Identifying SQL issues using AWR, cardinality feedback and analytic functions

Image courtesy of Forschungszentrum Jülich / Seitenplan, with material from NASA, ESA and AURA/Caltech

Eric Grancher
eric.grancher@cern.ch
CERN IT department

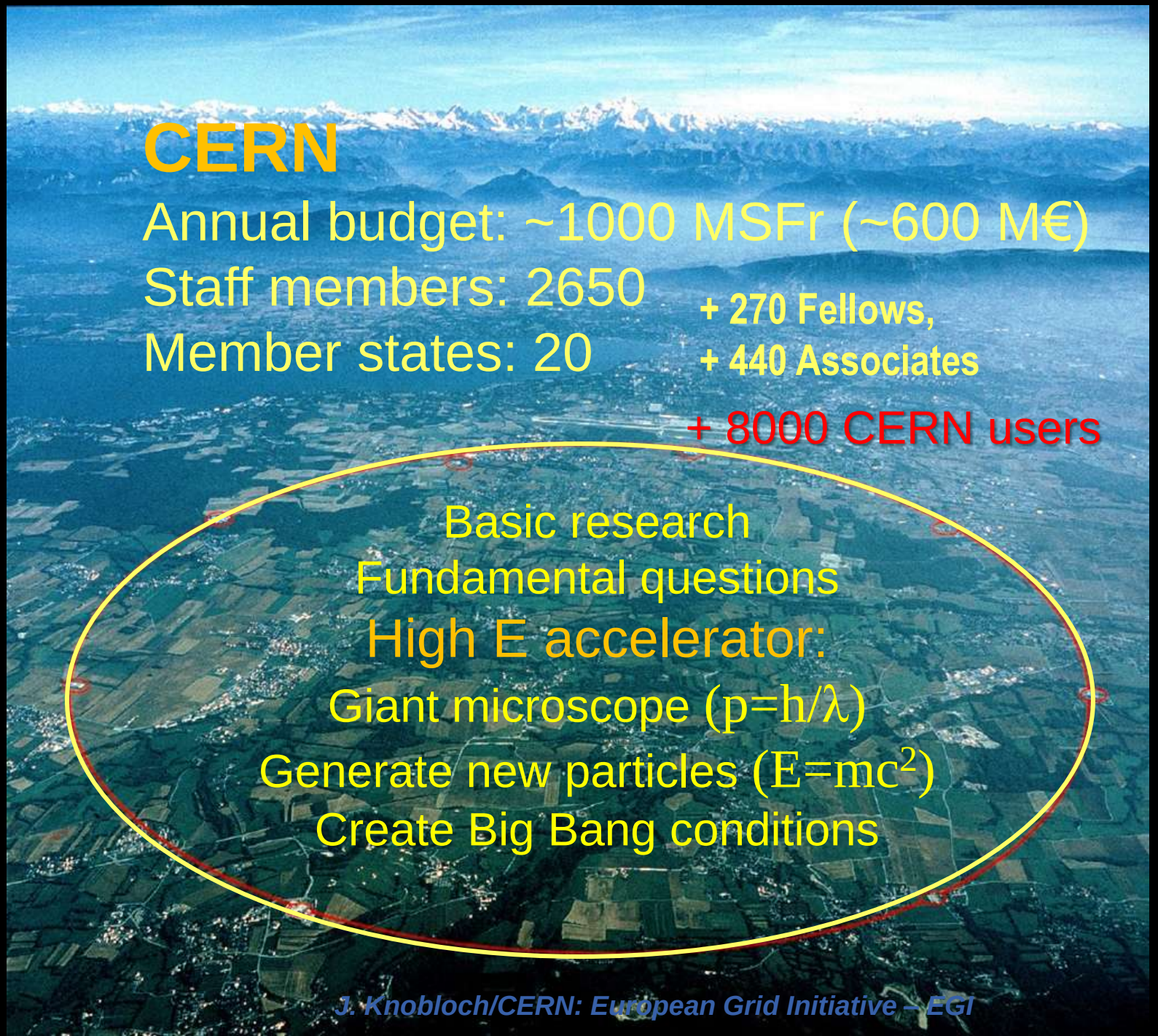
https://edms.cern.ch/document/1053734/1

Outline

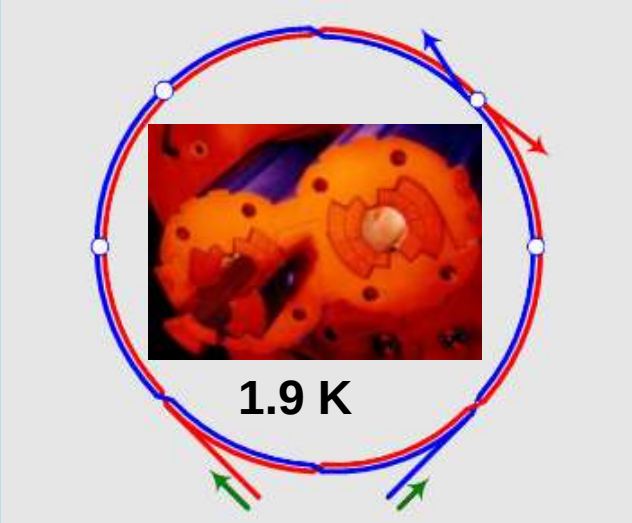
- Few words about CERN and computing challenge

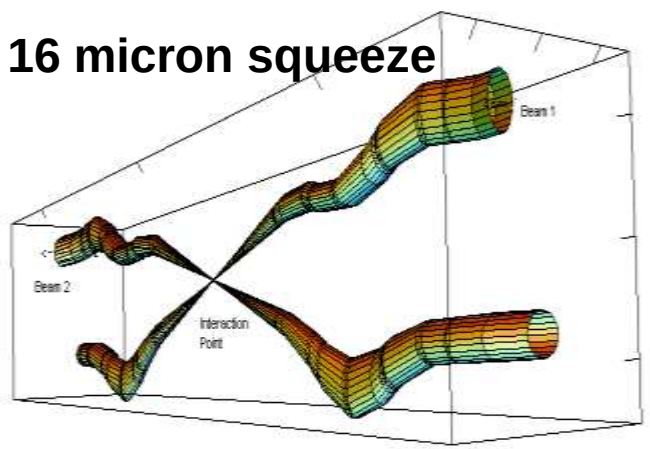
- Analytics
- Automatic Workload Repository
- Cardinality feedback
- Active Session History

- Conclusions
- References


CERN
Annual budget: ~1000 MSFr (~600 M€)
Staff members: 2650
Member states: 20
+ 270 Fellows,
+ 440 Associates
+ 8000 CERN users
Basic research
Fundamental questions
High E accelerator:
Giant microscope (p=h/λ)
Generate new particles (E=mc²)
Create Big Bang conditions

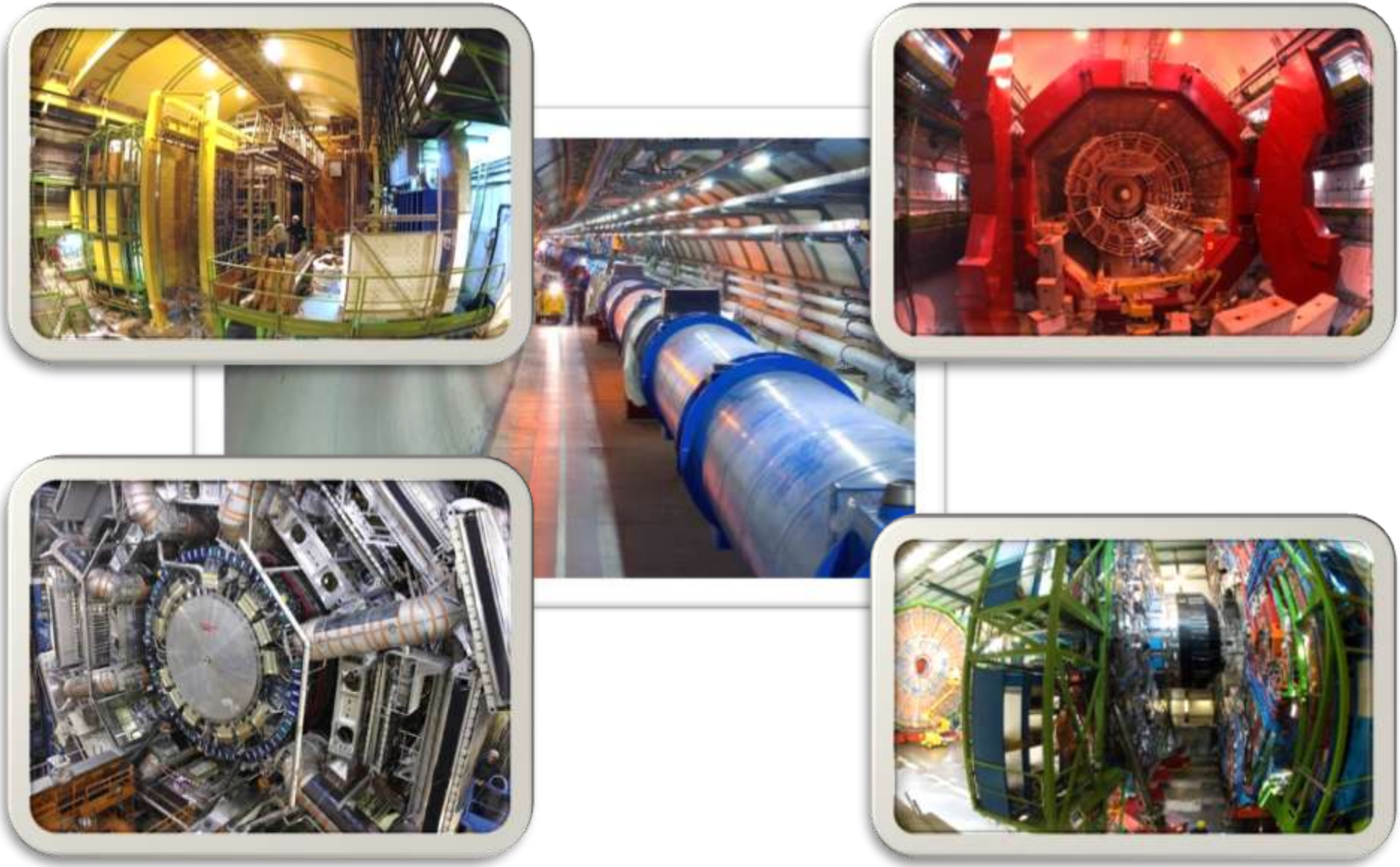
Large Hadron Collider - LHC





- 27 km circumference
- Cost ~ 3000 M€ (+ detectors)
- Proton-proton (or lead ion) collisions at 7+7 TeV
- Bunches of 10^{11} protons cross every 25 nsec
- 600 million collisions/sec
- Physics questions
  - Origin of mass (Higgs?)
  - Dark matter?
  - Symmetry matter-antimatter
  - Forces – supersymmetry
  - Early universe – quark-gluon plasma
  - …

LHC is ready again …



LHC Computing Challenge

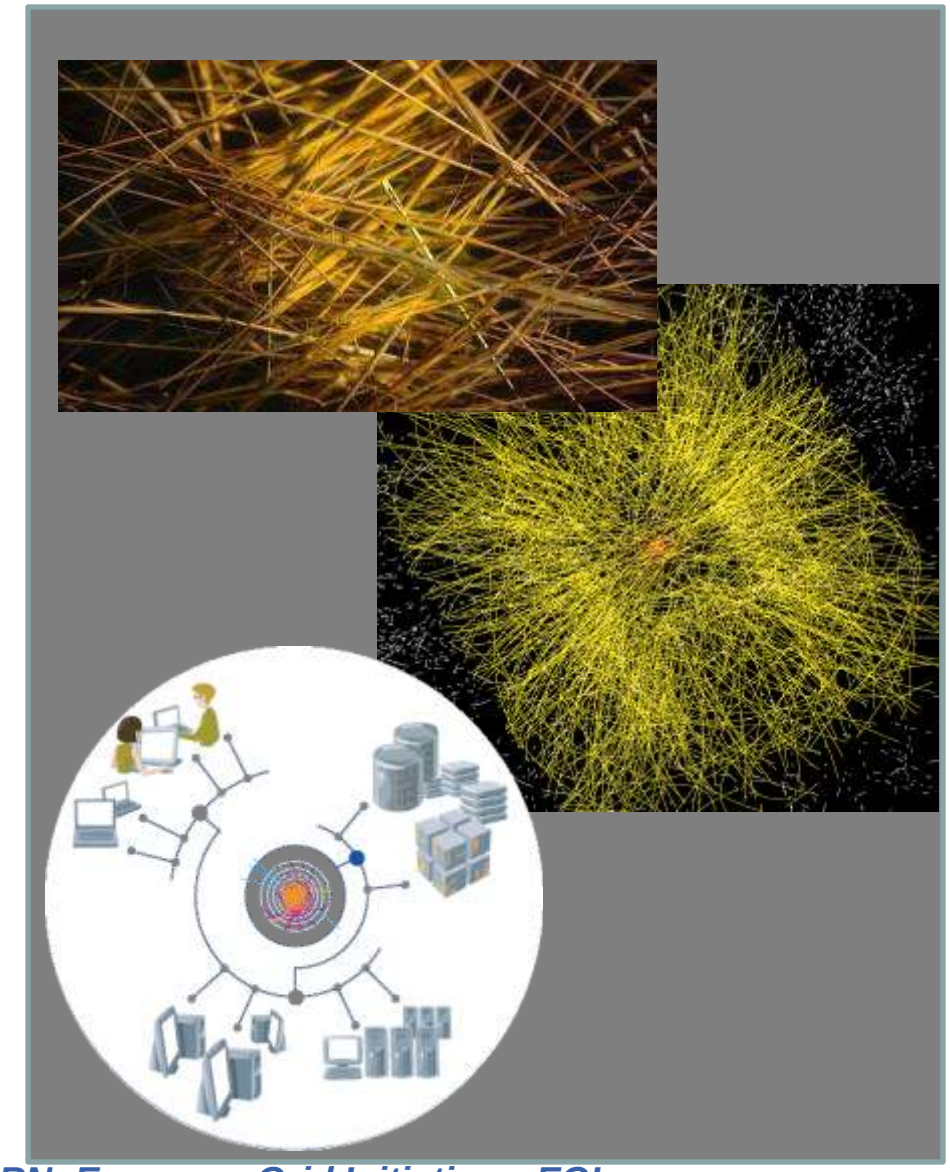
CERN IT Department

- Signal/Noise 10^{-9}
- Data volume
  - High rate * large number of channels * 4 experiments

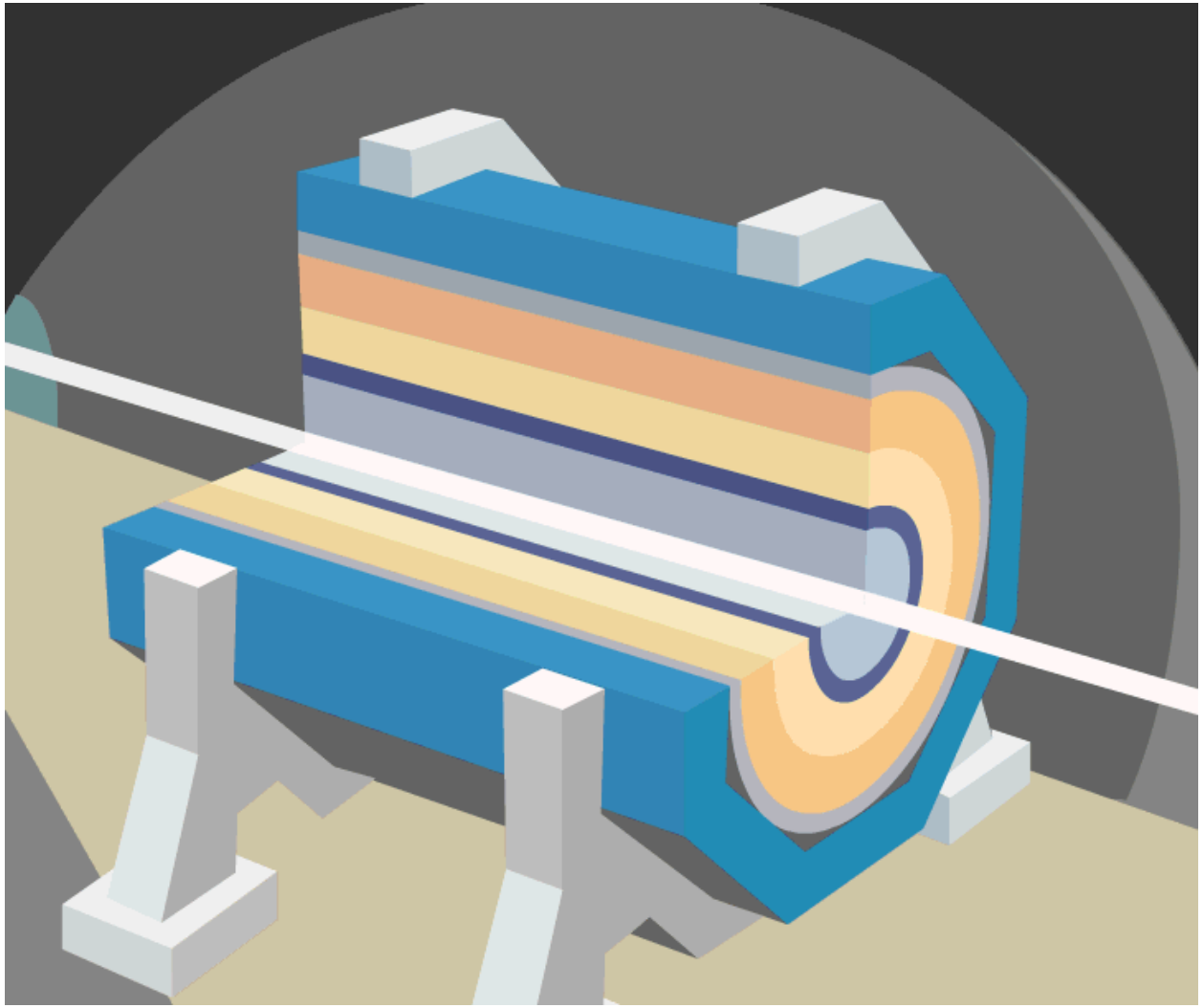
  ➔ 15 PetaBytes of new data each year
- Compute power
  - Event complexity * Nb. events * thousands users

  ➔ 100 k CPUs (cores)
- Worldwide analysis & funding
  - Computing funding locally in major regions & countries
  - Efficient analysis everywhere

  ➔ GRID technology



Events at LHC



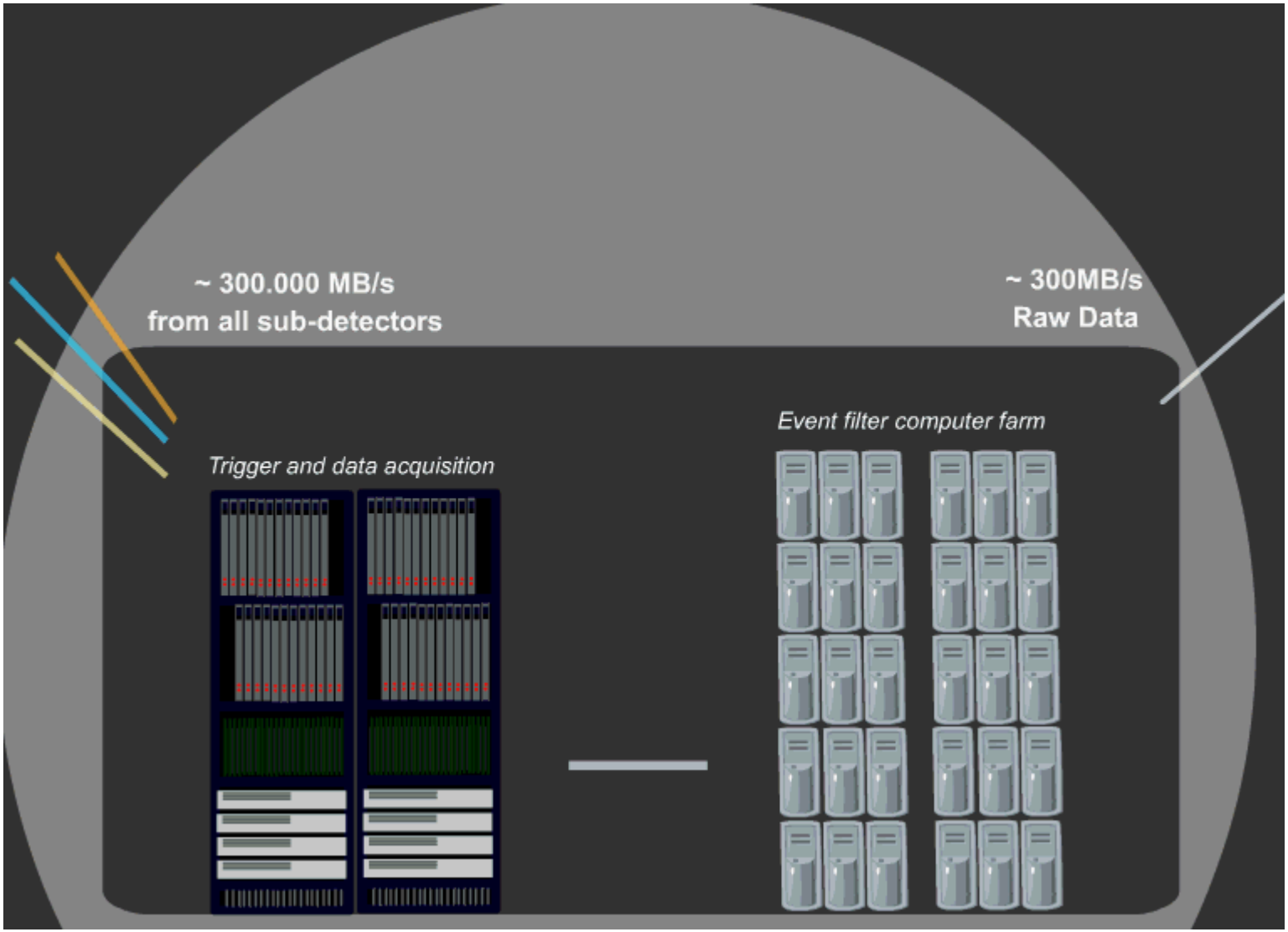
Luminosity :
$10^{34}cm^{-2}\ s^{-1}$

40 MHz – every 25 ns

20 events overlaying

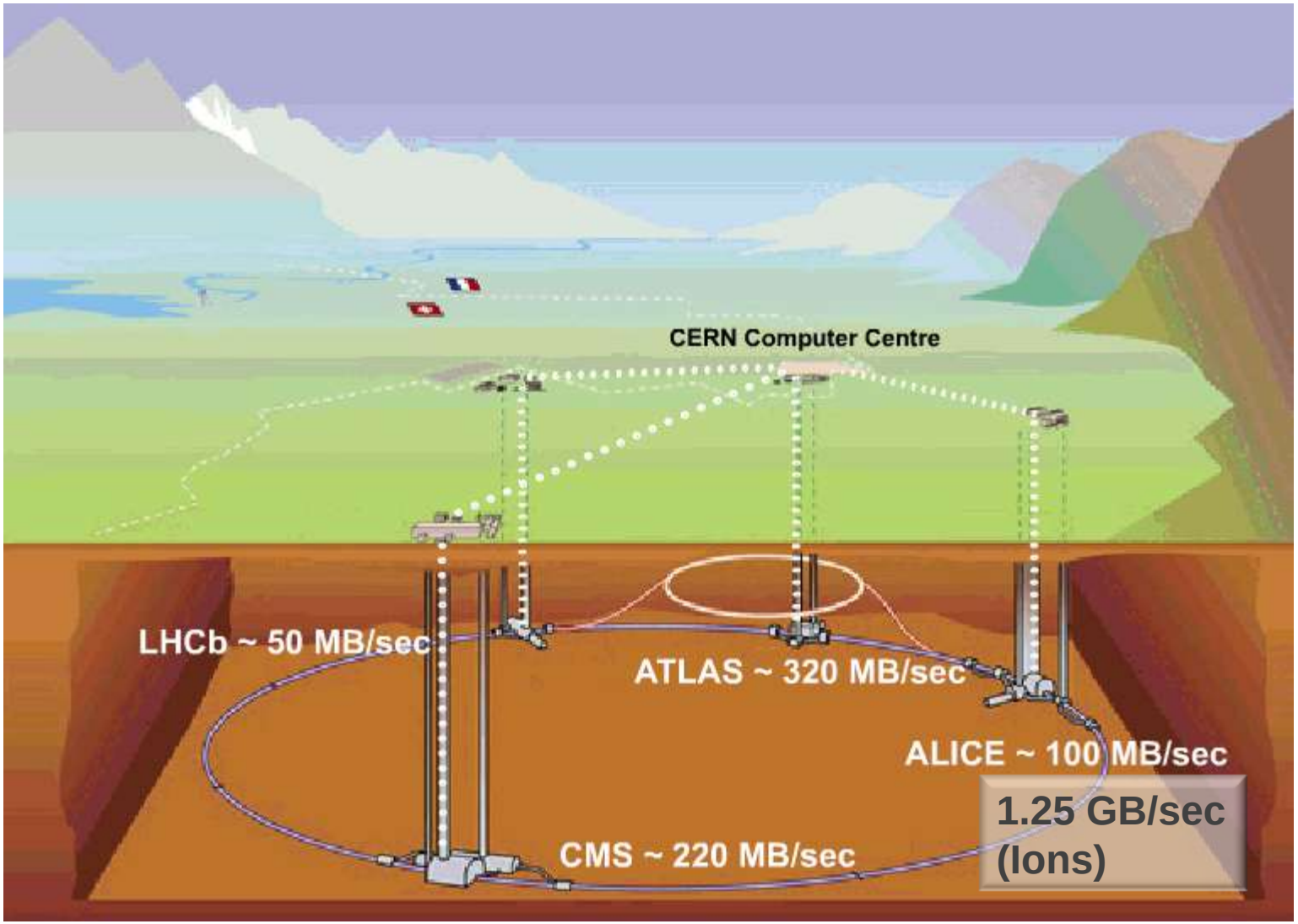
Trigger & Data Acquisition

CERN IT Department



J. Knobloch/CERN: European Grid Initiative – EGI

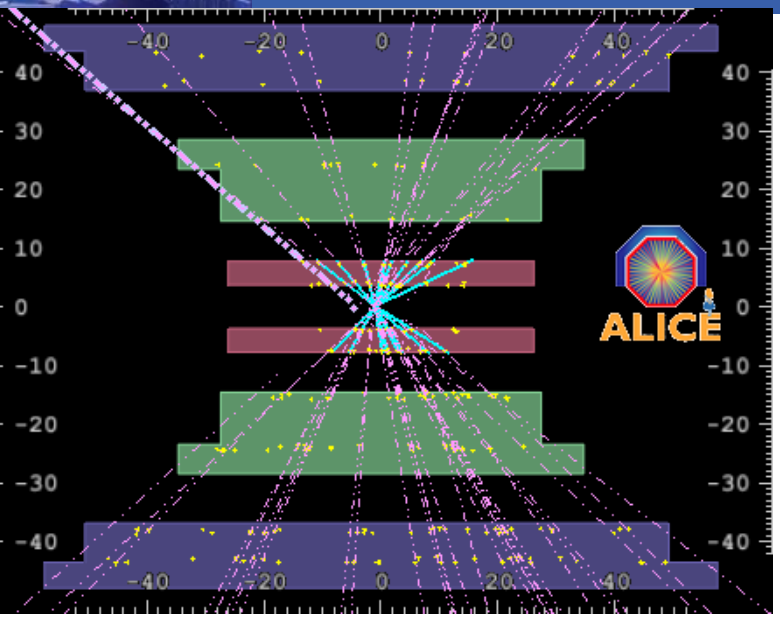
Data Recording

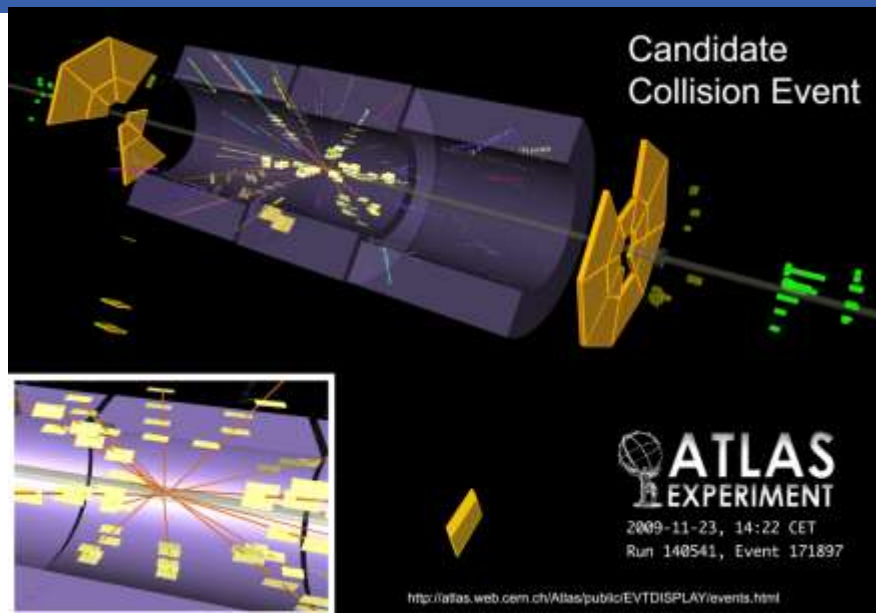


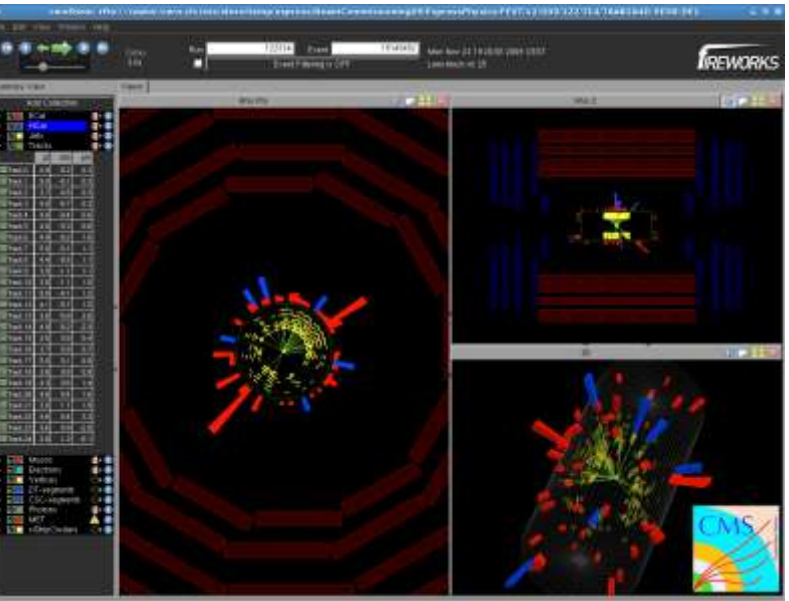
J. Knobloch/CERN: European Grid Initiative – EGI

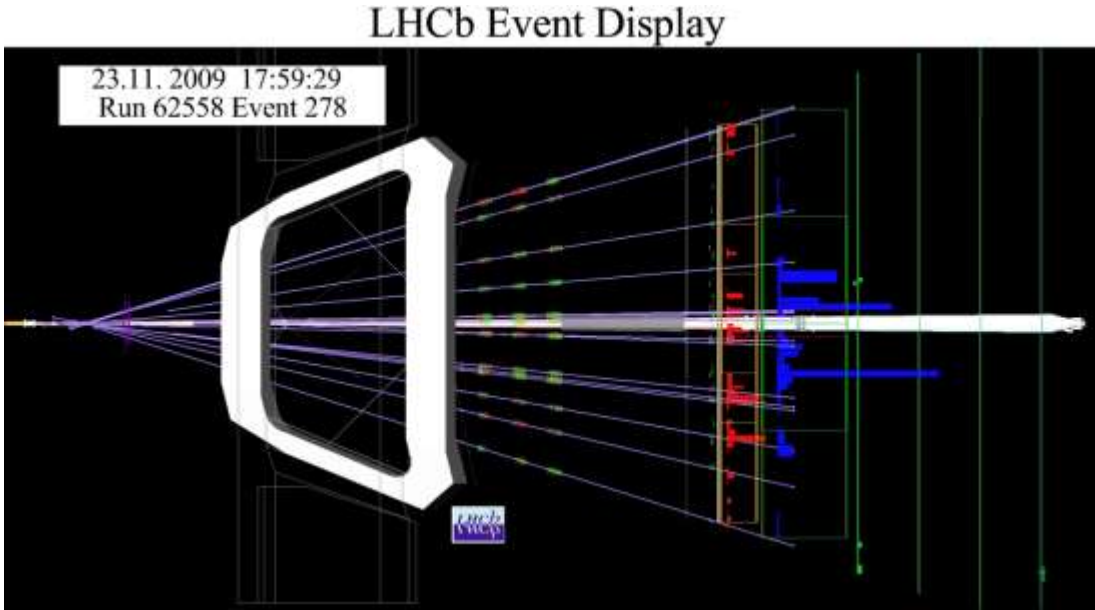
LHC starting... 23 Nov first collisions

CERN IT Department









LHC starting... new world record

Geneva, 30 November 2009. CERN's Large Hadron Collider has today become the world's highest energy particle accelerator, having accelerated its twin beams of protons to an energy of **1.18TeV** in the early hours of the morning. […] It marks another important milestone on the road to first physics at the LHC in 2010. […]

Identifying problems and SQL

- The Oracle database system collects a lot of information (statistics, wait event interface...)
- Some tools provide a visual layer on top (Enterprise Manager for example)
- A GUI, as good as it can be, does not replace raw data and numbers (more precise, own graphing, comparison, ...)
- Ideas presented here **complement** Oracle scripts and GUIs

Identifying problems and SQL

- Global analysis can often not identify local issues, local and precise understanding needed as well
- Analytic SQL functions are key for efficient processing, some examples...
- Graphing tools are not discussed

Introduction to analytics

- Function(arg1,..., argn) OVER ([PARTITION BY <...>] [ORDER BY <....>] [<window_clause>])
- Different than aggregating

```
SQL> select snap_id,END_INTERVAL_TIME from dba_hist_snapshot where
END_INTERVAL_TIME>sysdate-1/24 order by END_INTERVAL_TIME;

   SNAP_ID END_INTERVAL_TIME
---------- ---------------------------------
       454 23-NOV-09 10.40.01.054 PM
       455 23-NOV-09 10.50.01.506 PM
       456 23-NOV-09 11.00.02.090 PM
       457 23-NOV-09 11.10.02.695 PM
       458 23-NOV-09 11.20.03.164 PM
       459 23-NOV-09 11.30.03.641 PM
```

454 23-NOV-09 10.40.01.054 PM

We need the previous entry

- Which was the previous one?

Analytics

- We need to perform differences of timing information / wait statistics between snapshots

```
select snap_id,END_INTERVAL_TIME,
lead(snap_id,1) over (order by snap_id desc) previous_snap_id
from dba_hist_snapshot where END_INTERVAL_TIME>sysdate-1/24 order
by END_INTERVAL_TIME;

   SNAP_ID END_INTERVAL_TIME                 PREVIOUS_SNAP_ID
---------- ------------------------------ ----------------
       454 23-NOV-09 10.40.01.054 PM
       455 23-NOV-09 10.50.01.506 PM                   454
       456 23-NOV-09 11.00.02.090 PM                   455
       457 23-NOV-09 11.10.02.695 PM                   456
       458 23-NOV-09 11.20.03.164 PM                   457
       459 23-NOV-09 11.30.03.641 PM                   458
```

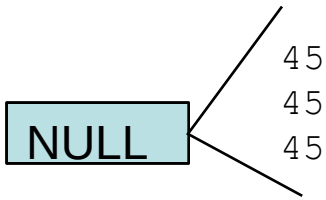
NULL

```
select snap_id,END_INTERVAL_TIME, lead(snap_id,1) over (order by snap_id desc)
previous_snap_id, STARTUP_TIME from dba_hist_snapshot where
END_INTERVAL_TIME>sysdate-1/24 order by END_INTERVAL_TIME;

   SNAP_ID END_INTERVAL_TIME                 PREVIOUS_SNAP_ID STARTUP_TIME
---------- --------------------------------- ---------------- ---------------------------
       457 23-NOV-09 11.10.02.695 PM                         12-NOV-09 02.47.55.000 PM
       458 23-NOV-09 11.20.03.164 PM                     457 12-NOV-09 02.47.55.000 PM
       459 23-NOV-09 11.30.03.641 PM                     458 12-NOV-09 02.47.55.000 PM
       460 23-NOV-09 11.40.04.106 PM                     459 12-NOV-09 02.47.55.000 PM
       461 23-NOV-09 11.53.17.377 PM                     460 23-NOV-09 11.42.11.000 PM
       462 24-NOV-09 12.00.17.646 AM                     461 23-NOV-09 11.42.11.000 PM

select snap_id,END_INTERVAL_TIME,
lead(snap_id,1) over (partition by STARTUP_TIME order by snap_id desc)
previous_snap_id, STARTUP_TIME
from dba_hist_snapshot where END_INTERVAL_TIME>sysdate-1/24 order by
END_INTERVAL_TIME;

   SNAP_ID END_INTERVAL_TIME                 PREVIOUS_SNAP_ID STARTUP_TIME
---------- --------------------------------- ---------------- ---------------------------
       457 23-NOV-09 11.10.02.695 PM                         12-NOV-09 02.47.55.000 PM
       458 23-NOV-09 11.20.03.164 PM                     457 12-NOV-09 02.47.55.000 PM
       459 23-NOV-09 11.30.03.641 PM                     458 12-NOV-09 02.47.55.000 PM
       460 23-NOV-09 11.40.04.106 PM                     459 12-NOV-09 02.47.55.000 PM
       461 23-NOV-09 11.53.17.377 PM                         23-NOV-09 11.42.11.000 PM
       462 24-NOV-09 12.00.17.646 AM                     461 23-NOV-09 11.42.11.000 PM
```



Automatic Workload Repository

- On a regular basis, the instances collect statistics and stores in SYSAUX tablespace (SYS.WRM$_SNAPSHOT, SYS.WRH$_SQLSTAT, SYS.WRH$_FILESTATXS, SYS.WRH$_DATAFILE...):
  snapshots of many activity related views
- This is performed every x minutes and the snapshots are kept y minutes (unless space pressure)

- @?/rdbms/admin/awrrpt to create reports
- Enterprise Manager visualisation

AWR and IO

CERN IT Department

- First example: IO response time
- v$filestat

FILE#	NUMBER	Number of the file
PHYRDS	NUMBER	Number of physical reads done
PHYWRTS	NUMBER	Number of times DBWR is required to write
PHYBLKRD	NUMBER	Number of physical blocks read
PHYBLKWRT	NUMBER	Number of blocks written to disk, which may be the same as PHYWRTS if a
SINGLEBLKRDS	NUMBER	Number of single block reads

- DBA_HIST_FILESTATXS is made of snapshots of V$FILESTAT

SNAP_ID	NUMBER	Unique snapshot ID
DBID	NUMBER	Database ID for the snapshot
INSTANCE_NUMBER	NUMBER	Instance number for the snapshot
FILE#	NUMBER	File identification number
CREATION_CHANGE#	NUMBER	Change number at which the datafile was cre
FILENAME	VARCHAR2(513)	Name of the datafile
TS#	NUMBER	Tablespace number
TSNAME	VARCHAR2(30)	Name of the tablespace
BLOCK_SIZE	NUMBER	Block size of the datafile
PHYRDS	NUMBER	Number of physical reads done

AWR and IO

CERN IT Department

```
select instance_number, snap_id,phyrds,phywrts,singleblkrds,
readtim  from DBA_HIST_FILESTATXS where  snap_id between 43381
and 43384  and file#=1286 order by instance_number, snap_id;
```

INSTANCE_NUMBER	SNAP_ID	PHYRDS	PHYWRTS	SINGLEBLKRDS	READTIM	
1	43381	416	430	11	1171	
1	43382	417	431	11	1173	
1	43383	425	432	18	1185	
1	43384	427	434	18	1192	
2	43381	68092	8102972	67758	267253	←
2	43382	86149	8363390	85814	321604	←
2	43383	94910	8613536	94574	346178	←
2	43384	98032	8826146	97695	355154	←

AWR and IO

- Perform differences (deltas)

```
select snap_id,instance_number,file#,ts#,
   phyrds - lead(phyrds, 1) OVER (PARTITION BY
instance_number,file# ORDER BY snap_id DESC) phyrds,
   READTIM - lead(READTIM, 1) OVER (PARTITION BY
instance_number,file# ORDER BY snap_id DESC) READTIM,
  PHYWRTS - lead(PHYWRTS, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) PHYWRTS,
   WRITETIM - lead(WRITETIM, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) WRITETIM,
   SINGLEBLKRDS - lead(SINGLEBLKRDS, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC)
SINGLEBLKRDS,
   SINGLEBLKRDTIM - lead(SINGLEBLKRDTIM, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id
DESC) SINGLEBLKRDTIM,
  lead(snap_id, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) previous_snap_id
from DBA_HIST_FILESTATXS where snap_id between 43380 and 43384 and file# in (4,7,1286)
order by snap_id,file#;
```

SNAP_ID	INSTANCE_NUMBER	FILE#	TS#	PHYRDS	READTIM	PHYWRTS	WRITETIM	SINGLEBLKRDS	SINGLEBLKRDTIM	PREVIOUS_SNAP_ID
43380	1	4	4							
43380	2	4	4							
43380	1	7	7							
43380	2	7	7							
43380	1	1286	1281							
43380	2	1286	1281							
43381	1	4	4	1	1	1	0	0	0	43380
43381	2	4	4	1	0	1	0	0	0	43380
43381	2	7	7	1	0	10492	2802	0	0	43380
43381	1	7	7	1	0	1	0	0	0	43380
43381	2	1286	1281	4403	13267	245566	20438	4402	13266	43380

AWR and IO

CERN IT Department

```
select snap_id,file#,ts#,
sum(phyrds) phyrds,sum(READTIM) READTIM, CASE WHEN sum(phyrds) > 0 THEN round(10*SUM(READTIM) / SUM(phyrds), 2) END
avg_phyrdtime_ms,
sum(PHYWRTS) PHYWRTS,sum(WRITETIM) WRITETIM, CASE WHEN SUM(PHYWRTS) > 0 THEN round(10*SUM(WRITETIM) / SUM(PHYWRTS),
2) END avg_phywrttime_ms,
sum(SINGLEBLKRDS) SINGLEBLKRDS,sum(SINGLEBLKRDTIM) SINGLEBLKRDTIM, CASE WHEN SUM(SINGLEBLKRDS) > 0 THEN
round(10*SUM(SINGLEBLKRDTIM) / SUM(SINGLEBLKRDS), 2) END avg_singleblkrdtime_ms
from
            (select snap_id,instance_number,file#,ts#,
            phyrds - lead(phyrds, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) phyrds,
            READTIM - lead(READTIM, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) READTIM,
            PHYWRTS - lead(PHYWRTS, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) PHYWRTS,
            WRITETIM - lead(WRITETIM, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) WRITETIM,
            SINGLEBLKRDS - lead(SINGLEBLKRDS, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC)
SINGLEBLKRDS,
            SINGLEBLKRDTIM - lead(SINGLEBLKRDTIM, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC)
SINGLEBLKRDTIM,
            lead(snap_id, 1) OVER (PARTITION BY instance_number,file# ORDER BY snap_id DESC) previous_snap_id
            from DBA_HIST_FILESTATXS  where
            snap_id between 43380 and 43384 and file#=1286
            )
where previous_snap_id is not null
group by snap_id,file#,ts#
order by snap_id desc;
```

SNAP_ID	FILE#	TS#	PHYRDS	READTIM	AVG_PHYRDTIME_MS	PHYWRTS	WRITETIM	AVG_PHYWRTTIME_MS	SINGLEBLKRDS	SINGLEBLKRDTIM	AVG_SINGLEBLKRDTIME_MS
43381	1286	1281	4404	13270	30.13	245567	20441	.83	4402	13266	30.14
43382	1286	1281	18058	54353	30.1	260419	21300	.82	18056	54351	30.1
43383	1286	1281	8769	24586	28.04	250147	20008	.8	8767	24583	28.04
43384	1286	1281	3124	8983	28.75	212612	36098	1.7	3121	8976	28.76

AWR and IO

CERN IT Department

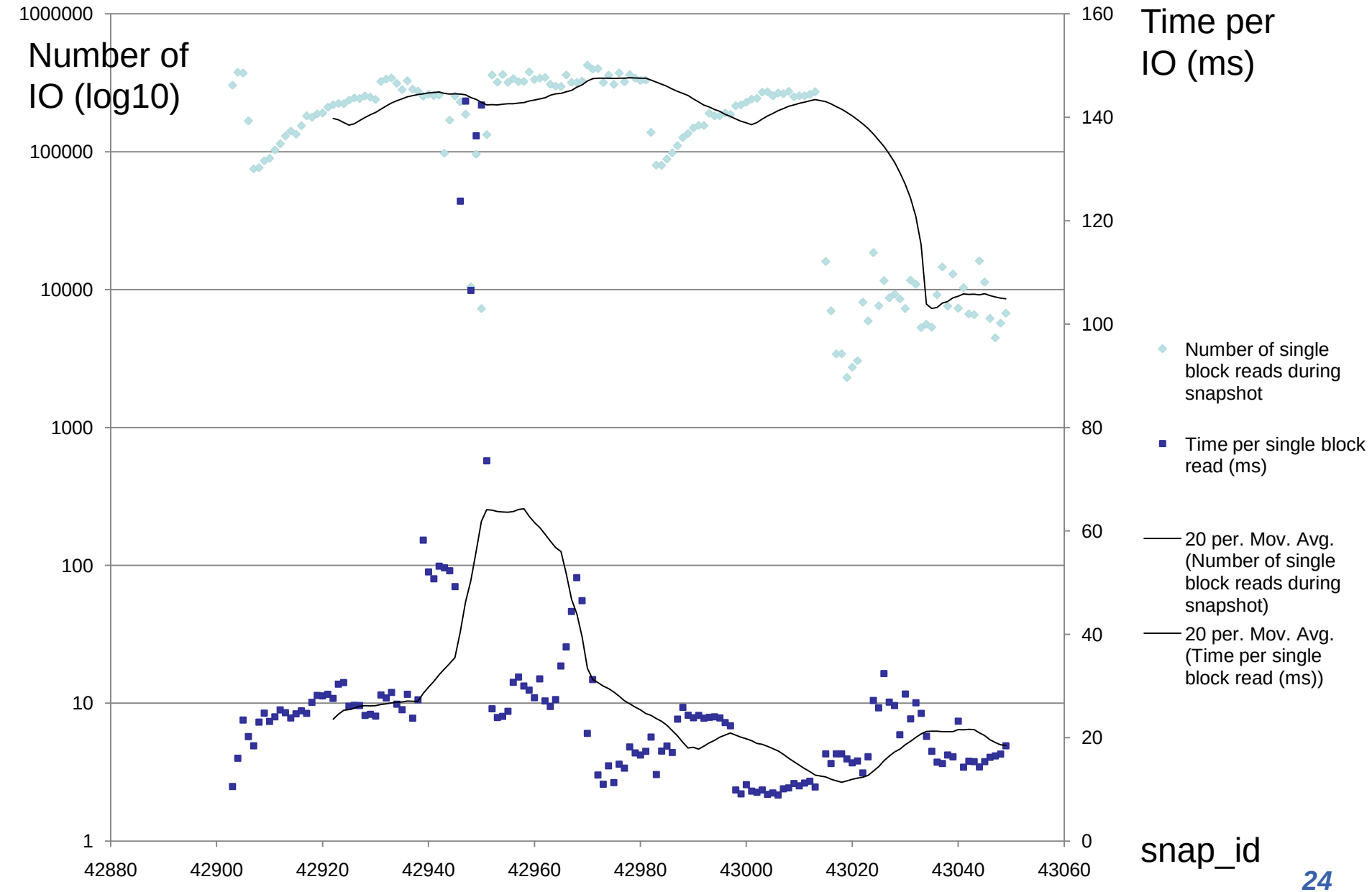
```
select * from (
select snap_id,file#,ts#,
               sum(phyrds) phyrds,sum(READTIM) READTIM, CASE WHEN sum(phyrds) > 0 THEN round(10*SUM(READTIM) / SUM(phyrds), 2) END avg_phyrdtime_ms,
               sum(PHYWRTS) PHYWRTS,sum(WRITETIM) WRITETIM, CASE WHEN SUM(PHYWRTS) > 0 THEN round(10*SUM(WRITETIM) / SUM(PHYWRTS), 2) END avg_phywrttime_ms,
               sum(SINGLEBLKRDS) SINGLEBLKRDS,sum(SINGLEBLKRDTIM) SINGLEBLKRDTIM, CASE WHEN SUM(SINGLEBLKRDS) > 0 THEN round(10*SUM(SINGLEBLKRDTIM) /
SUM(SINGLEBLKRDS), 2) END avg_singleblkrdtime_ms,
          round(100*sum(PHYRDS+PHYWRTS+SINGLEBLKRDS)/sum(sum(PHYRDS)+sum(PHYWRTS)+sum
(SINGLEBLKRDS)) over (PARTITION BY snap_id),2) ratio,
          rank() over (PARTITION BY snap_id order by sum(PHYRDS+PHYWRTS+SINGLEBLKRDS)
desc) file_rank from
               (select snapshot.snap_id,snapshot.instance_number,file#,ts#,
               phyrds - lead(phyrds, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) phyrds,
               READTIM - lead(READTIM, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) READTIM,
               PHYWRTS - lead(PHYWRTS, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) PHYWRTS,
               WRITETIM - lead(WRITETIM, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) WRITETIM,
SINGLEBLKRDS - lead(SINGLEBLKRDS, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) SINGLEBLKRDS,
SINGLEBLKRDTIM - lead(SINGLEBLKRDTIM, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) SINGLEBLKRDTIM,
               lead(snapshot.snap_id, 1) OVER (PARTITION BY snapshot.instance_number,snapshot.startup_time,file# ORDER BY snapshot.snap_id DESC) previous_snap_id
               from dba_hist_filestatxs  filestatxs ,dba_hist_snapshot snapshot where
               filestatxs.snap_id between 43380 and 43384
               and snapshot.snap_id=filestatxs.snap_id
               and snapshot.instance_number=filestatxs.instance_number
               )
where previous_snap_id is not null group by snap_id,file#,ts#) where file_rank <3 order by snap_id ,file_rank;
```

SNAP_ID	FILE#	TS#	PHYRDS	READTIM	AVG_PHYRDTIME_MS	PHYWRTS	WRITETIM	AVG_PHYWRTTIME_MS	SINGLEBLKRDS	SINGLEBLKRDTIM	AVG_SINGLEBLKRDTIME_MS	RATIO	FILE_RANK
43381	1286	1281	4404	13270	30.13	245567	20441	.83	4402	13266	30.14	90.21	1
43381	7	7	2	0	0	10493	2802	2.67	0	0		3.72	2
43382	1286	1281	18058	54353	30.1	260419	21300	.82	18056	54351	30.1	87.32	1
43382	7	7	2	0	0	11434	2952	2.58	0	0		3.37	2
43383	1286	1281	8769	24586	28.04	250147	20008	.8	8767	24583	28.04	85.35	1
43383	7	7	2	3	15	11289	2877	2.55	0	0		3.6	2
43384	1286	1281	3124	8983	28.75	212612	36098	1.7	3121	8976	28.76	74.85	1
43384	4	4	15960	4602	2.88	5	7	14	15957	4597	2.88	10.92	2

IO response time evolution

- Identify IO problems
- Compare old and new IO subsystems
- Trend of usage, plan for more IO spindles

Number of IO and time per IO



SQL execution time evolution

- A lot of information in dba_hist_sqlstat

```
SQL> desc dba_hist_sqlstat
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SNAP_ID                                   NOT NULL NUMBER
 DBID                                      NOT NULL NUMBER
 INSTANCE_NUMBER                           NOT NULL NUMBER
 SQL_ID                                    NOT NULL VARCHAR2(13)
 PLAN_HASH_VALUE                           NOT NULL NUMBER
[...]
 EXECUTIONS_TOTAL                                   NUMBER
 EXECUTIONS_DELTA                                   NUMBER
[...]
 DISK_READS_TOTAL                                   NUMBER
 DISK_READS_DELTA                                   NUMBER
```

  - Total versus delta. My recommendation is to not used delta but rather make difference between totals (instance startup, skip some of the snapshots)
- Plan can go out of shared pool (and come back!)

SQL with rank, different exec plans

CERN IT Department

```
select * from (
select snap_id,sql_id,plan_hash_value , sum(executions_delta) executions_delta, sum(elapsed_time_delta)
elapsed_time_delta,
            round(100*sum(elapsed_time_delta)/sum(sum(elapsed_time_delta)) over (PARTITION BY snap_id),2) ratio,
            rank() over (PARTITION BY snap_id order by sum(elapsed_time_delta) desc) sql_rank,
            o_plan_hash_value, sum(o_executions_delta) o_executions_delta, sum(o_elapsed_time_delta)
o_elapsed_time_delta,PARSING_SCHEMA_NAME
from
            (select snapshot.snap_id,snapshot.instance_number,sql_id,plan_hash_value,
            elapsed_time_delta,
            executions_delta,
            lead(plan_hash_value,1) OVER (PARTITION BY
snapshot.instance_number,snapshot.startup_time,sql_id,snapshot.snap_id order by plan_hash_value desc)
o_plan_hash_value,
            lead(elapsed_time_delta,1) OVER (PARTITION BY
snapshot.instance_number,snapshot.startup_time,sql_id,snapshot.snap_id order by
plan_hash_value desc) o_elapsed_time_delta,
            lead(executions_delta,1) OVER (PARTITION BY
snapshot.instance_number,snapshot.startup_time,sql_id,snapshot.snap_id order by plan_hash_value desc)
o_executions_delta, PARSING_SCHEMA_NAME
            from dba_hist_sqlstat sqlstat,dba_hist_snapshot snapshot where
            snapshot.snap_id=sqlstat.snap_id
            and snapshot.instance_number=sqlstat.instance_number
            )
where  executions_delta>0
group by snap_id,sql_id,PARSING_SCHEMA_NAME, plan_hash_value,o_plan_hash_value)
where o_plan_hash_value is not null
order by sql_id,snap_id, plan_hash_value desc;
```

SNAP_ID	SQL_ID	PLAN_HASH_VALUE	EXECUTIONS_DELTA	ELAPSED_TIME_DELTA	RATIO	SQL_RANK	O_PLAN_HASH_VALUE	O_EXECUTIONS_DELTA	O_ELAPSED_TIME_DELTA
303	59g3mgh0yfaj8	3232516354	1	8415	.09	75	698531012	3	7676
461	59g3mgh0yfaj8	3232516354	1	18810	.1	72	698531012	10	10392

SQL execution plan evolution

CERN IT Department

```
select snap_id, sql_id,
listagg(plan_hash_value,', ') within group
(order by elapsed_time_delta) plan_hash_value,
listagg(elapsed_time_delta,', ') within group
(order by elapsed_time_delta) elapsed_time_delta,
listagg(executions_delta,', ') within group
(order by elapsed_time_delta) executions_delta
from dba_hist_sqlstat where executions_delta>0
group by snap_id,sql_id
order by snap_id;
```

SNAP_ID	SQL_ID	PLAN_HASH_VALUE	ELAPSED_TIME_DELTA	EXECUTIONS_DELTA
300	59g3mgh0yfaj8	3232516354	28458	10
301	59g3mgh0yfaj8	3232516354	29031	10
302	59g3mgh0yfaj8	3232516354	28615	10
303	59g3mgh0yfaj8	698531012, 32325163	7676, 8415	3, 1
304	59g3mgh0yfaj8	698531012	5609	10
305	59g3mgh0yfaj8	698531012	3261	10

SQL execution plan evolution


```
select snap_id,plan_hash_value,elapsed_time_delta,executions_delta from
            (select
            snap_id,
                        lead(plan_hash_value,1,-1) over (order by snap_id ) le,
                        lag(plan_hash_value,1,-1) over (order by snap_id ) la,
                        plan_hash_value,elapsed_time_delta,executions_delta
            from
                        (select snap_id, sql_id,listagg(plan_hash_value,', ') within group (order by
elapsed_time_delta) plan_hash_value,
                        listagg(elapsed_time_delta,', ') within group (order by elapsed_time_delta)
elapsed_time_delta,
                        listagg(executions_delta,', ') within group (order by elapsed_time_delta) executions_delta
                        from dba_hist_sqlstat where sql_id='59g3mgh0yfaj8' and executions_delta>0
                        group by snap_id,sql_id
                        )
            )
where plan_hash_value<>le or plan_hash_value<>la
order by snap_id;
```

SNAP_ID	PLAN_HASH_VALUE	ELAPSED_TIME_DELTA	EXECUTIONS_DELTA
240	3232516354	2088	5
302	3232516354	28615	10
303	698531012, 3232516354	7676, 8415	3, 1
304	698531012	5609	10
460	698531012	4336	10
461	698531012, 3232516354	10392, 18810	10, 1
462	698531012	2483	7
847	698531012	3626	10

The plan has changed!

Cardinality feedback

- Introduced to me by Wolfgang Breitling at Trivadis CBO days in 2006. Never stopped using it since.
- Main idea is that if the optimiser makes a wrong estimate for the cardinality, it can lead to a sub-optimal execution plan.
- **"Where Cardinality Can Go Wrong (Greg Rahn, see reference)**

  There are several common scenarios that can lead to inaccurate cardinality estimates. Some of those on the list are:
  - **Data skew**: *Is the NDV (Number Distinct Values) inaccurate due to data skew and a poor dbms_stats sample?*
  - **Data correlation**: *Are two or more predicates related to each other?*
  - **Out-of-range values**: *Is the predicate within the range of known values?*
  - **Use of functions in predicates**: *Is the 5% cardinality guess for functions accurate?*
  - **Stats gathering strategies**: *Is your stats gathering strategy yielding representative stats?"*

Cardinality feedback

- [g]v$sql_plan_statistics_all
  - last_output_rows
  - round(OUTPUT_ROWS/executions) "AVG ROWS“
  - cardinality "EST ROWS“
- dbms_xplan.display%(...'ALLSTATS LAST');

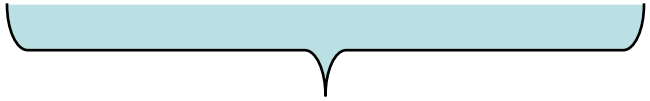
- (from my experience) often useful to refer to [g]v$sql_plan_statistics_all diectly rather than dbms_xplan

Cardinality feedback

CERN IT Department

```
SELECT  depth,LPAD (' ', DEPTH) || operation operation,
options, object_name,
         round(STARTS/executions,2) "AVG starts", last_output_rows "LAST ROWS",
         round(OUTPUT_ROWS/executions) "AVG ROWS",
         cardinality "EST ROWS",
         executions
FROM      gv$sql_plan_statistics_all
WHERE
        sql_id ='4d56v07vcnh59'
ORDER BY inst_id,sql_id,child_number,id;
```

DEPTH	OPERATION	OPTIONS	OBJECT_NAME	AVG starts	LAST ROWS	AVG ROWS	EST ROWS	EXECUTIONS
0	SELECT STATEMENT			1	1	1		1
1	SORT	AGGREGATE		1	1	1	1	1
2	TABLE ACCESS	BY INDEX ROWID	EMP2	1	1	1	1	1
3	INDEX	RANGE SCAN	EMP2_ENAME	1	2187	2187	1	1



discrepancy

Cardinality feedback

CERN IT Department

```
SELECT  id,depth,LPAD (' ', DEPTH) || operation operation,
options, object_name,
         round(STARTS/executions,2) "AVG starts", last_output_rows "LAST ROWS",
         round(OUTPUT_ROWS/executions) "AVG ROWS", cardinality "EST ROWS",
         round((OUTPUT_ROWS/executions)/cardinality,2) avg_over_est,
         CASE WHEN (OUTPUT_ROWS/executions)/cardinality > 1 THEN
         rpad('+',round(log(10, ((OUTPUT_ROWS/executions)/cardinality) )),'+') else
         rpad('-',round(-log(10,(OUTPUT_ROWS/executions/cardinality))),'-') END
magnitude
FROM     gv$sql_plan_statistics_all
WHERE
       sql_id ='5ymxdhjdtzd3y'
ORDER BY inst_id,sql_id,child_number,id;
```

Makes it easier to identify the rowsource from which the problem comes from

ID	DEPTH	OPERATION	OPTIONS	OBJECT_NAME	AVG starts	LAST ROWS	AVG ROWS	EST ROWS	AVG_OVER_EST	MAGNITUDE
1	1	SORT	ORDER BY		1	47397	47397	152358	.31	-
2	2	NESTED LOOPS			1	47397	47397	152358	.31	-
3	3	NESTED LOOPS			1	47397	47397	150394	.32	-
4	4	NESTED LOOPS			1	47397	47397	150475	.31	-
5	5	TABLE ACCESS	BY INDEX ROWID	DOC_SEARCH_RECENT	1	47397	47397	150475	.31	-
6	6	DOMAIN INDEX		I_DOC_XML_XML_RECENT	1	47397	47397	278740	.17	-
7	5	TABLE ACCESS	BY INDEX ROWID	EDFELBISTOOLS	47397	47397	47397	1	47397	+++++
8	6	INDEX	RANGE SCAN	I_DIDEDFELBIS	47397	47397	47397	1	47397	+++++
9	4	INDEX	RANGE SCAN	TYPTOD	47397	47397	47397	1	47397	+++++
10	3	TABLE ACCESS	BY INDEX ROWID	EDHPER	47397	47397	47397	1	47397	+++++
11	4	INDEX	UNIQUE SCAN	I_EDHPER_IDN	47397	47397	47397	1	47397	+++++

How to have cardinality feedback

- Actual number of rows for each row source (last and aggregated)
- select /*+ **gather_plan_statistics** */ ...
- **alter session set statistics_level=all**;
  (ALL | TYPICAL | BASIC, on-logon trigger)
- Individual sql (sql_id) tagging (Miladin Modrakovic, “Oracle event SQL_TRACE in 11g”)

  alter system set events '**sql_trace** [sql:02hc70wxyugg0]';
  - Cost can be reduced by setting it for only a few executions
    - It has a cost
    - One can not disable it while the statement is running, be careful with the additional resource required for long statements

“Live” cardinality feedback

- At the next parse (soft or hard) operation,
  - if sql is tagged for sql_trace or statistics_level is set to all
  - A new child cursor is created
- Example
  - Java / jdbc program running in a loop
    - Open connection
    - parse
    - execute 5 times
    - parse
    - execute 5 times
    - Closes connection and exits
  - Enable sql_trace on the given sql_id
  - Change statistics_level

AWR and cardinality feedback

CERN IT Department

- AWR does not (as far as I know) collect cardinality feedback, no AWR extension scheme (again AFAIK)
- But... we can build the extension
  - create table **dba_hist_sqlplan_statistics** as select * from **v$sql_plan_statistics_all** where rownum<1;
  - alter table **dba_hist_sqlplan_statistics** add (**snap_id number,dbid number,instance_number number**);
  - Procedure which runs every minute, checks for new AWR report, if any, copy data from **v$sql_plan_statistics_all** to **dba_hist_sqlplan_statistics**

Cardinality feedback evolution

CERN IT Department

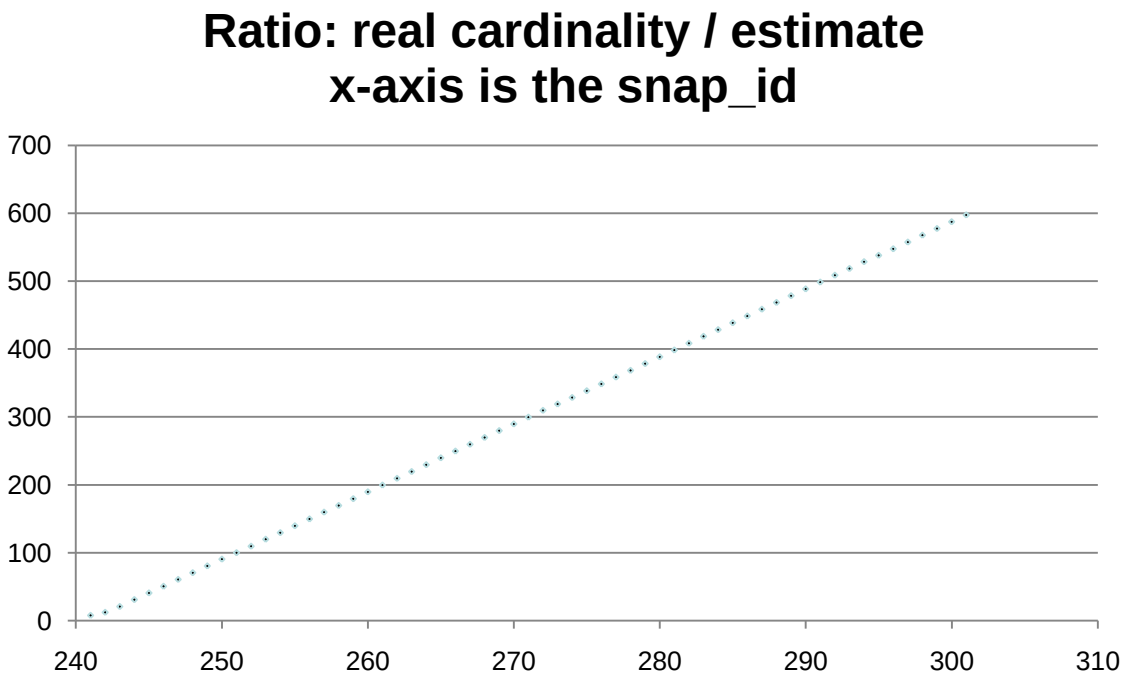
```
SELECT  snap_id,child_number,depth,LPAD (' ', DEPTH) || operation operation,
options, object_name,
         last_output_rows "LAST ROWS", round(OUTPUT_ROWS/executions) "AVG ROWS",
         cardinality "EST ROWS",
          executions, PLAN_HASH_VALUE
FROM      dba_hist_sqlplan_statistics
WHERE
       sql_id ='59g3mgh0yfaj8' and snap_id in (240,241,301,758) and executions>1
ORDER BY snap_id,instance_number,sql_id,child_number,id;
```

SNAP_ID	CHILD_NUMBER	DEPTH	OPERATION	OPTIONS	OBJECT_NAME	LAST ROWS	AVG ROWS	EST ROWS	EXECUTIONS	PLAN_HASH_VALUE	MAGNITUDE
240	0	0	SELECT STATEMENT			1	1		6	3232516354	
240	0	1	SORT	AGGREGATE		1	1	1	6	3232516354	
240	0	2	TABLE ACCESS	BY INDEX ROWID	EMP2	1	1	1	6	3232516354	
240	0	3	INDEX	RANGE SCAN	EMP2_ENAME	6	3	1	6	3232516354	
241	0	0	SELECT STATEMENT			1	1		9	3232516354	
241	0	1	SORT	AGGREGATE		1	1	1	9	3232516354	
241	0	2	TABLE ACCESS	BY INDEX ROWID	EMP2	1	1	1	9	3232516354	
241	0	3	INDEX	RANGE SCAN	EMP2_ENAME	8	4	1	9	3232516354	+
301	0	0	SELECT STATEMENT			1	1		603	3232516354	
301	0	1	SORT	AGGREGATE		1	1	1	603	3232516354	
301	0	2	TABLE ACCESS	BY INDEX ROWID	EMP2	1	1	1	603	3232516354	
301	0	3	INDEX	RANGE SCAN	EMP2_ENAME	602	301	1	603	3232516354	++
758	1	0	SELECT STATEMENT			1	1		2962	698531012	
758	1	1	SORT	AGGREGATE		1	1	1	2962	698531012	
758	1	2	TABLE ACCESS	BY INDEX ROWID	EMP2	1	1	1	2962	698531012	
758	1	3	INDEX	RANGE SCAN	EMP2_JOB	1	1	1	2962	698531012	

Cardinality feedback evolution

- Evolution of the Ratio real over estimate

```
select snap_id,round(OUTPUT_ROWS_delta/executions_delta/cardinality,5) ratio_real_over_estimate
FROM
             (select
             snap_id,sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner,
             OUTPUT_ROWS - lead(OUTPUT_ROWS, 1) OVER (PARTITION BY
sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner ORDER BY snap_id DESC) OUTPUT_ROWS_delta,
             executions -lead(executions, 1) OVER (PARTITION BY
sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner ORDER BY snap_id DESC) executions_delta,
             cardinality
             from dba_hist_sqlplan_statistics where sql_id ='59g3mgh0yfaj8'  and snap_id  between 240 and 301 and executions>1)
where executions_delta is not null and id=3
ORDER BY snap_id,sql_id,id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner;
```



Cardinality feedback evolution

- Linear regression, slope

```
SELECT
sql_id,child_number,depth,LPAD (' ', DEPTH) || operation operation,
options, object_name,
round(REGR_SLOPE(OUTPUT_ROWS_delta/executions_delta/cardinality, snap_id),2) slope
FROM
  (select
  snap_id,sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner,
  OUTPUT_ROWS - lead(OUTPUT_ROWS, 1) OVER (PARTITION BY
sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner ORDER BY snap_id
DESC) OUTPUT_ROWS_delta,
  executions -lead(executions, 1) OVER (PARTITION BY
sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner ORDER BY snap_id
DESC) executions_delta,
  cardinality
  from dba_hist_sqlplan_statistics where sql_id ='59g3mgh0yfaj8'  and snap_id  between 240 and 301 and
executions>1)
where executions_delta is not null
group by sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner
ORDER BY sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner;
```

DEPTH	OPERATION	OPTIONS	OBJECT_NAME	SLOPE
0	SELECT STATEMENT			
1	SORT	AGGREGATE		0
2	TABLE ACCESS	BY INDEX ROWID	EMP2	0
3	INDEX	RANGE SCAN	EMP2_ENAME	9.94

Cardinality feedback evolution

CERN IT Department

- Linear regression, identify high and low slope value

```
with s as
(SELECT
sql_id,child_number,plan_hash_value,depth,LPAD (' ', DEPTH) || operation operation,
options, object_name, object_owner, id,
REGR_SLOPE(OUTPUT_ROWS_delta/executions_delta/cardinality, snap_id) slope
FROM
              (select
              snap_id,sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner,
              OUTPUT_ROWS - lead(OUTPUT_ROWS, 1) OVER (PARTITION BY
sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner ORDER BY snap_id DESC) OUTPUT_ROWS_delta,
              executions -lead(executions, 1) OVER (PARTITION BY
sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner ORDER BY snap_id DESC) executions_delta,
              cardinality
              from dba_hist_sqlplan_statistics where
              executions>1
              )
where executions_delta >0
group by sql_id,plan_hash_value,child_number,id,depth,operation,options,object_name,object_owner)
select s1.sql_id,s1.child_number,s1.plan_hash_value,s1.depth,LPAD (' ', s1.DEPTH) || s1.operation operation,
s1.options, s1.object_name, s1.object_owner, round(s1.slope,2) slope
from s s1, s s2 where s1.sql_id=s2.sql_id and s1.child_number=s2.child_number and s1.plan_hash_value=s2.plan_hash_value and
(s2.slope >5 or s2.slope < 0.2) and s2.slope<>0
ORDER BY s1.sql_id,s1.plan_hash_value,s1.child_number,s1.id,s1.depth,s1.operation,s1.options,s1.object_name,s1.object_owner;
```

SQL_ID	CHILD_NUMBER	PLAN_HASH_VALUE	DEPTH	OPERATION	OPTIONS	OBJECT_NAME	OBJECT_OWNER	SLOPE
59g3mgh0yfaj8	0	3232516354	0	SELECT STATEMENT				
59g3mgh0yfaj8	0	3232516354	1	SORT	AGGREGATE			0
59g3mgh0yfaj8	0	3232516354	2	TABLE ACCESS	BY INDEX ROWID	EMP2	GRANCHERDBA	0
59g3mgh0yfaj8	0	3232516354	3	INDEX	RANGE SCAN	EMP2_ENAME	GRANCHERDBA	9.94

Timed event interface

- Oracle database code is instrumented
- When a session is “active”, the database software publishes what it is waiting on
- Session 1

SQL> create table a (a1 number primary key,a2 number);

SQL> insert into a values (0,0);

- Session 2

SQL> insert into a values (0,1);

Hangs

- Session 3

TRAINING:SQL> select sql_id,status,event,state from v$session where sid=1629;

```
SQL_ID         STAT    EVENT                                    STATE
-------------- ------  ---------------------------------------- ------
2s69gqm9yp53s  ACTIVE  enq: TX -row lock contention             WAITING
```

Active Session History

- See Graham Wood's presentation
- Naive mockup:

from v$session where status='ACTIVE'

||||
vvvv

v$active_session_history → → DBA_HIST_ACTIVE_SESS_HISTORY →

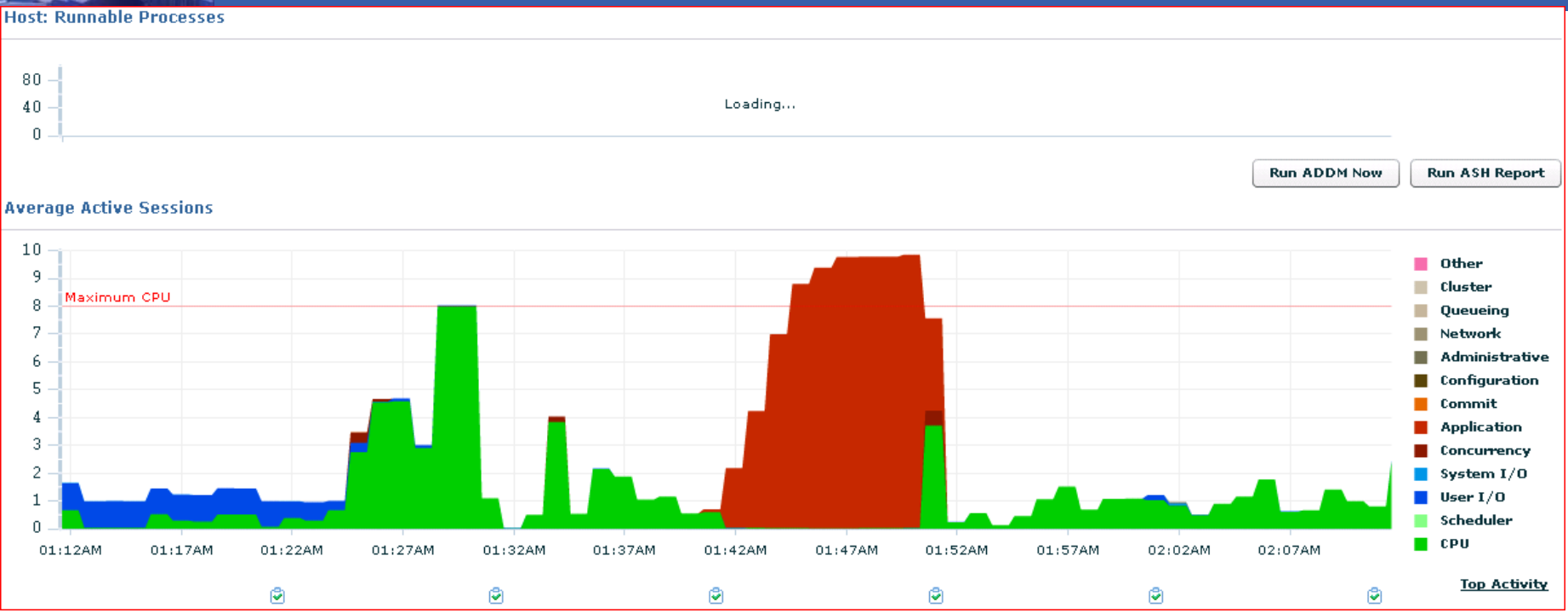
Active Session History

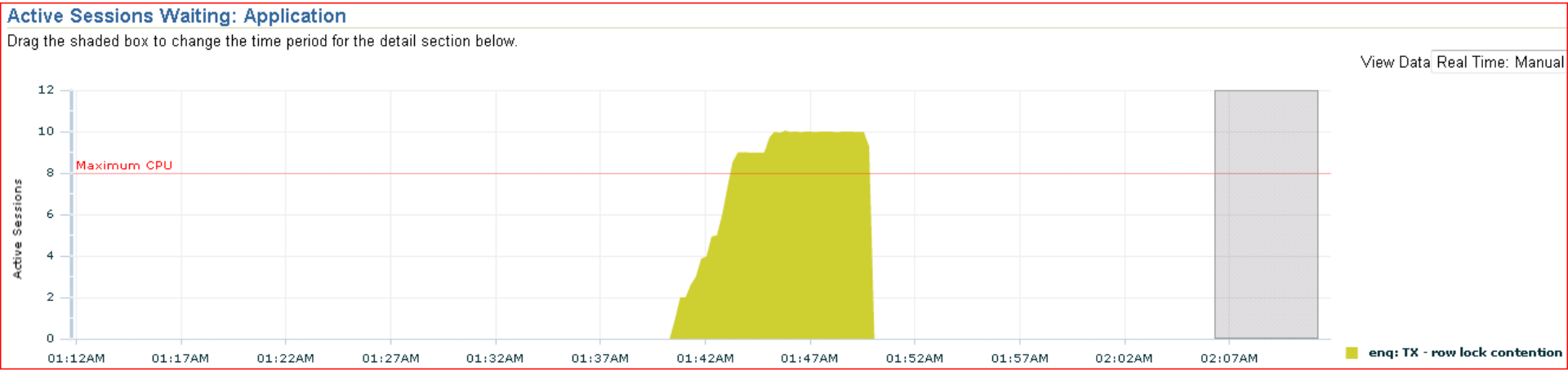
CERN IT Department

```
SQL> desc v$active_session_history
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 SAMPLE_ID                                          NUMBER
 SAMPLE_TIME                                        TIMESTAMP(3)
 IS_AWR_SAMPLE                                      VARCHAR2(1)
 SESSION_ID                                         NUMBER
 SESSION_SERIAL#                                    NUMBER
[...]
 SQL_ID                                             VARCHAR2(13)
[...]
 EVENT                                              VARCHAR2(64)
 EVENT_ID                                           NUMBER
 EVENT#                                             NUMBER
 SEQ#                                               NUMBER
 P1TEXT                                             VARCHAR2(64)
 P1                                                 NUMBER
 P2TEXT                                             VARCHAR2(64)
 P2                                                 NUMBER
 P3TEXT                                             VARCHAR2(64)
 P3                                                 NUMBER
 WAIT_CLASS                                         VARCHAR2(64)
```

Active Session History

CERN IT Department





What happened at...?

CERN IT Department

- In 10.2

```
select session_id,session_serial#,seq#,event#,min(sample_time),
max(sample_time),count(*),CURRENT_OBJ#,CURRENT_FILE#,CURRENT_BLOCK#
from v$active_session_history where sample_time between
to_date('29112009:0140','DDMMYYYY:HH24MI') and
to_date('29112009:0150','DDMMYYYY:HH24MI')
and event='enq: TX - row lock contention' and session_state='WAITING'
group by
session_id,session_serial#,seq#,event#,CURRENT_OBJ#,CURRENT_FILE#,CURRENT_BLOCK#;
```

SESSION_ID	SESSION_SERIAL#	SEQ#	EVENT#	MIN(SAMPLE_TIME)	MAX(SAMPLE_TIME)	COUNT(*)	CURRENT_OBJ#	CURRENT_FILE#	CURRENT_BLOCK#
247	34277	40	186	29-NOV-09 01.42.59.403 AM	29-NOV-09 01.49.47.879 AM	409	741226	65	12574
130	37146	40	186	29-NOV-09 01.42.37.373 AM	29-NOV-09 01.49.47.879 AM	431	741226	65	12574
152	3552	40	186	29-NOV-09 01.40.33.228 AM	29-NOV-09 01.49.47.879 AM	555	741226	65	12574
206	44660	40	186	29-NOV-09 01.40.21.218 AM	29-NOV-09 01.49.47.879 AM	567	741226	65	12574
354	4460	40	186	29-NOV-09 01.43.11.413 AM	29-NOV-09 01.49.47.879 AM	397	741226	65	12574
208	49197	40	186	29-NOV-09 01.42.05.318 AM	29-NOV-09 01.49.47.879 AM	463	741226	65	12574
93	63540	40	186	29-NOV-09 01.44.53.537 AM	29-NOV-09 01.49.47.879 AM	295	741226	65	12574
133	37033	40	186	29-NOV-09 01.41.36.288 AM	29-NOV-09 01.49.47.879 AM	492	741226	65	12574
310	34391	40	186	29-NOV-09 01.42.46.383 AM	29-NOV-09 01.49.47.879 AM	422	741226	65	12575
374	59895	40	186	29-NOV-09 01.41.10.268 AM	29-NOV-09 01.49.47.879 AM	518	741226	65	12574

What happened at...?

- In 11.2, additional column added

```
CURRENT_OBJ#                    NUMBER
CURRENT_FILE#                   NUMBER
CURRENT_BLOCK#                  NUMBER
CURRENT_ROW#                    NUMBER
```

CURRENT_OBJ#	CURRENT_FILE#	CURRENT_BLOCK#	CURRENT_ROW#
69502	4	134	3
69502	4	134	5
69502	4	134	0
69502	4	134	6
69502	4	134	4
69502	4	134	7
69502	4	134	2
69502	4	134	1
69502	4	134	8
69502	4	134	9

```
DBMS_ROWID.ROWID_CREATE (
   rowid_type    IN NUMBER,
   object_number IN NUMBER,
   relative_fno  IN NUMBER,
   block_number  IN NUMBER,
   row_number    IN NUMBER)
  RETURN ROWID;
```

What happened at...

- Retrieve object, then create ROWID
- Flashback query to have the data “as of the problem timestamp!”

```
SQL> select owner,object_name from dba_objects where object_id=69502;
OWNER                          OBJECT_NAME
------------------------------ ---------------
GRANCHER                       A

SQL> select * from grancher.a where
rowid=DBMS_ROWID.ROWID_CREATE(1,69502,4,134,3);
        A1 MODIF_DATE          MODIF_REASON
---------- ------------------- --------------------
         4 2009-11-21 16:42:16 8157

SQL> select * from
grancher.a as of timestamp to_date ('21112009:1622','DDMMYYYY:HH24MI')
where rowid = DBMS_ROWID.ROWID_CREATE(1,69502,4,134,3);
        A1 MODIF_DATE          MODIF_REASON
---------- ------------------- --------------------
         4 2009-11-21 16:21:44 8157
```

Conclusions

- The Oracle database produces a lot of information (AWR turned “on typical” by default).
- Raw data access is key for understanding, analytics to the rescue. Graphing can be done on top.
  - Evolution of IO usage / response time
  - Evolution of SQL timing / resource as well as execution plan evolution
  - Cardinality feedback and evolution (with “AWR extension”) of rowsource real versus estimated ratio
  - Active Session History identification of enqueue

References

- Graham Wood, “Sifting through the ASHes” http://www.oracle.com/technology/products/manageability/database/pdf/twp03/PPT_active_session_history.pdf
- Shouvik Basu, “Analytic functions by Example” http://www.orafaq.com/node/55
- Greg Rahn, “Troubleshooting Bad Execution Plans” http://structureddata.org/2007/11/21/troubleshooting-bad-execution-plans/
- Miladin Modrakovic, “Oracle event SQL_TRACE in 11g” http://oraclue.com/2009/03/24/oracle-event-sql_trace-in-11g/

Q&A

session 25, Eric Grancher

Eric.Grancher@cern.ch

Glad to discuss / share ideas at the “Meet the speaker”, UKOUG Lounge, exhibition hall in a few minutes

DEPTH	OPERATION	OPTIONS	OBJECT_NAME	AVG starts	LAST ROWS	AVG ROWS	EST ROWS	AVG_OVER_EST	EXECUTIONS	EST TIME(s)	AVG TIME(s)	LAST TIME(s)
1	SORT	ORDER BY		1	40312	40312	235827	.17	1	14246	3.16	3.16
2	NESTED LOOPS			1	40312	40312	235827	.17	1	14212	1.39	1.39
3	NESTED LOOPS			1	40312	40312	252678	.16	1	11179	.87	.87
4	NESTED LOOPS			1	40312	40312	252898	.16	1	11179	.67	.67
5	TABLE ACCESS	BY INDEX ROWID	DOC_SEARCH_RECENT	1	40312	40312	252898	.16	1	2071	.1	.1
6	DOMAIN INDEX		I_DOC_XML_XML_RECENT	1	40312	40312	277197	.15	1	2058	.06	.06
5	TABLE ACCESS	BY INDEX ROWID	EDFELBISTOOLS	40312	40312	40312	1	40312	1	1	.49	.49
6	INDEX	RANGE SCAN	I_DIDEDFELBIS	40312	40312	40312	1	40312	1	1	.3	.3
4	INDEX	RANGE SCAN	TYPTOD	40312	40312	40312	1	40312	1		.14	.14
3	TABLE ACCESS	BY INDEX ROWID	EDHPER	40312	40312	40312	1	40312	1	1	.45	.45
4	INDEX	UNIQUE SCAN	I_EDHPER_IDN	40312	40312	40312	1	40312	1		.16	.16

DEPTH	OPERATION	OPTIONS	OBJECT_NAME	EXECUTIONS	EST TIME(s)	AVG TIME(s)	LAST TIME(s)
1	SORT	ORDER BY		1	14246	3.16	3.16
2	NESTED LOOPS			1	14212	1.39	1.39
3	NESTED LOOPS			1	11179	.87	.87
4	NESTED LOOPS			1	11179	.67	.67
5	TABLE ACCESS	BY INDEX ROWID	DOC_SEARCH_RECENT	1	2071	.1	.1
6	DOMAIN INDEX		I_DOC_XML_XML_RECENT	1	2058	.06	.06
5	TABLE ACCESS	BY INDEX ROWID	EDFELBISTOOLS	1	1	.49	.49
6	INDEX	RANGE SCAN	I_DIDEDFELBIS	1	1	.3	.3
4	INDEX	RANGE SCAN	TYPTOD	1		.14	.14
3	TABLE ACCESS	BY INDEX ROWID	EDHPER	1	1	.45	.45
4	INDEX	UNIQUE SCAN	I_EDHPER_IDN	1		.16	.16

DEPTH	OPERATION	OPTIONS	OBJECT_NAME	AVG starts	LAST ROWS	AVG ROWS	EST ROWS	AVG_OVER_EST
1	SORT	ORDER BY		1	40312	40312	235827	.17
2	NESTED LOOPS			1	40312	40312	235827	.17
3	NESTED LOOPS			1	40312	40312	252678	.16
4	NESTED LOOPS			1	40312	40312	252898	.16
5	TABLE ACCESS	BY INDEX ROWID	DOC_SEARCH_RECENT	1	40312	40312	252898	.16
6	DOMAIN INDEX		I_DOC_XML_XML_RECENT	1	40312	40312	277197	.15
5	TABLE ACCESS	BY INDEX ROWID	EDFELBISTOOLS	40312	40312	40312	1	40312
6	INDEX	RANGE SCAN	I_DIDEDFELBIS	40312	40312	40312	1	40312
4	INDEX	RANGE SCAN	TYPTOD	40312	40312	40312	1	40312
3	TABLE ACCESS	BY INDEX ROWID	EDHPER	40312	40312	40312	1	40312
4	INDEX	UNIQUE SCAN	I_EDHPER_IDN	40312	40312	40312	1	40312