



# Using Oracle Database In-Memory feature to Speed-up CERN Applications

Emil Pilecki

# About Emil

- Senior DBA at CERN
  - First joined CERN in 2000, staff member as of 2012
- Previously DBA team lead at Hewlett-Packard Poland
- 16 years of experience with Oracle databases
- Specializes in:
  - High availability solutions – RAC, Data Guard
  - Database performance and testing
  - Oracle In-Memory
  - Data warehousing



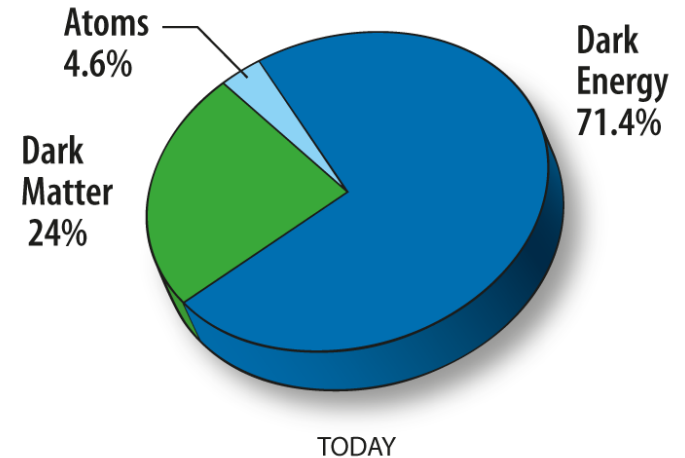
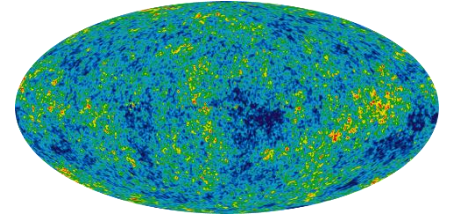
# About CERN

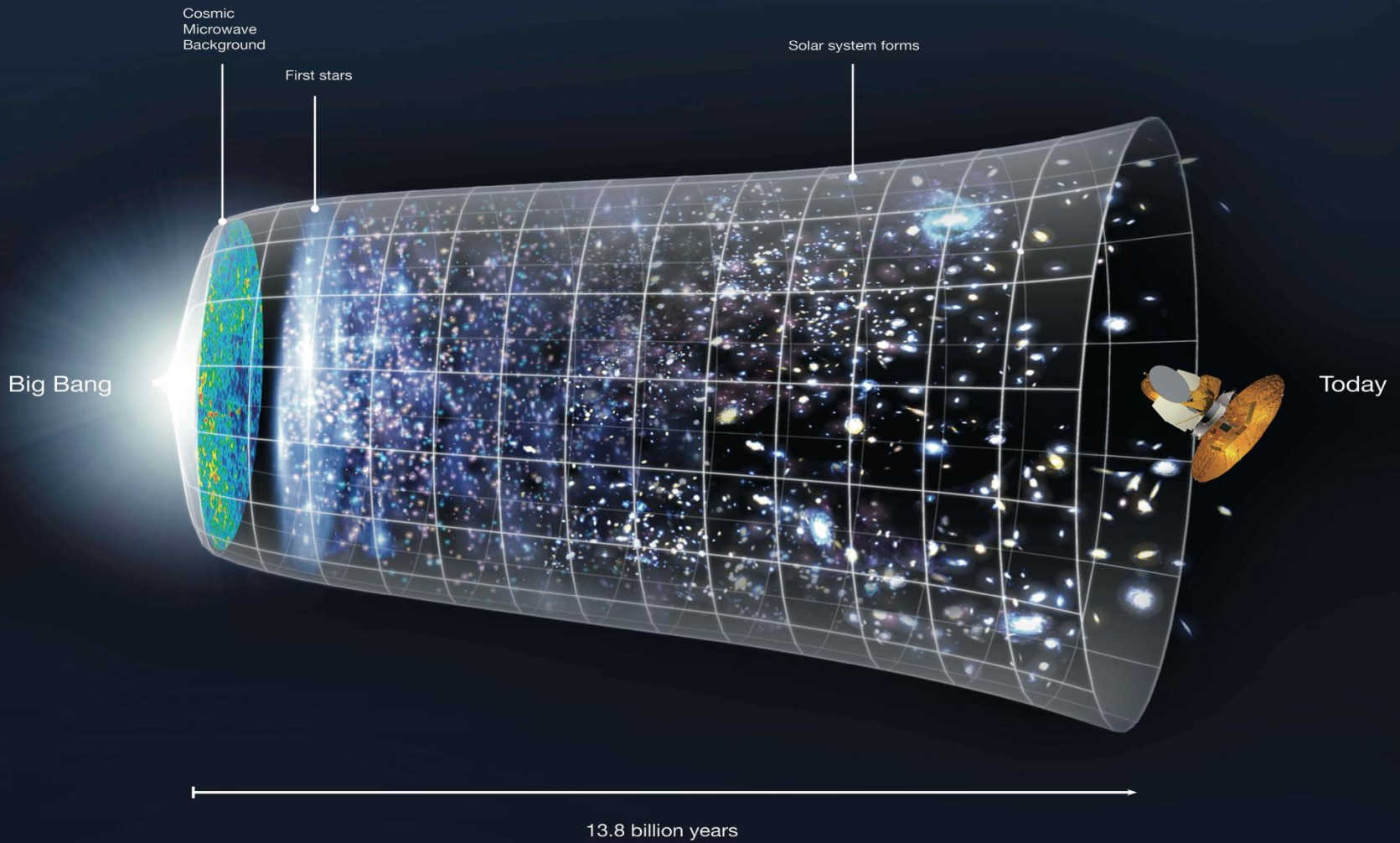
- CERN - **European Laboratory for Particle Physics**
- Founded in 1954 by 12 countries for fundamental physics research in the post-war Europe
- Today **22 member states + world-wide collaborations**
  - About ~1000 MCHF yearly budget
  - 2'300 CERN personnel
  - 10'000 users from 110 countries



# Fundamental Research

- What is 95% of the Universe made of?
- Why do particles have mass?
- Why is there no antimatter left in the Universe?
- What was the Universe like, just after the "Big Bang"?





# The Large Hadron Collider (LHC)



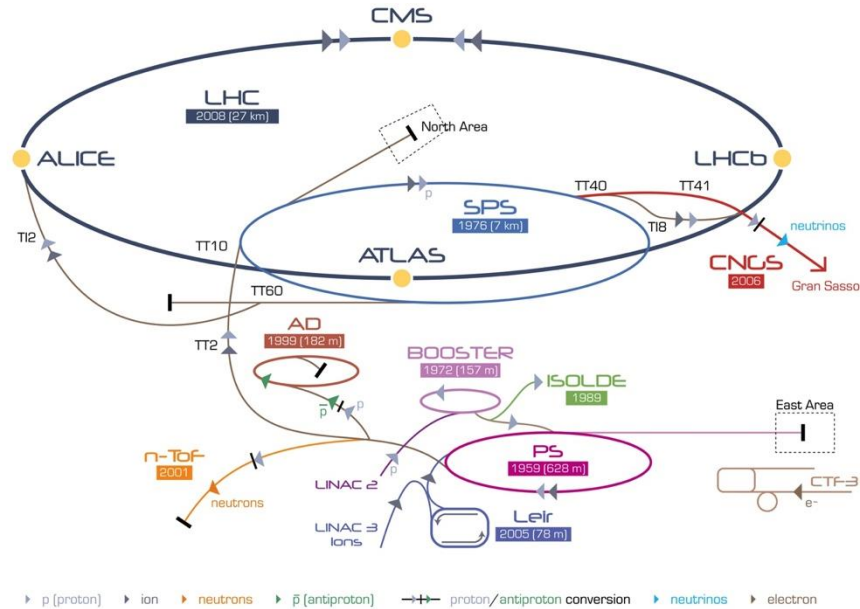
**Largest machine** in the world  
27km, 6000+ superconducting magnets

**Fastest racetrack** on Earth  
Protons circulate 11245 times/s (99.9999991% the speed of light)

**Emptiest** place in the solar system  
High vacuum inside the magnets

**Hottest spot** in the galaxy  
During Lead ion collisions create temperatures 100 000x hotter than the heart of the sun

# CERN's Accelerator Complex



LHC Large Hadron Collider SPS Super Proton Synchrotron PS Proton Synchrotron

AD Antiproton Decelerator CTF-3 Clic Test Facility CNGS Cern Neutrinos to Gran Sasso ISOLDE Isotope Separator OnLine DEvice  
 LeIR Low Energy Ion Ring LINAC LINear ACcelerator n-Tof Neutrons Time Of Flight



# ATLAS Detector



**150 Million** sensors  
Control and detection sensors

**Massive 3D camera**  
Capturing 600 million collisions per second  
Data rate hundreds TB per second

# CMS Detector



## Raw Data

- Was a detector element hit?
- How much energy?
- What time?

## Reconstructed Data

- Particle Type
- Origin
- Momentum of tracks (4 vectors)
- Energy in cluster (jets)
- Calibration Information

# LHC Computing Grid-WLCG



- 600 million events per second, 1 PB raw data per second before filtering, 30 PB of filtered data annually
- 500,000 cores in >40 countries, >170 computer centers around the world

<http://cern.ch/about/computing>  
<http://cern.ch/lhcatome/>

# CERN Database Services

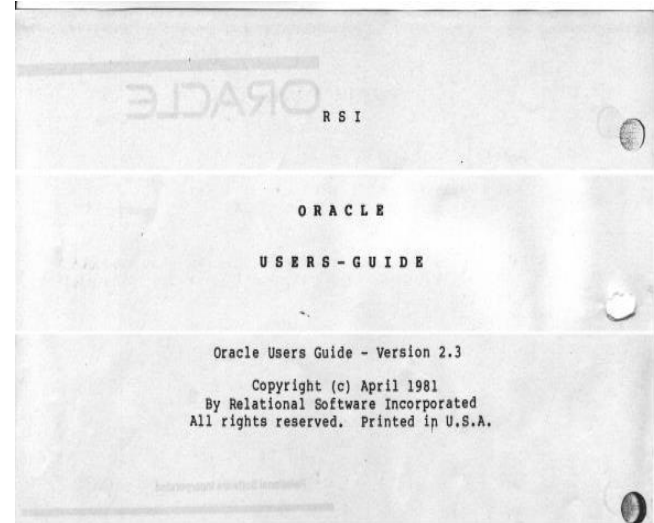
- Over 100 Oracle databases, most of them RAC
  - Running Oracle 11.2.0.4 and **12.1.0.2**
  - 750 TB of data files for production DBs in total, NAS as storage
  - Oracle In-Memory in production since July 2015
- Examples of critical production DBs:
  - Quench Protection System: 150'000 changes/s
  - LHC logging database: 430 TB, growing 180 TB/year
- But also DB as a Service (single instances)
  - 310 MySQL, 70 PostgreSQL, 9 Oracle, 5 InfluxDB

ORACLE®



# Oracle at CERN

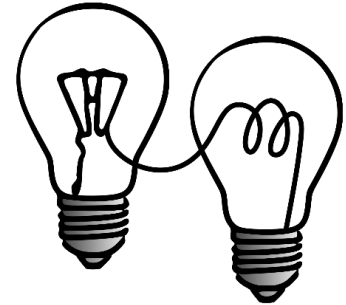
- Since 1982 – version 2.3
  - Initially used for accelerator controls
- Currently supports **hundreds of applications** in different domains
  - LHC experiments metadata
  - Accelerator control and logging
  - Engineering applications
  - Other (smaller) experiments
  - Administrative support: HR, ERT, ERP, Finance, WMS



Source: N. Segura Chinchilla, CERN

# CERN Computing Challenges

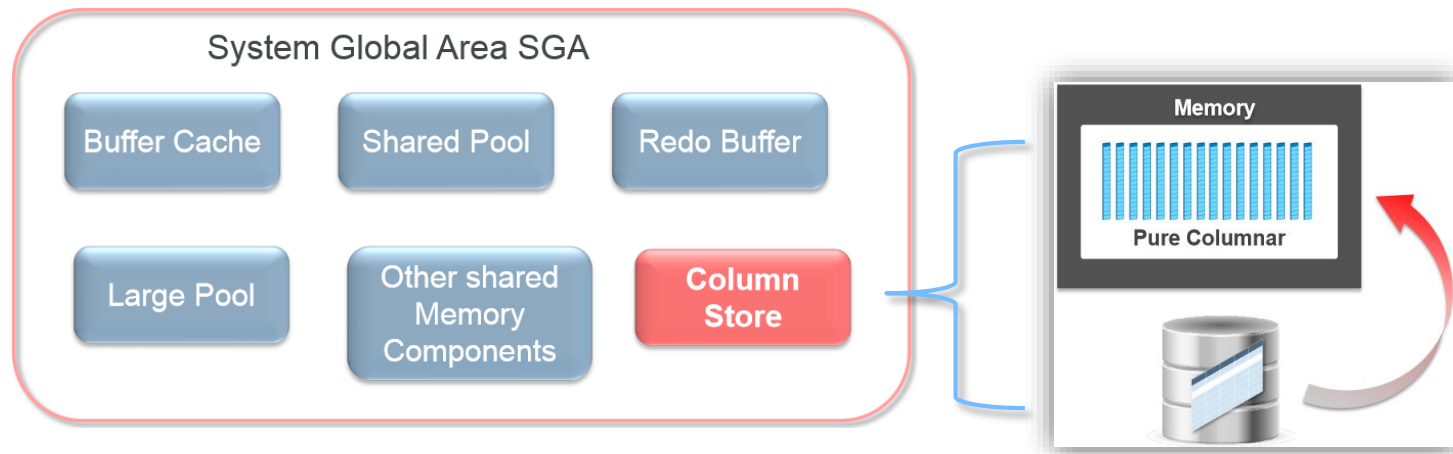
- The amount of CERN data increases quickly
  - Data increase rate will greatly accelerate after LHC luminosity upgrade (planned around 2020)
- HW performance increase over time not fast enough
  - Moore's law, even if holds true, won't save us
- Need to rethink our computing model
- Consider novel database technologies
  - Scalable databases – Hadoop, NoSQL
  - In-Memory databases
  - Columnar data stores



# In-Memory Column Store



- New pool in System Global Area
- Data in memory and in columnar format
- Huge performance boost for full table scans!



# In-Memory Column Store

- Both row and columnar format simultaneously in memory
  - Buffer Cache for OLTP workload and data modifications (DML)
  - In-Memory columnar cache for analytics and reporting queries
  - Guaranteed transactional consistency
  - Distribute memory between IMC and Buffer Cache



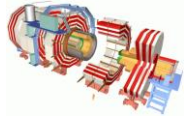


# In-Memory Column Store

- Very simple setup – only one parameter + restart
- Transparent for applications – no code change needed
- Optimizer automatically uses In-Memory cache
- No storage overhead – only row format on disk
- In-Memory compression
  - Reduces the amount of extra memory needed
  - No negative performance impact on queries\*

# CERN In-Memory Use-Cases

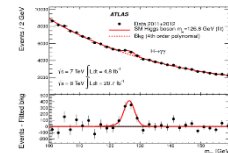
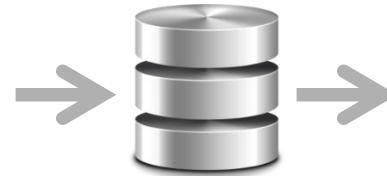
- **Physics Data Analysis**
  - Data from LHC collisions, gathered by all 4 detectors
- **Administrative Data Warehouse**
  - HR data and personal records, financial data, orders / purchases, resource usage planning, electronic recruitment and many others
- **LHCb DIRAC bookkeeping system**
  - Metadata catalogue for experiment data sets
    - files and jobs



# Physics Data Analysis

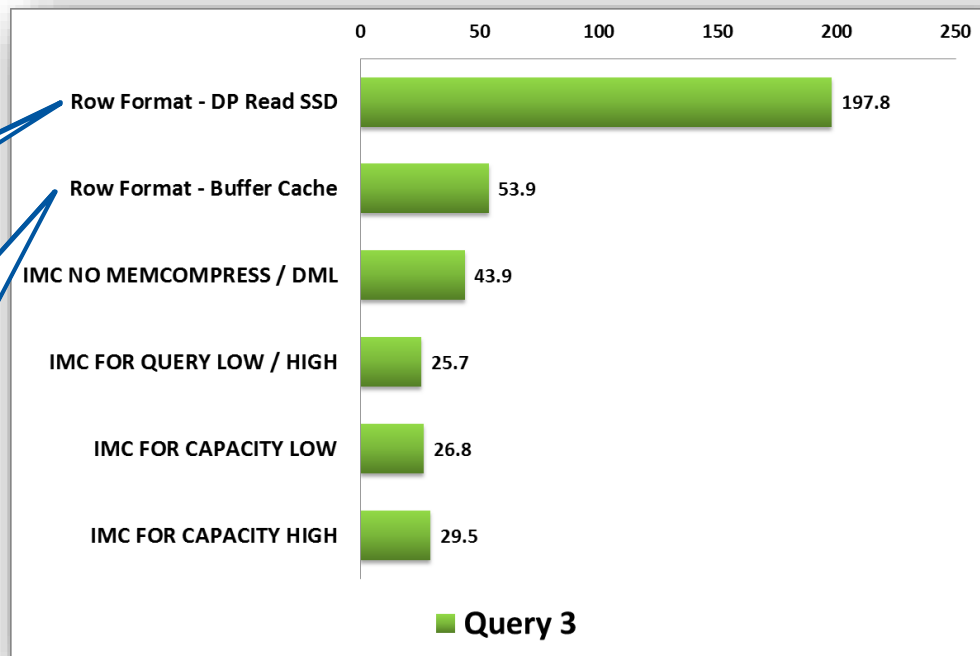
- Currently C++ and flat file based (ROOT)
  - WLCG grid used for running analysis jobs
- Analysis inside the database **not possible without IMC**
  - Large data sets have to be scanned for each query
  - Hundreds of columns, with each query using a unique subset
  - Cannot index all possible combinations of columns
  - Query performance typically limited by IO reads

```
with
"sel_muon" as (select "muon_id", "RunNumber", "EventNumber", "E", "px", "py", "pz", "charge", "pt", "phi", "eta" from DATA12_STEV."muon")
where "pts" > 10000. and abs("eta") < 2.7 and "tightly" = 1 and ("id_d0"<10. or abs("eta")>2.5) and "ptcoone20"<0.1"pts)
select "RunNumber", "EventNumber", mu_sel_n,
muon0."muon_id" as mu_id0, muon1."muon_id" as mu_id1,
muon0."charge" as mu_charge0, muon1."charge" as mu_charge1,
muon0."pts"/1000. as mu_pt0, muon1."pts"/1000. as mu_pt1,
muon0."eta" as mu_eta0, muon1."eta" as mu_eta1,
(case when abs(muon0."phi"-muon1."phi")<acos(-1.) then sqrt(POWER(abs(muon0."phi"-muon1."phi"),2)+POWER(abs(muon0."eta"-muon1."eta"),2))
else sqrt(POWER(2.*acos(-1.) - abs(muon0."phi"-muon1."phi"),2)+POWER(abs(muon0."eta"-muon1."eta"),2)) end) as DELTA,
ANALYSTOOLS.FITANALYSIS11V_HASS_LEFTORS(muon0."E", muon1."E", muon0."px", muon1."px", muon0."py", muon1."py", muon0."pz", muon1."pz")/1000. as IFV_HASS
from DATA12_STEV."periodAllYear_v47-pro13-01"
INNER JOIN (select "RunNumber", "EventNumber", COUNT(*) as mu_sel_n from "sel_muon" group by ("RunNumber", "EventNumber")) USING ("RunNumber", "EventNumber")
INNER JOIN "sel_muon" USING ("RunNumber", "EventNumber") INNER JOIN "sel_muon" USING ("RunNumber", "EventNumber")
where muon0."muon_id"<muon1."muon_id" and muon0."charge" != muon1."charge" and mu_sel_n=2;
```



# Physics Analysis – Benchmark

Configuration	Query 1	Query 2	Query 3
Row Format - DP Read SSD	47.1	287.2	197.8
Row Format - Buffer Cache	34.3	252.0	53.9
IMC NO MEMCOMPRESS / DML	0.4	17.7	43.9
<b>IMC FOR QUERY LOW / HIGH</b>	0.4	17.6	25.7
<b>IMC FOR CAPACITY LOW</b>	0.5	17.7	26.8
IMC FOR CAPACITY HIGH	2.1	17.2	29.5




IMC vs SSD Direct Path  
**16x faster!!**

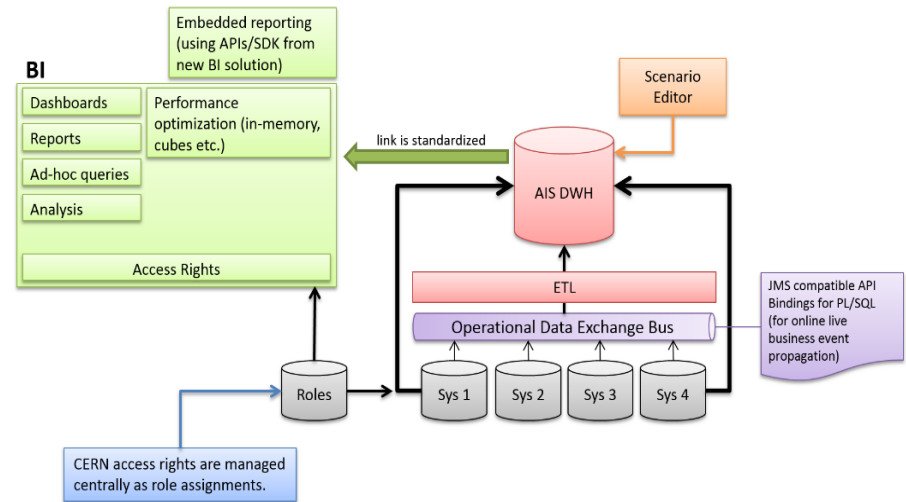
IMC vs Buffer Cache  
**14x faster!!**

# Physics Data Analysis with IMC

- Testing done in 2014 – very positive results!
- In-Memory DB processing much faster than file based data analysis
- This was more of an academic study
  - Too late to redesign LHC data processing for the current LHC run
  - Challenges remain:
    - 30 PB of experiment data per year – cannot fit in memory
    - Preloading data subsets before analysis – load time critical
    - Many Oracle instances with a lot of RAM needed

# Administrative Data Warehouse

- Currently in production since July 2015
- Supports CERN reports, dashboards and data analytics
- Pentaho BI Suite  pentaho as the application layer
  - Mondrian OLAP



# Administrative Data Warehouse

- Unique data source for all BI applications
  - Assures data consistency across all systems
- Designed to be used with In-Memory Column Store
  - Data set of 170GB, can fit entirely in memory
- Bi-temporal data model
  - Preserves full history of changes
  - Can view data „as of timestamp”



# IMC Tests – Methodology

- Real life queries from BI applications
  - Captured from DB SQL history, and provided by application users
  - Covering 1 week of DB activity – reporting queries only
  - All captured query types were used in the test
    - Grouped together if differ only by bind variables / WHERE conditions
    - And only if the execution plan and response time were comparable
- Performance testing – query response time
  - 9 runs of each query for each DB configuration
  - Results averaged out of 7 test runs
    - Discarding the best and the worst run





# IMC Tests – DB Configurations

- Row format only
  - Big and small Buffer Cache – 180GB / 32GB
    - Pre-warmed by 1 run of each test query
    - *ALTER SESSION SET "\_serial\_direct\_read"=never;*
  - Forced Direct IO (direct path read)
    - *ALTER SESSION SET "\_serial\_direct\_read"=always;*
- In-Memory columnar format
  - With different compression levels
  - And pre-warmed 32GB Buffer Cache



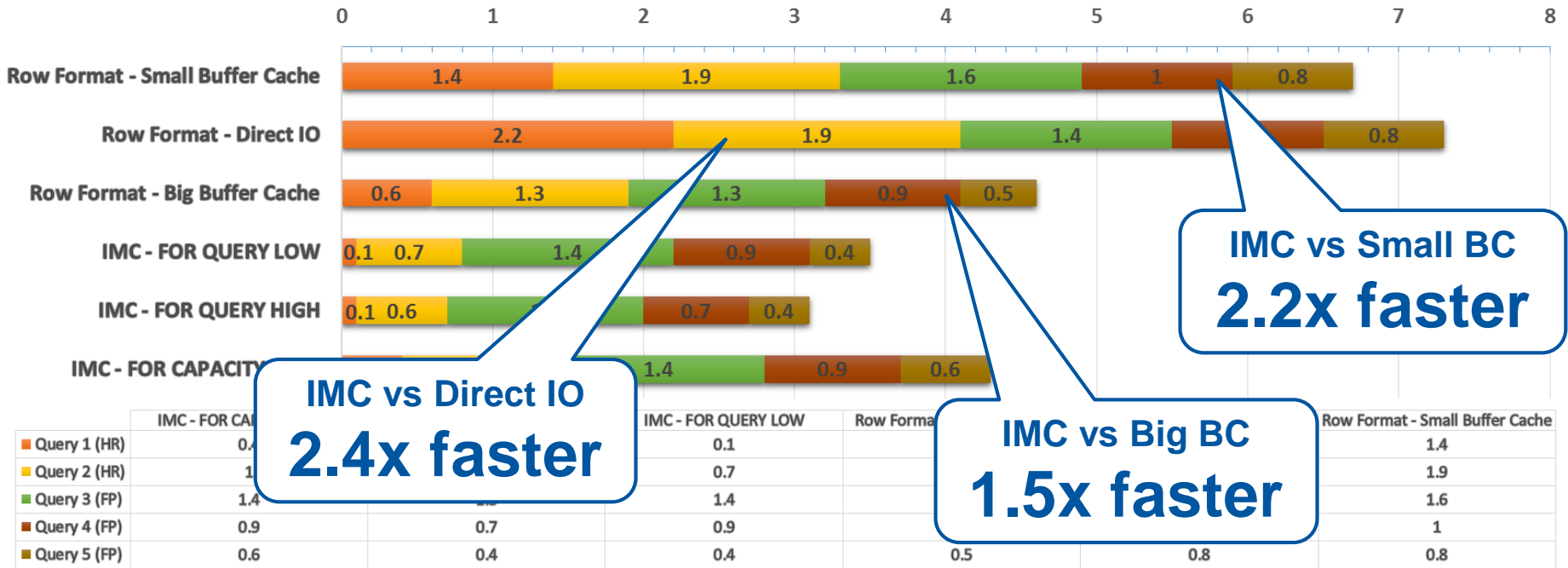
# IMC Tests – Environment

- Server: ACTINA SOLAR 820 X5 (Intel S2600WP)
- CPU: Intel Xeon E5-2650 v2 - 2.60GHz  
16 cores, 32 threads
- Memory: DDR3 256GB
- Storage: NetApp NAS FAS8040  
8 cores, 64GB RAM and 3.7TB SSD cache
- OS: Red Hat Enterprise Linux Server 6.8
- RDBMS: 12.1.0.2 Enterprise Edition - 64bit Production



# ADW Queries – Small Datasets

Queries on small data sets <1GB - response time (s)



# Example – Query 1 HR Domain

```
--Q1 HR
select co
select
sum
co
from co
im
on
im
on
im
on
im
on
where f
and f
and as
and in
and ex
and (e
and sy
and sy
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1			1415 (6)	00:00:01
1	SORT AGGREGATE		1				
2	VIEW		1			1415 (6)	00:00:01
3	SORT AGGREGATE		1	120			
* 4	HASH JOIN		20216	2369K	2064K	1415 (6)	00:00:01
* 5	TABLE ACCESS INMEMORY FULL	COR_D_PERSON_EXP_PARTICIPATION	65800	1285K		29 (7)	00:00:01
* 6	HASH JOIN		18998	1855K		1185 (7)	00:00:01
7	JOIN FILTER CREATE	:BF0000	18688	1496K		843 (8)	00:00:01
* 8	HASH JOIN		18688	1496K		843 (8)	00:00:01
9	JOIN FILTER CREATE	:BF0001	18688	985K		402 (10)	00:00:01
* 10	TABLE ACCESS INMEMORY FULL	COR_F_PERSON_BRIDGE	18688	985K		402 (10)	00:00:01
11	JOIN FILTER USE	:BF0001	514K	13M		439 (5)	00:00:01
* 12	TABLE ACCESS INMEMORY FULL	COR_D_PERSON_ASSIGNMENT	514K	13M		439 (5)	00:00:01
13	JOIN FILTER USE	:BF0000	766K	13M		340 (6)	00:00:01
* 14	TABLE ACCESS INMEMORY FULL	COR_D_PERSON_IN...	766K	13M		340 (6)	00:00:01

In-Memory

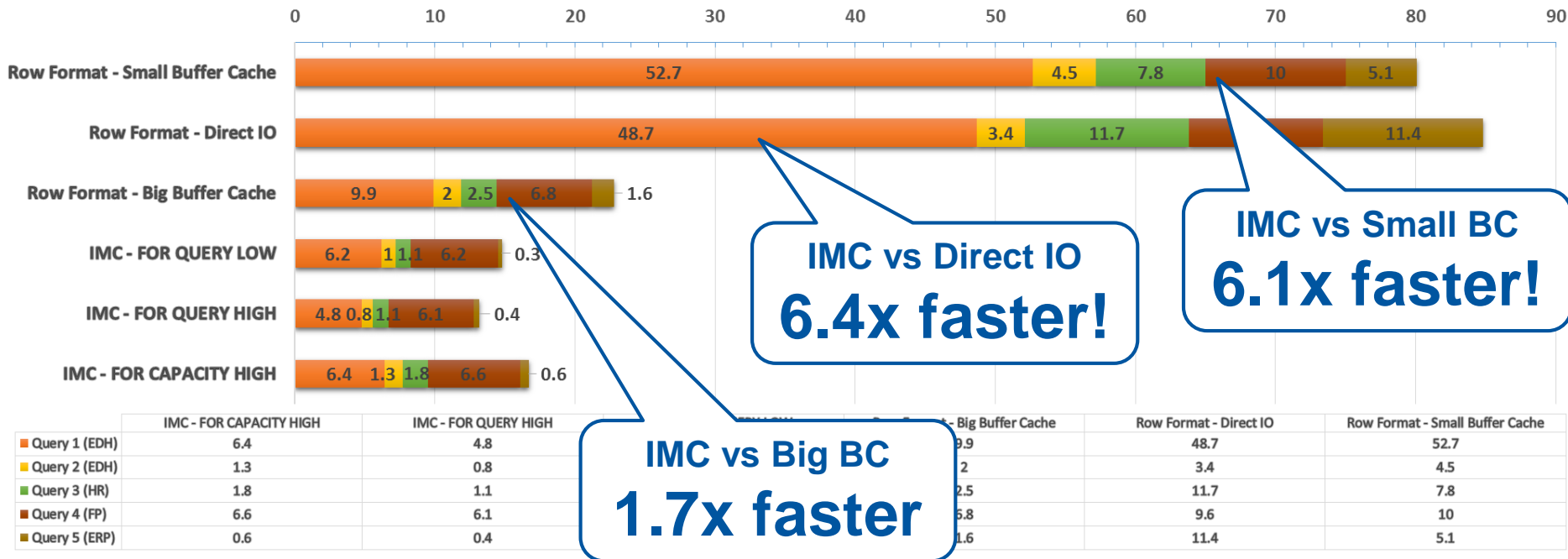
In-Memory used

Bloom filters used

```
0 recursive calls
9 db block gets
30 consistent gets
0 physical reads
0 redo size
542 bytes sent via SQL*Net to client
552 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
1 rows processed
```

# ADW Queries – Medium Datasets

Queries on medium data sets <10GB - response time (s)



	IMC - FOR CAPACITY HIGH	IMC - FOR QUERY HIGH	Big Buffer Cache	Row Format - Direct IO	Row Format - Small Buffer Cache
Query 1 (EDH)	6.4	4.8	9.9	48.7	52.7
Query 2 (EDH)	1.3	0.8	2	3.4	4.5
Query 3 (HR)	1.8	1.1	2.5	11.7	7.8
Query 4 (FP)	6.6	6.1	6.8	9.6	10
Query 5 (ERP)	0.6	0.4	1.6	11.4	5.1

# Example – Query 4 FP Domain

--QUERY4 - FP domain

```
select count(*) from (
SELECT
  TO_CHAR(TO_DATE(MOISENCOURS, 'YYYYMMDD'), 'MM-YYYY') AS "Month-Year",
  ROUND(PREV_MONTH_TOTAL_CASH / 1000000) AS "Opening Cash",
  ROUND(INFLOW_CURR_MONTH / 1000000, 0) AS "Inflows",
  ROUND(OUTFLOWCHF / 1000000, 0) AS "Outflows",
  ROUND(CURR_MONTH_TOTAL_CASH / 1000000) AS "Closing Cash "
FROM
  AISDWH_READ.DWH_TREA_CASH_LAST_12_MONTHS);
```

0 recursive calls  
328 db block gets  
1124364 consistent gets  
48 physical reads  
0 redo size  
542 bytes sent via SQL\*Net to client  
608 bytes received via SQL\*Net from client  
2 SQL\*Net roundtrips to/from client  
8 sorts (memory)  
0 sorts (disk)  
1 rows processed

Buffer Cache

0 recursive calls  
0 db block gets  
28882 consistent gets  
0 physical reads  
0 redo size  
542 bytes sent via SQL\*Net to client  
552 bytes received via SQL\*Net from client  
2 SQL\*Net roundtrips to/from client  
6 sorts (memory)  
0 sorts (disk)  
1 rows processed

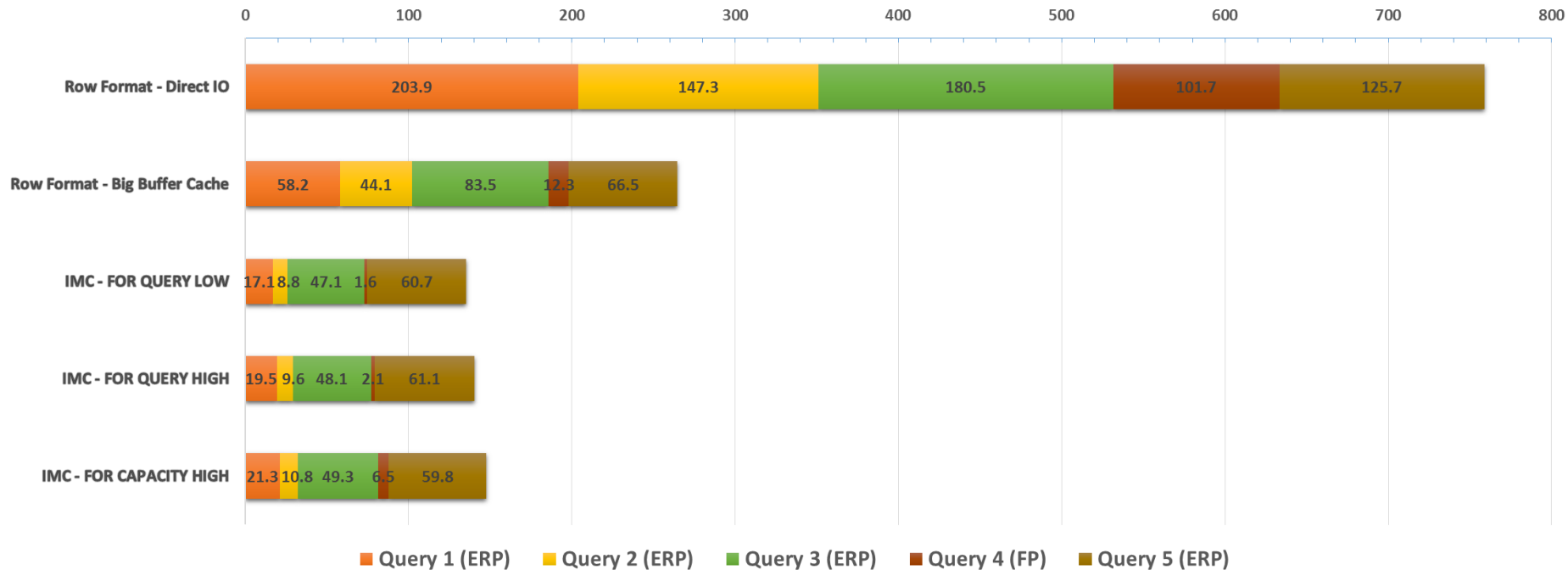
In-Memory

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1	2031		150K (9)	00:00:06
1	SORT AGGREGATE		1	2031			
* 2	HASH JOIN		4539M	85866	26M	150K (9)	00:00:06
3	VIEW		269K	257M		20892 (2)	00:00:01
4	HASH GROUP BY		269K	262M		20892 (2)	00:00:01
5	VIEW		269K	262M		20892 (2)	00:00:01
6	SORT UNIQUE		269K	33M	37M	20885 (2)	00:00:01
7	UNION-ALL						
* 8	HASH JOIN RIGHT OUTER		66	71742		343 (5)	00:00:01
* 9	TABLE ACCESS FULL	COR_D_KTP_COMPTE_BANQUE	121	2057		3 (0)	00:00:01
10	VIEW	DWH_INVESTMENT_LAST_12_MONTHS	62	66340		340 (5)	00:00:01
* 11	FILTER						
12	HASH GROUP BY		62	8866		340 (5)	00:00:01
* 13	HASH JOIN		627	89661		339 (5)	00:00:01
* 14	TABLE ACCESS FULL	COR_D_KTP_COMPTE_BANQUE	69	1656		3 (0)	00:00:01
* 15	FILTER						
* 16	HASH JOIN RIGHT OUTER		1104	128K		336 (5)	00:00:01
			912	102		102 (0)	00:00:01
			115K	234		7 (7)	00:00:01
			5917	131		12 (12)	00:00:01
			26	99		11 (1)	00:00:01
			2250	51		12 (12)	00:00:01
			3750	51		12 (12)	00:00:01
			3750	50		10 (10)	00:00:01
			1659K	48		7 (7)	00:00:01
			3360	32		16 (16)	00:00:01
			152K	183		1 (1)	00:00:01
			10M	4852		1 (1)	00:00:01
			2057	3		0 (0)	00:00:01
			9971K	4849		1 (1)	00:00:01
				987K	11M	4849 (1)	00:00:01
			9H	3871		1 (1)	00:00:01
			5917	131		12 (12)	00:00:01
			5917	131		12 (12)	00:00:01
			26	99		11 (1)	00:00:01
			2250	51		12 (12)	00:00:01
			3750	51		12 (12)	00:00:01
* 39	FILTER						
* 40	TABLE ACCESS INMEMORY FULL	COR_D_DATE	250	3750		50 (10)	00:00:01
* 41	TABLE ACCESS INMEMORY FULL	COR_D_DATE	99986	1659K		48 (7)	00:00:01
* 42	TABLE ACCESS INMEMORY FULL	COR_F_EXCHANGE_RATE	96	3360		32 (16)	00:00:01
* 43	VIEW		139K	6108K		3734 (1)	00:00:01
* 44	HASH GROUP BY		139K	7873K	10M	3734 (1)	00:00:01
* 45	JOIN FILTER USE	:BF0000	139K	7873K		1785 (1)	00:00:01
* 46	HASH JOIN OUTER		139K	7873K		1785 (1)	00:00:01
* 47	TABLE ACCESS FULL	COR_D_KTP_COMPTE_BANQUE	67	1688		3 (0)	00:00:01
* 48	TABLE ACCESS FULL	COR_F_KTP_HISTO_REGLEMENT	232K	7735K		1782 (1)	00:00:01
* 49	HASH JOIN RIGHT OUTER		259K	22M		7925 (4)	00:00:01
* 50	TABLE ACCESS FULL	COR_D_KTP_COMPTE_BANQUE	121	2057		3 (0)	00:00:01
* 51	VIEW	DWH_INFLOWS_LAST_12_MONTHS	243K	7921		4 (4)	00:00:01
* 52	HASH GROUP BY		243K	24M	27M	7921 (4)	00:00:01
* 53	HASH JOIN		243K	24M		2124 (13)	00:00:01
* 54	TABLE ACCESS FULL	COR_D_KTP_COMPTE_BANQUE	119	2023		3 (0)	00:00:01
* 55	HASH JOIN		230K	19M		2120 (13)	00:00:01

Buffer Cache is less clogged with In-Memory enabled and can better cache small tables accessed by index

# ADW Queries – Large Datasets

Queries on large data sets >10GB - response time (s)

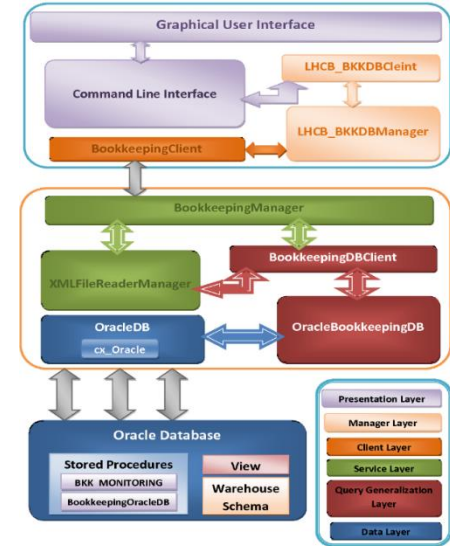






# LHCb Bookkeeping System

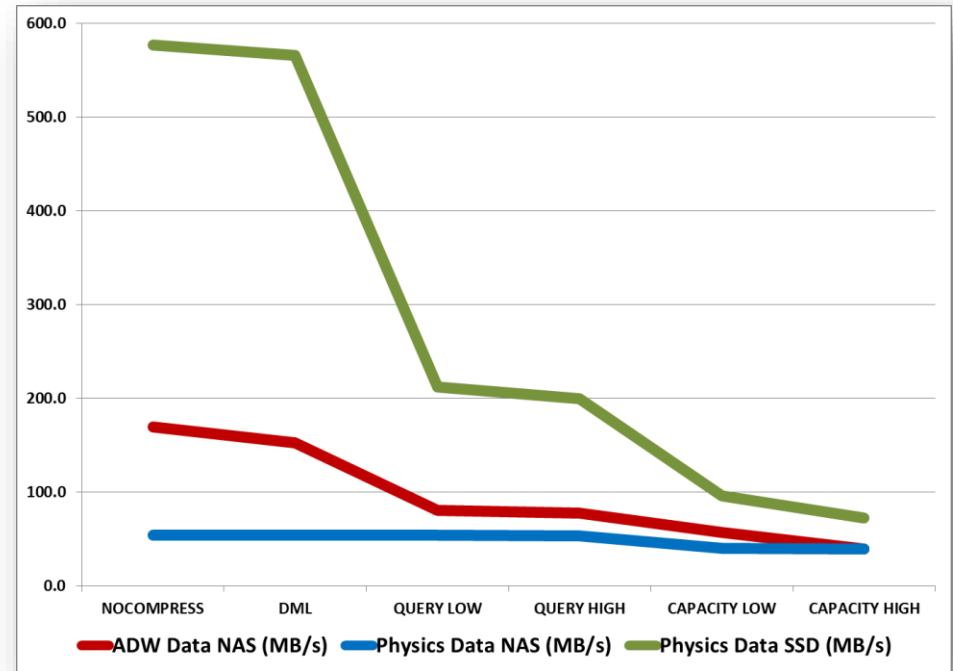
- Metadata repository for WLCG grid (LHCb detector)
- Allows browsing and retrieval of experiment data, as well as result sets produced by grid jobs
- Metadata for almost 1 billion data sets/files; over 700GB in total
- Active data of ~120GB can fit entirely in memory (IMC supports partitioning!)
- 12.1 performance testing not conclusive
- Awaiting 12.2 final release to continue...



# IMC Cache Population Speed

- 32 populate servers
- Primarily IO bound for low compression
- Primarily CPU bound for high compression

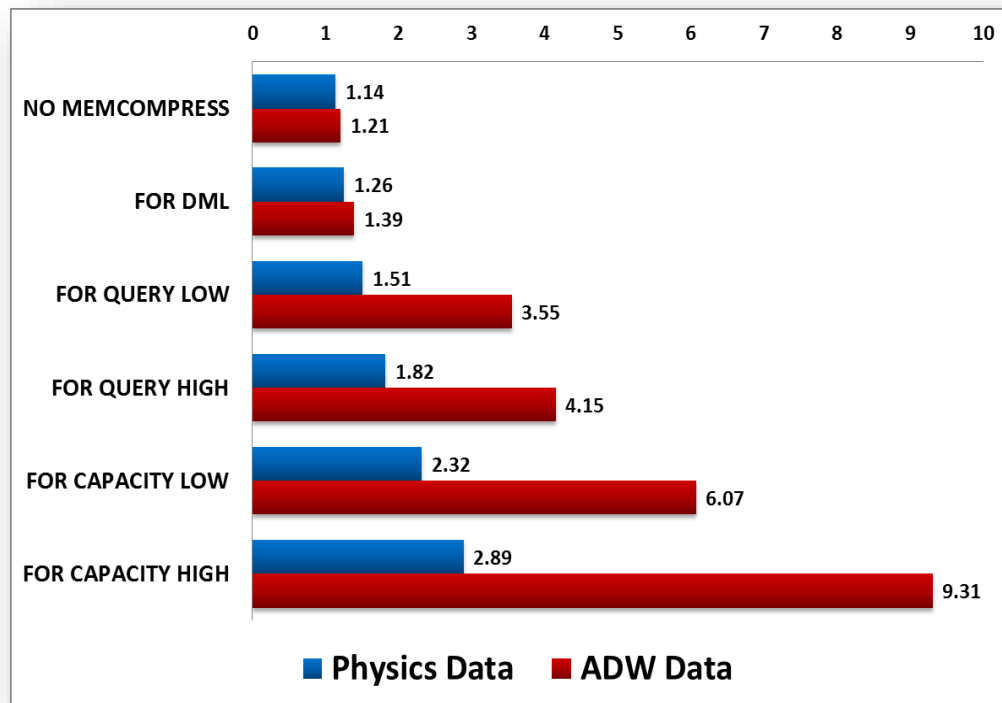
Compression Level	Physics Data		ADW Data
	SSD (MB/s)	NAS (MB/s)	NAS (MB/s)
NO MEMCOMPRESS	576.7	54.5	169.2
FOR DML	565.6	54.4	153.0
FOR QUERY LOW	212.0	54.0	80.3
FOR QUERY HIGH	200.1	53.7	77.9
FOR CAPACITY LOW	96.4	40.4	57.2
FOR CAPACITY HIGH	72.8	39.7	39.7



# IMC Compression Ratios

- Physics data mostly random numbers, hard to compress
- Much better compression for administrative data

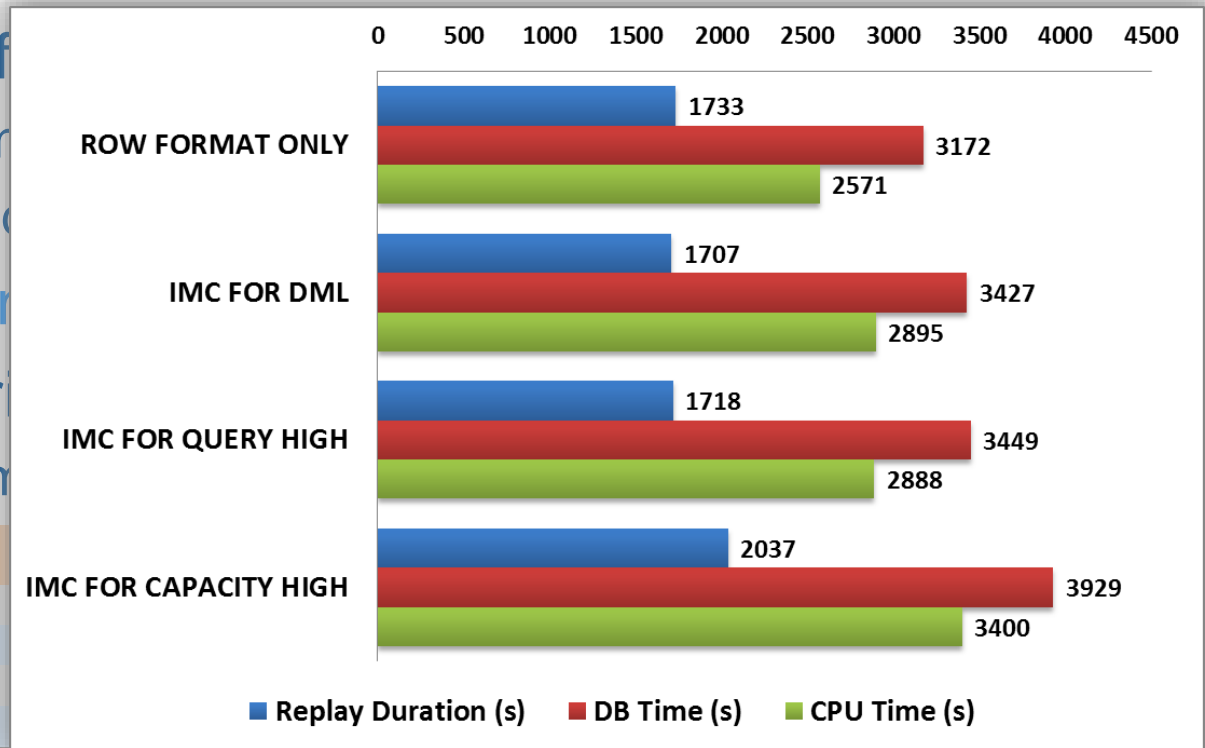
IMC LAYOUT	Physics Data	ADW Data
NO MEMCOMPRESS	1.14	1.21
FOR DML	1.26	1.39
FOR QUERY LOW	1.51	3.55
<b>FOR QUERY HIGH</b>	1.82	4.15
<b>FOR CAPACITY LOW</b>	2.32	6.07
<b>FOR CAPACITY HIGH</b>	2.89	9.31



# IMC Tests – OLTP Benchmark

- OLTP schema f
- High concurrer
- Mixed DML and
- Real Application
- Replay compar
- IMC with 3 com

Configuration
ROW FORMAT
IMC FOR DML
IMC FOR QUERY HIGH
IMC FOR CAPACITY HIGH



# In-Memory 12.2 Features

- In-Memory on [Active Data Guard](#)
- [Automatic Data Optimization \(ADO\)](#)
  - Policy based mode
  - Heat map based mode (fully automatic)
- [Elastic scans](#) within the IMC store
- [Join Groups](#) for faster hash joins
- In-Memory [Expressions](#)
- Better RAC support – `DISTRIBUTE FOR SERVICE`

# Recommended IMC Use-Cases

- **Very wide tables** - hundreds of columns
- Queries select only few out of many columns
- Queries need to scan **big data sets** – full table scan
- **Avoid joins** of multiple big tables
  - Bloom filters can help if only few rows selected
- Data sets should **fit entirely in memory** (compressed)
- **Business Intelligence** and **Data Warehousing** domain
- **Data Analytics** and **Reporting**, including ad-hoc

# Do You Really Need In-Memory?

- Very interesting and useful feature!
- But not an universal panacea for all performance issues!
- Performance of BI, analytics and reporting applications should improve greatly with IMC cache
- OLTP query performance likely won't improve significantly
  - But also should not deteriorate
- No development needed (consider HW and license costs)
- Test it for yourself and see how much you can gain...

# Summary

- Oracle 12.1.0.2 is a stable release
  - In production at CERN since February 2014 (7 databases)
- In-Memory feature very useful for CERN applications
  - Administrative DW platform already in production
  - One database currently being tested – LHCb experiment
  - Plans to introduce In-Memory for other CERN experiments
    - ALICE, CMS
  - Deployment is fast and straightforward
  - Design for In-Memory to get maximum benefits



# Thank you for your attention!



# Q & A

**E-mail:** [Emil.Pilecki@cern.ch](mailto:Emil.Pilecki@cern.ch)



[www.cern.ch](http://www.cern.ch)