CERN

Using the In-Memory Columnar Store to Perform Real-Time Analysis of CERN Data

Maaike Limper
Emil Pilecki
Manuel Martín Márquez

About the speakers

- Maaike Limper
  - Physicist and project leader
- Manuel Martín Márquez
  - Data Scientist at CERN
  - Interests: Data Analytics and Big Data
- Emil Pilecki
  - Oracle DBA at CERN
  - Interests: Database High Availability and Performance
  - Previously: DBA Team Leader at Hewlett-Packard

CERN

- CERN - **European Laboratory for Particle Physics**
- Founded in **1954** by **12 Countries** for fundamental physics research in a post-war Europe
  - Major milestone in the post-World War II recovery/reconstruction process

YEARS / ANS CERN

CERN openlab

- Public-private partnership between CERN and leading ICT companies
- Accelerate cutting-edge solutions to be used by the worldwide LHC community
- Train the next generation of top engineers and scientists.

Partners

HUAWEI

intel

ORACLE

SIEMENS

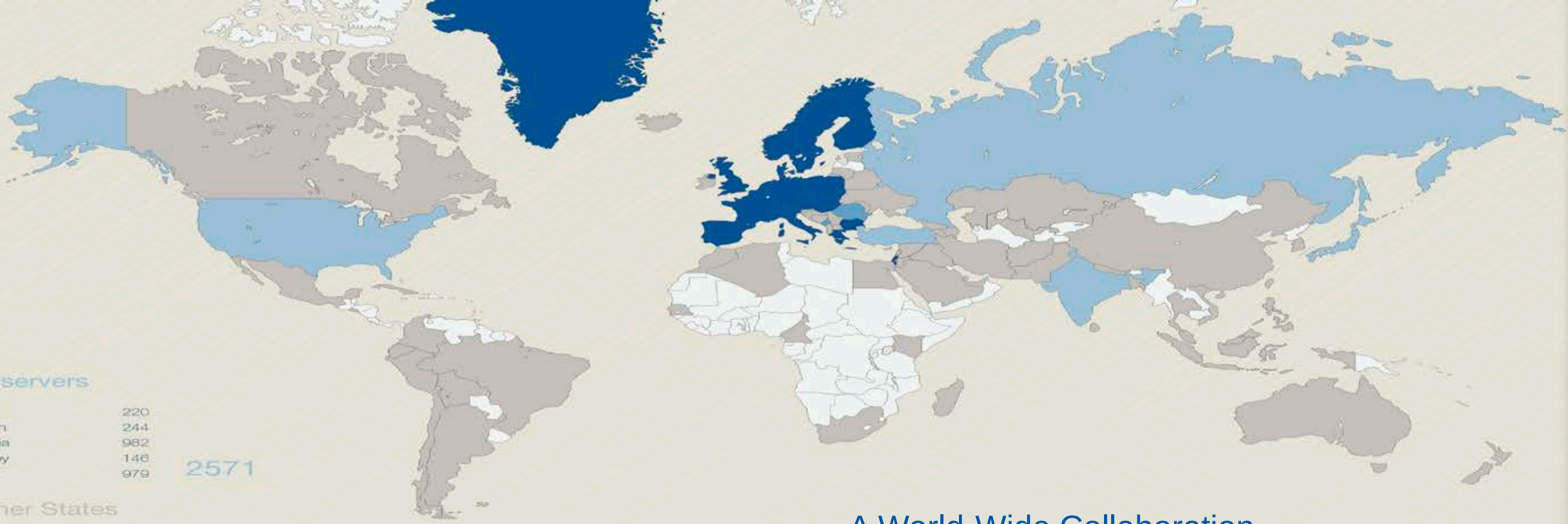
Contributors

rackspace
the open cloud company

Associates

Yandex

A World-Wide Collaboration



Observers

India	220
Japan	244
Russia	982
Turkey	146
USA	979

2571

Other States

Afghanistan	1	El Salvador	1	Pakistan	41
Albania	2	Estonia	16	Palestine (O.T.)	4
Algeria	8	Georgia	36	Peru	8
Argentina	11	Gibraltar	1	Philippines	1
Armenia	25	Hong Kong	1	Saudi Arabia	3
Australia	25	Iceland	4	Senegal	1
Azerbaijan	8	Indonesia	1	Singapore	2
Bangladesh	4	Iran	28	Sint Maarten	2
Belarus	47	Ireland	22	Slovenia	27
Bolivia	3	Jordan	2	South Africa	16
Bosnia & Herzegovina	1	Kenya	1	Sri Lanka	5
		Korea, D.P.R.	1	Syria	2
Brazil	108	Korea Rep.	117	Thailand	12
Cameroon	1	Kuwait	1	T.F.Y.R.O.M.	1
Canada	134	Lebanon	12	Tunisia	6
Cape Verde	1	Lithuania	19	Ukraine	55
Chile	12	Luxembourg	4	Uzbekistan	4
China	280	Madagascar	4	Venezuela	9
China (Taipei)	45	Malaysia	15	Viet Nam	9
Colombia	30	Mauritius	1	Zimbabwe	2
Croatia	35	Mexico	64		
Cuba	7	Montenegro	3		
Cyprus	16	Morocco	12		
Ecuador	3	Nepal	5		
Egypt	19	New Zealand	7		

1415

Member States

Austria	99	Greece	152	Slovakia	88
Belgium	106	Hungary	68	Spain	337
Bulgaria	75	Israel	51	Sweden	75
Czech Republic	202	Italy	1686	Switzerland	180
Denmark	53	Netherlands	153	United Kingdom	640
Finland	87	Norway	61		
France	751	Poland	229		
Germany	1150	Portugal	109		

6352

Candidate for Accession

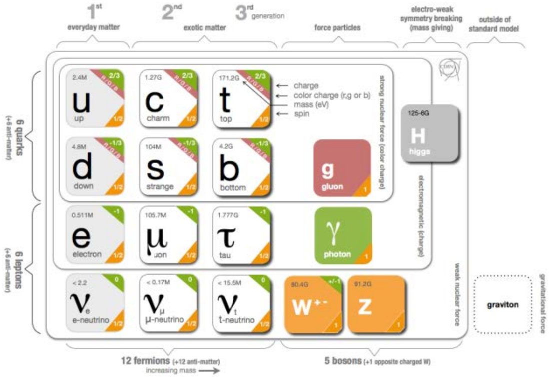
Romania	118

Associate Members in the Pre-stage to Membership

Serbia	41

Distribution of All CERN Users by Nationality on 14 January 2014

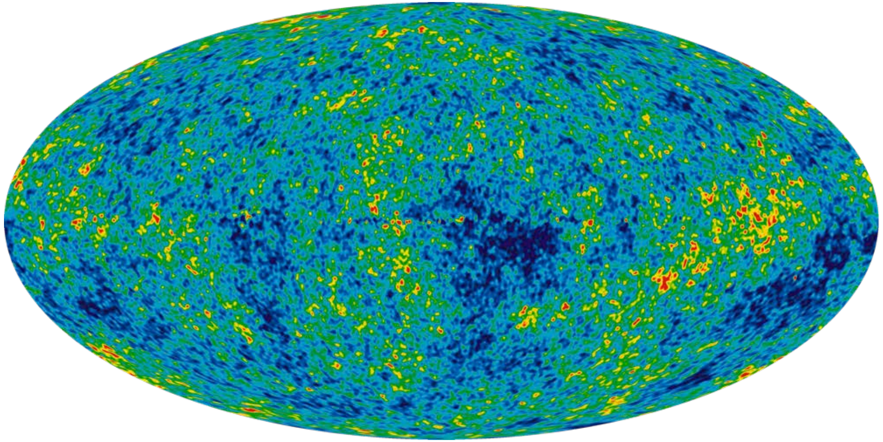
Fundamental Research

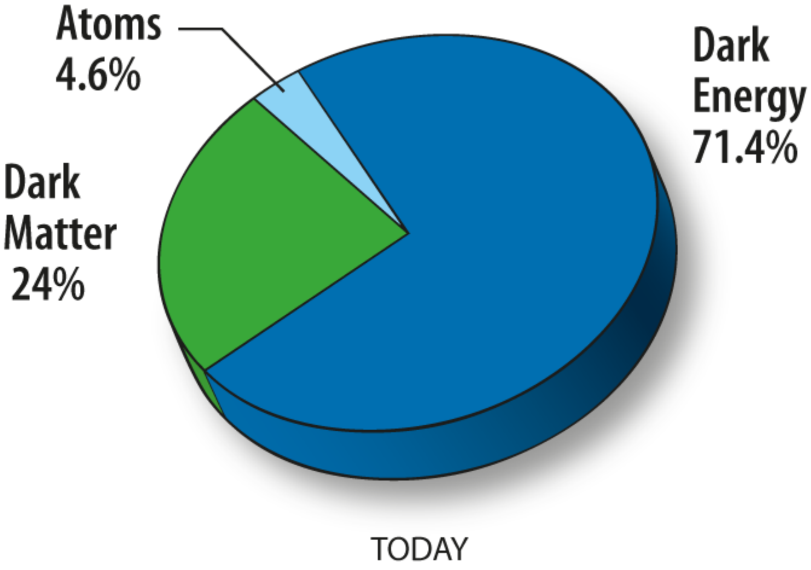


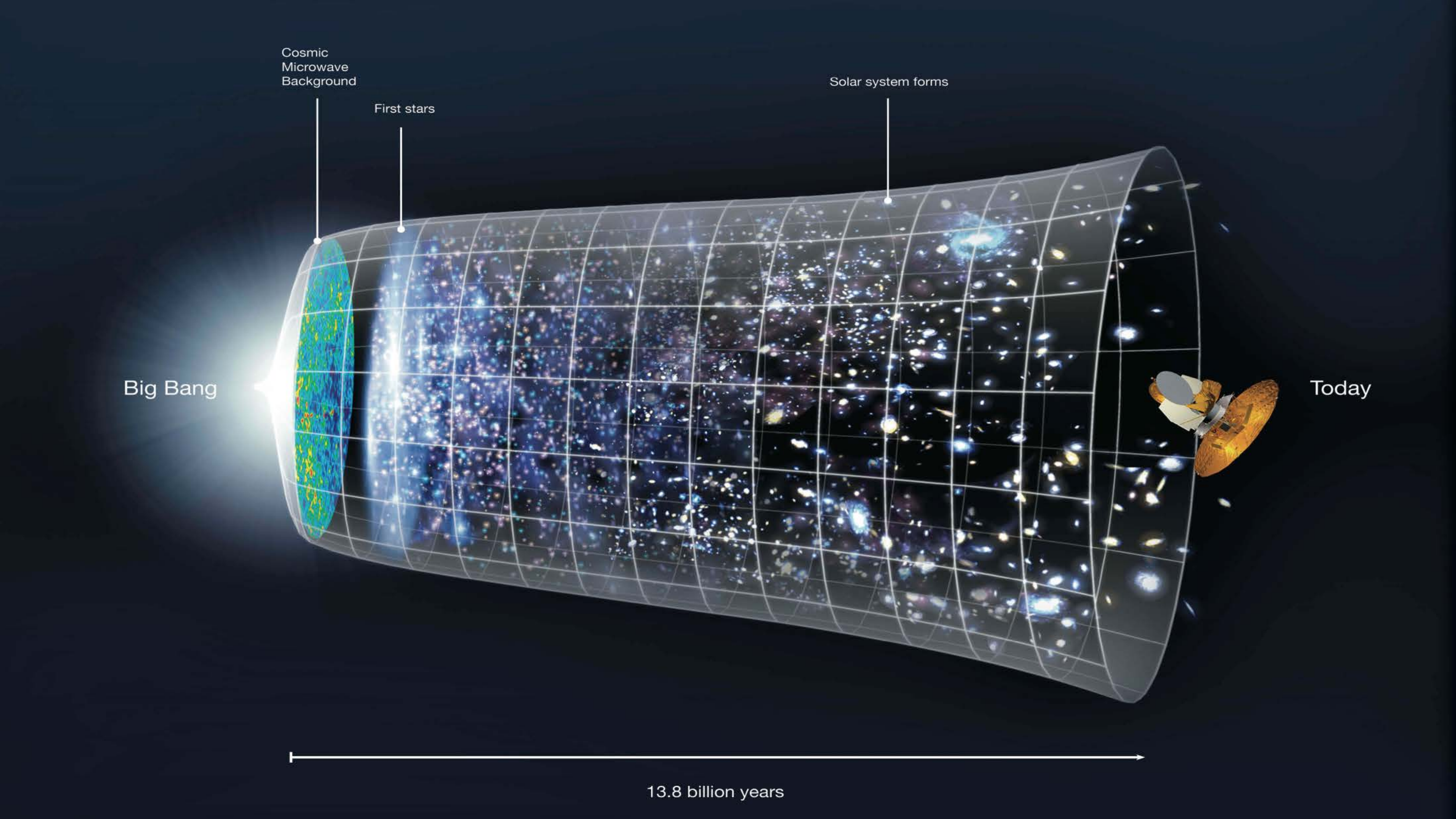
- Why do particles have mass?
  - Higgs Mechanism
- Why is there no antimatter left in the Universe?
  - Nature should be symmetrical
- What was matter like during the first second of the Universe, right after the "Big Bang"?
  - A journey towards the beginning of the Universe gives us deeper insight.

Fundamental Research

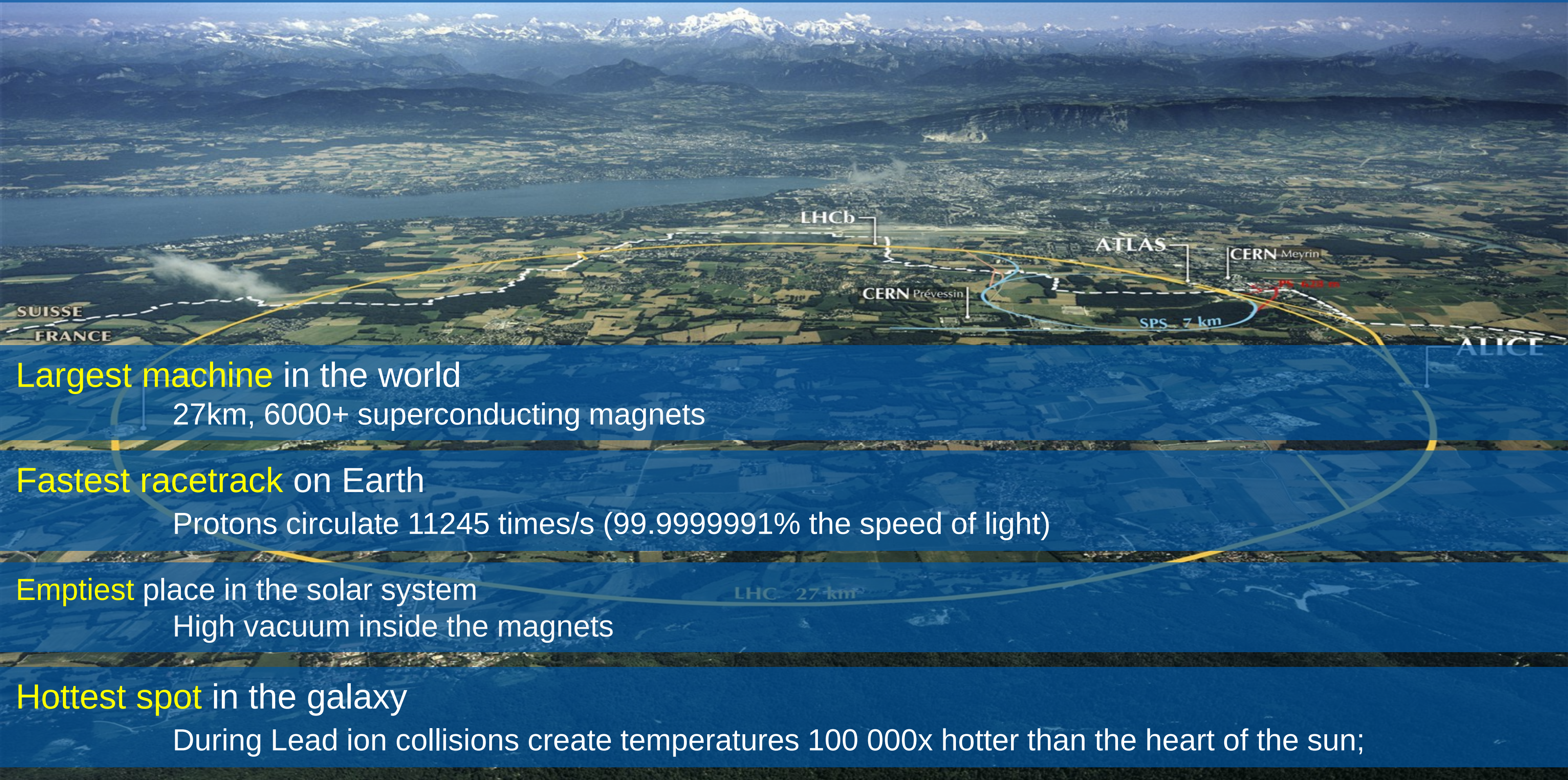
- What is 95% of the Universe made of?






Cosmic Microwave Background
First stars
Solar system forms
Big Bang
Today
13.8 billion years

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)

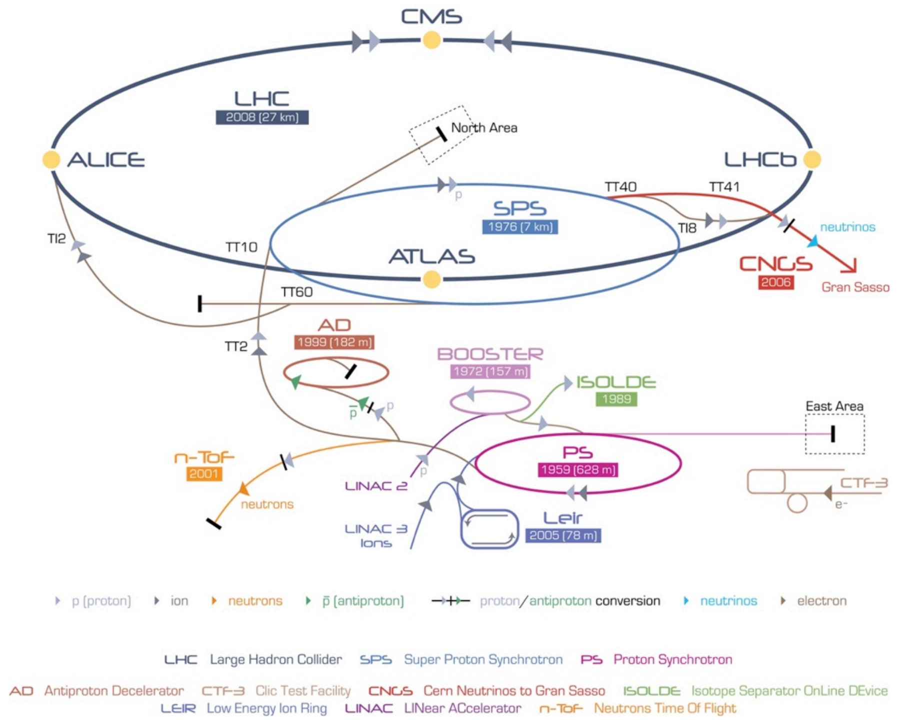
Emptiest place in the solar system

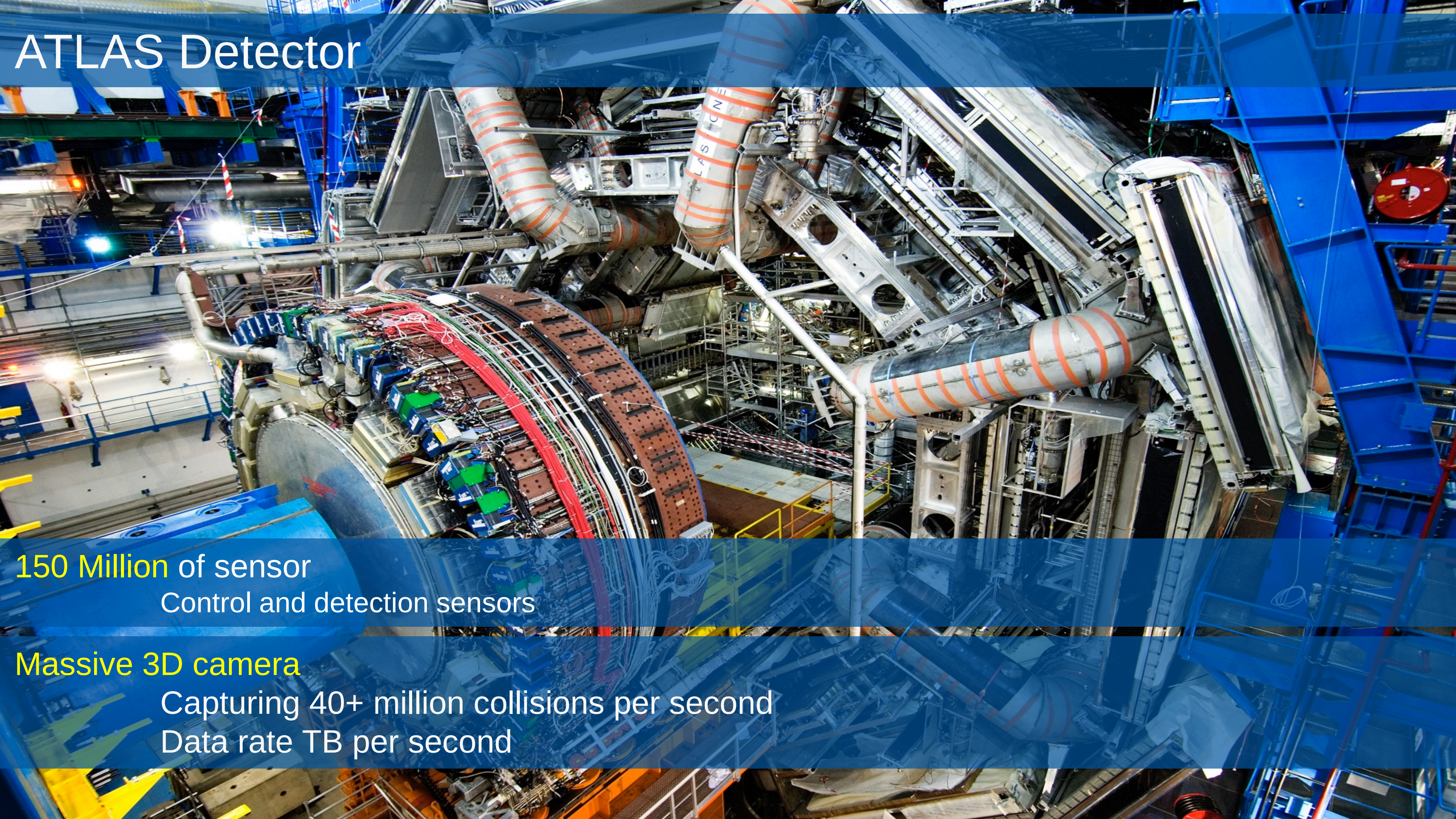
High vacuum inside the magnets

Hottest spot in the galaxy

During Lead ion collisions create temperatures 100 000x hotter than the heart of the sun;

CERN’s Accelerator Complex





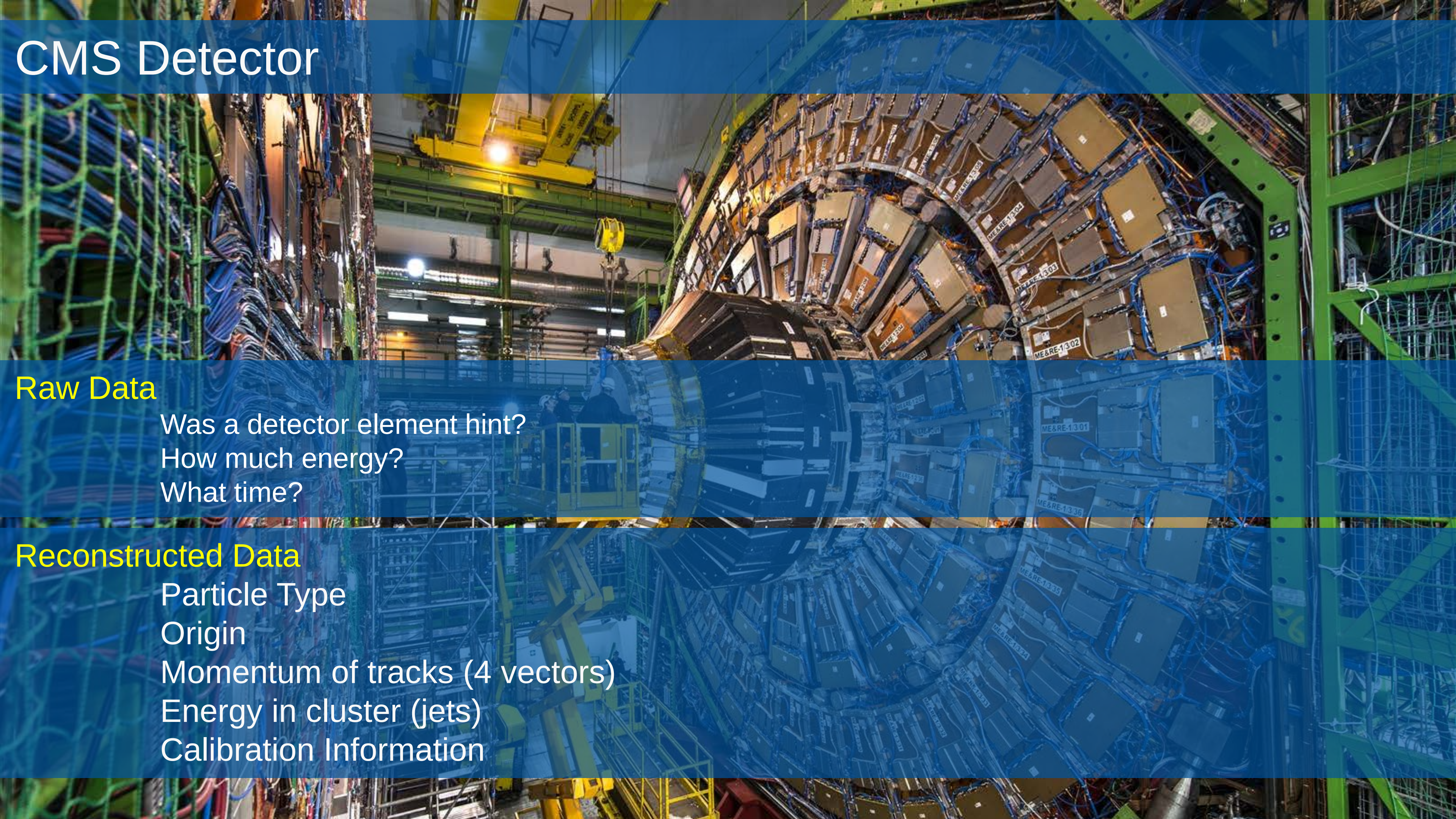
ATLAS Detector

150 Million of sensor

- Control and detection sensors

Massive 3D camera

- Capturing 40+ million collisions per second
- Data rate TB per second



CMS Detector

Raw Data

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

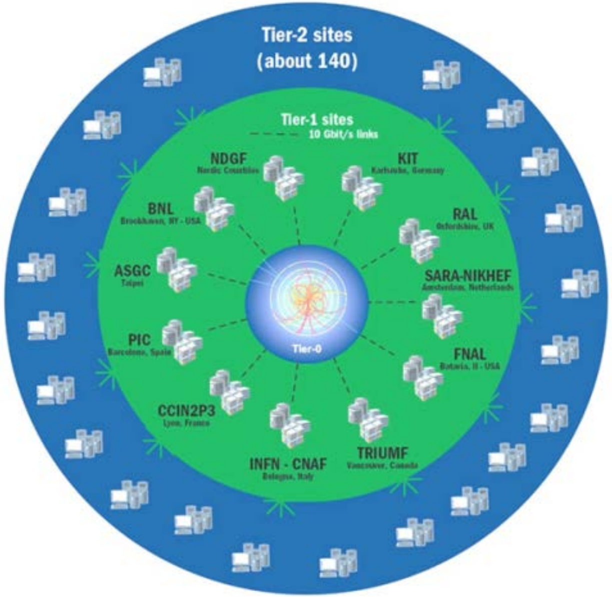
Reconstructed Data

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

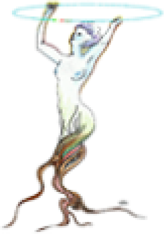
Worldwide LHC Computing Grid

- Provides Global computing resources
  - Store, distribution and analysis

- Physics Analysis using ROOT
  - Dedicated analysis framework
  - Plotting, fitting, statistics and analysis



Data Analysis in Practice

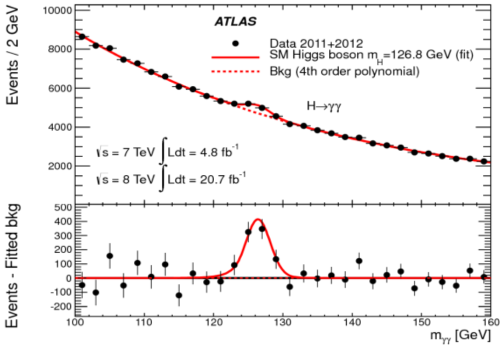


- ROOT works with N-tuples
  - Specifically produced by physics groups
    - Physics objects, level of details, filter and pre-analysis steps
- Small Datasets
  - Copy files and run locally
- Large Datasets
  - WLCG infrastructure split the analysis in multiple jobs
  - Each job is sent to Grid site where input files are available

The Challenge

Can we replace the file based analysis with a model where the data is analyzed inside a **centrally accessible Oracle Database?**

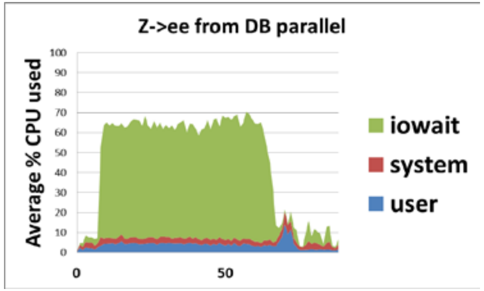
```sql
with
"sel_muon" as (select "muon_i","RunNumber","EventNumber","E","px","py","pz","charge","pt","phi","eta" from DATA12_8TEV."muon"
where "pt" > 10000. and abs("eta") < 2.7 and "tight" = 1  and ("id_d0"<10. or abs("eta")>2.5) and "ptcone20"<0.1*"pt")
select "RunNumber","EventNumber",mu_sel_n,
       muon0."muon_i" as mu_id0, muon1."muon_i" as mu_id1,
       muon0."charge" as mu_charge0, muon1."charge" as mu_charge1,
       muon0."pt"/1000. as mu_pt0, muon1."pt"/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 DELTAR,
ANALYSISTOOLS.PHYSANALYSIS.INV_MASS_LEPTONS(muon0."E",muon1."E",muon0."px",muon1."px",muon0."py",muon1."py",muon0."pz",muon1."pz")/1000. as INV_MASS
from DATA12_8TEV."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" muon0 USING ("RunNumber","EventNumber") INNER JOIN "sel_muon" muon1 USING ("RunNumber","EventNumber")
where muon0."muon_i"<muon1."muon_i" and muon0."charge" != muon1."charge" and mu_sel_n=2;
```



The Challenge

- Physics-objects described by **hundreds of variables**
- Each query uses unique subset of variables
  - Cannot index all possible combinations
- Query performance typically limited by I/O reads
  - Root N-tuples Optimized to reduce I/O

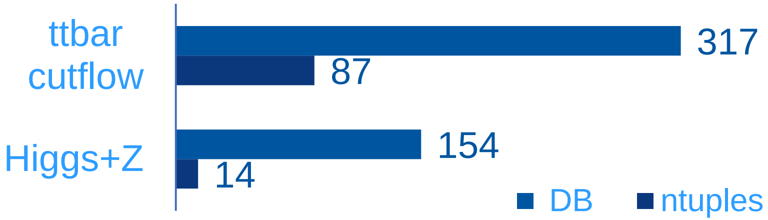
Table name	*columns*	*M rows*	*size in GB*
photon	216	89.9	**114.4**
electron	340	49.5	**94.6**
jet	171	26.8	**26.3**
muon	251	7.7	**14.2**
primary_vertex	25	89.5	**11.9**
EF (trigger)	490	7.2	**7.9**
MET_RefFinal	62	6.6	**2.3**
eventData	52	7.2	**1.4**

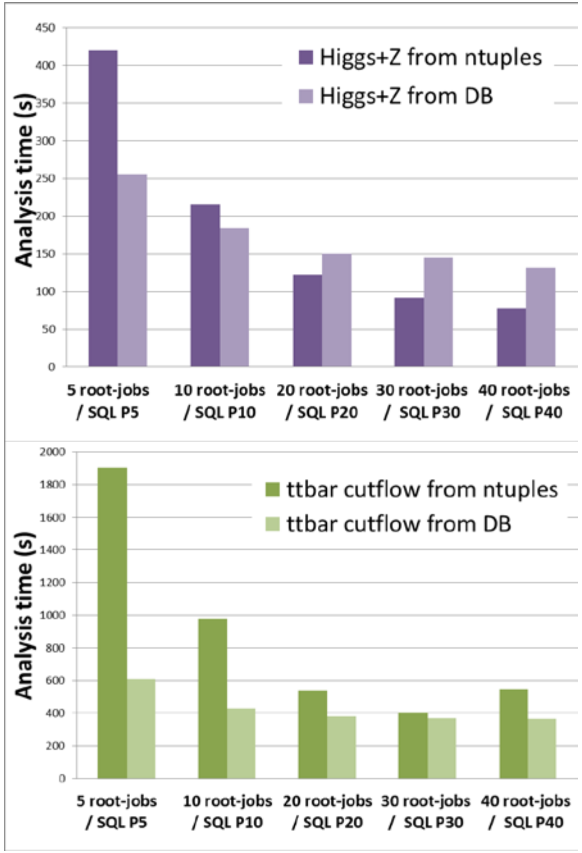


The Challenge

- **Higgs+Z:** 40 variables
- **TTbar cutflow:** 262 variables

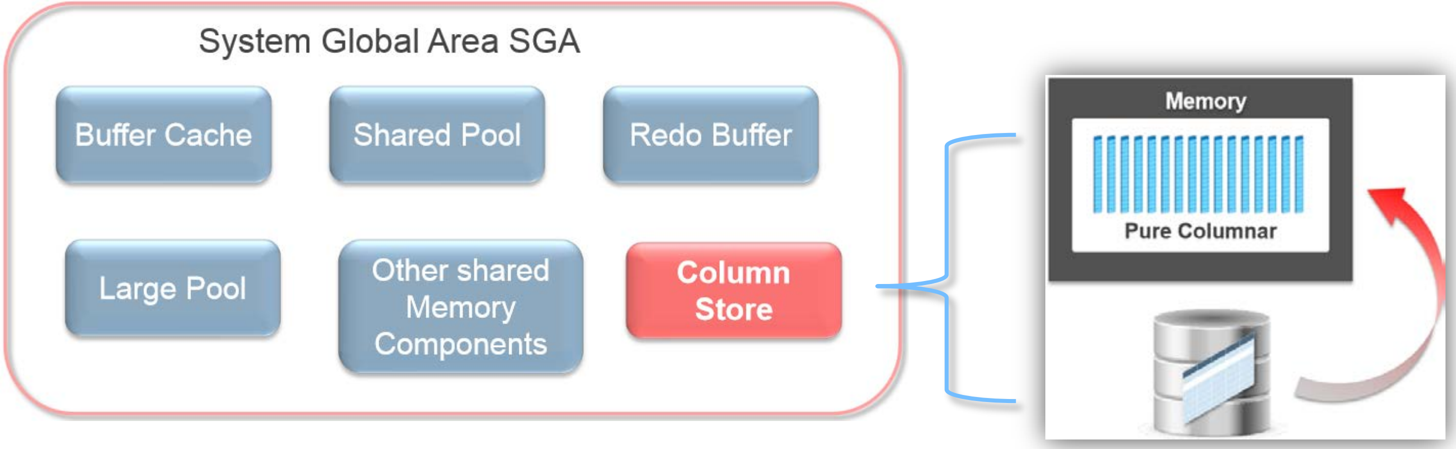
Total volume of I/O reads in GB:





In-Memory Column Store

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



In-Memory Column Store

- Both row and columnar format simultaneously in memory
  - Buffer Cache for OLTP workload and data modifications (DML)
  - IM Column Store for analytics and reporting queries
  - Guaranteed transactional consistency
- Transparent for applications
  - No code change needed
- No storage overhead
  - Row format on disk

IMC – Setup

- Very simple setup

  alter system set inmemory_size=128G scope = spfile;
- Database restart required
- Other parameters: data population, optimizer awareness…

```
SQL> show parameter inmemory

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------
inmemory_clause_default              string
inmemory_force                       string      DEFAULT
inmemory_max_populate_servers        integer     32
inmemory_query                       string      ENABLE
inmemory_size                        big integer 128G
inmemory_trickle_repopulate_servers_ integer     1
percent
optimizer_inmemory_aware             boolean     TRUE
SQL> _
```

IMC – Memory

- Distribute memory between IMC and Buffer Cache

```
MEMORY_COMPONENT                   SIZE_GB
------------------------------ ----------
SGA Target                            186
In-Memory Area                        128
DEFAULT buffer cache                   38
shared pool                            12
java pool                             3.5
large pool                            3.5
```

- Data pool (1M chunks) and Metadata pool (64K chunks)
- V$INMEMORY_AREA view to monitor IMC pools

```sql
select POOL, ROUND(ALLOC_BYTES/1024/1024/1024,2) as "ALLOC_BYTES_GB",
   ROUND(USED_BYTES/1024/1024/1024,2) as "USED_BYTES_GB",
   populate_status
 from v$inmemory_area;
```

```
POOL       ALLOC_BYTES_GB USED_BYTES_GB POPULATE_STATUS
---------- -------------- ------------- ---------------
1MB POOL           101.99            .1 DONE
64KB POOL           25.98           .02 DONE
```

IMC – Data Population

ALTER TABLE „electron” INMEMORY;

- Table can be loaded immediately (on database startup) or on-demand, when first accessed
- Data populated and re-populated in the background
- New processes IMCO, SMCO, Wnnn
- Optimizer automatically uses In-Memory table scans for fully populated tables

IMC – Data Population

- V$IM_SEGMENTS view to monitor in-memory area contents on segment level

```
select segment_name, ROUND(SUM(BYTES)/1024/1024/1024,2) "Orig. Bytes GB",
       ROUND(SUM(INMEMORY_SIZE)/1024/1024/1024,2) "In-memory GB",
       ROUND(SUM(BYTES-BYTES_NOT_POPULATED)*100/SUM(BYTES),2) "% In-memory",
       ROUND(SUM(BYTES-BYTES_NOT_POPULATED)/SUM(INMEMORY_SIZE),2) "Compression Ratio"
 from v$IM_SEGMENTS group by owner,segment_name order by 2 desc;
```

SEGMENT_NAME	Orig. Bytes GB	In-memory GB	% In-memory	Compression Ratio
electron	65.66	24.45	51.76	1.39
jet	32.27	15.08	98.45	2.11
muon	11.08	10.22	97.75	1.06
EF	3.22	.07	52.16	23.23
MET_RefFinal	2.53	2.7	100	.94
periodAllYear_v47-pro13-01_LEO	.13	.09	100	1.38

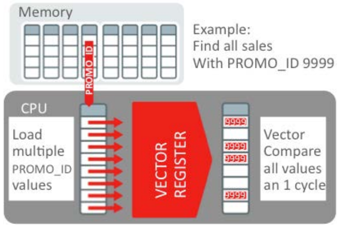
IMC - Compression

- Reduces the amount of extra memory needed for IMC
- 5 levels of compression

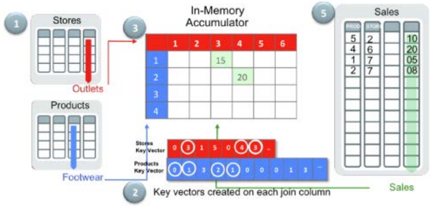
  NOCOMPRESS → FOR DML → FOR QUERY LOW|HIGH → FOR CAPACITY LOW|HIGH
- 2x-20x compression
  - Depends on data type and distribution (number of unique values)
- Filters for In-memory table scans applied directly on compressed data
  - Except FOR CAPACITY compression
- Very low performance impact on queries

IMC – Cool Features

- In-memory Storage Index – like Exadata!
  - Skip scanning table sections based on filter predicates
- SIMD Vector Processing
  - Process multiple values in single CPU instruction



- In-memory Joins
  - With Bloom Filter transformation – replace join with a filter
- In-memory Aggregation
  - Special „Accumulator” aggregates data on the fly during the IMC big table scan



Graphics: www.oracle.com, Oracle In-Memory white paper

IMC – Questions

- How much performance can we gain with In-Memory Column Store for physics queries?
- How does it cope with numeric data used in physics?
  - int, float, double
  - Very high cardinality columns – almost every value is unique
- Is it really transparent?
  - e.g. no negative impact on DML operations
- Can the analysis now be performed in real-time?

IMC Tests – Methodology

- A workload consisting of all 4 benchmark queries
- Queries run in Parallel and some also in Serial mode
- Sample 115GB set of physics experiment data
- Multiple runs of the workload for each DB configuration
  - 4 undisturbed workload runs with a stable execution plan
  - Results averaged out of all test runs
  - 3-4 warmup runs before the actual test

IMC Tests - Environment

- Server: Dell PowerEdge R820
- CPU: Intel Xeon 2.7Ghz
  32 cores / 64 threads
- Memory: DDR3 256GB
- Storage: local SSD drive and NetApp NAS
- OS: Red Hat Enterprise Linux Server 6.5
- RDBMS: 12.1.0.2 Enterprise Edition - 64bit Production

IMC Tests – DB Configurations

- Row format only – Buffer Cache
  - With empty buffer (flushed before each query)
  - With pre-warmed big buffer – 160GB
  - With pre-warmed reduced buffer – 32/80GB
- In-Memory Columnar format – IM Column Store
  - In addition to Buffer Cache
  - All possible compression levels
  - BC/IMC pool sizes dependent on compression: total = 160GB

IMC Tests – Compression Rates

- For tables used in physics analysis – 115GB in row format
  - „electron”, „jet”, „muon” → typical physics data: mixture of int, float, double
  - „Event Filter” → only boolean columns (mostly false), best compression
  - „Missing Energy” → table with float & double columns, worst compression

Table:	electron	jet	muon	Event Filter	Missing Energy	Total (all tables)	Size GB (all tables)
NO MEMCOMPRESS	1.07	1.41	1.10	0.70	1.42	**1.14**	**101.2**
FOR DML	1.17	1.62	1.12	0.95	1.46	**1.26**	**91.5**
FOR QUERY LOW	1.39	2.11	1.06	21.75	0.94	**1.51**	**76.2**
FOR QUERY HIGH	1.60	2.77	1.30	30.08	1.49	**1.82**	**63.0**
FOR CAPACITY LOW	2.00	3.71	1.80	39.44	1.54	**2.32**	**49.5**
FOR CAPACITY HIGH	2.44	4.83	2.44	56.28	1.71	**2.89**	**39.7**

IMC Tests – Compression Rates

- For tables used in physics analysis – 115GB in row format
  - „electron”, … int, float, double
  - „Event Filt… compression
  - „Missing E… rst compression

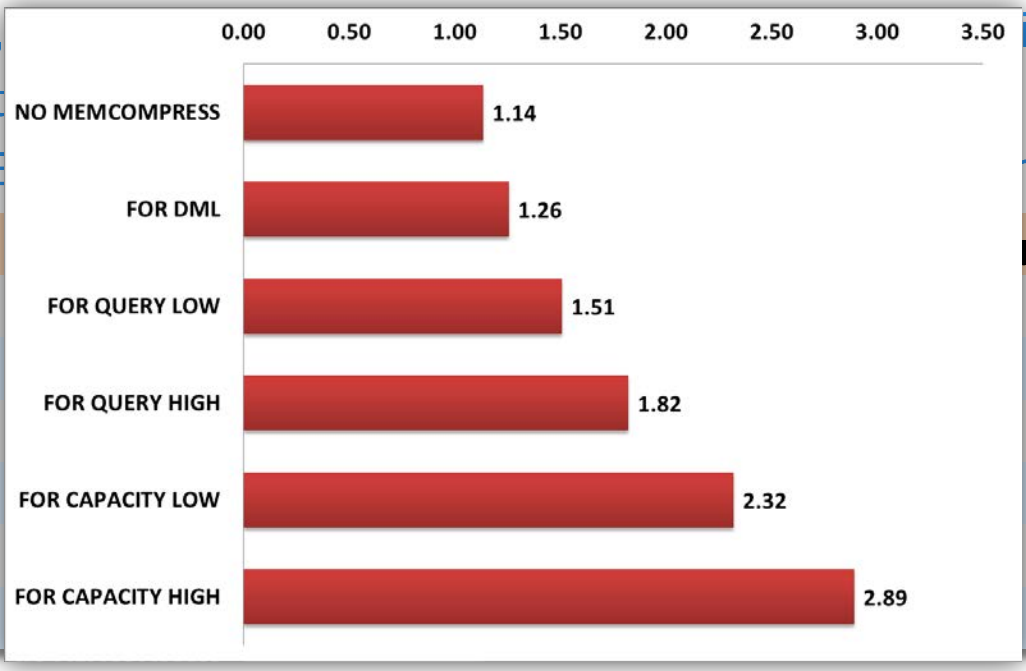
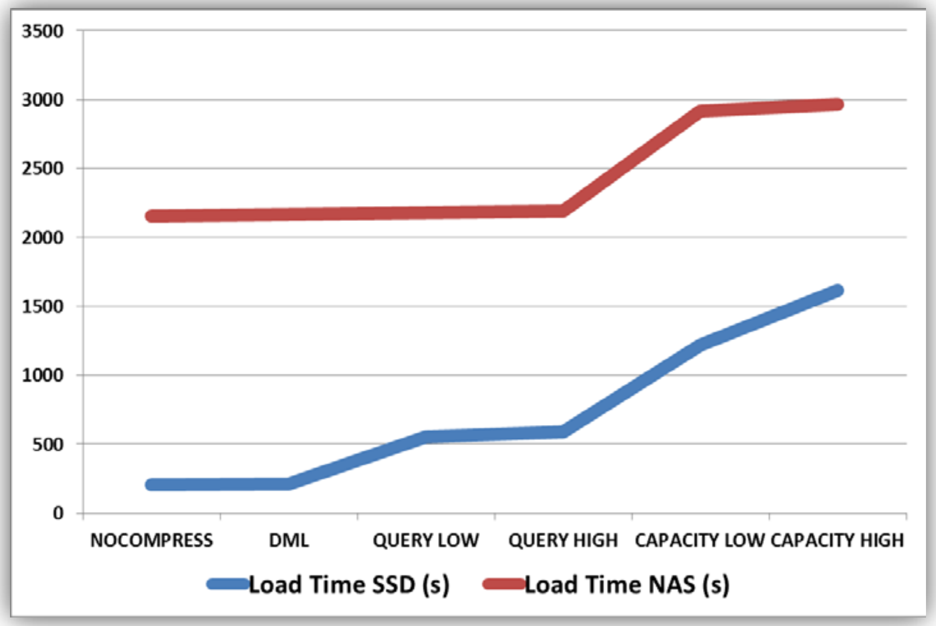


Table:	…les)	Size GB (all tables)
NO MEMCOMPRESS		101.2
FOR DML		91.5
FOR QUERY LOW		76.2
FOR QUERY HIGH		63.0
FOR CAPACITY LOW		49.5
FOR CAPACITY HIGH		39.7

IMC Tests – Population Time

- For tables used in physics analysis – 115GB in row format
- 32 populate processes
- SSD vs NAS storage

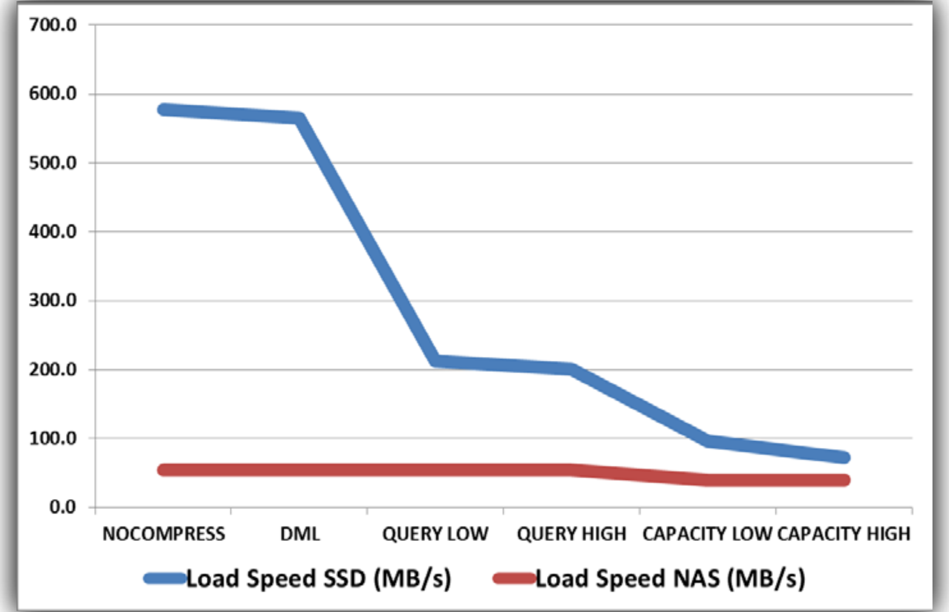
Compression Level	Load Time SSD (s)	Load Time NAS (s)
NO MEMCOMPRESS	3:24	35:57
FOR DML	3:28	36:04
FOR QUERY LOW	9:15	36:20
FOR QUERY HIGH	9:48	36:30
FOR CAPACITY LOW	20:20	48:34
FOR CAPACITY HIGH	26:57	49:24



IMC Tests – Population Time

- For tables used in physics analysis – 115GB in row format
- 32 populate processes
- SSD vs NAS storage

Compression Level	Load Speed SSD (MB/s)	Load Speed NAS (MB/s)
NO MEMCOMPRESS	576.7	54.5
FOR DML	565.6	54.4
FOR QUERY LOW	212.0	54.0
FOR QUERY HIGH	200.1	53.7
FOR CAPACITY LOW	96.4	40.4
FOR CAPACITY HIGH	72.8	39.7



IMC Tests – Population CPU

- **INMEMORY_MAX_POPULATE_SERVERS** parameter controls resources dedicated to IMC Store population
  - By default ½ of available CPU cores

```
top - 16:02:43 up 43 days,  3:04,  3 users,  load average: 21.67, 15.32, 11.15
Tasks: 1464 total,  31 running, 1433 sleeping,   0 stopped,   0 zombie
Cpu(s): 40.6%us,  3.3%sy,  0.0%ni, 55.3%id,  0.7%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:  264548792k total, 248388696k used, 16160096k free,   350608k buffers
Swap:  1999992k total,        0k used,  1999992k free, 35801544k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
57628 oracle    20   0  187g 637m  19m R 100.0  0.2   2:11.90 ora_w00i_imcpro
57640 oracle    20   0  187g 617m  19m R 100.0  0.2   2:41.59 ora_w00m_imcpro
57594 oracle    20   0  187g 805m  22m R 100.0  0.3   3:45.96 ora_w002_imcpro
57596 oracle    20   0  187g 586m  21m R 100.0  0.2   3:27.39 ora_w003_imcpro
57599 oracle    20                                          5 ora_w004_imcpro
57603 oracle    20                                          4 ora_w006_imcpro
57609 oracle    20                                          7 ora_w009_imcpro
57615 oracle    20                                          8 ora_w00c_imcpro
57617 oracle    20   0  187g 743m  20m R 100.0  0.3   3:02.50 ora_w00d_imcpro
57623 oracle    20   0  187g 754m  22m R 100.0  0.3   3:03.99 ora_w00g_imcpro
57630 oracle    20   0  187g 459m  21m R 100.0  0.2   3:34.24 ora_w00j_imcpro
57632 oracle    20   0  187g 617m  20m R 100.0  0.2   3:57.38 ora_w00k_imcpro
57638 oracle    20   0  187g 521m  20m R 100.0  0.2   3:56.26 ora_w00l_imcpro
57646 oracle    20   0  187g 600m  19m R 100.0  0.2   2:27.77 ora_w00p_imcpro
57648 oracle    20   0  187g 587m  20m R 100.0  0.2   3:30.33 ora_w00q_imcpro
```

SSD 41% CPU

```
top - 19:58:41 up 44 days,  7:00,  3 users,  load average: 5.50, 2.99, 2.81
Tasks: 1454 total,  12 running, 1442 sleeping,   0 stopped,   0 zombie
Cpu(s): 17.7%us,  0.4%sy,  0.0%ni, 78.6%id,  3.2%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  264548792k total, 232140772k used, 32408020k free,   442512k buffers
Swap:  1999992k total,        0k used,  1999992k free, 28127592k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
10502 oracle    20   0  186g 411m  23m R 100.0  0.2   1:04.42 ora_w000_imcpro
10906 oracle    20   0  187g 694m  19m R 100.0  0.3   0:42.02 ora_w008_imcpro
10921 oracle    20   0  186g 431m  18m R 100.0  0.2   0:22.82 ora_w00f_imcpro
10909 oracle    20   0  186g 421m  18m R 99.9  0.2   0:40.28 ora_w009_imcpro
10913 oracle    20                                          8 ora_w00b_imcpro
10927 oracle    20                                          4 ora_w00i_imcpro
10901 oracle    20                                          0 ora_w006_imcpro
10935 oracle    20                                          6 ora_w00k_imcpro
10899 oracle    20   0  187g 293m  20m R 88.7  0.1   0:45.29 ora_w005_imcpro
10897 oracle    20   0  187g 422m  20m R 88.4  0.2   0:49.56 ora_w004_imcpro
10911 oracle    20   0  186g 408m  18m S 73.2  0.2   0:35.92 ora_w00a_imcpro
10938 oracle    20   0  186g  85m  19m R 65.0  0.0   0:02.04 ora_w00l_imcpro
10929 oracle    20   0  186g  34m  21m S 32.6  0.0   0:05.57 ora_w00j_imcpro
 6278 root      20   0     0    0    0 S 10.6  0.0  29:11.59 rpciod/25
10940 oracle    20   0  186g  20m  16m S  2.0  0.0   0:00.06 ora_w00m_imcpro
```

NAS 18% CPU

IMC Tests – Physics Benchmark

- Query 1: „Electron Counter”
  - Count number of electrons meeting certain criteria
- Query 2: „Electron Filter”
  - Find good quality electron-positron pairs, calculate properties
- Query 3: „Interesting Events” (TTbar cutflow)
  - Find interesting collision events, meeting certain criteria
- Query 4: „Higgs Boson” (Higgs+Z)
  - Find collision events in which Higgs boson was produced

IMC Tests – Benchmark Query 1

- „Electron Counter”
  - Big table scan (66GB) with filters

```
select count(*) from DATA12_8TEV."electron" where "pt">18000. and "eta"<1.5;
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3317014141

-------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                        | Name     | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |    TQ  |IN-OUT| PQ Distrib |
-------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                 |                                                  |       |       |        |      |            |
|   1 |  SORT AGGREGATE                  |                                                  |       |       |        |      |            |
|   2 |   PX COORDINATOR                 |                                                  |       |       |        |      |            |
|   3 |    PX SEND QC (RANDOM)           |                                                  |       |       |  Q1,00 | P->S | QC (RAND)  |
|   4 |     SORT AGGREGATE               |                                                  |       |       |  Q1,00 | PCWP |            |
|   5 |      PX BLOCK ITERATOR           |          |  6718K|    51M| 62386   (5)| 00:00:03 |     1 |1048575|  Q1,00 | PCWC |            |
|*  6 |       TABLE ACCESS INMEMORY FULL | electron |  6718K|    51M| 62386   (5)| 00:00:03 |     1 |1048575|  Q1,00 | PCWP |            |
-------------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   6 - inmemory("pt">1.8E+004F AND "eta"<1.5E+000F)
       filter("pt">1.8E+004F AND "eta"<1.5E+000F)
```

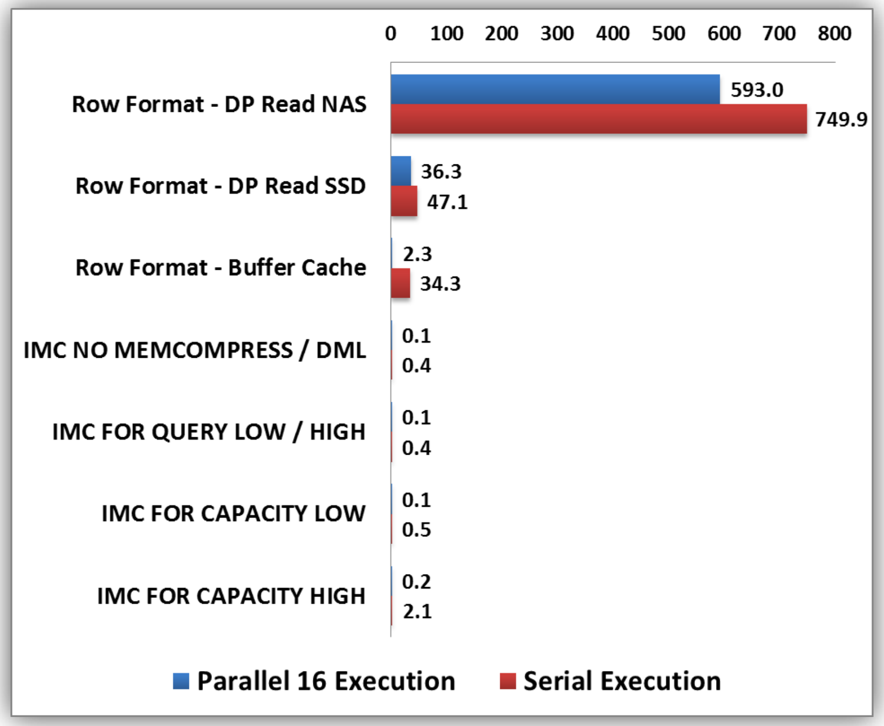
In-Memory scan: electron
TABLE ACCESS INMEMORY FULL

Fraction of a second

IMC Tests – Benchmark Results Q1

- „Electron Counter”

Configuration	Parallel 16 Execution	Serial Execution
Row Format - Direct Path Read NAS	593.0	749.9
Row Format - Direct Path Read SSD	36.3	47.1
Row Format - Buffer Cache	2.3	34.3
IMC NO MEMCOMPRESS / DML	0.1	0.4
IMC FOR QUERY LOW / HIGH	0.1	0.4
IMC FOR CAPACITY LOW	0.1	0.5
IMC FOR CAPACITY HIGH	0.2	2.1



IMC Tests – Benchmark Results Q1

- „Electron Counter”

IMC vs Buffer Cache serial

86x faster!!!

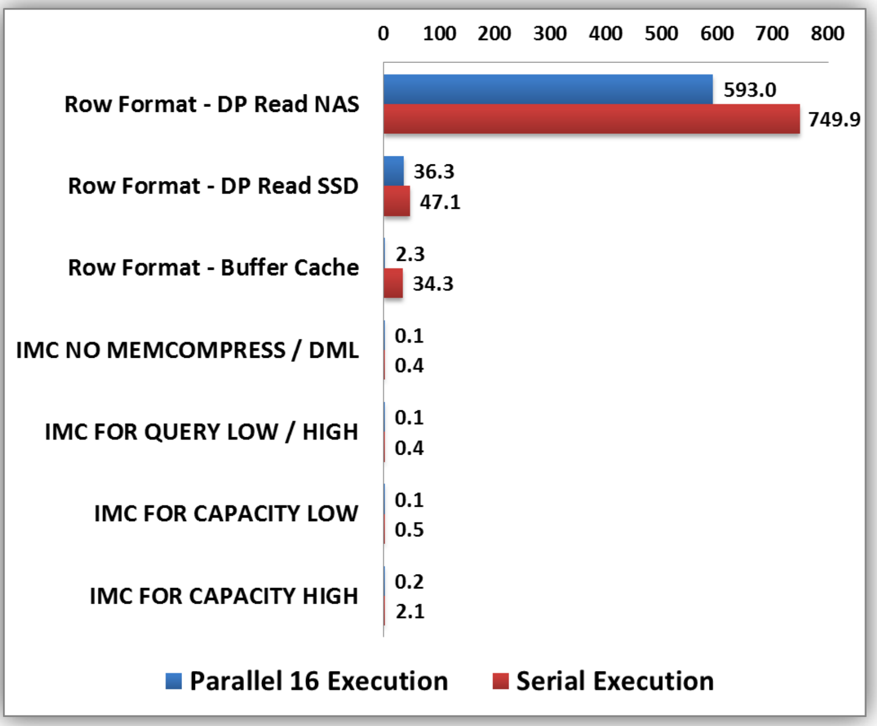
Row Format - Buffer Cache	2.3	34.3

IMC vs SSD Direct Path serial

118x faster!!!

IMC vs NAS Direct Path serial

1900x faster!!!



IMC Tests – Benchmark Query 2

- „Electron Filter”
    - Big table scan with 2 self-joins and a join to time dimension
    - physics analytic calculations of electron properties

```
select "RunNumber","EventNumber",el_sel_n,
    electron0."electron_i" as mu_id0, electron1."electron_i" as mu_id1,
    electron0."charge" as mu_charge0, electron1."charge" as mu_charge1,
    electron0."pt"/1000. as mu_pt0, electron1."pt"/1000. as mu_pt1,
    electron0."eta" as el_eta0, electron1."eta" as el_eta1,
   (case when abs(electron0."phi"-electron1."phi")<acos(-1.) then sqrt(POWER(abs(electron0."phi"-electron1."phi"),2)
    +POWER(abs(electron0."eta"-electron1."eta"),2)) else sqrt(POWER( 2.*acos(-1.) - abs(electron0."phi"-electron1."phi"),2)
    +POWER(abs(electron0."eta"-electron1."eta"),2)) end) as DELTAR,
    ANALYSISTOOLS.PHYSANALYSIS.INV_MASS_LEPTONS(electron0."E",electron1."E",electron0."px",electron1."px",
    electron0."py",electron1."py",electron0."pz",electron1."pz")/1000. as INV_MASS
 from DATA12_8TEV."periodAllYear_v47-pro13-01"
      INNER JOIN (select "RunNumber","EventNumber",COUNT(*) as el_sel_n
                  from "sel_electron_v" group by ("RunNumber","EventNumber")) USING ("RunNumber","EventNumber")
      INNER JOIN "sel_electron_v" electron0 USING ("RunNumber","EventNumber")
      INNER JOIN "sel_electron_v" electron1 USING ("RunNumber","EventNumber")
where electron0."electron_i"<electron1."electron_i" and electron0."charge" != electron1."charge" and el_sel_n=2;
```

IMC Tests – Benchmark Query 2

- „Electron Filter”
  - Big table scan with 2 self-joins and a join to time dimension
  - physics analytic calculations of electron properties

```
Execution Plan
----------------------------------------------------------
Plan hash value: 950320988

--------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                             | Name                     | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |   TQ  |IN-OUT| PQ Distrib |
--------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                                      |                          |     1 |   196 | 74334  (20)| 00:00:03 |       |       |       |      |            |
|                                                             |                          |       |       |            |          |       |       |       |      |            |
|                                                             | :TQ10002                 |     1 |   196 | 74334  (20)| 00:00:03 |       |       | Q1,02 | P->S | QC (RAND)  |
|                                                             |                          |     1 |   196 | 74334  (20)| 00:00:03 |       |       | Q1,02 | PCWP |            |
|                                                             | :BF0000                  |     1 |   185 | 74199  (20)| 00:00:03 |       |       | Q1,02 | PCWP |            |
|                                                             |                          |     1 |   185 | 74199  (20)| 00:00:03 |       |       | Q1,02 | PCWP |            |
|                                                             | :TQ10001                 |     1 |   185 | 74199  (20)| 00:00:03 |       |       | Q1,01 | P->P | BROADCAST  |
|                                                             |                          |     1 |   185 | 74199  (20)| 00:00:03 |       |       | Q1,01 | PCWP |            |
|                                                             |                          |     1 |   185 | 74199  (20)| 00:00:03 |       |       | Q1,01 | PCWP |            |
|                                                             |                          |     1 |   112 | 74199  (20)| 00:00:03 |       |       | Q1,01 | PCWP |            |
|                                                             |                          |     1 |    39 | 74199  (20)| 00:00:03 |       |       | Q1,01 | PCWP |            |
|                                                             |                          |       |       |            |          |       |       | Q1,01 | PCWC |            |
|                                                             |                          |     1 |    38 | 74199  (20)| 00:00:03 |       |       | Q1,01 | PCWP |            |
|                    PX RECEIVE                               |                          |     1 |    38 | 74199  (20)| 00:00:03 |       |       | Q1,01 | PCWP |            |
|  14 |                 PX SEND HASH                          | :TQ1                                                   :03 |       |       | Q1,00 | P->P | HASH       |
|  15 |                  HASH GROUP BY                        |                                                         03 |       |       | Q1,00 | PCWP |            |
|  16 |                   PX BLOCK ITERATOR                   |                                                         03 |     1 |1048575| Q1,00 | PCWC |            |
|* 17 |                    TABLE ACCESS INMEMORY FULL         | ele                                                     03 |     1 |1048575| Q1,00 | PCWP |            |
|* 18 |             TABLE ACCESS BY GLOBAL INDEX ROWID BATCHED| ele                                                     01 | ROWID | ROWID | Q1,01 | PCWP |            |
|* 19 |              INDEX RANGE SCAN                         | IX                                                      01 |       |       | Q1,01 | PCWP |            |
|* 20 |            INDEX RANGE SCAN                                                                                      01 |       |       | Q1,01 | PCWP |            |
|* 21 |           TABLE ACCESS BY GLOBAL INDEX ROWID                                                                     01 | ROWID | ROWID | Q1,01 | PCWP |            |
|  22 |       JOIN FILTER USE                                 | :                                                       01 |       |       | Q1,02 | PCWP |            |
|  23 |        PX BLOCK ITERATOR                              |                                                         01 |       |       | Q1,02 | PCWC |            |
|* 24 |         TABLE ACCESS INMEMORY FULL                    | period                                                  01 |       |       | Q1,02 | PCWP |            |
--------------------------------------------------------------------------------------------------------------------------------------------------------------
```

In-Memory scan: electron
TABLE ACCESS
INMEMORY FULL

Combined with 2 index
scans (buffer cache)
INDEX RANGE SCAN

IMC Tests – Benchmark Query 2

- „Electron Filter"
  - Big table scan w
  - physics analytic

```
select "RunNumber","EventNumber",
  electron0."electron_i" as

  +POWER(abs(electron0."eta"-electro
  ANALYSISTOOLS.PHYSANALYSIS.INV_
  electron0."py",electron1."py",electr
from DATA12_8TEV."periodAllYear_v47-
  INNER JOIN (select "RunNumber","
        from "sel_electron_v" g
  INNER JOIN "sel_electron_v" electr
  INNER JOIN "sel_electron_v" electr
where electron0."electron_i"<electron1.
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3145764102

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                           | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |    TQ  |IN-OUT| PQ Distrib |
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                                |     1 |   192 | 74340  (20)| 00:00:03 |       |       |        |      |            |
|   1 |  TEMP TABLE TRANSFORMATION               |                                |       |       |            |          |       |       |        |      |            |
|   2 |   PX COORDINATOR                         |                                |       |       |            |          |       |       |        |      |            |
|   3 |    PX SEND QC (RANDOM)                   | :TQ10000                       | 1849K|   128M| 74199  (20)| 00:00:03 |       |       |  Q1,00 | P->S | QC (RAND)  |
|   4 |     LOAD AS SELECT (TEMP SEGMENT MERGE)| SYS_TEMP_0FD9D668B_4B7AC6      |       |       |            |          |       |       |  Q1,00 | PCWP |            |
|   5 |      PX BLOCK ITERATOR                   |                                | 1849K|   128M| 74199  (20)| 00:00:03 |     1 |1048575|  Q1,00 | PCWC |            |
|*  6 |       TABLE ACCESS INMEMORY FULL         |                                | 1849K|   128M| 74199  (20)| 00:00:03 |     1 |1048575|  Q1,00 | PCWP |            |
|   7 |   X COORDINATOR                          |                                |       |       |            |          |       |       |        |      |            |
|     |    PX SEND QC (RANDOM)                   | :TQ2                           |     1 |   192 |   142  (87)| 00:00:01 |       |       |  Q2,06 | P->S | QC (RAND)  |
|     |     HASH JOIN BUFFERED                   |                                |       |   192 |   142  (87)| 00:00:01 |       |       |  Q2,06 | PCWP |            |
|     |      PX RECEIVE                          |                                |       |       |            |          |       |       |        |      |            |
|  11 |       PX SEND HYBRID HASH                |                                |       |       |            |          |       |       |        |      |        HASH|
|     |        STATISTICS COLLECTOR              |                                |       |       |            |          |       |       |        |      |            |
|     |         HASH JOIN BUFFERED               |                                |       |       |            |          |       |       |        |      |            |
|     |          HASH JOIN                       |                                |       |       |            |          |       |       |        |      |            |
|     |           PX RECEIVE                     |                                |       |       |            |          |       |       |        |      |            |
|     |            PX SEND HASH                  |                                |       |       |            |          |       |       |        |      |            |
|     |             VIEW                         |                                |       |       |            |          |       |       |        |      |            |
|     |              FILTER                      |                                |       |       |            |          |       |       |        |      |            |
|     |               HASH GROUP BY              |                                |       |       |            |          |       |       |        |      |            |
|     |                PX RECEIVE                |                                |     1 |    11 |     2   (0)| 00:00:01 |       |       |  Q2,01 | PCWP |            |
|     |                 PX SEND HASH             | :TQ20000                       |     1 |    11 |     2   (0)| 00:00:01 |       |       |  Q2,00 | P->P | HASH       |
|  22 |                  HASH GROUP BY           |                                |     1 |    11 |     2   (0)| 00:00:01 |       |       |  Q2,00 | PCWP |            |
|  23 |                   VIEW                   |                                |     1 |    11 |     2   (0)| 00:00:01 |       |       |  Q2,00 | PCWP |            |
|  24 |                    PX BLOCK ITERATOR     |                                |     1 |    46 |     2   (0)| 00:00:01 |       |       |  Q2,00 | PCWC |            |
|  25 |                     TABLE ACCESS FULL    | SYS_TEMP_0FD9D668B_4B7AC6      |     1 |    46 |     2   (0)| 00:00:01 |       |       |  Q2,00 | PCWP |            |
|  26 |           PX RECEIVE                     |                                |     1 |    71 |     2   (0)| 00:00:01 |       |       |  Q2,04 | PCWP |            |
|  27 |            PX SEND HASH                  | :TQ20002                       |     1 |    71 |     2   (0)| 00:00:01 |       |       |  Q2,02 | P->P | HASH       |
|  28 |             VIEW                         |                                |     1 |    71 |     2   (0)| 00:00:01 |       |       |  Q2,02 | PCWP |            |
|  29 |              PX BLOCK ITERATOR           |                                |     1 |    46 |     2   (0)| 00:00:01 |       |       |  Q2,02 | PCWC |            |
|  30 |               TABLE ACCESS FULL          | SYS_TEMP_0FD9D668B_4B7AC6      |     1 |    46 |     2   (0)| 00:00:01 |       |       |  Q2,02 | PCWP |            |
|  31 |          PX RECEIVE                      |                                |     1 |    71 |     2   (0)| 00:00:01 |       |       |  Q2,04 | PCWP |            |
|  32 |           PX SEND HASH                   | :TQ20003                       |     1 |    71 |     2   (0)| 00:00:01 |       |       |  Q2,03 | P->P | HASH       |
|  33 |            VIEW                          |                                |     1 |    71 |     2   (0)| 00:00:01 |       |       |  Q2,03 | PCWP |            |
|  34 |             PX BLOCK ITERATOR            |                                |     1 |    46 |     2   (0)| 00:00:01 |       |       |  Q2,03 | PCWC |            |
|  35 |              TABLE ACCESS FULL           | SYS_TEMP_0FD9D668B_4B7AC6      |     1 |    46 |     2   (0)| 00:00:01 |       |       |  Q2,03 | PCWP |            |
|  36 |      PX RECEIVE                          |                                | 6961K|   73M|   102  (88)| 00:00:01 |       |       |  Q2,06 | PCWP |            |
|  37 |       PX SEND HYBRID HASH                | :TQ20005                       | 6961K|   73M|   102  (88)| 00:00:01 |       |       |  Q2,05 | P->P | HYBRID HASH|
|  38 |        PX BLOCK ITERATOR                 |                                | 6961K|   73M|   102  (88)| 00:00:01 |       |       |  Q2,05 | PCWC |            |
|  39 |         TABLE ACCESS INMEMORY FULL       | periodAllYear_v47-pro13-01_LEO | 6961K|   73M|   102  (88)| 00:00:01 |       |       |  Q2,05 | PCWP |            |
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

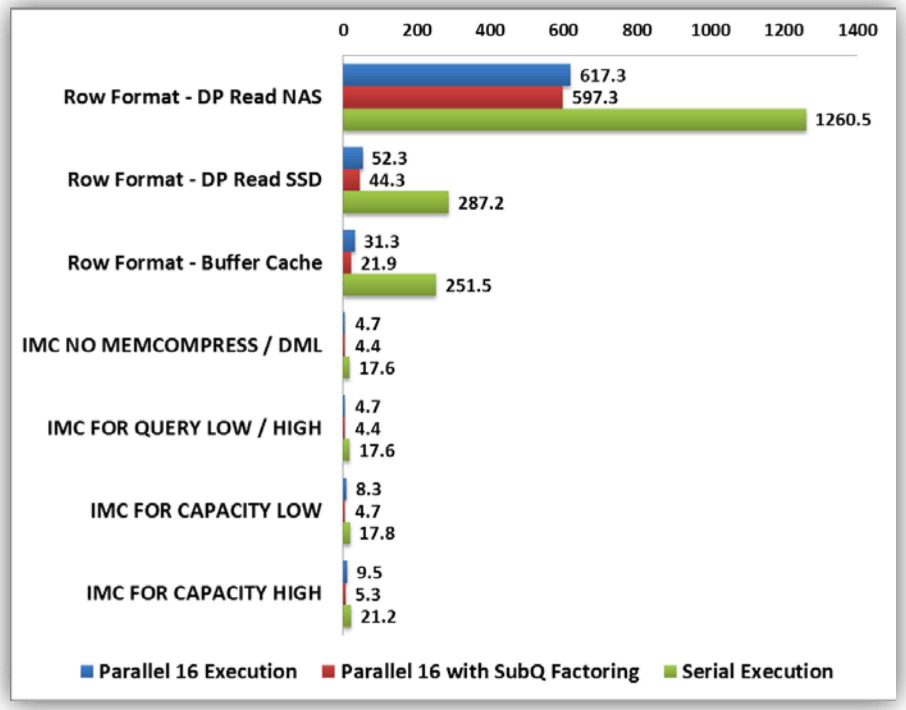
In-Memory scan: electron
TABLE ACCESS INMEMORY FULL

Sub-query factoring – temporary table
LOAD AS SELECT (TEMP SEGMENT MERGE)

IMC Tests – Benchmark Results Q2

- „Electron Filter"

Configuration	Parallel 16 Execution	Parallel 16 with SubQ Factoring	Serial Execution
Row Format - DP Read NAS	617.3	597.3	1260.5
Row Format - DP Read SSD	52.3	44.34	287.2
Row Format - Buffer Cache	31.3	21.9	251.5
IMC NO MEMCOMPRESS / DML	4.7	4.4	17.6
IMC FOR QUERY LOW / HIGH	4.7	4.4	17.6
IMC FOR CAPACITY LOW	8.3	4.7	17.8
IMC FOR CAPACITY HIGH	9.5	5.3	21.2



IMC Tests – Benchmark Results Q2

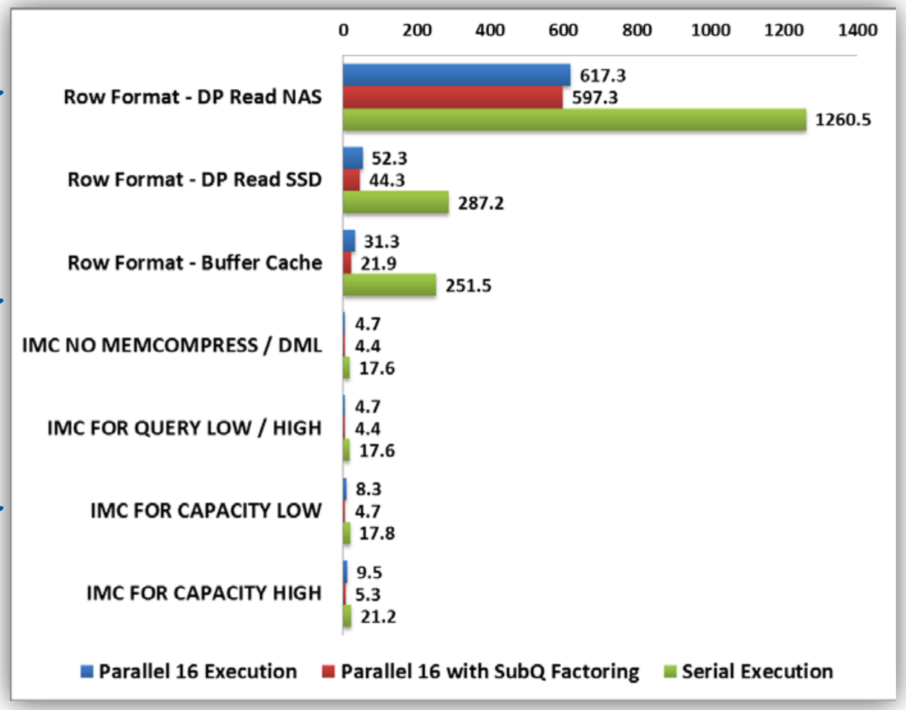
- „Electron Filter”

Confi…			…ution
Row …			…60.5
Row …			…37.2
Row Format - Buffer Cache	31.3	21.9	251.5
IMC …			…7.6
IMC …			….6
IMC …			…7.8
IMC …			…1.2

IMC vs Buffer Cache serial
14x faster!!

IMC vs SSD Direct Path serial
16x faster!!

IMC vs NAS Direct Path serial
72x faster!!



IMC Tests – Benchmark Query 3

- „Interesting Events” (TTbar cutflow)
  - 3 scans of a big table with filters and aggregations
  - 2x self-join, 100GB of data scanned
  - Sub-queries converted into views for simplicity

```
create or replace view "allJets" as (select /*+ FULL("jet") */ "RunNumber","EventNumber","jet_i","E","emscale_eta","emscale_phi","emscale_pt","emscale_E","EtaOrigin",
        "PhiOrigin","MOrigin","EMJES_EtaCorr","eta","phi","pt","isBadLoose","BCH_CORR_JET","BCH_CORR_CELL","jvtxf","fl_w_SV0","fl_w_JetFitterCOMBNN",
        ANALYSISTOOLS.MV1.mv1Eval_java("fl_w_IP3D","fl_w_SV1","fl_w_JetFitterCOMBNN", "pt","eta") as "MV1","E" as "correctedE","pt" as "correctedPt",0 as "isMC"
 from DATA12_8TEV."jet" );
create or replace view "noJvfJets" as (select  * from "allJets" where abs("emscale_eta"+"EMJES_EtaCorr")<2.5 and "correctedPt">25000.0 and "correctedE">0.);
...
create or replace view "badJets_count" as (select "RunNumber","EventNumber",COUNT(*) as N from "badJets" GROUP BY ("RunNumber","EventNumber") );

select "RunNumber",COUNT(1) as "GRL_Events",
         COUNT(case when 1=1 and "goodJets_count".N>=4  then 1 END) as "GRL_4goodJ_events",
         COUNT(case when 1=1 and "goodJets_count".N>=4 and "badJets_count".N is NULL then 1 END) as "GRL_4goodJ_noBadJ_events",
         COUNT(case when 1=1 and "goodJets_count".N>=4 and "badJets_count".N is NULL and "taggedJets_count".N>=1 then 1 END) as "GRL_4goodJ_noBadJ_1TagJ_events"
 from DATA12_8TEV."periodAllYear_v47-pro13-01" LEFT OUTER JOIN "goodJets_count" USING ("RunNumber","EventNumber")
        LEFT OUTER JOIN "taggedJets_count" USING ("RunNumber","EventNumber")
        LEFT OUTER JOIN "badJets_count" USING ("RunNumber","EventNumber") group by "RunNumber";
```

IMC Tests – Benchmark Query 3

- „Interesting Events” (TTbar cutflow)

Execution Plan

Plan hash value: 598906710

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time	Pstart	Pstop	TQ	IN-OUT	PQ Distrib
0	SELECT STATEMENT		3	249		96716 (10)	00:00:04					
1	PX COORDINATOR											
2	PX SEND QC (RANDOM)	:TQ10									P->S	QC (RAND)
3	HASH GROUP BY										PCWP	
4	PX RECEIVE										PCWP	
5	PX SEND HASH	:TQ10									P->P	HASH
6	HASH GROUP BY										PCWP	
* 7	HASH JOIN RIGHT OUTER										PCWP	
8	PX RECEIVE										PCWP	
9	PX SEND BROADCAST	:TQ10									P->P	BROADCAST
10	VIEW	badJets_count								Q1,03	PCWP	
11	HASH GROUP BY			1724K	2856K	31161 (7)	00:00:02			Q1,03	PCWP	
12	PX RECEIVE		80255	1724K		31161 (7)	00:00:02			Q1,03	PCWP	
13	PX SEND HASH	:TQ10000	80255	1724K		31161 (7)	00:00:02			Q1,00	P->P	HASH
14	HASH GROUP BY		80255	1724K	2856K	31161 (7)	00:00:02			Q1,00	PCWP	
15	PX BLOCK ITERATOR		80255	1724K		31158 (7)	00:00:02	1	1048575	Q1,00	PCWC	
* 16	TABLE ACCESS INMEMORY FULL	jet	80255	1724K		31158 (7)	00:00:02	1	1048575	Q1,00	PCWP	
* 17	HASH JOIN RIGHT OUTER		6961K	391M		65217 (11)	00:00:03			Q1,06	PCWP	
18	PX RECEIVE		30068	704K		31891 (9)	00:00:02			Q1,06	PCWP	
19	PX SEND BROADCAST	:TQ10004	30068	704K		31891 (9)	00:00:02			Q1,04	P->P	BROADCAST
20	VIEW	count	30068	704K		31891 (9)	00:00:02			Q1,04	PCWP	
21	HASH GROUP BY		30068	910K	1544K	31891 (9)	00:00:02			Q1,04	PCWP	
22	PX RECEIVE		30068	910K		31891 (9)	00:00:02			Q1,04	PCWP	
23	PX SEND HASH	:TQ10001	[illegible]	910K		31891 (9)	00:00:02			Q1,01	P->P	HASH
24	HASH GROUP BY									Q1,01	PCWP	
25	PX BLOCK ITERATOR							1	1048575	Q1,01	PCWC	
* 26	TABLE ACCESS INMEMORY FULL	jet						1	1048575	Q1,01	PCWP	
* 27	HASH JOIN RIGHT OUTER									Q1,06	PCWP	
28	PX RECEIVE									Q1,06	PCWP	
29	PX SEND BROADCAST	:TQ10								Q1,05	P->P	BROADCAST
30	VIEW	tagge								Q1,05	PCWP	
31	HASH GROUP BY									Q1,05	PCWP	
32	PX RECEIVE									Q1,05	PCWP	
33	PX SEND HASH	:TQ10002	1503	70641		33157 (12)	00:00:02			Q1,02	P->P	HASH
34	HASH GROUP BY		1503	70641		33157 (12)	00:00:02			Q1,02	PCWP	
35	PX BLOCK ITERATOR		1503	70641		33156 (12)	00:00:02	1	1048575	Q1,02	PCWC	
* 36	TABLE ACCESS INMEMORY FULL	jet	1503	70641		33156 (12)	00:00:02	1	1048575	Q1,02	PCWP	
37	PX BLOCK ITERATOR		6961K	73M		104 (86)	00:00:01			Q1,06	PCWC	
38	TABLE ACCESS INMEMORY FULL	periodAllYear_v47-pro13-01_LE0	6961K	73M		104 (86)	00:00:01			Q1,06	PCWP	

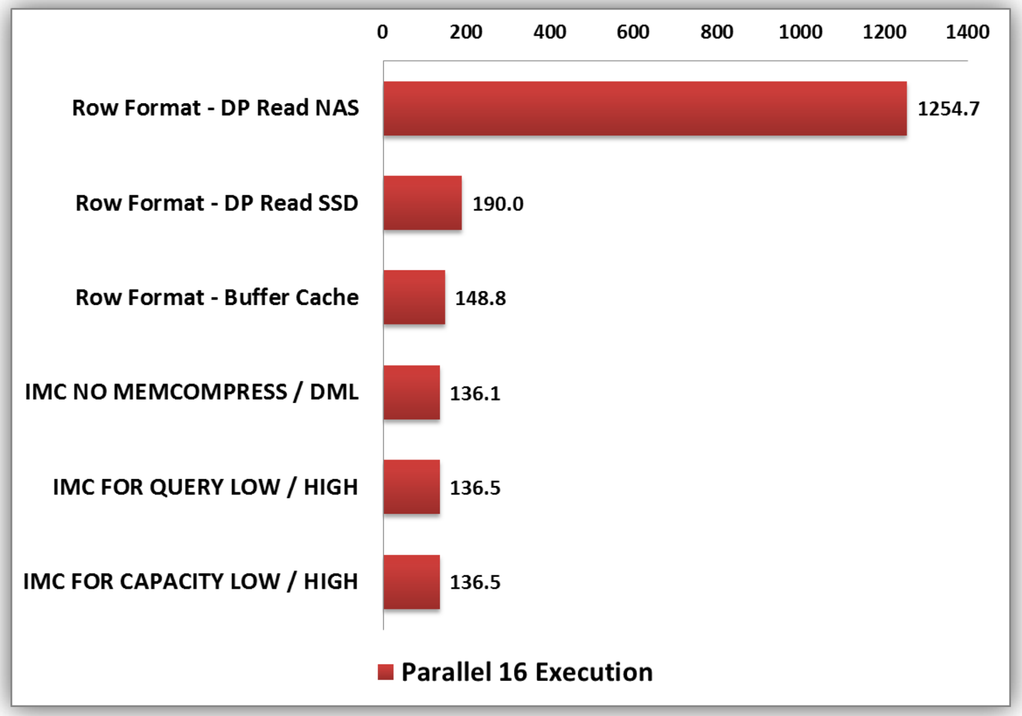
3x In-Memory scan: jet
TABLE ACCESS INMEMORY FULL

No In-Memory joins
HASH JOIN RIGHT OUTER

IMC Tests – Benchmark Results Q3

- „Interesting Events”

Configuration	Parallel 16 Execution
Row Format - DP Read NAS	1254.7
Row Format - DP Read SSD	190.0
Row Format - Buffer Cache	148.8
IMC NO MEMCOMPRESS / DML	136.1
IMC FOR QUERY LOW / HIGH	136.5
IMC FOR CAPACITY LOW / HIGH	136.5



IMC Tests – Benchmark Results Q3

- „Interesting Events”

IMC vs Buffer Cache

1.1x faster

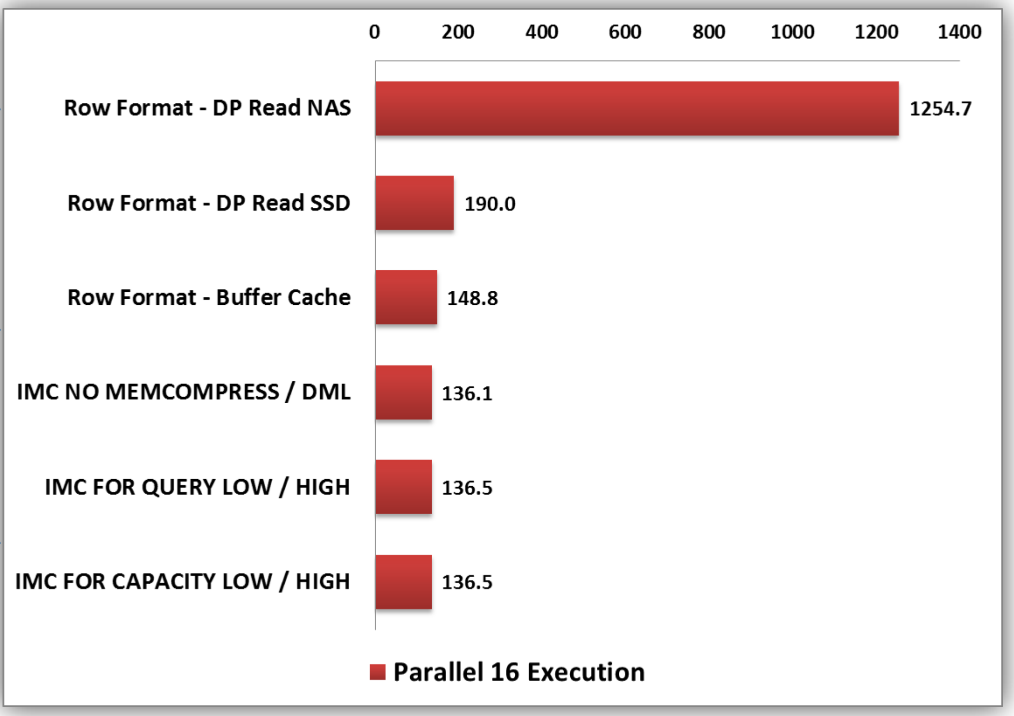
Row Format - Buffer Cache	148.8

IMC vs SSD Direct Path

1.4x faster

IMC vs NAS Direct Path

9.2x faster



IMC Tests – Benchmark Query 4

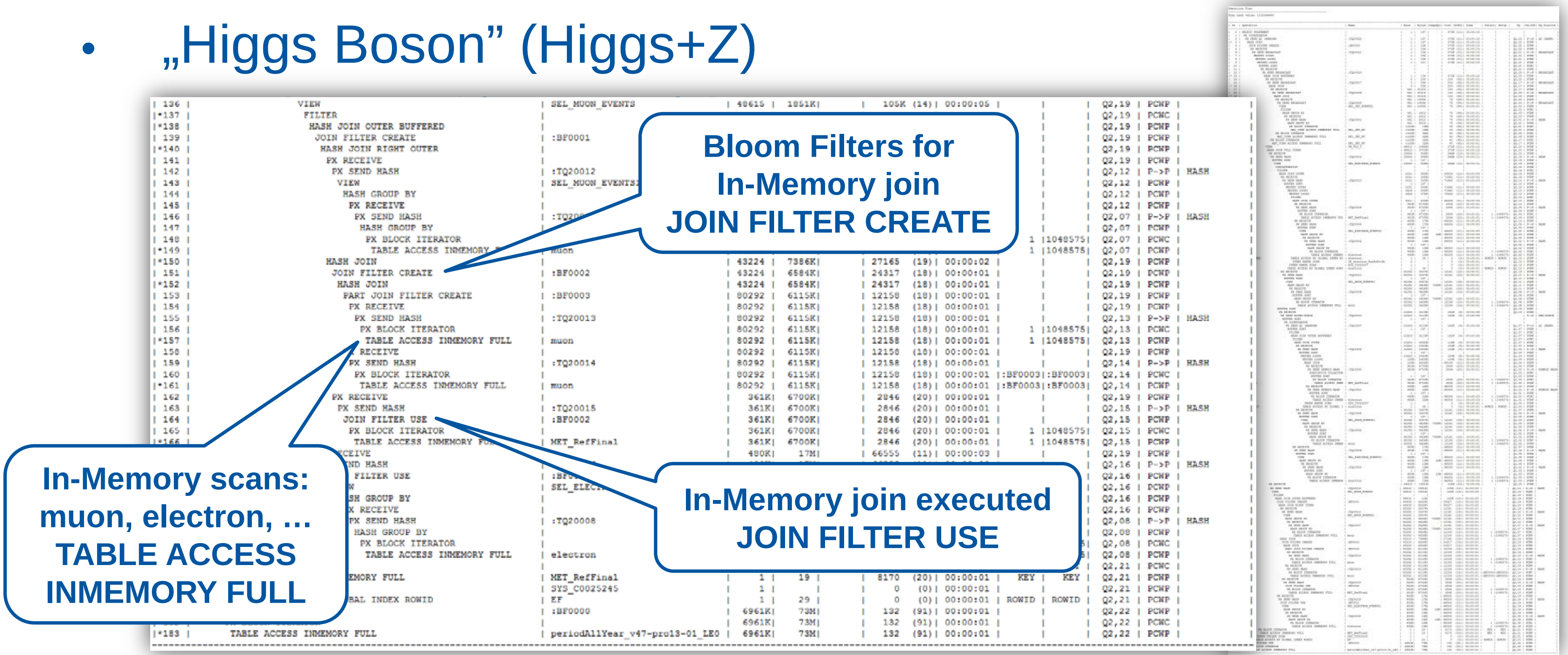
- „Higgs Boson” (Higgs+Z)
  - Join of 6 tables + 2 self joins
  - Over 200GB of data to be scanned
  - Very complex query with analytic calculations
  - Multiple layers of views for simplicity

```
create or replace view sel_EF_events as (select /*+ FULL("EF") */ "RunNumber","EventNumber"
  from DATA12_8TEV_NAS."EF" where ("e24vhi_medium1"=1 or "e60_medium1"=1 or "2e12Tvh_loose1"=1 or "mu24i_tight"=1 or "mu36_tight"=1 or "2mu13"=1 ));
...
create or replace view sel_muon_events as (select "RunNumber","EventNumber",ANALYSISTOOLS.PHYSANALYSIS.INV_MASS_LEPTONS(
       muon0."E",muon1."E",muon0."px",muon1."px",muon0."py",muon1."py",muon0."pz",muon1."pz")/1000. as "DiMuonMass"
  from sel_mu_el_events INNER JOIN sel_muon muon0 USING ("RunNumber","EventNumber") INNER JOIN sel_muon muon1
USING ("RunNumber","EventNumber") where muon0."muon_i"<muon1."muon_i" );

select "RunNumber","EventNumber","EventNumber","RunNumber","DiMuonMass","DiElectronMass","DiJetMass"
 from sel_muon_events FULL OUTER JOIN sel_electron_events USING ("RunNumber","EventNumber")
       INNER JOIN sel_jet_events USING ("RunNumber","EventNumber") INNER JOIN sel_MET_events USING ("RunNumber","EventNumber")
       INNER JOIN sel_EF_events USING("RunNumber","EventNumber")
       INNER JOIN DATA12_8TEV_NAS."periodAllYear_v47-pro13-01" USING("RunNumber","EventNumber");
```

IMC Tests – Benchmark Query 4

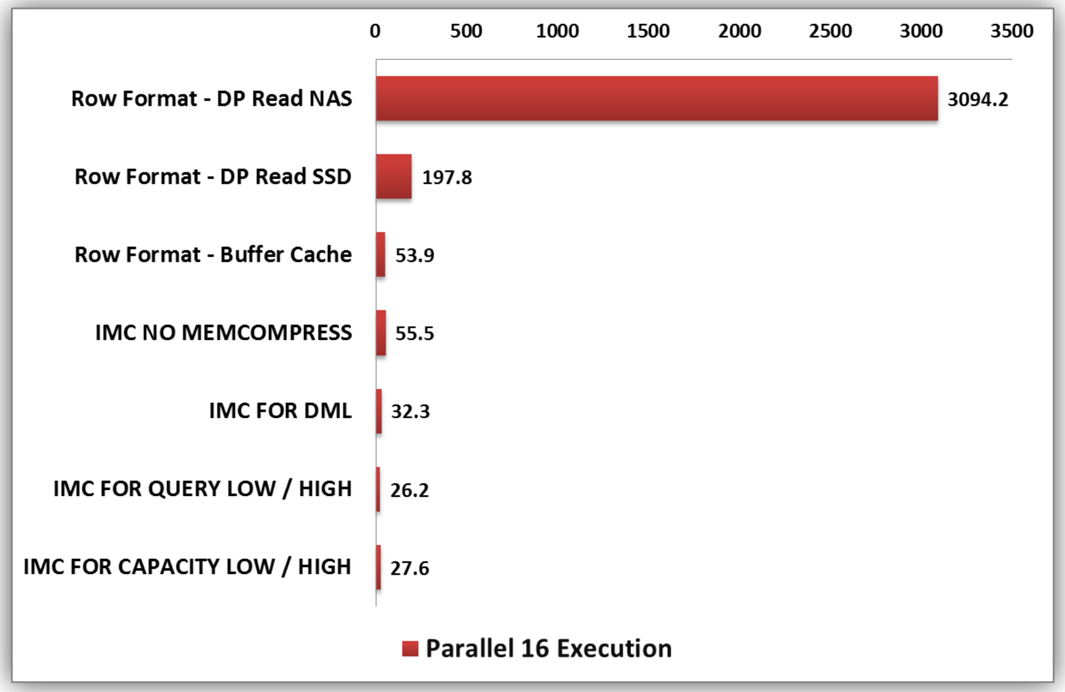
- „Higgs Boson” (Higgs+Z)



IMC Tests – Benchmark Results Q4

- „Higgs Boson"

Configuration	Parallel 16 Execution
Row Format - DP Read NAS	3094.2
Row Format - DP Read SSD	197.8
Row Format - Buffer Cache	53.9
IMC NO MEMCOMPRESS	55.5
IMC FOR DML	32.3
IMC FOR QUERY LOW / HIGH	26.2
IMC FOR CAPACITY LOW / HIGH	27.6



IMC Tests – Benchmark Results Q4

- „Higgs Boson"

IMC vs Buffer Cache

2.1x faster!

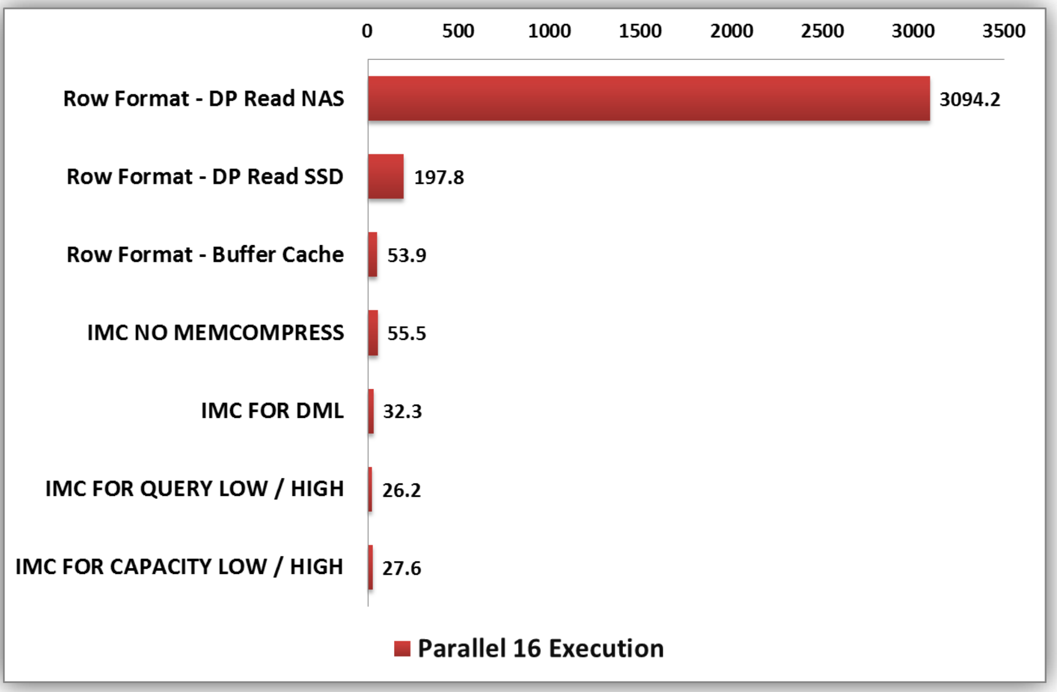
Row Format - Buffer Cache	53.9

IMC vs SSD Direct Path

7.6x faster!

IMC vs NAS Direct Path

118x faster!



IMC Tests – OLTP Benchmark

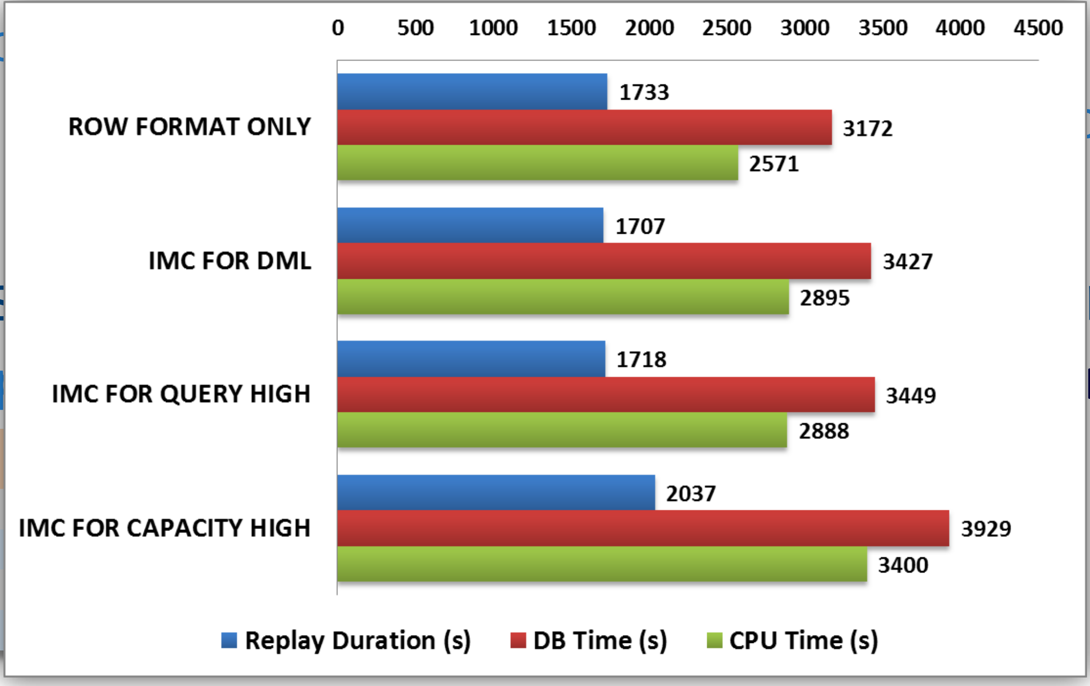
- OLTP schema from one of the physics databases (CMS)
  - High concurrency workload with ~300 simultaneous sessions
  - Mixed DML and SELECT queries – mostly with index access path
- Real Application Testing – 1h of peak activity captured
- Replay comparison: row format (BC) vs In-Memory Store
  - IMC with 3 compression levels: **DML** / **QUERY HIGH** / **CAPACITY HIGH**

Configuration	Replay Duration (s)	DB Time (s)	CPU Time (s)	Physical Reads (GB)
ROW FORMAT	1733	3172	2571	97.9
IMC FOR DML	1707	3427	2895	7.9
IMC FOR QUERY HIGH	1718	3449	2888	7.5
IMC FOR CAPACITY HIGH	2037	3929	3400	7.1

IMC Tests – OLTP Benchmark

- OLTP schema from one of the physics databases (CMS)
  - High concurrenc
  - Mixed DML and … ath
- Real Application
- Replay comparis … re
  - IMC with 3 com … H

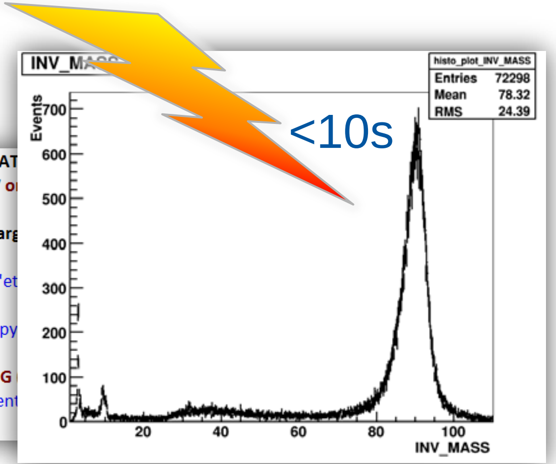
Configuration
ROW FORMAT
IMC FOR DML
IMC FOR QUERY HIGH
IMC FOR CAPACITY HIGH



Physics Analysis NOW – with IMC

- In-memory Column Store provides significant performance boost for physics queries
- Very positive benchmark results

```
with "sel_electron" as (select /*+ PARALLEL(16) */ "electron_i","RunNumber","EventNumber","E","px","py","pz","charge","pt","phi","eta" from DAT
      where ("author"=1 or "author"=3) and "cl_E"/cosh("tracketa") >10000. and (abs("cl_eta")!=0 and abs("cl_eta")<2.47) and (abs("cl_eta")<1.37 or
         and BITAND("OQ",1446)=0 and abs("trackz0pvunbiased") < 2. and "tightPP"=1)
select "RunNumber","EventNumber",el_sel_n, electron0."electron_i" as mu_id0, electron1."electron_i" as mu_id1, electron0."charge" as mu_charg
    electron0."pt"/1000. as mu_pt0, electron1."pt"/1000. as mu_pt1, electron0."eta" as el_eta0, electron1."eta" as el_eta1,
   (case when abs(electron0."phi"-electron1."phi")<acos(-1.) then sqrt(POWER(abs(electron0."phi"-electron1."phi"),2)+POWER(abs(electron0."et
    else sqrt(POWER( 2.*acos(-1.) - abs(electron0."phi"-electron1."phi"),2)+POWER(abs(electron0."eta"-electron1."eta"),2)) end) as DELTAR,
    ANALYSISTOOLS.PHYSANALYSIS.INV_MASS_LEPTONS(electron0."E",electron1."E",electron0."px",electron1."px",electron0."py",electron1."py
from DATA12_8TEV."periodAllYear_v47-pro13-01"
      INNER JOIN (select "RunNumber","EventNumber",COUNT(*) as el_sel_n from "sel_electron" group by ("RunNumber","EventNumber")) USING
      INNER JOIN "sel_electron" electron0 USING ("RunNumber","EventNumber") INNER JOIN "sel_electron" electron1 USING ("RunNumber","Event
where electron0."electron_i"<electron1."electron_i" and electron0."charge" != electron1."charge" and el_sel_n=2;
```



Conclusions

- In-Memory Colum Store is both powerful and easy to use
- Significant performance improvements of analytic-type queries on the same hardware
    - On average 10x for fast SSD storage
    - On average 100x for slower spindles
- No negative impact on OLTP, high concurrency workloads
- More data can go to memory due to the columnar format with compression
    - Very low performance impact of higher compression levels

Conclusions

Do we plan to use In-Memory Column Store at CERN?

Will definitely consider it in future projects!

Many good use-cases identified

Q & A

Maaike: Maaike.Limper@cern.ch

Manuel: Manuel.Martin.Marquez@cern.ch

Emil: Emil.Pilecki@cern.ch

www.cern.ch