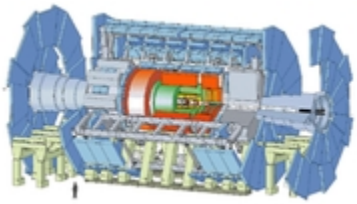
Big data processing and analysis workshop
Moscow, 29th Jan 2015

the ATLAS Experiment

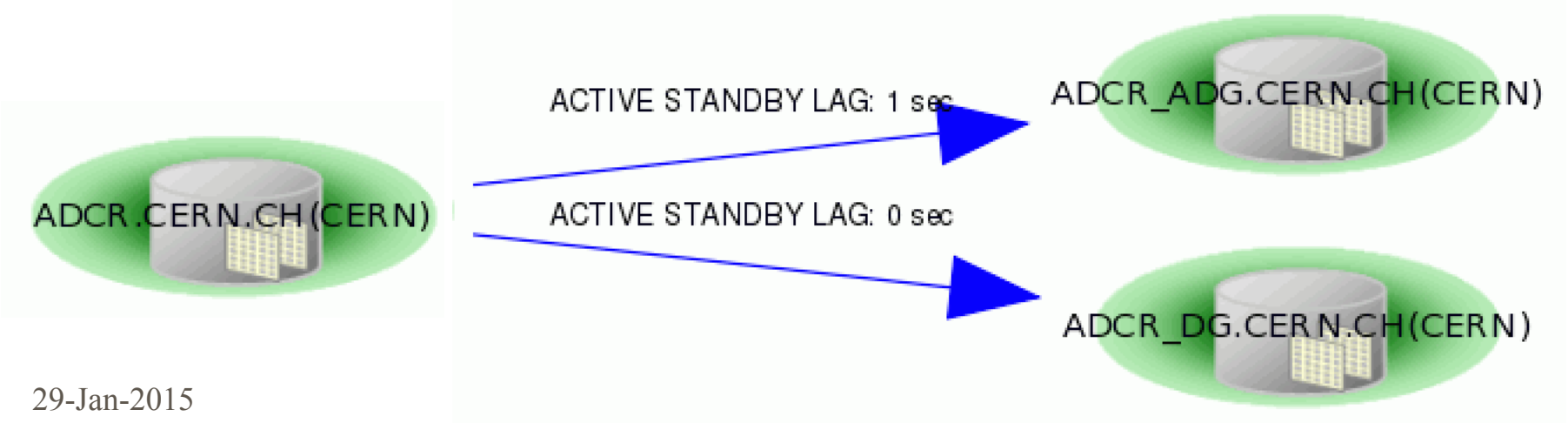


PanDA database schema: an overview, current state and possible improvements for LHC Run2

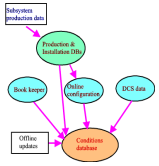
Gancho Dimitrov (CERN)

HW specifications of the ADC Oracle cluster

Database / info	ADCR
Main DB role	RDBMS for the Grid jobs and file distribution and management
Oracle ver.	11.2.0.4
# DB nodes	4
DB volume	25 TB
HW specs	CPU Intel E5-2650@ 2GHz - 16 cores per node, RAM 256 GB 10 GigE for storage and cluster access NetApp NAS storage with 1.5 TB SSD cache



PanDA system



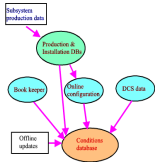
- **The PanDA system relies on the Oracle RDBMS since 2008**. Proper schema design, data management and index strategy are key factors due to the importance of the DB layer for the ATLAS workload management system.

- **DB system has to deal efficiently with two different workloads**: transactional (PanDA server) and data warehouse load (PanDA monitor).

- Due to the different workloads the PanDA data are logically split into **'operational' (**400 GB**)** and **'archive' (**8 TB**)** types and respectively stored into separate DB schemes.

- **The newly introduced JEDI (Job Execution and Definition Interface) component introduced new set of tables** which need extra disk space and handling. Currently at 240 GB

Why split PanDA data?



- PanDA server interacts with small manageable '**operational**' data and slice of the very recent history (~ 400GB). The recent history is defined to be 3 days but can be easily changed to any other period that could be considered appropriate.

- On the PanDA '**operational**' data are defined different indices than on the '**archive**' (7 TB) as the workload is different.

- The '**archive**' partitions are with a compression setting and for good reasons are set to reside in yearly based tablespaces.

This makes easy and with minimum resources the partition exchanges to dedicated tables on the production server and further move to different DB (with the TTS method) for long-term data preservation.

The PanDA 'operational' and 'archive' data

- All information relevant to a single job is stored in four fact tables.
- The '**operational**' data is kept in a separate schema which hosts all active jobs plus finished jobs of the most recent 3 days.
- Jobs that get status 'finished' or 'failed' and are older than 3 days are moved to an archive PANDA schema (PanDA => PanDAARCH).
- **In the first implementation the data move to the archive schema was done by the PanDA server by transactional rows insert and delete statements.**

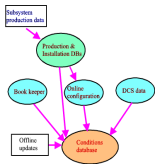
  However the negative impact from that approach was:

  - increased transactional workload on the database

  - highly fragmented PanDA tables due to the row deletion from the '**operational**' tables.

  - the hot operational data and indexes could not fit into the buffer pool

Better PanDA data segments organization

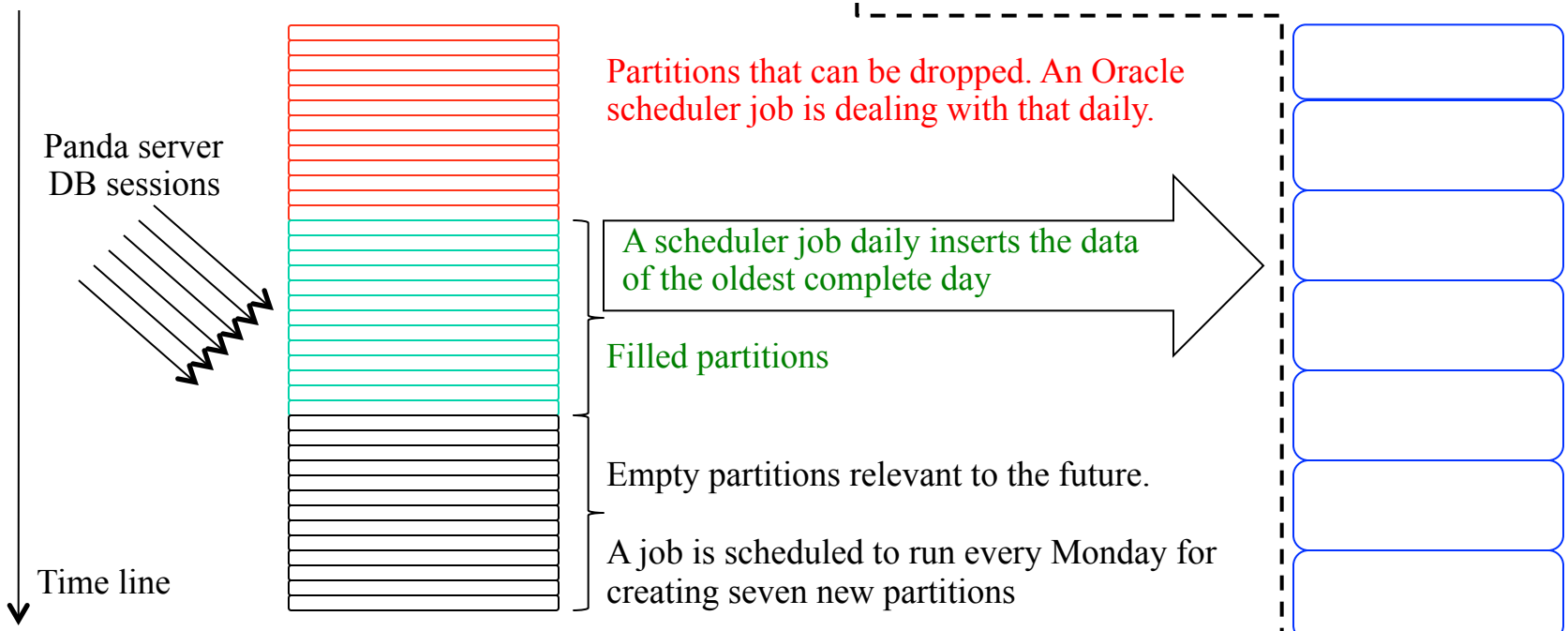


‘Operational’ data

Tables are range partitioned on a column of type “Date”
Each partition covers a time range of a day.

‘Archive’ data

Tables are range partitioned on a column of type “Date”
Some tables are with 3 days interval partitions others with monthly partitions



PanDA => PanDA archive data flow machinery

- The PanDA => PanDAARCH data flow is **sustained by a set of scheduler jobs** on the Oracle server which execute a logic encoded in PL/SQL procedures

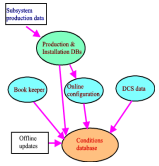
Weekly job creates daily partitions relevant to the near future days

Daily job copies data of a set of partitions from PanDA to PanDAARCH

JOB_NAME	JOB_CLASS	RUN_DURATION	LAST_START_DATE	LAST_RUN_DURATION	LAST_STATUS	STATE	NEXT_RUN_DATE
ADD_DAILYPART_PANDA	PANDA_JOB_CLASS	-	27-OCT-2014 10:00	0 00:00:03.0	SUCCEEDED	SCHEDULED	03-NOV-2014 10:00
BULKCOPY_PANDAPART_JOB	PANDA_JOB_CLASS	-	30-OCT-2014 20:00	0 00:55:18.0	SUCCEEDED	SCHEDULED	31-OCT-2014 20:00
JEDI_REFR_MINTASKIDS_JOB	PANDA_JOB_CLASS	-	31-OCT-2014 15:05	0 00:00:00.0	SUCCEEDED	SCHEDULED	31-OCT-2014 15:06
PANDA_DATASETS_90DAYS_SLWINDOW	PANDA_JOB_CLASS	-	05-OCT-2014 08:00	0 00:00:01.0	SUCCEEDED	SCHEDULED	05-NOV-2014 08:00
PANDA_PANDALOG_SLWINDOW	PANDA_JOB_CLASS	-	31-OCT-2014 08:30	0 00:00:01.0	SUCCEEDED	SCHEDULED	01-NOV-2014 08:30
PANDA_TAB_INDICES_REBUILD	PANDA_JOB_CLASS	-	27-OCT-2014 11:00	0 00:00:28.0	SUCCEEDED	SCHEDULED	03-NOV-2014 11:00
UPDATE_JOBSACTIVE_STATS_JOB	PANDA_JOB_CLASS	-	31-OCT-2014 15:04	0 00:00:01.0	SUCCEEDED	SCHEDULED	31-OCT-2014 15:06
VERIF_AND_DROP_PANDAPART_JOB	PANDA_JOB_CLASS	-	31-OCT-2014 09:00	0 00:11:37.0	SUCCEEDED	SCHEDULED	01-NOV-2014 09:00

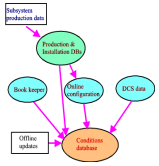
Daily job verifies that all rows of certain partition have been copied to PanDAARCH and drops the PanDA partition if the above is true.

Benefits from the data removal based on partitions



- **Scalability**: the PanDA bulk data copy and deletion is done on **table partition level instead on row level**
- **Not IO demanding:** Removing the already copied data is not IO demanding as this is a simple DDL operation over a table segment and its relevant index segments (alter table … drop partition)
- **Avoids fragmentation** in the PanDA '**operational**' table. Much better space utilization and caching in the buffer pool
- **No need for indices rebuild or coalesce operations** for these partitioned tables.

Applied policies on the 'archive' data



- Table partitions by **design are set to reside into dedicated yearly based Oracle tablespaces.**

- Activated **basic compression**

  With this option on, we can fit 50% more rows within a data block of 8KB into the 'jobs' tables and 60% more rows into the "files" table blocks.

- **Sustained data sliding window of 12 months** performing partition exchange operations from the primary tables to new yearly based tables.

  It does not require IO resources (and resp. CPU) as it is a simple Oracle data dictionary change.

- The new yearly based tables can be easily moved to another database cluster **as they are self-contained in dedicated tablespaces**.

Archive partitions management

- The PanDA archive data segmentation makes very simple the data archiving by performing partition exchanges from the current tables to new dedicated yearly based tables: **results in fast operations (avoids physical row movement).**

Source table partition exchange with an auxiliary table :

ALTER TABLE FILESTABLE_ARCH **EXCHANGE PARTITION** FILESTABLE_ARCH_JAN_2009 WITH TABLE AUX_FILESTABLE_ARCH **INCLUDING INDEXES WITHOUT VALIDATION;**

Destination table partition exchange with the auxiliary table :

ALTER TABLE Y2009_FILESTABLE_ARCH **EXCHANGE PARTITION** FILESTABLE_ARCH_JAN_2009 WITH TABLE AUX_FILESTABLE_ARCH **INCLUDING INDEXES WITHOUT VALIDATION;**

Snapshot of the partitions states

	TABLE_NAME	PARTITION_NAME	NUM_ROWS	LAST_ANALYZED
1	FILESTABLE_ARCH	FILESTABLE_ARCH_JAN_2009	18853782	02.07.13 22:27:36
2	FILESTABLE_ARCH	FILESTABLE_ARCH_FEB_2009	21816573	03.07.13 02:02:23
3	FILESTABLE_ARCH	FILESTABLE_ARCH_MAR_2009	32323317	02.07.13 23:05:50
4	FILESTABLE_ARCH	FILESTABLE_ARCH_APR_2009	56934389	02.07.13 21:41:22
5	FILESTABLE_ARCH	FILESTABLE_ARCH_MAY_2009	37405261	02.07.13 22:31:03
6	FILESTABLE_ARCH	FILESTABLE_ARCH_JUN_2009	44404565	03.07.13 03:21:51
7	FILESTABLE_ARCH	FILESTABLE_ARCH_JUL_2009	36353676	02.07.13 21:46:19
8	FILESTABLE_ARCH	FILESTABLE_ARCH_AUG_2009	17632867	03.07
9	FILESTABLE_ARCH	FILESTABLE_ARCH_SEP_2009	21106984	03.07
10	FILESTABLE_ARCH	FILESTABLE_ARCH_OCT_2009	30079052	03.07
11	FILESTABLE_ARCH	FILESTABLE_ARCH_NOV_2009	40702189	02.07
12	FILESTABLE_ARCH	FILESTABLE_ARCH_DEC_2009	44173542	03.07
13	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_JAN_2009	(null)	(null
14	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_FEB_2009	(null)	(null
15	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_MAR_2009	(null)	(null
16	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_APR_2009	(null)	(null
17	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_MAY_2009	(null)	(null
18	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_JUN_2009	(null)	(null
19	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_JUL_2009	(null)	(null
20	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_AUG_2009	(null)	(null
21	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_SEP_2009	(null)	(null
22	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_OCT_2009	(null)	(null
23	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_NOV_2009	(null)	(null
24	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_DEC_2009	(null)	(null

Before the partition exchange of the FILESTABLE

After the partitions exchange of the FILESTABLE

	TABLE_NAME	PARTITION_NAME	NUM_ROWS	LAST_ANALYZED
1	FILESTABLE_ARCH	FILESTABLE_ARCH_JAN_2009	(null)	(null)
2	FILESTABLE_ARCH	FILESTABLE_ARCH_FEB_2009	(null)	(null)
3	FILESTABLE_ARCH	FILESTABLE_ARCH_MAR_2009	(null)	(null)
4	FILESTABLE_ARCH	FILESTABLE_ARCH_APR_2009	(null)	(null)
5	FILESTABLE_ARCH	FILESTABLE_ARCH_MAY_2009	(null)	(null)
6	FILESTABLE_ARCH	FILESTABLE_ARCH_JUN_2009	(null)	(null)
7	FILESTABLE_ARCH	FILESTABLE_ARCH_JUL_2009	(null)	(null)
8	FILESTABLE_ARCH	FILESTABLE_ARCH_AUG_2009	(null)	(null)
9	FILESTABLE_ARCH	FILESTABLE_ARCH_SEP_2009	(null)	(null)
10	FILESTABLE_ARCH	FILESTABLE_ARCH_OCT_2009	(null)	(null)
11	FILESTABLE_ARCH	FILESTABLE_ARCH_NOV_2009	(null)	(null)
12	FILESTABLE_ARCH	FILESTABLE_ARCH_DEC_2009	(null)	(null)
13	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_JAN_2009	18853782	02.07.13 22:27:36
14	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_FEB_2009	21816573	03.07.13 02:02:23
15	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_MAR_2009	32323317	02.07.13 23:05:50
16	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_APR_2009	56934389	02.07.13 21:41:22
17	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_MAY_2009	37405261	02.07.13 22:31:03
18	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_JUN_2009	44404565	03.07.13 03:21:51
19	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_JUL_2009	36353676	02.07.13 21:46:19
20	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_AUG_2009	17632867	03.07.13 06:38:36
21	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_SEP_2009	21106984	03.07.13 03:24:13
22	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_OCT_2009	30079052	03.07.13 02:06:12
23	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_NOV_2009	40702189	02.07.13 21:51:34
24	Y2009_FILESTABLE_ARCH	FILESTABLE_ARCH_DEC_2009	44173542	03.07.13 01:04:47

Insert rates and data volumes

- **Rows insert rates:**
  JOBS archive tables: daily rates **0.8 -1.4 million rows, total 1.4 billion**
  FILES archive tables: daily rates **5-10 million, total 10.7 billion**
  JEDI tables: daily rates **5-10 million, total 0.7 billion**

- **On average used disk space in PanDA archive:**
  per day in 2012-2013 (data compression is off): **6 GB**
  per day in 2014 (data compression is on): **4.4 GB**

- **Current segments sizes:**
  PanDA objects: 0.26 TB data segments, 0.22 TB indices segments
  PanDA archive objects: 6 TB data segments, 1 TB indices segments

Room for improvement (1)

- **PanDA server and PanDA monitor workloads to be served by different DB nodes of the Oracle cluster.**

pros: **=> there won't be concurrency for CPU resources**

=> two data caches will be used (each of 200 GB)

For example:

PanDA monitor

PanDA server

Id	Name	CPU Usage	CPU Load
1	ADCR1	0%	0.40
2	ADCR2	7%	0.95
3	ADCR3	13%	1.91
4	ADCR4	0%	0.33

cons: => there might be waits on the blocks transfer between the two nodes (cache fusion)

Room for improvement (2)

- **Identify use cases of PanDA monitor for using the resources of Active Data Guard (ADG) database of the ADCR.**

ADG is a full copy of the ADCR and refreshed asynchronously with usual latency of a second.

ADCR_ADG has the same HW spec as the ADCR, expect that it has 128 GB RAM per machine.

pros: **=> available full data replica with minimal latency to the ADCR**

=> available 4 nodes x 16 CPU cores.

=> may use parallel slave processes for sequential and faster read of large amounts of data.

cons: => application has to support two connection pools (or rely on DB links and synonym objects) and an encoded logic has to decide which DB to serve the user query depending on some criteria.

e.g. large time span (> 6 months period), analytic statement, etc..

Cross DB requests in heterogeneous architecture?

- Identify the use cases that impose the need for a hybrid DB system.
- A reliable method must be present for adding/updating the data on the second DB backend with the primary DB.
- The application has to support connections to different DB backend.
- It has to have encoded criteria for deciding which database would serve the request faster/efficient.
- Requests have to be served by the different DB in a transparent for the user way.

Questions, suggestions, discussion.