH '@id',
        "DEEPVEC_HJ"        VARCHAR2(5) PATH 'rwsstats/stat[@id="11"]',
        "DEEPVEC_HJ_FLAGS"  VARCHAR2(5) PATH 'rwsstats/stat[@id="12"]') deepvec;
```

23c In-Memory Vectorized Join Enhancements

In-Memory Vectorization Framework: Multi-Level Joins and Aggregations

- In-Memory Deep Vectorization Framework was introduced on Oracle Database 21c
 - The first feature that leveraged the framework improved the performance of single level hash joins
- In Oracle Database 23c the In-Memory Deep Vectorization Framework adds more join support:
 - Multi-level hash joins
 - Multi join key
 - Semi joins
 - Outer joins
 - Full group by aggregation
- This translates to faster join performance by utilizing SIMD optimizations during join processing

Semi-Join Example

```
select /*+ MONITOR */ count(l.lo_custkey)
from lineorder l
where l.lo_partkey IN (select p.p_partkey from part p)
and l.lo_quantity <= 3;
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT					5113 (100)			
1	SORT AGGREGATE		1	18					
* 2	HASH JOIN RIGHT SEMI		1640K	28M	12M	5113 (8)	00:00:01		
3	TABLE ACCESS INMEMORY FULL	PART	800K	3906K		73 (3)	00:00:01		
4	PARTITION RANGE ALL		1640K	20M		2448 (15)	00:00:01	1	3
* 5	TABLE ACCESS INMEMORY FULL	LINEORDER	1640K	20M		2448 (15)	00:00:01	1	3

Semi-Join Example, Part 2

SQL*Plus way to determine Deep Vector usage (similar to Join Groups)

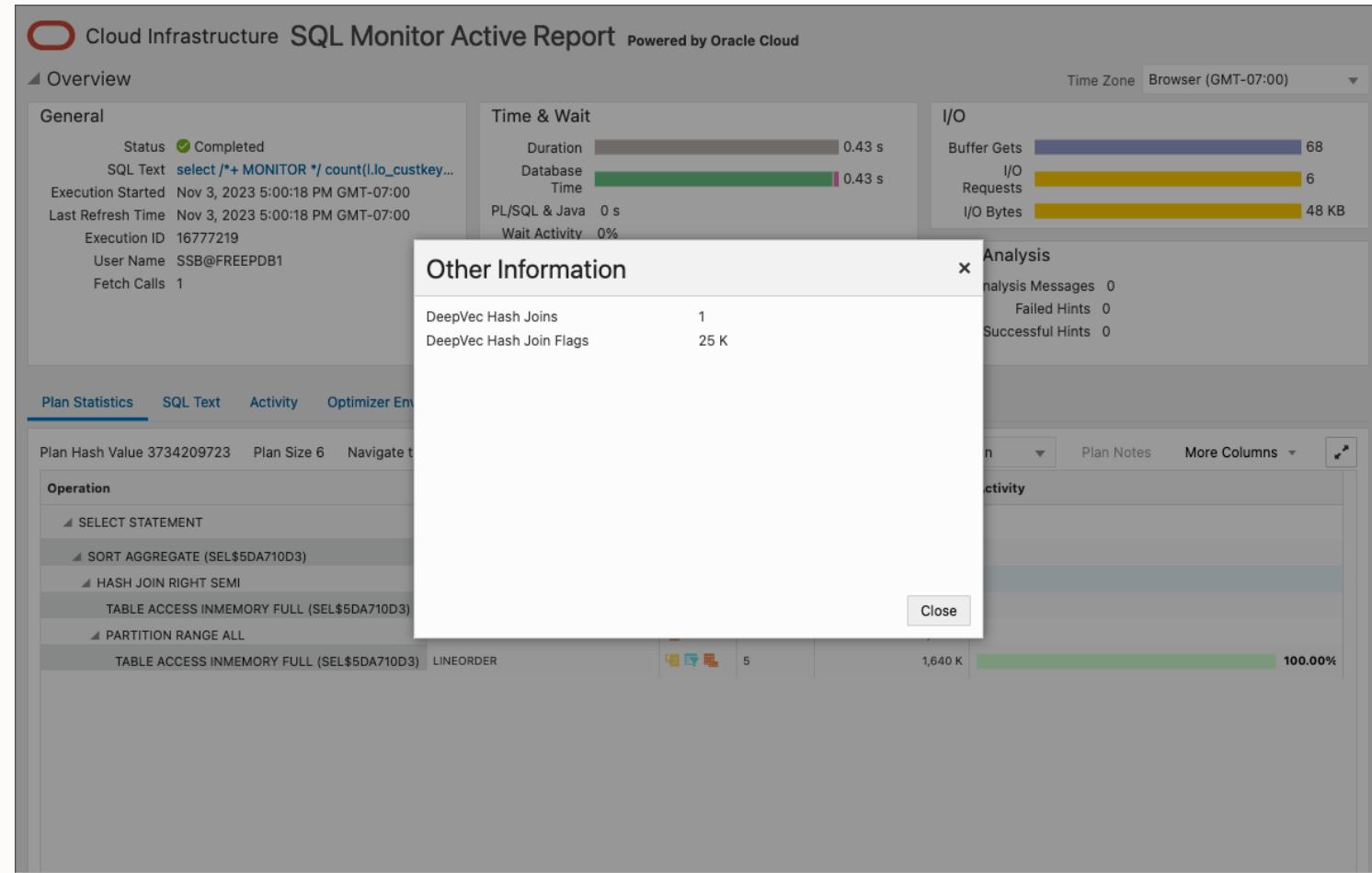
```
SELECT
  ' ' || deepvec.rowsource_id || ' - ' row_source_id,
CASE
  WHEN deepvec.deepvec_hj IS NOT NULL
  THEN
    'deep vector hash joins used: ' || deepvec.deepvec_hj || ', deep vector hash join flags: ' || deepvec.deepvec_hj_flags
  ELSE
    'deep vector HJ was NOT leveraged'
END deep_vector_hash_join_usage_info
FROM
  (SELECT EXTRACT(DBMS_SQL_MONITOR.REPORT_SQL_MONITOR_XML, q'#/operation[@name='HASH JOIN' and @parent_id]#') xmldata
   FROM DUAL) hj_operation_data,
XMLTABLE('/operation'
  PASSING hj_operation_data.xmldata
  COLUMNS
  "ROWSOURCE_ID"      VARCHAR2(5) PATH '@id',
  "DEEPVEC_HJ"        VARCHAR2(5) PATH 'rwsstats/stat[@id="11"]',
  "DEEPVEC_HJ_FLAGS"  VARCHAR2(5) PATH 'rwsstats/stat[@id="12"]') deepvec;
```

Deep Vectorization Usage:

2 - deep vector hash joins used: 1, deep vector hash join flags: 24576

Semi-Join Example, Part 3

SQL Monitor way to determine Deep Vector usage



Semi-Join Example, Part 4

Execution Plan Output

Plan hash value: 3734209723

Id	Operation	Name	Rows	Bytes	TempSpc	Cost	(%CPU)	Time	Pstart	Pstop
0	SELECT STATEMENT					5113	(100)			
1	SORT AGGREGATE		1	18						
* 2	HASH JOIN RIGHT SEMI		1640K	28M	12M	5113	(8)	00:00:01		
3	TABLE ACCESS INMEMORY FULL	PART	800K	3906K		73	(3)	00:00:01		
4	PARTITION RANGE ALL		1640K	20M		2448	(15)	00:00:01	1	3
* 5	TABLE ACCESS INMEMORY FULL	LINEORDER	1640K	20M		2448	(15)	00:00:01	1	3

Predicate Information (identified by operation id):

```
2 - access("L"."LO_PARTKEY"="P"."P_PARTKEY")
5 - inmemory("L"."LO_QUANTITY"><=3)
      filter("L"."LO_QUANTITY"><=3)
```

25 rows selected.

SQL>

SQL> set echo off

Deep Vectorization Usage:

```
2 -      deep vector hash joins used: 1, deep vector hash join flags: 24576
```

Where Can You Get More Information?

Learn More

Database In-Memory Blog

- [23c Deep Dive - In-Memory Vector Join Enhancements](#)
- [In-Memory Vectorized Joins](#)

Database In-Memory Ask TOM Office Hours

- [23c In-Memory Deep Vectorization Enhancement](#)
- [Database In-Memory Vectorized Joins in 21c](#)

Database In-Memory Hands-on-Lab

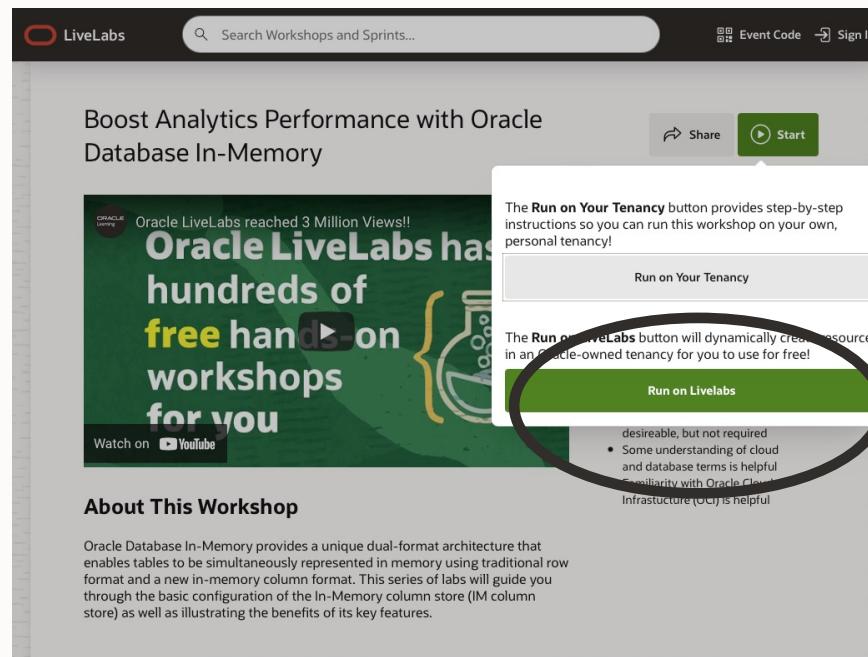
Database In-Memory Documentation

Database In-Memory LiveLabs

<http://bit.ly/golivelabs>

Boost Analytics Performance with Oracle Database In-Memory

Database In-Memory Advanced Features



The screenshot shows the Oracle LiveLabs website. At the top, there's a navigation bar with the 'LiveLabs' logo, a search bar, and a 'Sign In' button. Below the header, a banner reads 'Boost Analytics Performance with Oracle Database In-Memory'. A callout box highlights the 'Run on Your Tenancy' button, which is described as providing step-by-step instructions for running the workshop on personal tenancy. Another callout box highlights the 'Run on LiveLabs' button, which creates resources in a Oracle-owned tenancy for free. An arrow points from the 'Run on LiveLabs' button towards the 'Free Tenancy Reservation' section on the right.

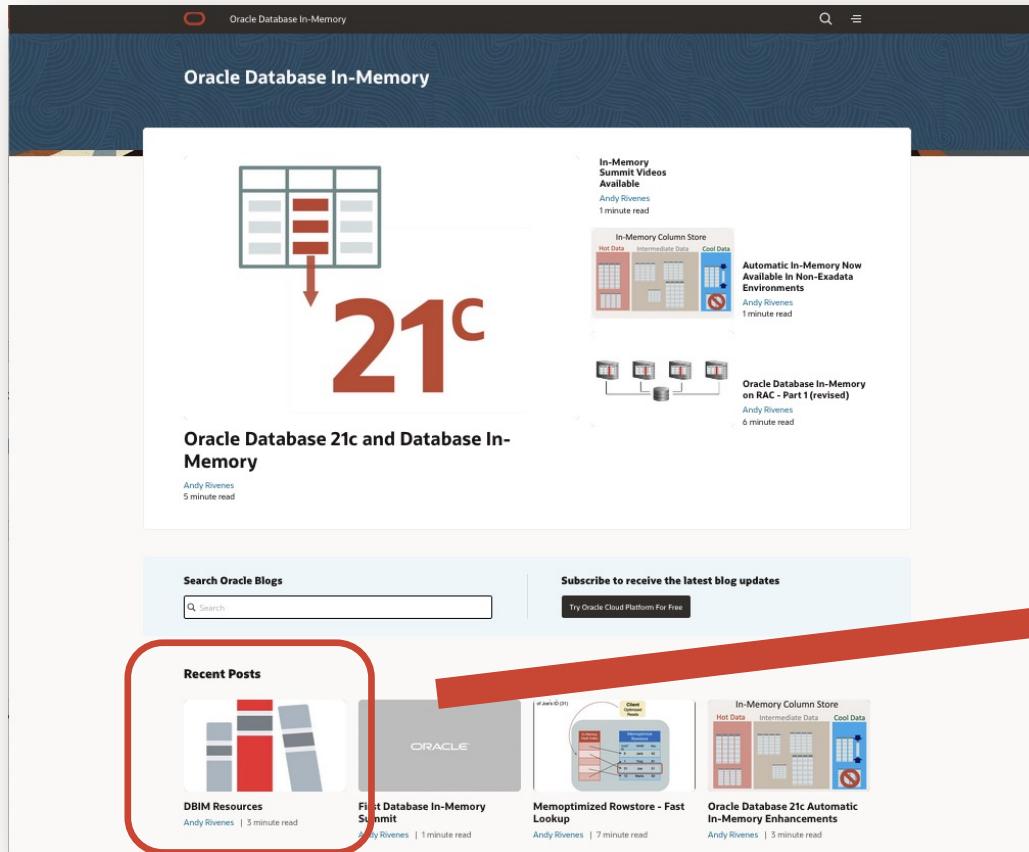
Free Tenancy Reservation

Try it out

No requirement to sign up
for the Cloud!



<https://blogs.oracle.com/in-memory/dbim-resources>



Oracle Database In-Memory

Oracle Database 21c and Database In-Memory

Andy Rivenes | 5 minute read

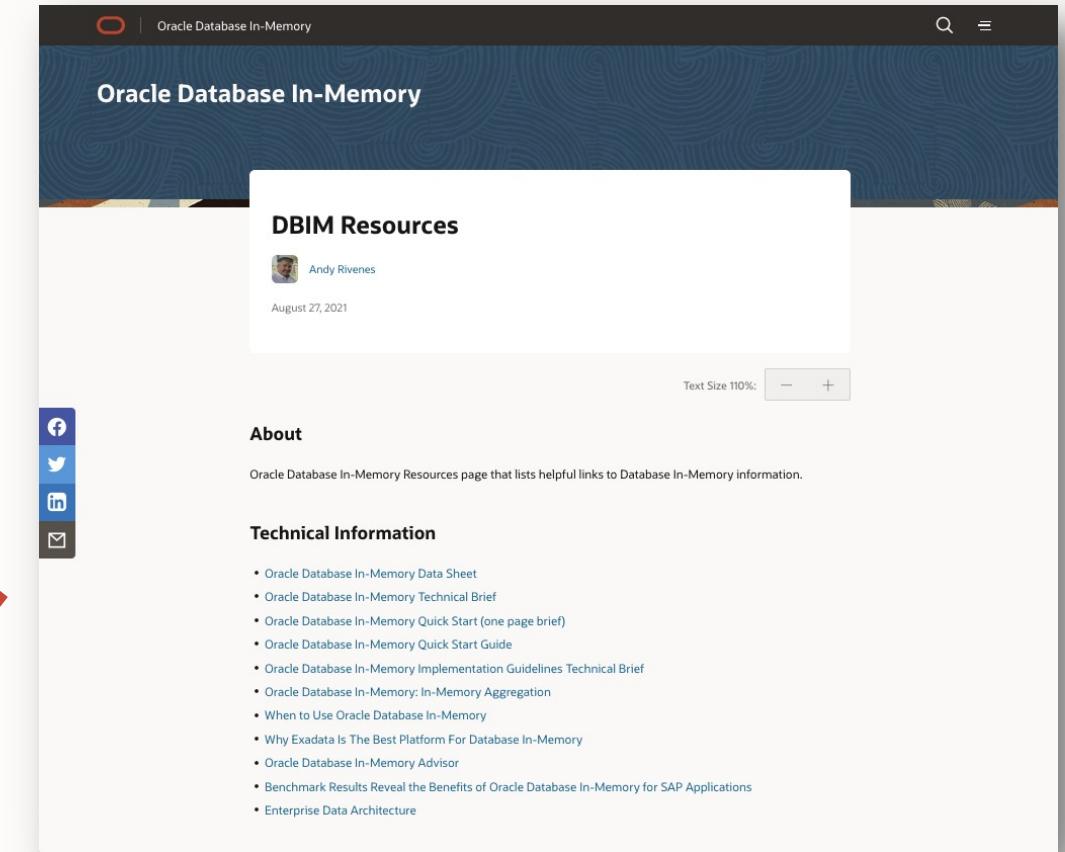
Recent Posts

- DBIM Resources (Red Box)
- First Database In-Memory Summit
- Memoptimized Rowstore - Fast Lookup
- In-Memory Column Store

Search Oracle Blogs

Subscribe to receive the latest blog updates

Try Oracle Cloud Platform For Free



Oracle Database In-Memory

DBIM Resources

Andy Rivenes | August 27, 2021

Text Size 110%: - +

About

Oracle Database In-Memory Resources page that lists helpful links to Database In-Memory information.

Technical Information

- Oracle Database In-Memory Data Sheet
- Oracle Database In-Memory Technical Brief
- Oracle Database In-Memory Quick Start (one page brief)
- Oracle Database In-Memory Quick Start Guide
- Oracle Database In-Memory Implementation Guidelines Technical Brief
- Oracle Database In-Memory: In-Memory Aggregation
- When to Use Oracle Database In-Memory
- Why Exadata Is the Best Platform For Database In-Memory
- Oracle Database In-Memory Advisor
- Benchmark Results Reveal the Benefits of Oracle Database In-Memory for SAP Applications
- Enterprise Data Architecture