# Notes About Autonomous DB

## Autotrace Stat

PLUSTRACE role does not exist, and stat works only for Admin user.

|  |
| --- |
| set autotrace traceonly exp stat |

## Autonomous DB Tuning

* AWR on PDB Level is available with NO Exadata stats

# Exadata Concept

Pre-built system that consists of :

* Number of (compute node) running Linux
  + Compute node can be virtualized or can run databases on bare metal.
* Storage Nodes
* All connected via low-latency network
* Exadata nodes are Intel – based

## Locations

* Oracle Cloud
* Exadata CC (Customer Cloud)

Physically it customer's DC, but managed by Oracle

* Exadata (on Prem.)

## Configuration Packages

* 1/8, ¼… full RAC
* ¼ package includes 2 compute nodes and 3 storage nodes

# Related Products

## Super Cluster

* Super Cluster (SC) is rare (but used in IFIS)
* SC runs on SPARC
* combines DB and App Domains
* Runs Exadata storage

## Golden Gate

Log based replication system:

* Minds logs of source system
* Finds SQL statements to execute based on some criteria
* Execute the statements on the target system

### No Downtime Upgrade

Methodology to use GG for DB upgrade:

* Use RMAN to recover target DB to some Point-in-Time
* Use Golden Gate to roll forward all transactions on target
* Switch source and target
* Use Golden Gate to roll forward all transactions on OLD source in case we need to switch back

# Internal Network

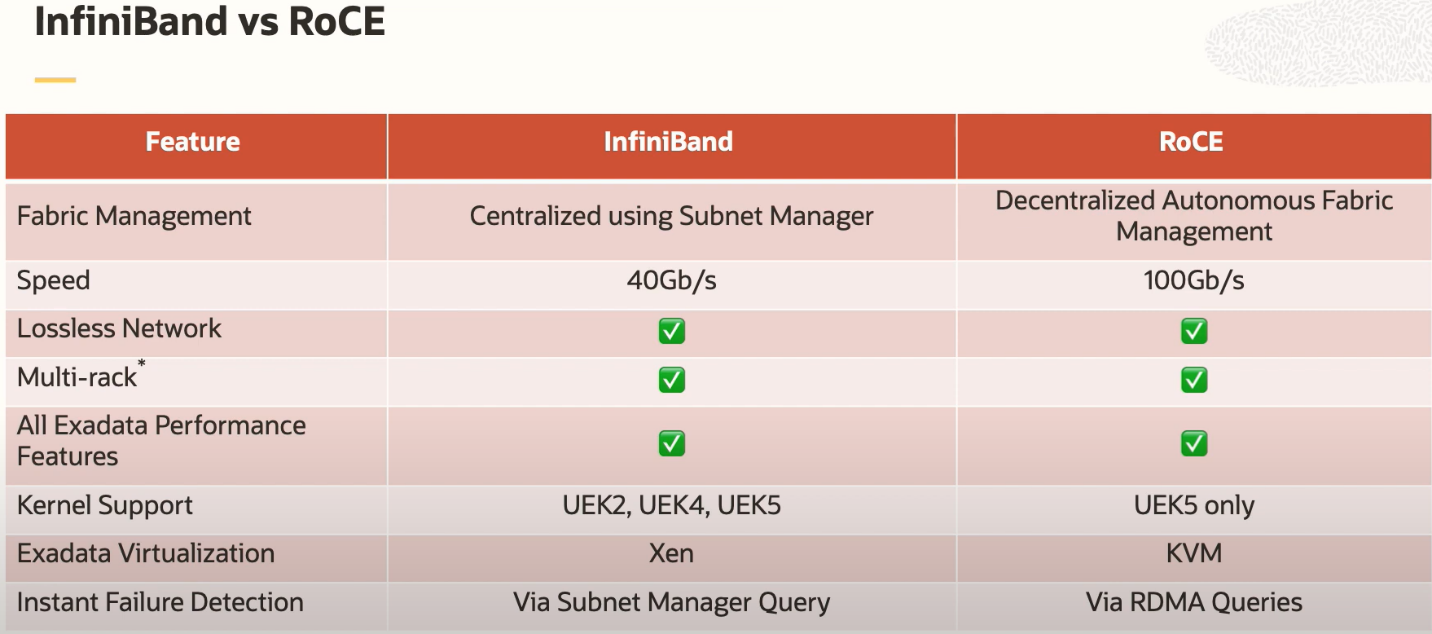
## Before X8X – InfiniBand

( which uses Remote Direct Memory Access ) – RDMA

Key **difference between Ethernet** and **Infiniband**, which **makes Infiniband** faster, is RDMA (Remote Direct Memory Access). ... RDMA is the same idea, but the direct memory access is done by a remote machine. More differences: Communication is done between QPs (Queue Pairs) instead of channels

## Since X8X – Ethernet

( RoCE = RDMA over Converged Ethernet)



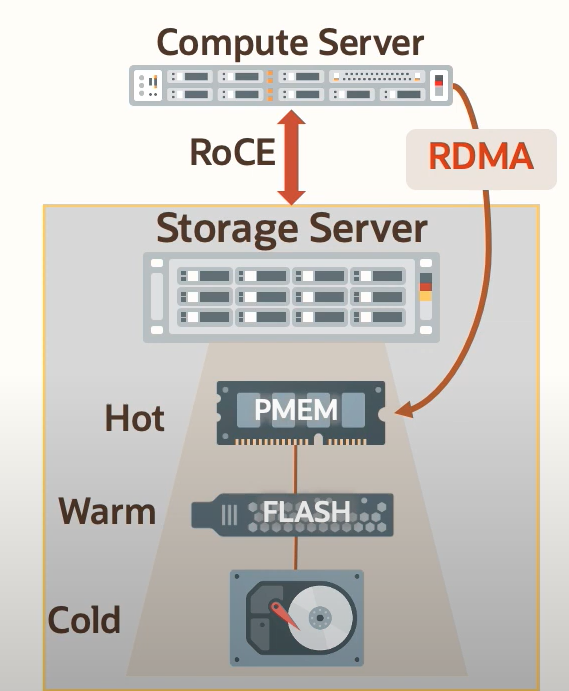
# Storage

## Data (DB Files)

Data files are stored on ASM located on Storage Server

The storage is THREE levels:

* PMEM ( hot)
* Flash PCI for LRU blocks ( warm)
* Disk ( cold)



Storage blocks are DB blocks

## Binaries and some logs

On compute nodes

## Backup

* On another ASM
* OR on Oracle ZFS appliance

# Smart Scan and Cell Offloading

## Cell Offload

Is offload of workload to Storage Node (a.k.a Storage Cell) from the database ( aka Compute Node)

## Smart Scan

Is one of types of Cell Offload. Database scan when query predicate are applied on storage level , thus dramatically improving query performance. Because is executed in parallel on multiple storage nodes .

### Predicate Filtering

Reduce number of blocks returned from storage based on query where clause:

|  |
| --- |
| SQL> select max(qc\_id) from bom\_item a where A.BOM\_ID >= 50000  2 /  Elapsed: 00:00:00.13  Execution Plan  ----------------------------------------------------------  Plan hash value: 3207179268  ---------------------------------------------------------------------------------------  | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |  ---------------------------------------------------------------------------------------  | 0 | SELECT STATEMENT | | 1 | 11 | 14 (15)| 00:00:01 |  | 1 | SORT AGGREGATE | | 1 | 11 | | |  |\* 2 | TABLE ACCESS STORAGE FULL| BOM\_ITEM | 37467 | 402K| 14 (15)| 00:00:01 |  ---------------------------------------------------------------------------------------  Predicate Information (identified by operation id):  ---------------------------------------------------  2 - storage("A"."BOM\_ID">=50000)  filter("A"."BOM\_ID">=50000) |

### Column Projection

Transfer from storage to compute node only required columns

### Join Filtering

Storage cell may reduce number of blocks generated in join operation.

Join:

|  |
| --- |
| select max(use\_code||qc\_id ) from  (  select /\*+ no\_merge \*/ a.\*, b.use\_code from  bom b,  bom\_item a  where a.BOM\_ID=b.bom\_id  and customer like 'A%'  ); |

Produces following plan showing STORAGE Access in both tables via Filter:

|  |
| --- |
| Execution Plan  ----------------------------------------------------------  Plan hash value: 2939182434  ------------------------------------------------------------------------------------------  | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |  ------------------------------------------------------------------------------------------  | 0 | SELECT STATEMENT | | 1 | 24 | 25 (12)| 00:00:01 |  | 1 | SORT AGGREGATE | | 1 | 24 | | |  | 2 | VIEW | | 8168 | 191K| 25 (12)| 00:00:01 |  |\* 3 | HASH JOIN | | 8168 | 271K| 25 (12)| 00:00:01 |  | 4 | JOIN FILTER CREATE | :BF0000 | 8168 | 271K| 25 (12)| 00:00:01 |  |\* 5 | TABLE ACCESS STORAGE FULL| BOM | 2800 | 64400 | 11 (10)| 00:00:01 |  | 6 | JOIN FILTER USE | :BF0000 | 79106 | 849K| 14 (15)| 00:00:01 |  |\* 7 | TABLE ACCESS STORAGE FULL| BOM\_ITEM | 79106 | 849K| 14 (15)| 00:00:01 |  ------------------------------------------------------------------------------------------  Predicate Information (identified by operation id):  ---------------------------------------------------  3 - access("A"."BOM\_ID"="B"."BOM\_ID")  5 - storage("CUSTOMER" LIKE 'A%')  filter("CUSTOMER" LIKE 'A%')  7 - storage(SYS\_OP\_BLOOM\_FILTER(:BF0000,"A"."BOM\_ID"))  filter(SYS\_OP\_BLOOM\_FILTER(:BF0000,"A"."BOM\_ID")) |

## Savings: SI savings, CC savings

# Performance Tuning on Exadata:

**Performance Tuning Fundamentals Do not change:**

1. Main Principles:
   1. Tuning methodology did not change
   2. Focus on the data ( stats, AWR, ASH)
   3. Keep good baselines
2. Set the goal
3. Keep baseline
4. Look at OS, DB Time, Wait events
5. Use AWR, ASH to understand the issue
6. Optimize schema
   1. Review partitioning
7. Avoid issues not addressed by Smart Scan for example:
   1. Extensive parsing
   2. Row-by-row processing of big amount of data

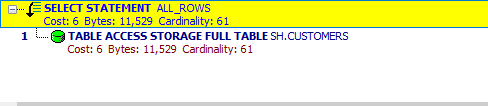
## Storage INDEX

Is a structure **in memory** on storage node that allows to eliminate some blocks based on values .

It's built automatically based on distribution of data in the table

## Allowing Smart Scan

Explain Plan will shows STORAGE keyword



## Invisible Indexes

In DW Mark indexes invisible to allow storage scan

**Marking PK or UK Constraint Index Invisible DOES NOT AFFECT CONSTRAINT. It's still enforced.**

Example:

|  |
| --- |
| create table t1 as  select \* from all\_objects  /  alter table t1 add constraint pk\_t1 primary key (object\_id)  / |

Statement:

|  |
| --- |
| select \* from t1 where object\_id < 100; |

Plan shows Range scan – no offloading:

|  |
| --- |
| SQL> select \* from t1 where object\_id < 100;  98 rows selected.  Elapsed: 00:00:00.20  Execution Plan  ----------------------------------------------------------  Plan hash value: 3772001702  ---------------------------------------------------------------------------------------------  | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |  ---------------------------------------------------------------------------------------------  | 0 | SELECT STATEMENT | | 88 | 11352 | 5 (0)| 00:00:01 |  | 1 | TABLE ACCESS BY INDEX ROWID BATCHED| T1 | 88 | 11352 | 5 (0)| 00:00:01 |  |\* 2 | INDEX RANGE SCAN | PK\_T1 | 88 | | 2 (0)| 00:00:01 |  ---------------------------------------------------------------------------------------------  Predicate Information (identified by operation id):  ---------------------------------------------------  2 - access("OBJECT\_ID"<100) |

Make index inviable

|  |
| --- |
| alter index pk\_t1 invisible  / |

Now plan shows STORAGE Access:

|  |
| --- |
| 98 rows selected.  Elapsed: 00:00:00.21  Execution Plan  ----------------------------------------------------------  Plan hash value: 3617692013  ----------------------------------------------------------------------------------  | Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |  ----------------------------------------------------------------------------------  | 0 | SELECT STATEMENT | | 88 | 11352 | 15 (7)| 00:00:01 |  |\* 1 | TABLE ACCESS STORAGE FULL| T1 | 88 | 11352 | 15 (7)| 00:00:01 |  ----------------------------------------------------------------------------------  Predicate Information (identified by operation id):  ---------------------------------------------------  1 - storage("OBJECT\_ID"<100)  filter("OBJECT\_ID"<100) |

**However, PK is still enforced:**

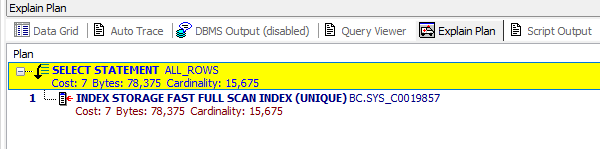
|  |
| --- |
| SQL> insert into t1  2 select \* from t1 where object\_id < 100 and rownum < 3;  insert into t1  \*  ERROR at line 1:  ORA-00001: unique constraint (BC.PK\_T1) violated |

## Create or Rebuild Indexes

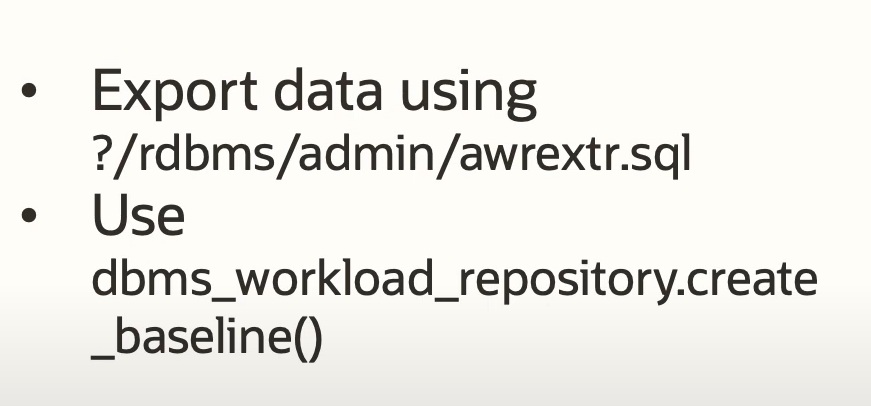
Is very fast, because it's done on Storage Node

## Index STORAGE Full Scan

Full index scan is a sequential operation and smart scan is much faster because it's massively parallel. Thus, full index scans should be avoided whenever possible UNLESS it's INDEX STORAGE scan.



## Keeping Baseline



## I/O Resource Manager (IORM)

* You cannot turn it off
* Used to prioritize databases in order to give them access to the same storage

# Known Issues

## DB Link between two Exadata databases

Before X8 DB link had to go to through external network because internal one was on InfiniBand

# Open Questions

## Persistent Memory (PMEM)

### Requires Database 19c and Exadata 19.3 (Exadata X8X)

PMEM is a new hardware solution, this is a memory that is almost as fast as DRAM, but is Persistent i.e. survives power shutdown. Tired in front of Flash and Disk

**Great for OLTP apps that require RANDOM READ**

OLTP that require random read was always a problem for hard drives because these operations are sequential in nature (require SEEK)

It was improved with hard drive cache first

And **more improved with FLASH which is parallel in nature. Flash allowed latency of <200 nanosec.**

* To get maximum performance Database connects **to PMEM via RDMA i.e. via network card, and WITHOUT invoking a process on a storage server**.
* Reads are increased 10x – to 19 nanosec for 8K block
* Used for Redo Logs

