Franky Weber Faust
May 2022

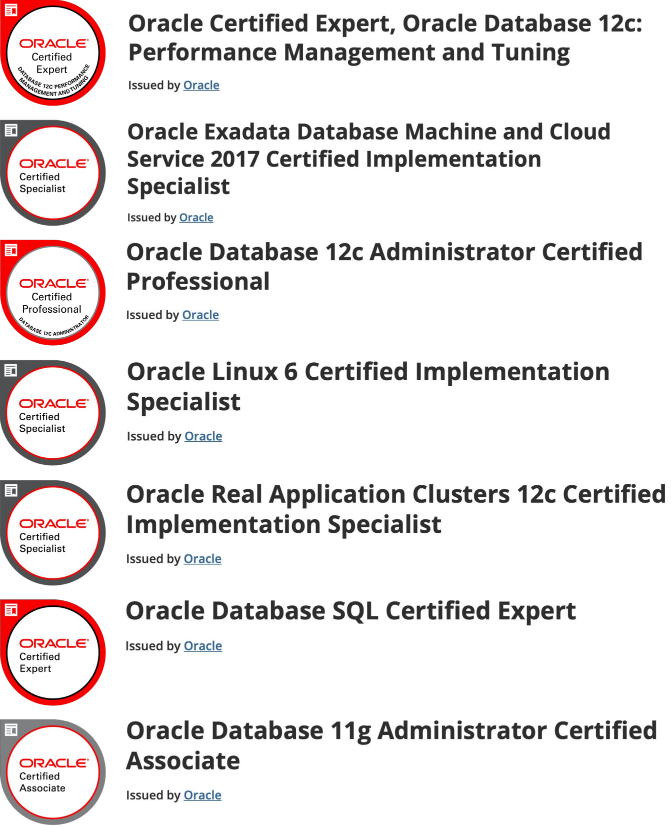
Pythian

L♥VE YOUR DATA

Exadata Smart Scan

~~Live~~ and uncensored

hroug '22
Spring | Proljeće
17. - 19.05.2022., Tuhelj



Oracle Certified Expert, Oracle Database 12c: Performance Management and Tuning

Issued by Oracle

Oracle Exadata Database Machine and Cloud Service 2017 Certified Implementation Specialist

Issued by Oracle

Oracle Database 12c Administrator Certified Professional

Issued by Oracle

Oracle Linux 6 Certified Implementation Specialist

Issued by Oracle

Oracle Real Application Clusters 12c Certified Implementation Specialist

Issued by Oracle

Oracle Database SQL Certified Expert

Issued by Oracle

Oracle Database 11g Administrator Certified Associate

Issued by Oracle

ORACLE ACE

FRANKY WEBER FAUST

- Lead Database Consultant at Pythian
- 31 years old
- Based in Brazil
- Writer at OTNLA and Lore Data Blog
- Speaker at conferences around the world
- High Availability specialist
- Performance researcher
- Exadata, RAC, DataGuard, GoldenGate
- AcroYoga practitioner
- Guitar player

loredata.com.br

tatiaraujo.arq
@CORPORISACRO
@MPANETONE
@FRANKYWEBER
COSTA DA LAGOA

WASHINGTON, DISTRICT OF COLUMBIA
9°C
@TATIARAUJO.ARQ
@CORPORISACRO

INUKSUK - VANCOUVER, BC
@tatiaraujo.arq

Keep in touch

E-mail: faust@pythian.com or franky@loredata.com.br

Blog: http://loredata.com.br/blog

Facebook: https://facebook.com/08Franky.Weber

Instagram: https://www.instagram.com/frankyweber/

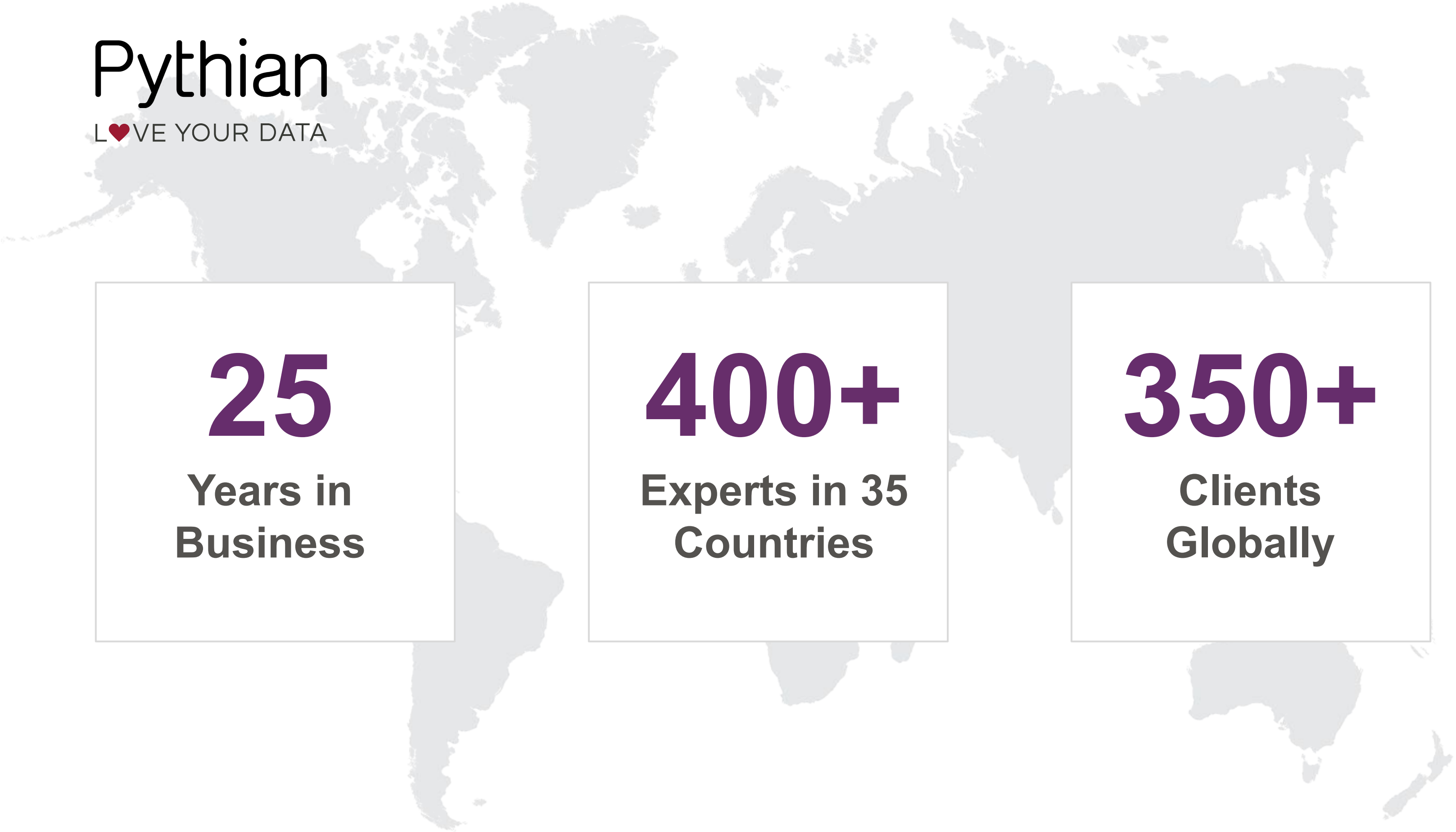
Twitter: https://twitter.com/frankyweber

LinkedIn: https://linkedin.com/in/frankyweber/en

Oracle ACE: https://bit.ly/2YxU6bK

Pythian

L♥VE YOUR DATA



25

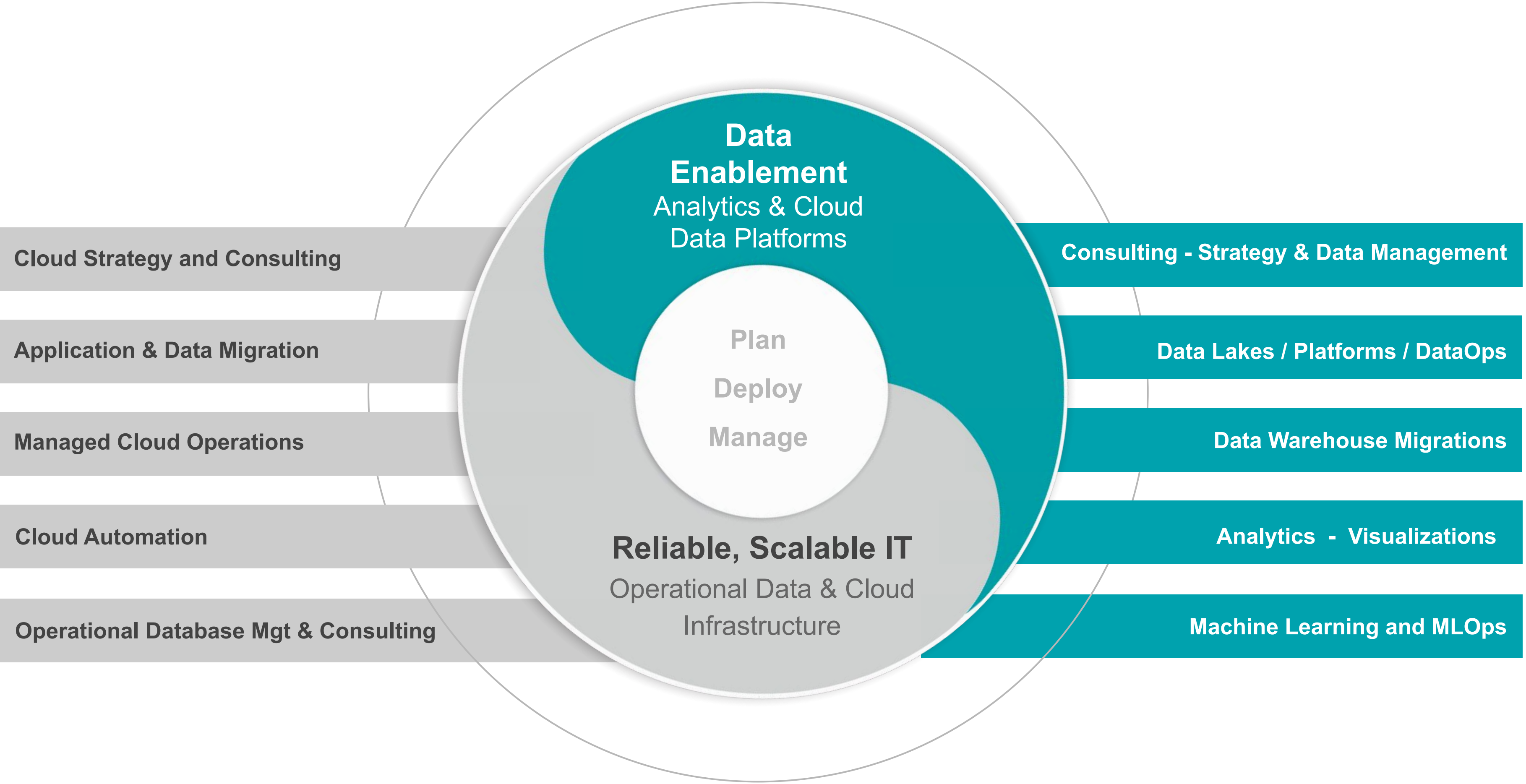
Years in Business

400+

Experts in 35 Countries

350+

Clients Globally



Data Enablement
Analytics & Cloud Data Platforms

Plan
Deploy
Manage

Reliable, Scalable IT
Operational Data & Cloud Infrastructure

- Cloud Strategy and Consulting
- Application & Data Migration
- Managed Cloud Operations
- Cloud Automation
- Operational Database Mgt & Consulting

- Consulting - Strategy & Data Management
- Data Lakes / Platforms / DataOps
- Data Warehouse Migrations
- Analytics - Visualizations
- Machine Learning and MLOps

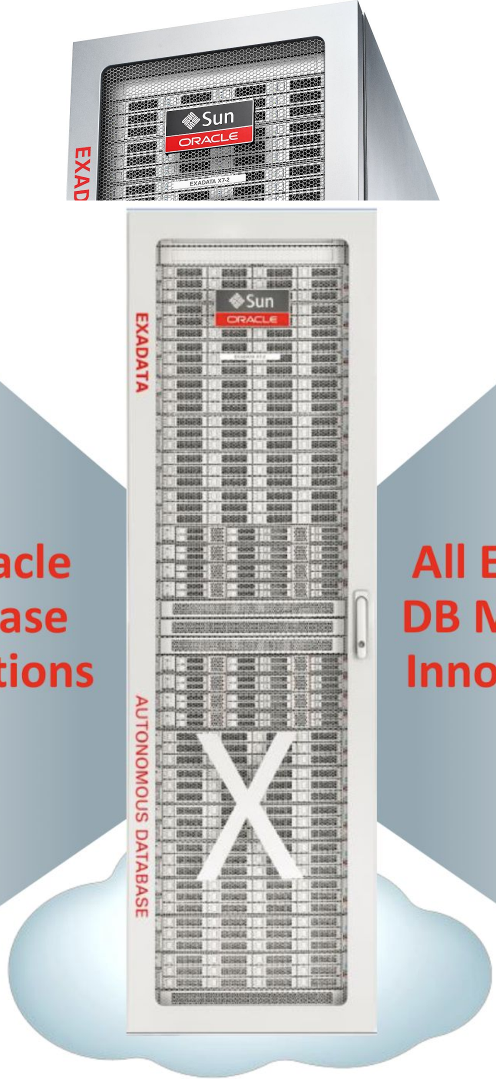
AGENDA

- Quick overview
- Understanding Smart Scan
- Live session

Quick Overview

- Multitenant
- In-Memory DB
- Real Application Clusters
- Active Data Guard
- Partitioning
- Advanced Compression
- Advanced Security, Label Security, DB Vault
- Real Application Testing
- Advanced Analytics, Spatial and Graph
- Management Packs for Oracle Database

All Oracle Database Innovations

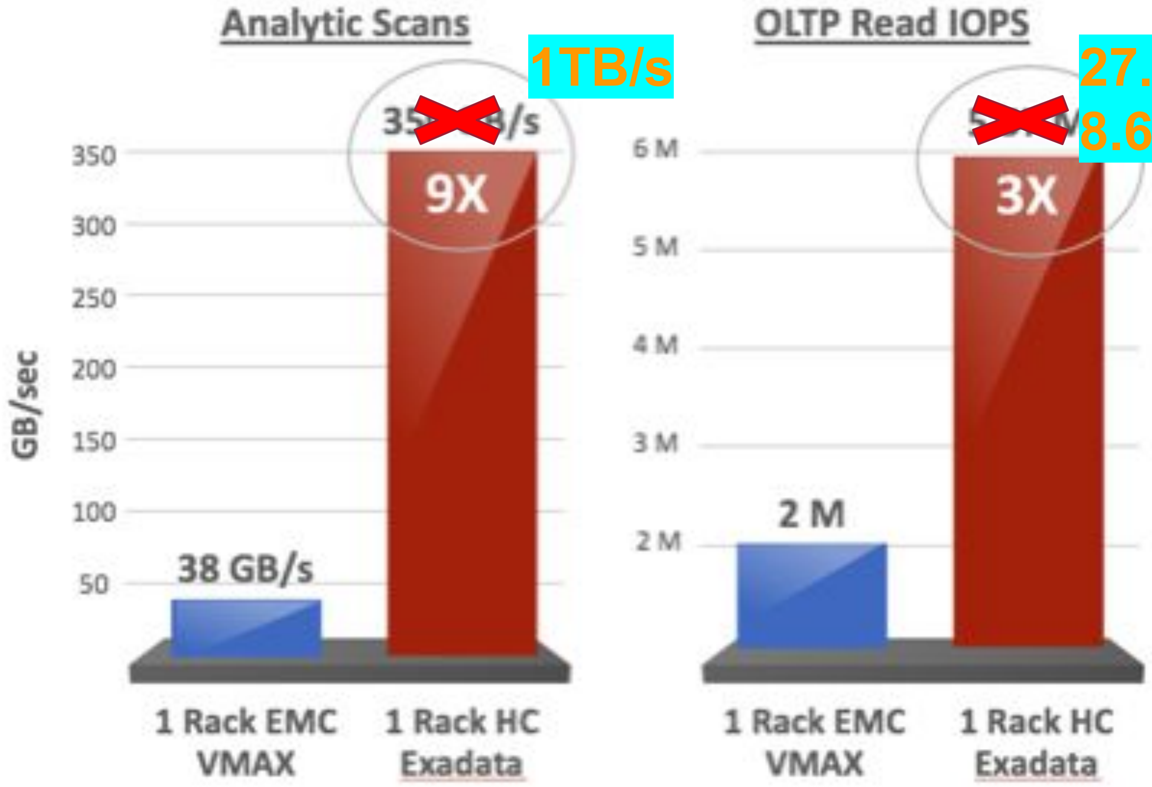


All Exadata DB Machine Innovations

- Offload SQL to Storage
- InfiniBand Fabric
- Smart Flash Cache, Log
- Storage Indexes
- Columnar Flash Cache
- Hybrid Columnar Compression
- I/O Resource Management
- Network Resource Management
- In-Memory Fault Tolerance
- Exafusion Direct-to-Wire Protocol

Features that leverage performance

- Offloading
- Smart Scan
- Storage Index
- IORM
- HCC
- ESFC



1TB/s

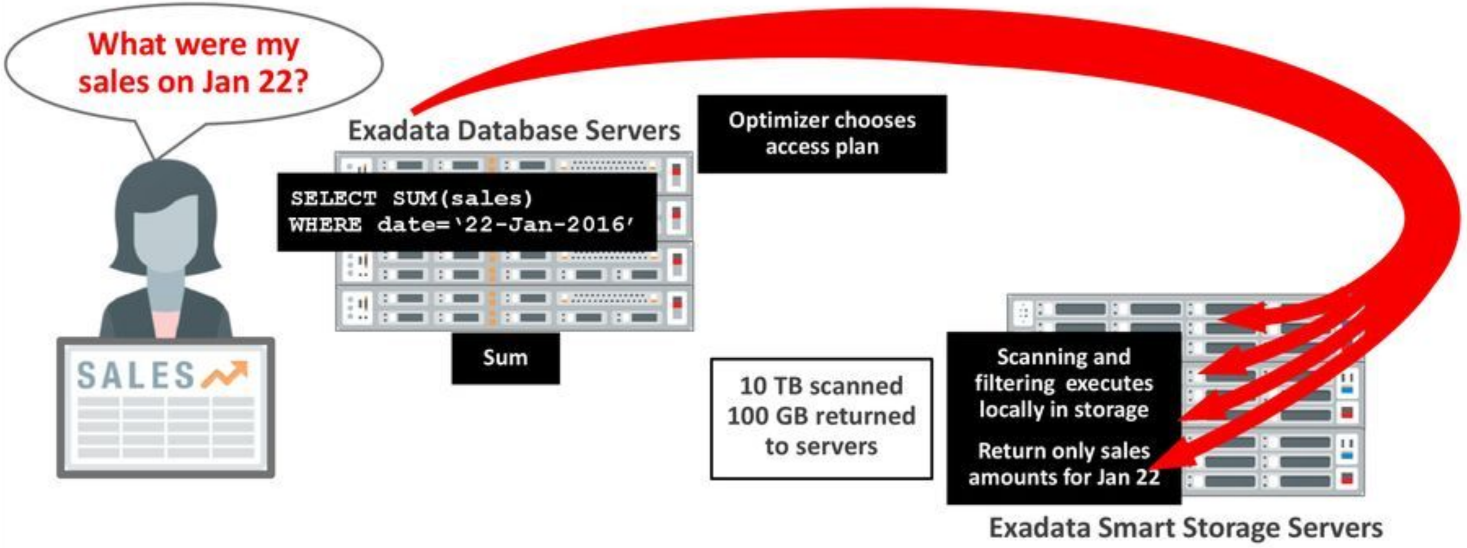
27.6M read IO/s
8.6M write IO/s

Features that leverage performance

- Offloading
- Smart Scan
- Storage Index
- IORM
- HCC
- ESFC

Exadata Smart Scan

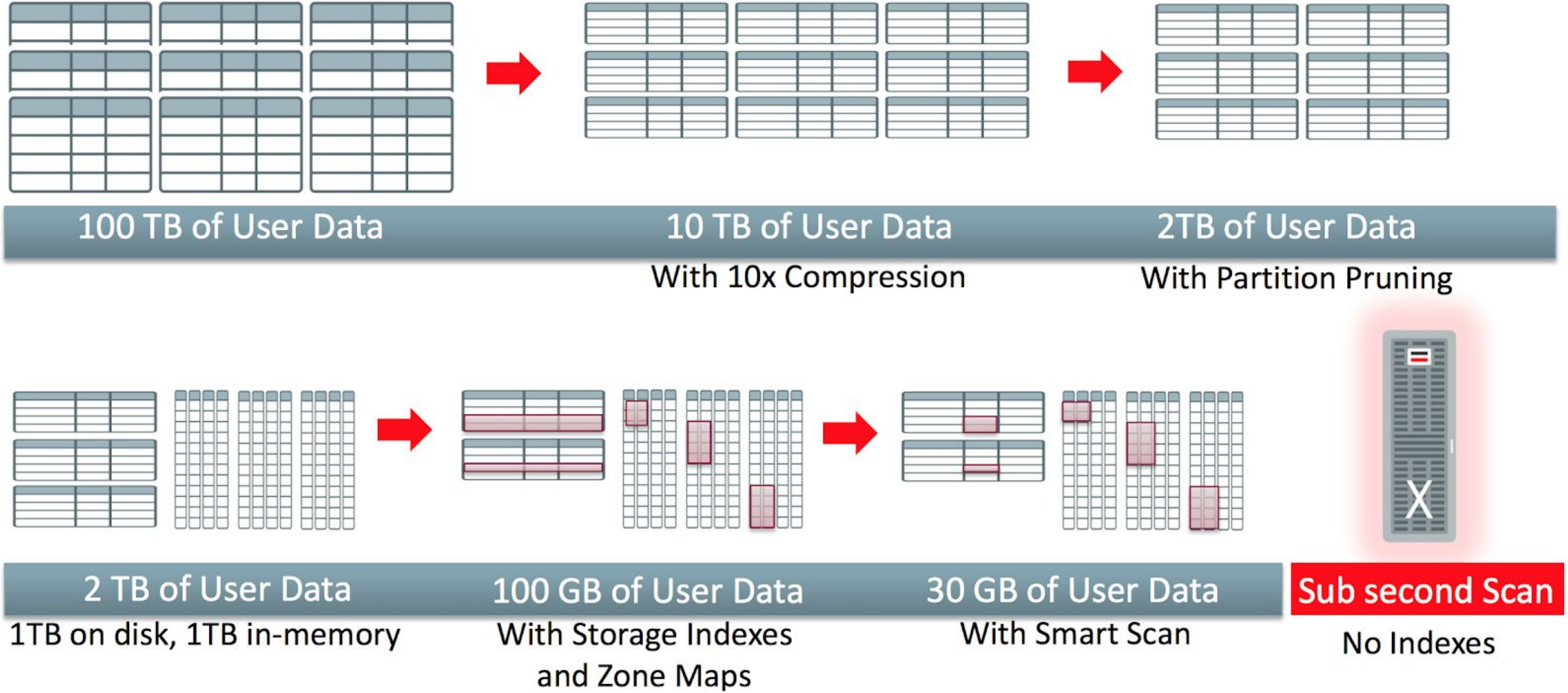
Move Queries to Data, Not Data to Queries



```
SELECT SUM(sales)
WHERE date='22-Jan-2016'
```

What were my sales on Jan 22?
SALES
Exadata Database Servers
Optimizer chooses access plan
Sum
10 TB scanned 100 GB returned to servers
Scanning and filtering executes locally in storage
Return only sales amounts for Jan 22
Exadata Smart Storage Servers

Features that leverage performance



100 TB of User Data

10 TB of User Data

With 10x Compression

2TB of User Data

With Partition Pruning

2 TB of User Data

1TB on disk, 1TB in-memory

100 GB of User Data

With Storage Indexes and Zone Maps

30 GB of User Data

With Smart Scan

Sub second Scan

No Indexes

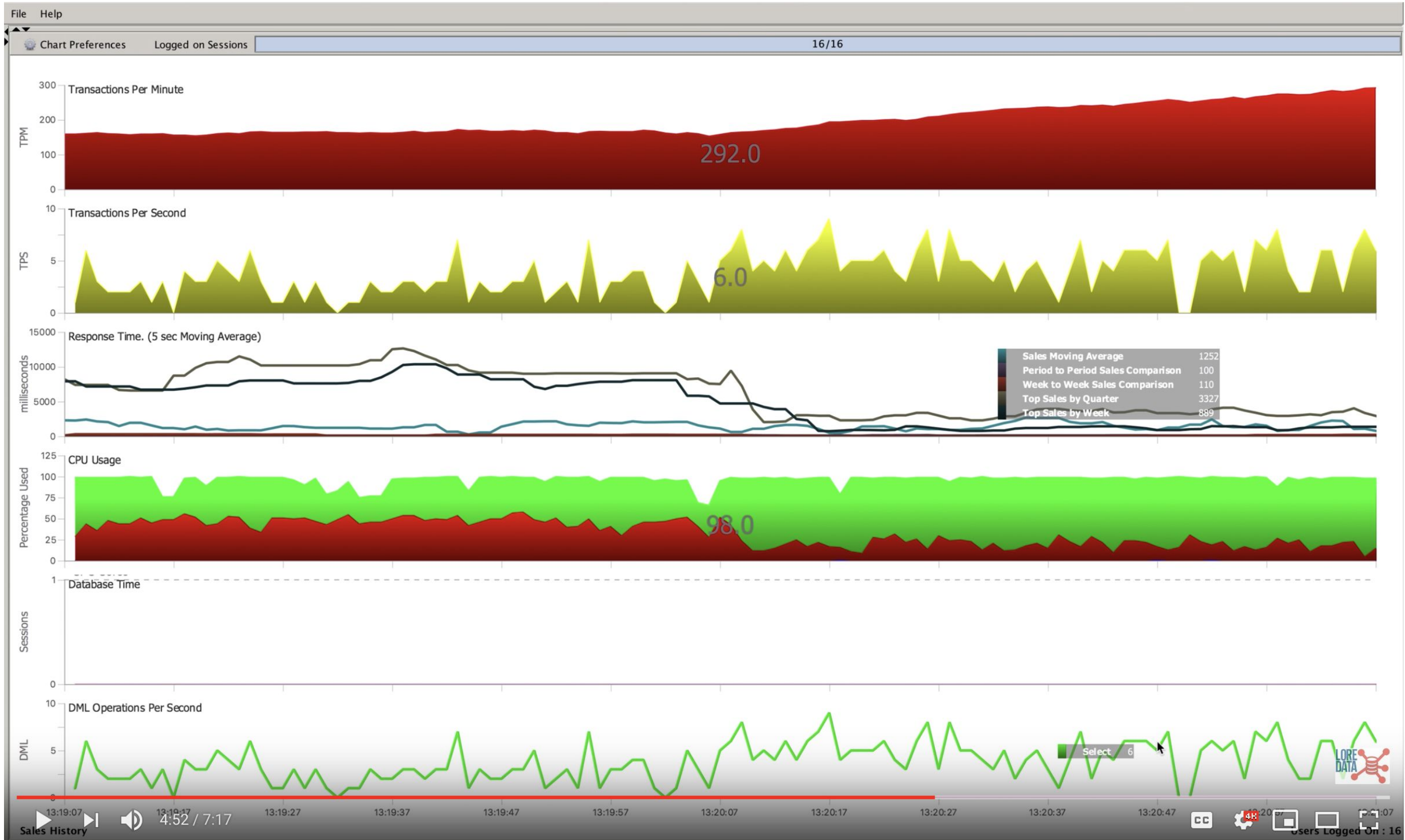
Offloading Goals

- Reduces data transfer from the storage system to the DB server
- Reduces the compute processing needed in the DB server
- Reduces the time needed to access data blocks on disks

Features that leverage performance

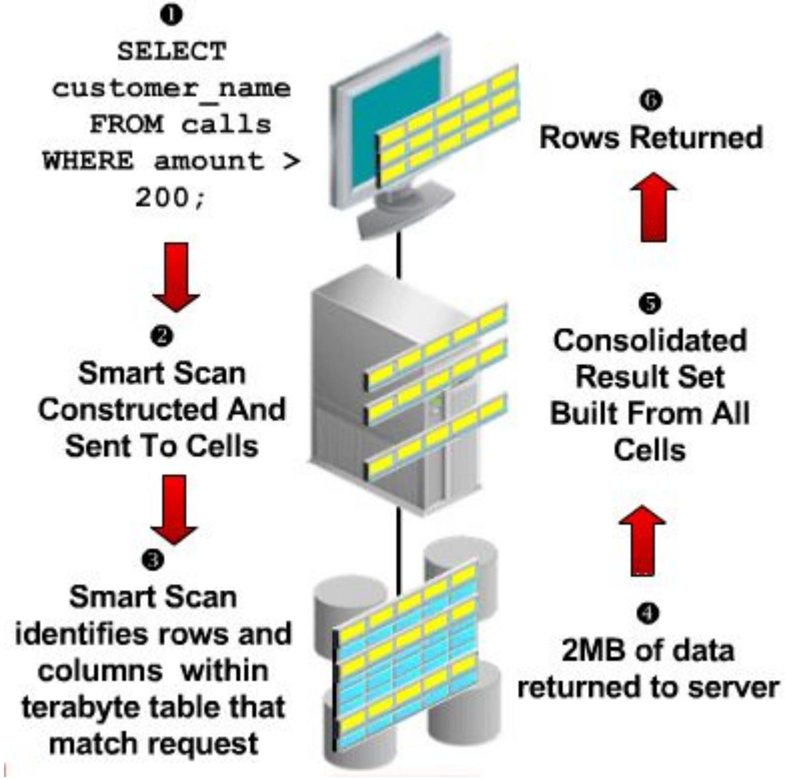
- https://www.youtube.com/watch?v=2lgoUL2eG2A

- Search YouTube for "Franky Weber Faust"



Smart Scan and Offloading

- Column Projection
- Predicate Filtering
- Bloom Filtering
- Simple Joins
- Storage Indexes
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading


❶
SELECT customer_name FROM calls WHERE amount > 200;
❷
Smart Scan Constructed And Sent To Cells
❸
Smart Scan identifies rows and columns within terabyte table that match request
❹
2MB of data returned to server
❺
Consolidated Result Set Built From All Cells
❻
Rows Returned

© The Pythian Group Inc., 2018

Smart Scan and Offloading

- Column Projection
- Predicate Filtering
- Bloom Filtering
- Simple Joins
- Storage Indexes
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

- "FTS is bad" biggest myth in SQL Tuning
- What is a Full Table Scan?
  - SQL Execution Access Method
    - Read data from table, while applying filters
  - Same mechanics apply to:
    - Full (sub)Partition Scan
    - Index Fast Full Scan
- Always available regardless of SQL construct

Smart Scan and Offloading

- Column Projection
- Predicate Filtering
- Bloom Filtering
- Simple Joins
- Storage Indexes
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

- How does FTS work?
  - Whole segment is read
    - From segment header up to (L)HWM
    - Blocks read regardless if empty or not
  - Several blocks read at once
    - This is key to understand full potential of FTS
  - Data goes into SGA => db file scattered read
  - Data goes into session PGA => direct path read

Smart Scan and Offloading

- Column Projection
- Predicate Filtering
- Bloom Filtering
- Simple Joins
- Storage Indexes
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

- Why FTS rocks?
  - Can crunch A LOT of data efficiently (*)
    - Couple of more data than index scans per disk
  - Full Scan (~200MB/s per disk)
    - Wait IO seek + latency per (large) chunk
    - Parallelizes well, increasing bandwidth (GB/s)
  - Index Scan (~1.5MB/s per disk)
    - Wait IO seek + latency per block

Smart Scan and Offloading

- Column Projection
- Predicate Filtering
- Bloom Filtering
- Simple Joins
- Storage Indexes
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

- Why FTS doesn’t rock?
  - FTS read ~100x faster than index
  - Needs to read the whole segment
    - GB to read just few rows
  - Concurrent users share bandwidth
    - More users less resource for each
  - Index faster if filters less than ~1% of data
    - Assuming data comes all from disk

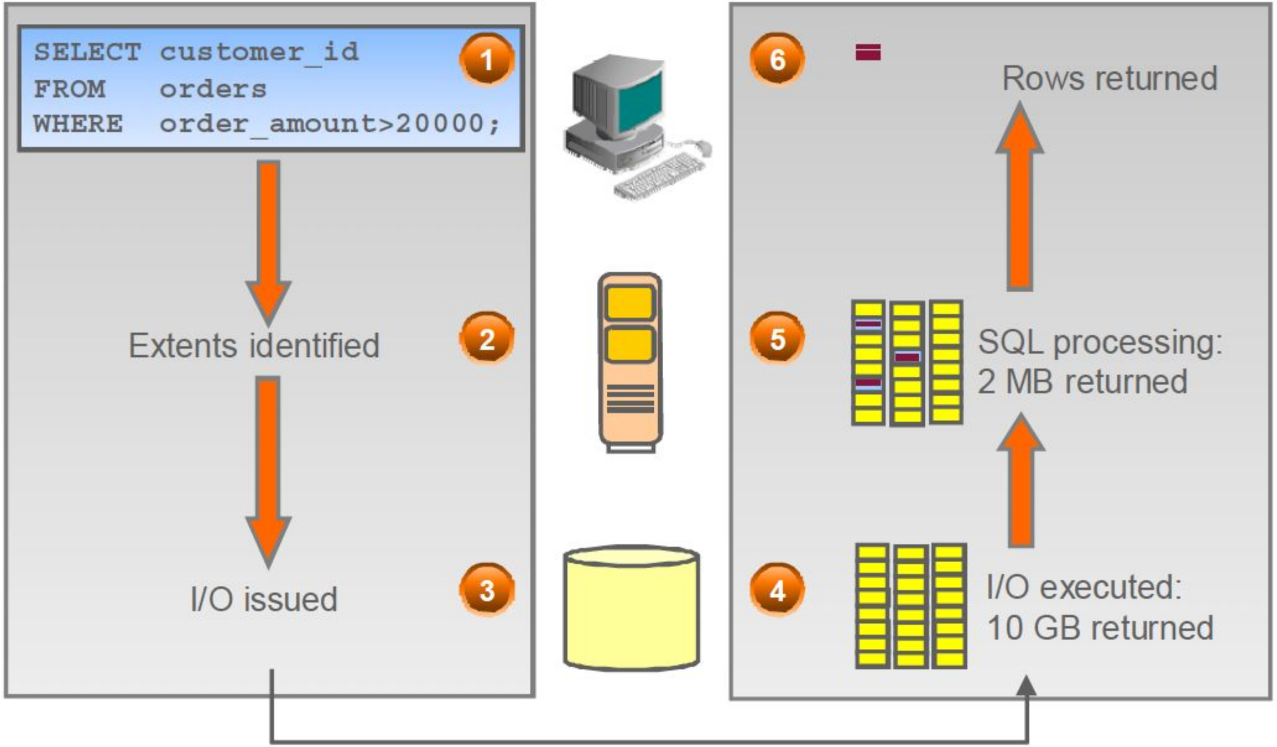
Smart Scan and Offloading

- Column Projection
- Predicate Filtering
- Bloom Filtering
- Simple Joins
- Storage Indexes
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

- What’s the challenge then?
  - If < 1% index otherwise FTS? Easy right?
  - NO!!!!
  - Buffer Cache complicates things a lot
    - It saves large % of disk reads
      - Especially for index blocks, touched often
    - Buffer Cache is transitory in nature
      - No guarantee block X will be there when needed
    - Complex for CBO to consider caching
      - Algorithms assume each read is a physical one (kind of)

Smart Scan and Offloading

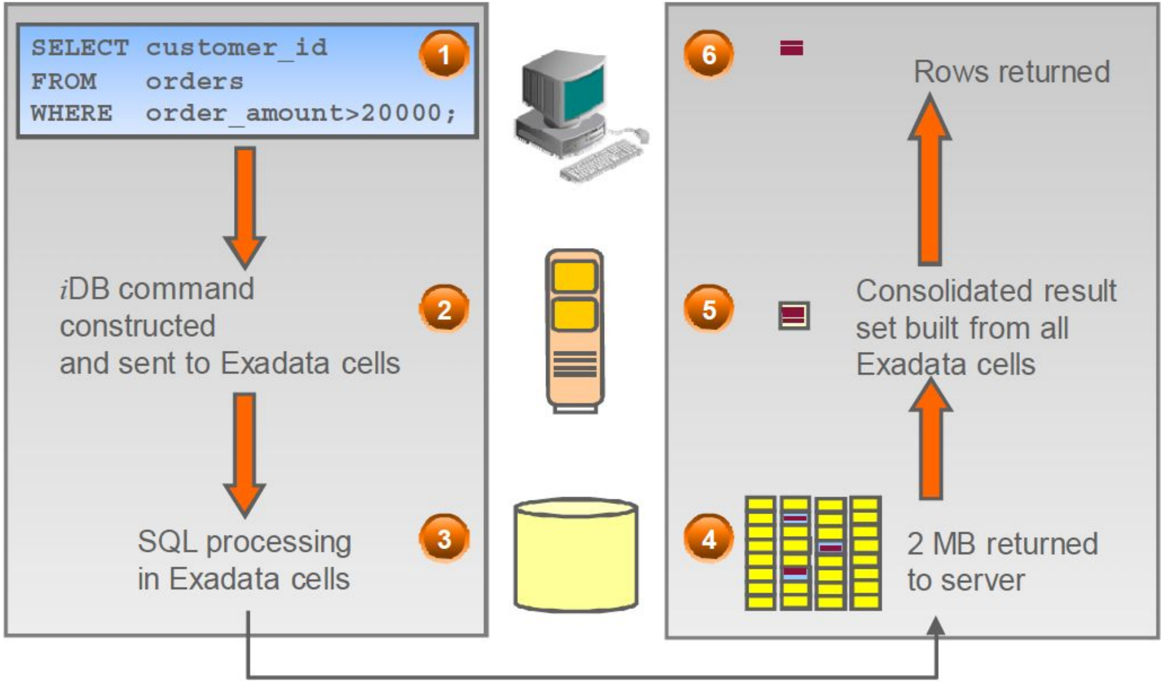
- Non-Exadata
- Classic Storage system



```
SELECT customer_id
FROM   orders
WHERE  order_amount>20000;
```

Smart Scan and Offloading

- Exadata
- Smart Storage system
- Smart Scan
  - Exadata
    - Compatible > 11.2
    - cell.smart_scan_capable = true
  - FTS or FFIS
  - Direct Path Read



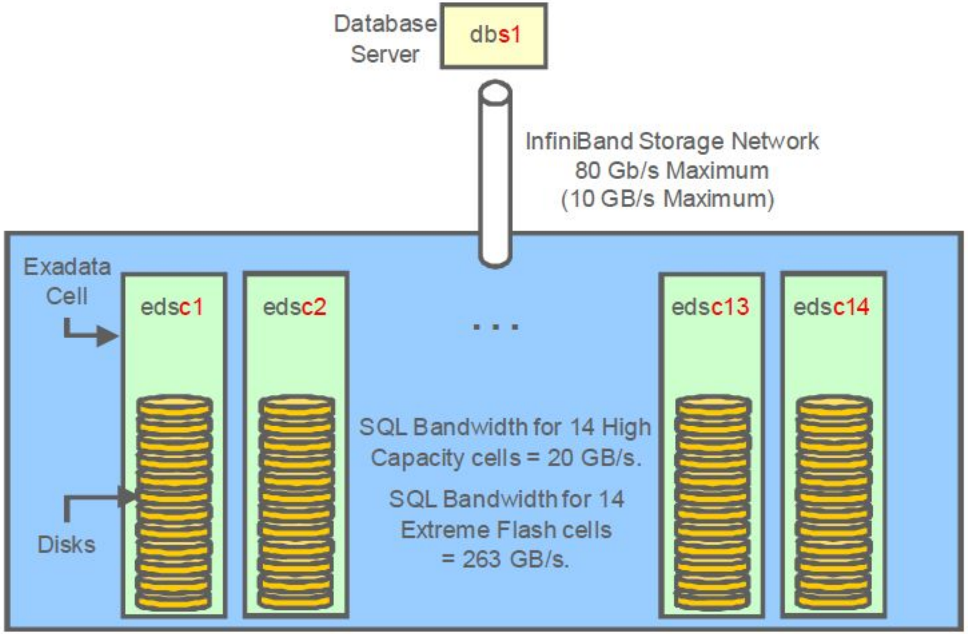
```
SELECT customer_id
FROM   orders
WHERE  order_amount>20000;
```

1
2
iDB command constructed and sent to Exadata cells
3
SQL processing in Exadata cells
4
2 MB returned to server
5
Consolidated result set built from all Exadata cells
6
Rows returned

Video 22:46

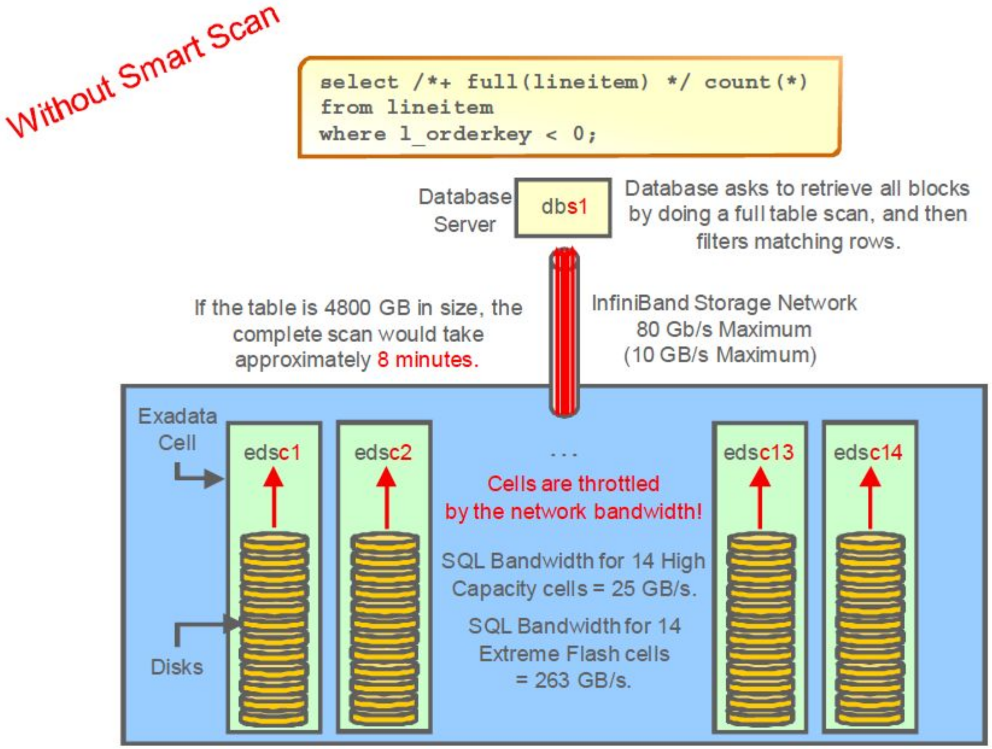
Smart Scan and Offloading

- InfiniBand Network is the pipe between Database and Storage Servers
- Maximum throughput is 80Gb/s or 10GB/s
- SQL Bandwidth for HC is 25GB/s on a full rack while for EF it is 263GB/s


Database Server
dbs1
InfiniBand Storage Network
80 Gb/s Maximum
(10 GB/s Maximum)
Exadata Cell
edsc1
edsc2
. . .
edsc13
edsc14
SQL Bandwidth for 14 High Capacity cells = 20 GB/s.
SQL Bandwidth for 14 Extreme Flash cells = 263 GB/s.
Disks

Smart Scan and Offloading

- Without Smart Scan the IB Network throttles th cells at 10GB/s
- LineItem table being fully scanned without sma scan will be returned to the database in 8 minutes
- `(4800GB) / (10GB/s) = 480s or 8mins`



Without Smart Scan

```
select /*+ full(lineitem) */ count(*)
from lineitem
where l_orderkey < 0;
```

Database asks to retrieve all blocks by doing a full table scan, and then filters matching rows.

If the table is 4800 GB in size, the complete scan would take approximately 8 minutes.

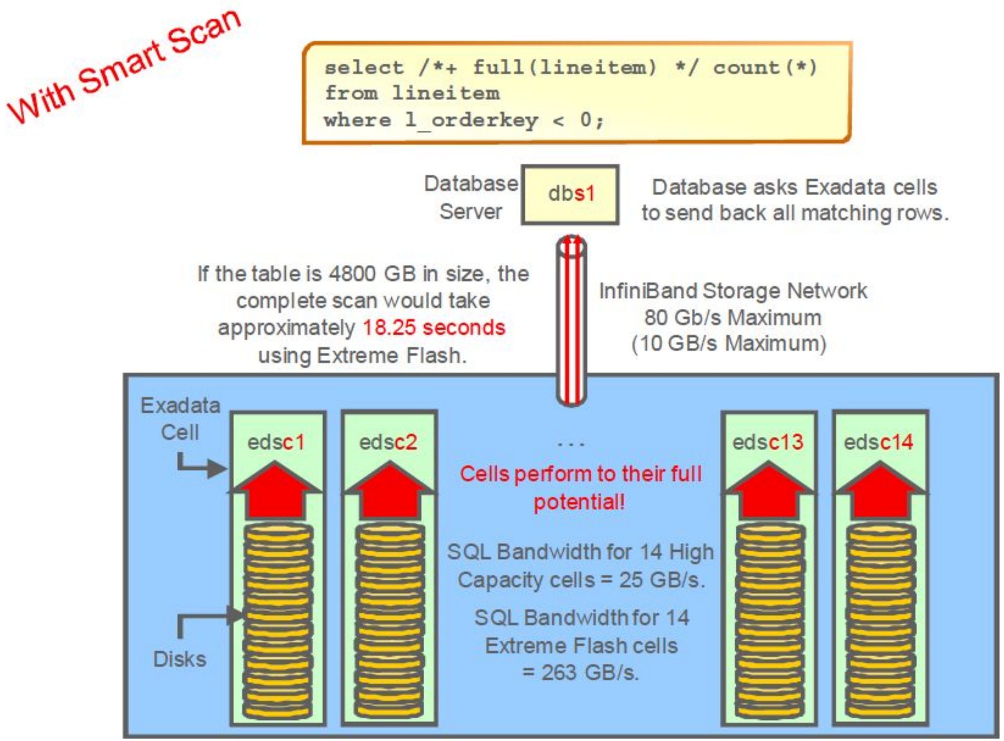
Cells are throttled by the network bandwidth!

SQL Bandwidth for 14 High Capacity cells = 25 GB/s.

SQL Bandwidth for 14 Extreme Flash cells = 263 GB/s.

Smart Scan and Offloading

- With Smart Scan Cells can perform with their full potential and only return the data that matters to the DB
- LineItem table being fully scanned with smart scan will be returned to the database in 3mins 12s in a full rack HC or 18s in a EF
- `(4800GB) / (25GB/s) = 192s or 3min12s`
- `(4800GB) / (263GB/s) = 18.25s`



With Smart Scan

```
select /*+ full(lineitem) */ count(*)
from lineitem
where l_orderkey < 0;
```

Database asks Exadata cells to send back all matching rows.

If the table is 4800 GB in size, the complete scan would take approximately 18.25 seconds using Extreme Flash.

Cells perform to their full potential!

SQL Bandwidth for 14 High Capacity cells = 25 GB/s.

SQL Bandwidth for 14 Extreme Flash cells = 263 GB/s.

Smart Scan and Offloading

```
select sum(amount_sold) from sales;
```

Offloading Capabilities

- **Column Projection**
- Predicate Filtering
- Storage Indexes
- Bloom Filtering
- Simple Joins
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

cust_id	product	order_date	quantity	amount_sold	ship_date
1025	xyz	20/07/2019	300	500	21/07/2019
3028	zyx	24/07/2019	150	800	25/07/2019
4823	yzx	24/07/2019	40	1200	25/07/2019
1239	xzy	25/07/2019	30	400	26/07/2019
2913	zxy	26/07/2019	80	300	27/07/2019

Smart Scan and Offloading

Offloading Capabilities

- Column Projection
- **Predicate Filtering**
- Storage Indexes
- Bloom Filtering
- Simple Joins
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

```
select * from sales
where ship_date = '21/07/2019';
```

cust_id	product	order_date	quantity	amount_sold	ship_date
1025	xyz	20/07/2019	300	500	21/07/2019
3028	zyx	24/07/2019	150	800	25/07/2019
4823	yzx	24/07/2019	40	1200	25/07/2019
1239	xzy	25/07/2019	30	400	26/07/2019
2913	zxy	26/07/2019	80	300	27/07/2019

Smart Scan and Offloading

Offloading Capabilities

- **Column Projection**
- **Predicate Filtering**
- Storage Indexes
- Bloom Filtering
- Simple Joins
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

```
select product from sales
where ship_date = '25/07/2019';
```

cust_id	product	order_date	quantity	amount_sold	ship_date
1025	xyz	20/07/2019	300	500	21/07/2019
3028	zyx	24/07/2019	150	800	25/07/2019
4823	yzx	24/07/2019	40	1200	25/07/2019
1239	xzy	25/07/2019	30	400	26/07/2019
2913	zxy	26/07/2019	80	300	27/07/2019

Smart Scan and Offloading

Offloading Capabilities

- Column Projection
- Predicate Filtering
- **Storage Indexes**
- Bloom Filtering
- Simple Joins
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- Fast File Creation
- Incremental Backup Offloading

```
select product from sales
where ship_date = '25/07/2019';
```

cust_id	product	order_date	quantity	amount_sold	ship_date
7813	xxy	19/07/2019	850	3000	20/07/2019
1025	xyz	20/07/2019	300	500	21/07/2019
3028	zyx	24/07/2019	150	800	25/07/2019
4823	yzx	24/07/2019	40	1200	25/07/2019
1239	xzy	25/07/2019	30	400	26/07/2019
2913	zxy	26/07/2019	80	300	27/07/2019
2018	xzy	28/07/2019	110	200	29/07/2019

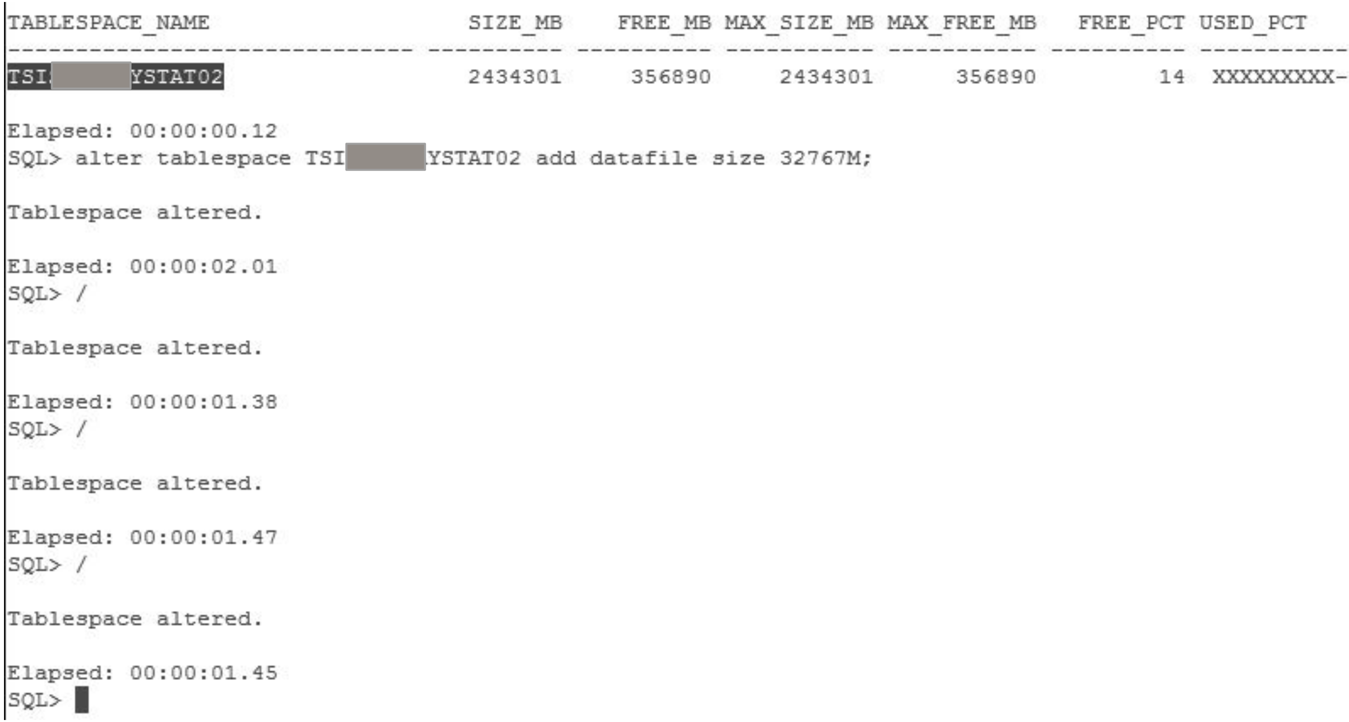
Let's NOT go LIVE
:(

Video 40:00

Smart Scan and Offloading

Practice Examples

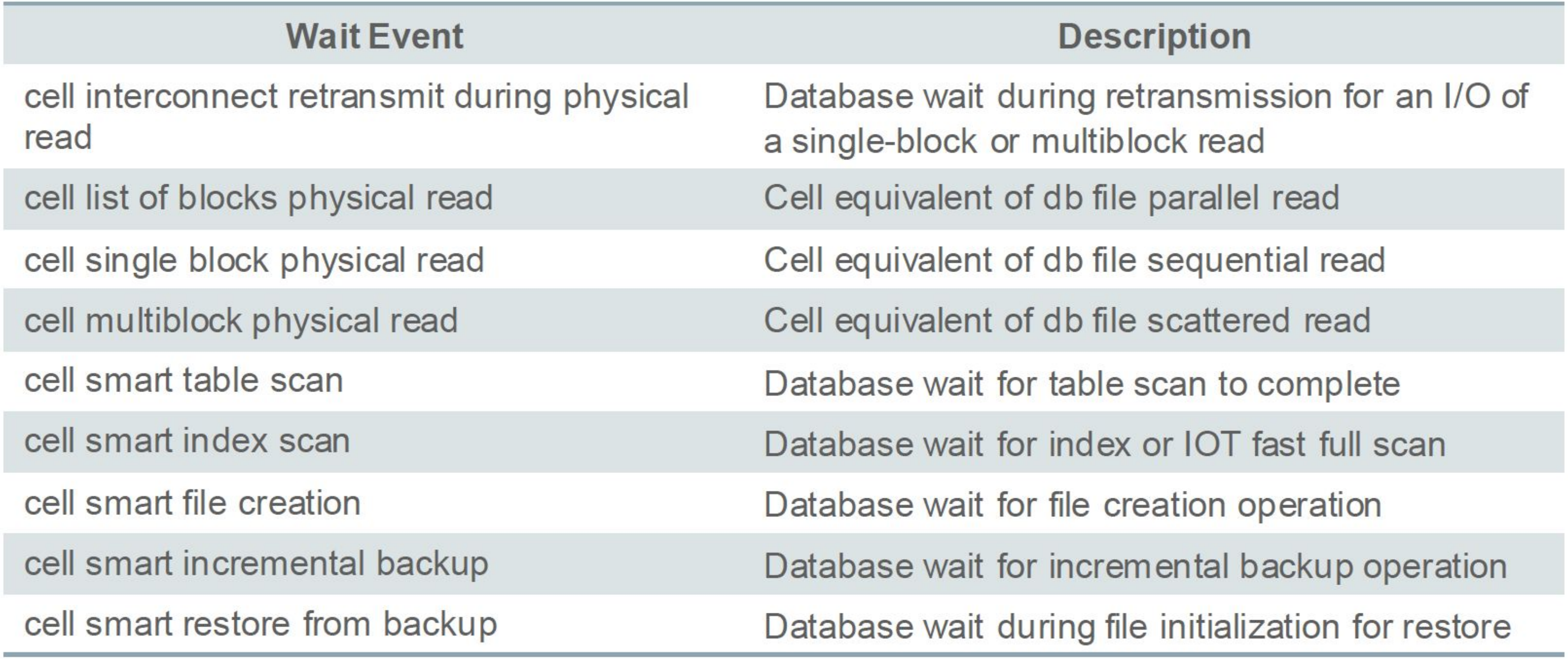
- Column Projection
- Predicate Filtering
- Storage Indexes
- Bloom Filtering
- Simple Joins
- Function Offloading
- Virtual Column Evaluation
- Decryption
- Decompression
- **Fast File Creation**
- Incremental Backup Offloading



```
TABLESPACE_NAME                   SIZE_MB    FREE_MB MAX_SIZE_MB MAX_FREE_MB   FREE_PCT USED_PCT
------------------------------ ---------- ---------- ----------- ----------- ---------- ----------
TSI[redacted]YSTAT02                 2434301     356890     2434301      356890         14 XXXXXXXXX-

Elapsed: 00:00:00.12
SQL> alter tablespace TSI[redacted]YSTAT02 add datafile size 32767M;

Tablespace altered.

Elapsed: 00:00:02.01
SQL> /

Tablespace altered.

Elapsed: 00:00:01.38
SQL> /

Tablespace altered.

Elapsed: 00:00:01.47
SQL> /

Tablespace altered.

Elapsed: 00:00:01.45
SQL> █
```

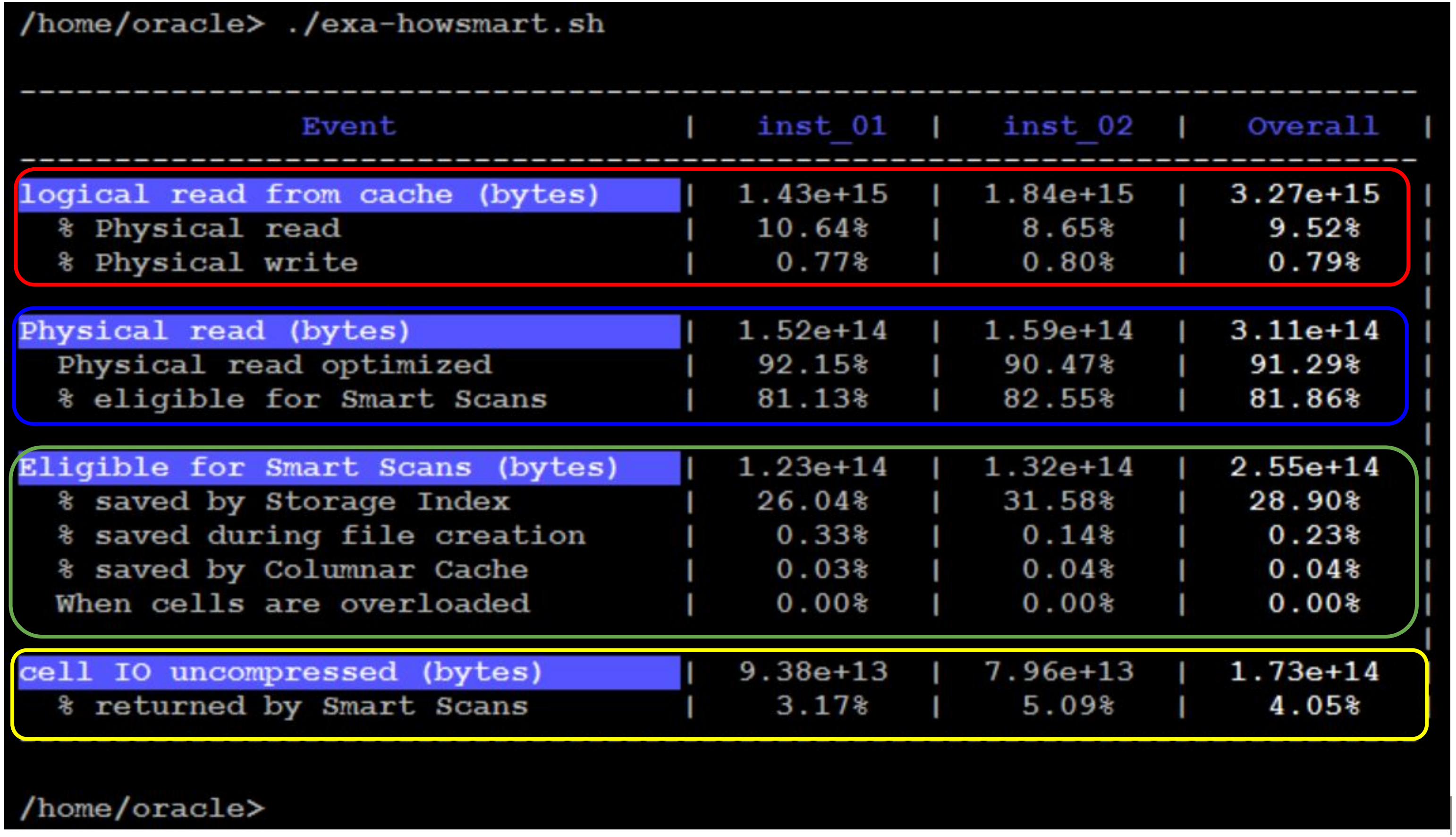
Smart Scan and Offloading

Exadata related Wait Events



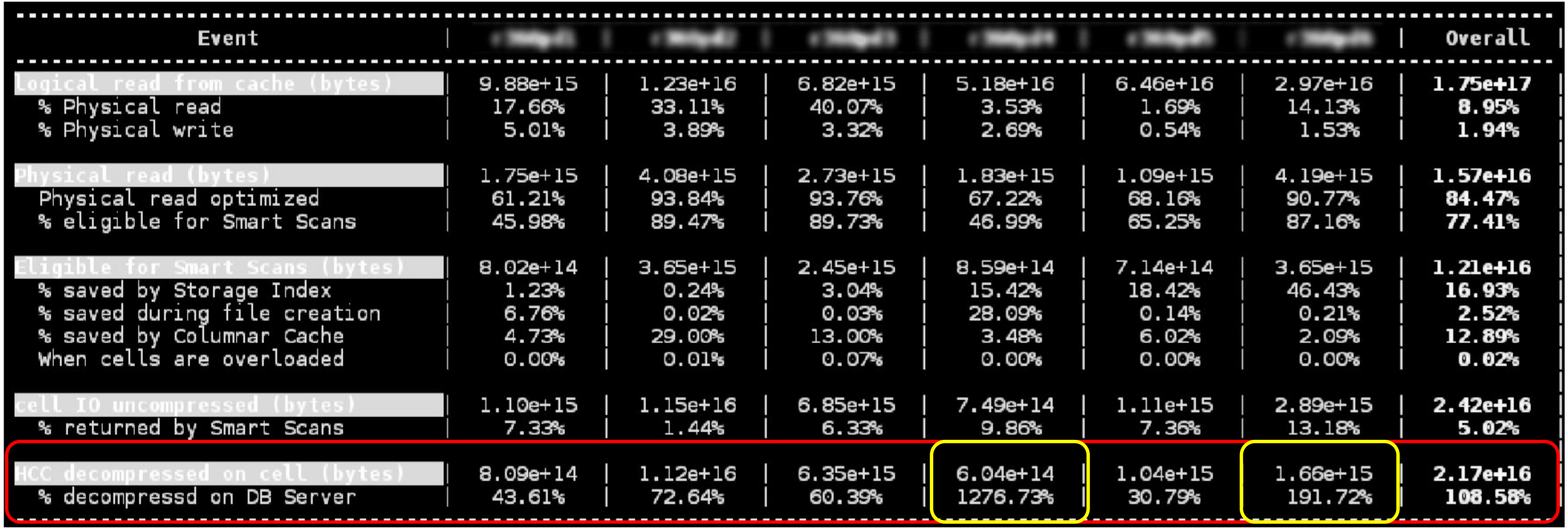
Wait Event	Description
cell interconnect retransmit during physical read	Database wait during retransmission for an I/O of a single-block or multiblock read
cell list of blocks physical read	Cell equivalent of db file parallel read
cell single block physical read	Cell equivalent of db file sequential read
cell multiblock physical read	Cell equivalent of db file scattered read
cell smart table scan	Database wait for table scan to complete
cell smart index scan	Database wait for index or IOT fast full scan
cell smart file creation	Database wait for file creation operation
cell smart incremental backup	Database wait for incremental backup operation
cell smart restore from backup	Database wait during file initialization for restore

exa-howsmart.sh



```
/home/oracle> ./exa-howsmart.sh

-------------------------------------------------------------------------
              Event              |   inst_01   |   inst_02   |   Overall   |
-------------------------------------------------------------------------
logical read from cache (bytes)  |   1.43e+15  |   1.84e+15  |   3.27e+15  |
  % Physical read                |    10.64%   |     8.65%   |     9.52%   |
  % Physical write               |     0.77%   |     0.80%   |     0.79%   |
                                                                           |
Physical read (bytes)            |   1.52e+14  |   1.59e+14  |   3.11e+14  |
  Physical read optimized        |    92.15%   |    90.47%   |    91.29%   |
  % eligible for Smart Scans     |    81.13%   |    82.55%   |    81.86%   |
                                                                           |
Eligible for Smart Scans (bytes) |   1.23e+14  |   1.32e+14  |   2.55e+14  |
  % saved by Storage Index       |    26.04%   |    31.58%   |    28.90%   |
  % saved during file creation   |     0.33%   |     0.14%   |     0.23%   |
  % saved by Columnar Cache      |     0.03%   |     0.04%   |     0.04%   |
  When cells are overloaded      |     0.00%   |     0.00%   |     0.00%   |
                                                                           |
cell IO uncompressed (bytes)     |   9.38e+13  |   7.96e+13  |   1.73e+14  |
  % returned by Smart Scans      |     3.17%   |     5.09%   |     4.05%   |


/home/oracle>
```

exa-howsmart.sh



```
------------------------------------------------------------------------------------------------------------------------
              Event                 | [redacted]  | [redacted]  | [redacted]  | [redacted]  | [redacted]  | [redacted]  |  Overall  |
------------------------------------------------------------------------------------------------------------------------
Logical read from cache (bytes)     |  9.88e+15   |  1.23e+16   |  6.82e+15   |  5.18e+16   |  6.46e+16   |  2.97e+16   | 1.75e+17  |
  % Physical read                   |   17.66%    |   33.11%    |   40.07%    |   3.53%     |   1.69%     |   14.13%    |  8.95%    |
  % Physical write                  |   5.01%     |   3.89%     |   3.32%     |   2.69%     |   0.54%     |   1.53%     |  1.94%    |

Physical read (bytes)               |  1.75e+15   |  4.08e+15   |  2.73e+15   |  1.83e+15   |  1.09e+15   |  4.19e+15   | 1.57e+16  |
  Physical read optimized           |   61.21%    |   93.84%    |   93.76%    |   67.22%    |   68.16%    |   90.77%    |  84.47%   |
  % eligible for Smart Scans        |   45.98%    |   89.47%    |   89.73%    |   46.99%    |   65.25%    |   87.16%    |  77.41%   |

Eligible for Smart Scans (bytes)    |  8.02e+14   |  3.65e+15   |  2.45e+15   |  8.59e+14   |  7.14e+14   |  3.65e+15   | 1.21e+16  |
  % saved by Storage Index          |   1.23%     |   0.24%     |   3.04%     |   15.42%    |   18.42%    |   46.43%    |  16.93%   |
  % saved during file creation      |   6.76%     |   0.02%     |   0.03%     |   28.09%    |   0.14%     |   0.21%     |  2.52%    |
  % saved by Columnar Cache         |   4.73%     |   29.00%    |   13.00%    |   3.48%     |   6.02%     |   2.09%     |  12.89%   |
  When cells are overloaded         |   0.00%     |   0.01%     |   0.07%     |   0.00%     |   0.00%     |   0.00%     |  0.02%    |

cell IO uncompressed (bytes)        |  1.10e+15   |  1.15e+16   |  6.85e+15   |  7.49e+14   |  1.11e+15   |  2.89e+15   | 2.42e+16  |
  % returned by Smart Scans         |   7.33%     |   1.44%     |   6.33%     |   9.86%     |   7.36%     |   13.18%    |  5.02%    |

HCC decompressed on cell (bytes)    |  8.09e+14   |  1.12e+16   |  6.35e+15   |  6.04e+14   |  1.04e+15   |  1.66e+15   | 2.17e+16  |
  % decompressd on DB Server        |   43.61%    |   72.64%    |   60.39%    |  1276.73%   |   30.79%    |   191.72%   |  108.58%  |
------------------------------------------------------------------------------------------------------------------------
```

Questions? Concerns? Where to go now?

Stay in touch!

- E-mail: franky@loredata.com.br or faust@pythian.com
- Blog: http://loredata.com.br/blog
- Facebook: https://facebook.com/08Franky.Weber
- Instagram: https://www.instagram.com/frankyweber/
- Twitter: https://twitter.com/frankyweber
- LinkedIn: https://linkedin.com/in/frankyweber/en
- Oracle ACE: https://bit.ly/2YxU6bK

Pythian

L♥VE YOUR DATA