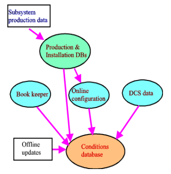
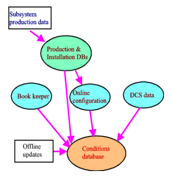
CHEP conference, Amsterdam Oct 2013



Next generation database relational solutions for ATLAS Distributed Computing

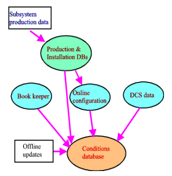
Gancho Dimitrov (CERN)
Tadashi Maeno (BNL)
Vincent Garonne (CERN)

on behalf of the ATLAS collaboration



Outline

- ATLAS databases topology
- Main ADC (**A**TLAS **D**istributed **C**omputing) systems hosted on the **A**TLAS **D**istributed **C**omputing Oracle **R**eal Application Cluster (ADCR)
- DB volumes, rates and challenges
- Data segments organization, technical solutions and specific DB performance and tuning techniques
- Development of new large scale applications
- Conclusions

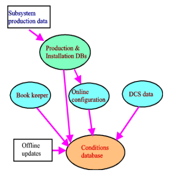


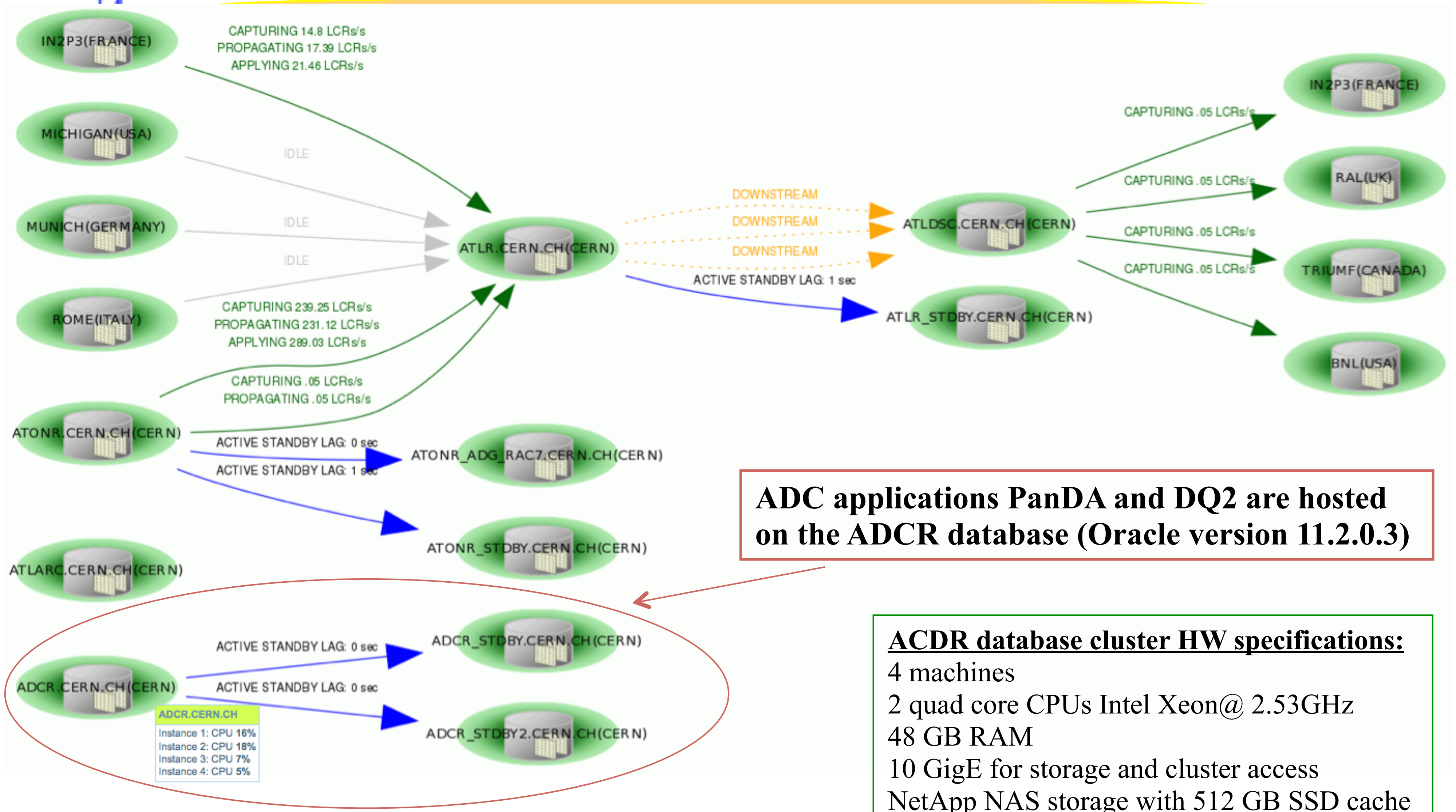
Main applications hosted on the ADCR DB

- **Production and Distributed Analysis (PanDA)** system is the ATLAS workload management system for production and user analysis jobs. Over the years it has demonstrated at a very large scale the value of automated dynamic brokering of diverse workloads across distributed computing resources.

- **Don Quijote 2 (DQ2)** is a Distributed Data Management (DDM) system with large scale data management capabilities. Currently manages more than 140 petabytes spread worldwide across 130 sites and serves more than 1000 active users.

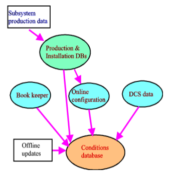
ATLAS databases topology



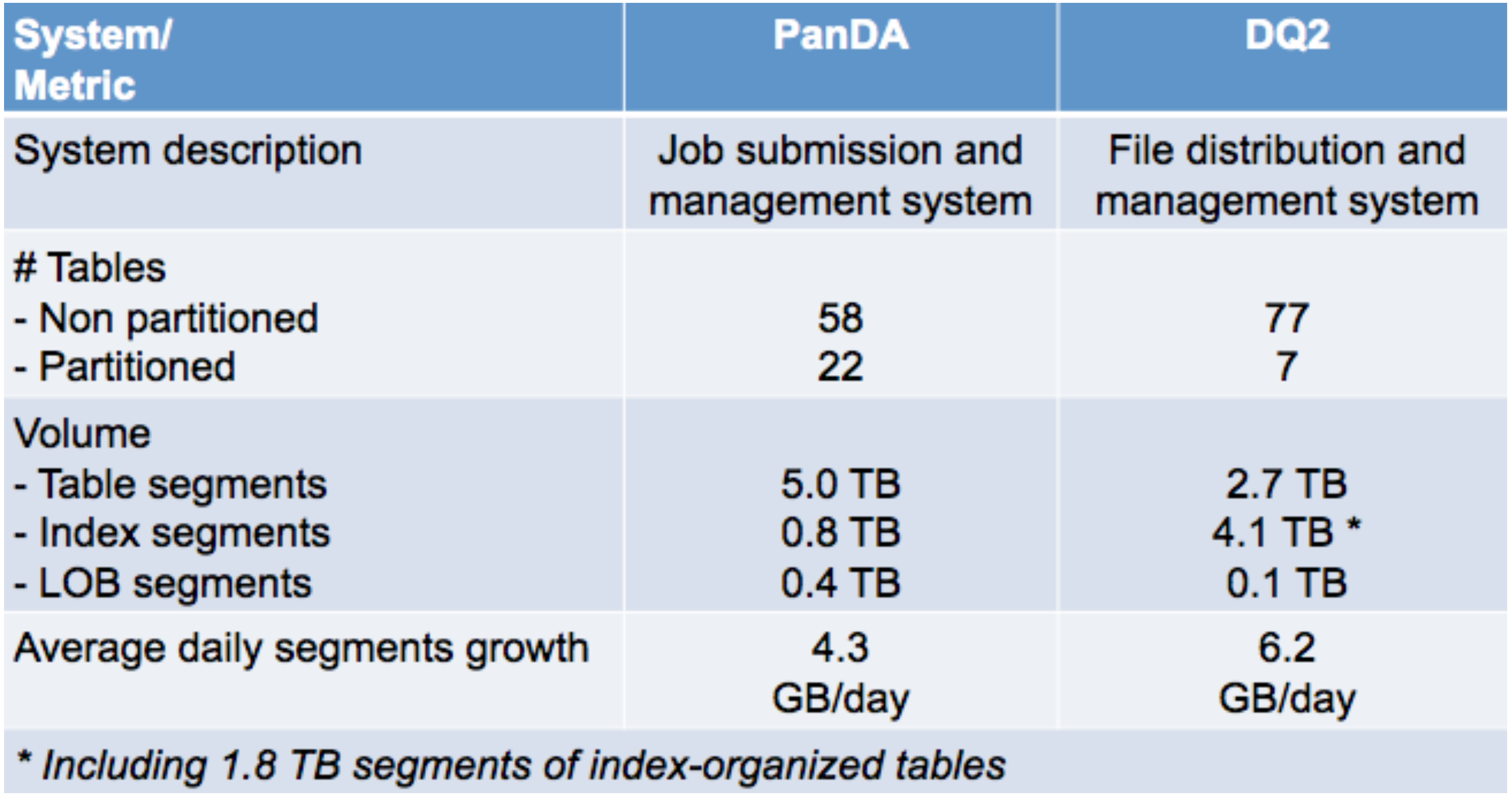


ADC applications PanDA and DQ2 are hosted on the ADCR database (Oracle version 11.2.0.3)

<u>ACDR database cluster HW specifications:</u>
4 machines
2 quad core CPUs Intel Xeon@ 2.53GHz
48 GB RAM
10 GigE for storage and cluster access
NetApp NAS storage with 512 GB SSD cache

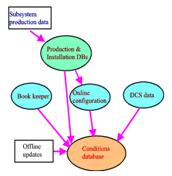


PanDA and DQ2 relational DB volumes



System/ Metric	PanDA	DQ2
System description	Job submission and management system	File distribution and management system
# Tables - Non partitioned - Partitioned	 58 22	 77 7
Volume - Table segments - Index segments - LOB segments	 5.0 TB 0.8 TB 0.4 TB	 2.7 TB 4.1 TB * 0.1 TB
Average daily segments growth	4.3 GB/day	6.2 GB/day

** Including 1.8 TB segments of index-organized tables*



PanDA system

- Originally based on a MySQL database hosted at BNL. Migrated in 2008 to Oracle relational database management system at CERN.

- **Challenges faced by PanDA and its database back end:**

  - PanDA must operate with high reliability and robustness.

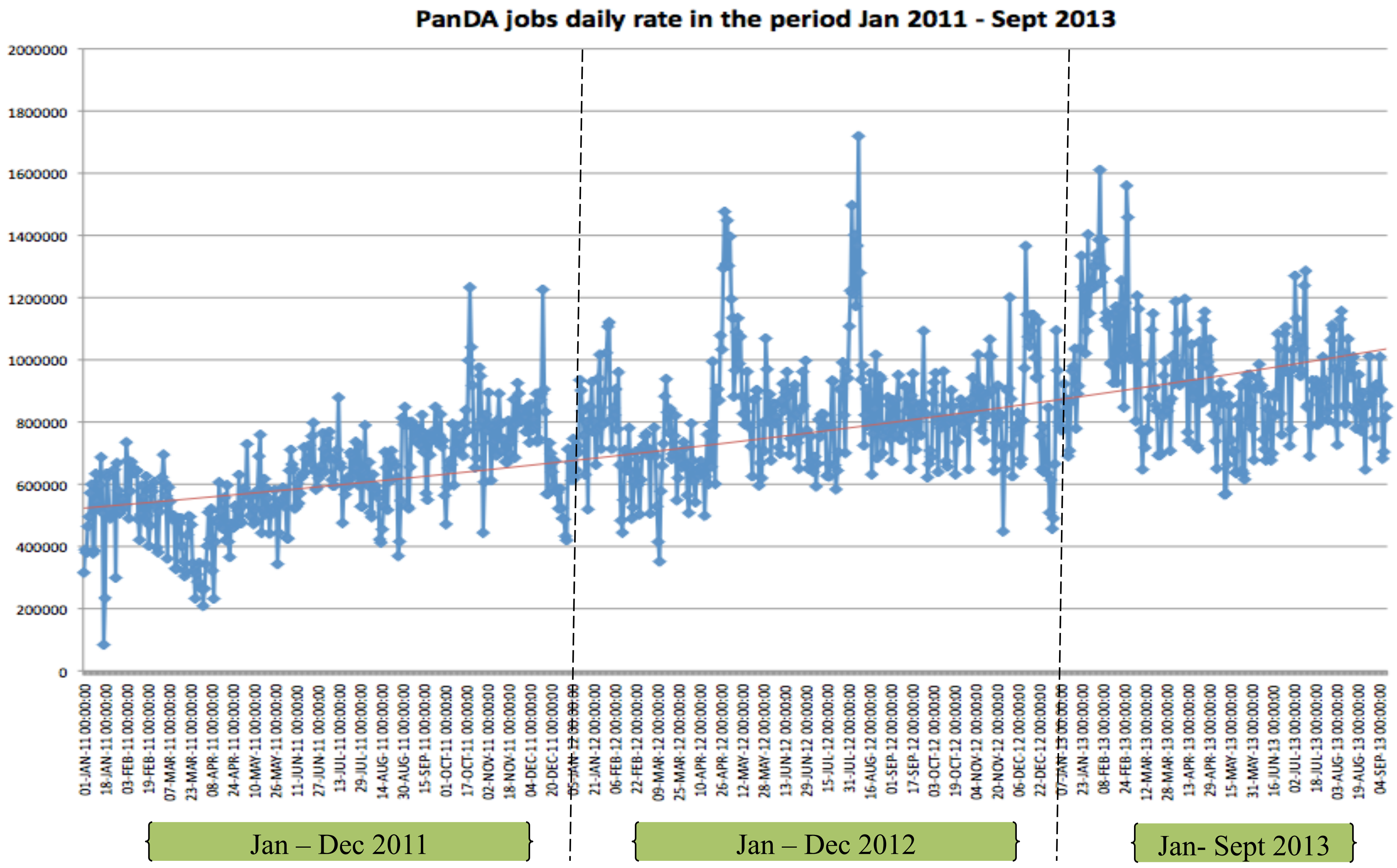
  - It has to manage millions of grid jobs daily.

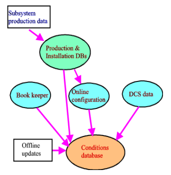
  - Changes of jobs status, site load and task progress have to be reflected on the database instantaneously. Fast data retrievals of the PanDA server and monitor are key requirements.

  - DB system must cope with spikes of user workload.

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

Trend in number of daily PanDA jobs





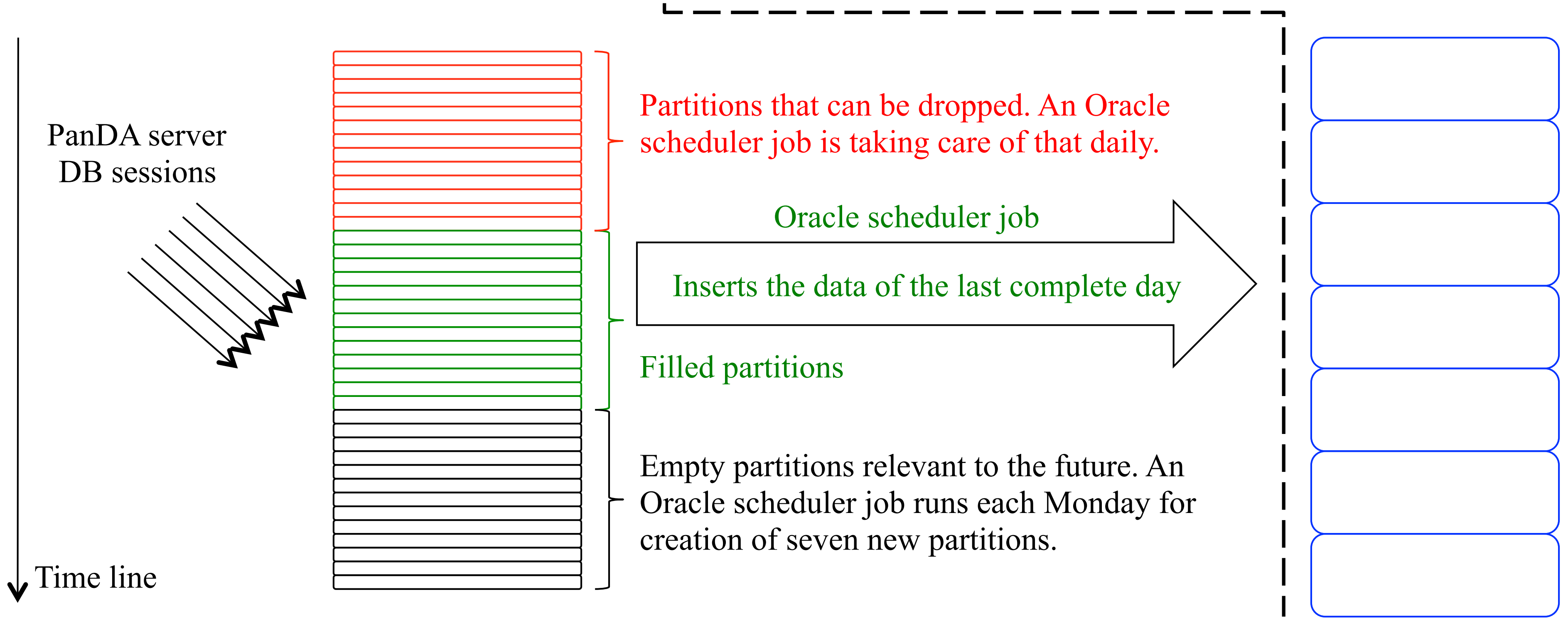
PanDA data flow and segment organization

PanDA operational data

PanDA jobs, attributes, input and output metadata are partitioned on column named 'modificationtime'. **Each partition covers a time range of a day.**

PanDA archive data

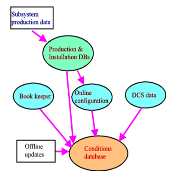
The archive data is partitioned based on the 'modificationtime'. **Certain tables have partitions of three days interval, others a time range of a month.**



Partitions that can be dropped. An Oracle scheduler job is taking care of that daily.

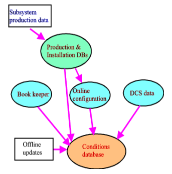
Empty partitions relevant to the future. An Oracle scheduler job runs each Monday for creation of seven new partitions.

This natural approach showed to be adequate and database resource efficient.



Advantages of the PanDA data segmentation

- **The 'operational' data is kept in a separate schema** which hosts active jobs plus finished ones of the most recent 3 days. Jobs that get status 'finished', 'failed' or 'cancelled' are moved to an archive PanDA schema.
- The PanDA jobs data copy and deletion is done on **table partition level instead on a row level** thus achieving high scalability.
- Removing the already copied data **is not IO demanding** (very little redo and does not produce undo) as this is a simple Oracle operation over a table segment and its relevant index.
- **Fragmentation in the table segments is avoided** resulting in much better space utilization and caching in the buffer pool.



Additional DB techniques used in PanDA

Automatic interval partitioning

Partition is created automatically when a user transaction imposes a need for it (e.g. user inserts a row with a timestamp for which a partition does not yet exist).
In PanDA and other ATLAS applications interval partitioning is very handy for transient type of data where **we impose a policy of agreed data sliding window**.

Result set caching

This technique was **used on well selected set of PanDA server queries** - useful in cases where data do not change often but is queried on a frequent basis.
Oracle sends back to the client a cached result if the result has not been changed meanwhile by any transaction, thus improving the performance and scalability.

Data aggregation for fast result delivery

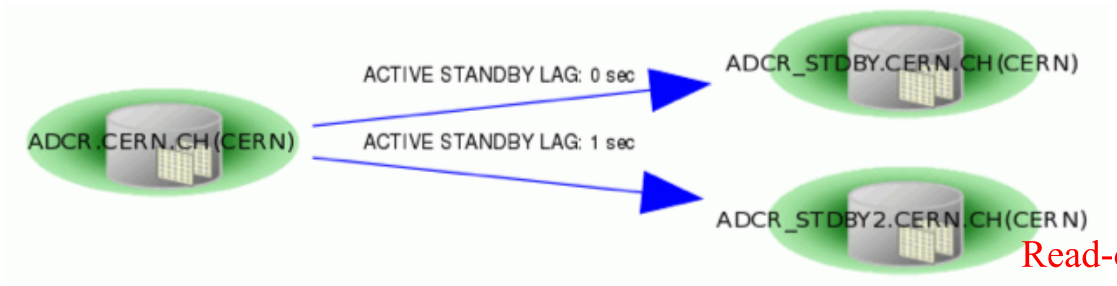
A table with aggregated stats is re-populated by a PL/SQL procedure on an interval of two minutes by an Oracle scheduler job (usual elapsed time 1-2 sec).

Customised table settings for the Oracle stats gathering

Oracle spends time and resources on collecting statistics only on partitions which are transactional active and computes the global table statistics using the previously ones in an incremental way.

Potential use of ADG from PanDA monitor

- PanDA complete archive now hosts information of more than 900 million jobs – all jobs since the system start in 2006.
- **ADCR database has two standby databases:**
  - Data Guard for disaster recovery and backup offloading.
  - Active Data Guard for read-only replica.


ADCR.CERN.CH (CERN)
ACTIVE STANDBY LAG: 0 sec
ACTIVE STANDBY LAG: 1 sec
ADCR_STDBY.CERN.CH (CERN)
ADCR_STDBY2.CERN.CH (CERN)

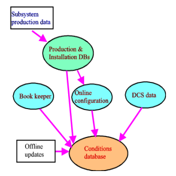
Read-only replica

- **PanDA monitor can benefit from the Active Data Guard (ADG) resources**

  => An option is to sustain two connection pools: one to the primary database ADCR and one to the ADCR's ADG.

  The idea is queries that span on time ranges larger than certain threshold to be resolved from the ADG where we can afford several parallel slave processes per user query.

  => Second option is to connect to the ADG only and fully rely on it.



JEDI (a component of PanDA)

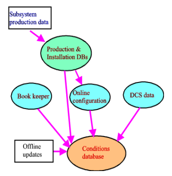
- JEDI (**J**ob **E**xecution and **D**efinition **I**nterface) is a new component of the PanDA server which **dynamically defines jobs from a task definition**. The main goal is to make PanDA task-oriented.

- **Tables of the new JEDI complement the existing PanDA tables**.

  New relations between the existing PanDA and the new set of JEDI tables are put in place.

  The transition from the current PanDA database schema to a new one was done in a **transparent way** in July 2013.

- Data segmenting is based on a range partitioning on the JEDI's TASKID column with interval of 100000 IDs (tasks) on six of the JEDI tables, thus **achieving uniform data partitioning**.



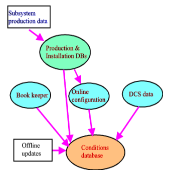
JEDI DB objects – physical implementation

- The JEDI data segments are placed on **dedicated Oracle tablespace** (data file) separate from existing PanDA tables.

- Thanks to the new CERN license agreement with Oracle, now **we take advantage of the Oracle advanced compression features** : compression of data within a data block (8KB) while application does row inserts or updates.

  Depending on the table’s data, the achieved **compression ratio varies between factor of 1.4 and 3.1.**

- According to the current analysis and production submission tasks rates of 10K to 15K tasks per day, the estimate for the JEDI needed disk space is in the range 2 to 3 TB per year (not considering the gain from the compression).



Rucio system

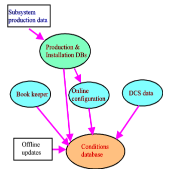
- **The Rucio project (rucio.cern.ch) is the next generation Distributed Data Management (DDM) system** for allowing the ATLAS collaboration to manage large volumes of data (tens of petabytes per year), both taken by the detector as well as generated in the ATLAS distributed computing system.
- **Challenges faced by Rucio and its back end database:**

  - It is a bookkeeping system that represents the current state and placement of production and user data files and datasets over the ATLAS Grid sites.

  - Because of the dynamic placement of files and datasets, large fraction of the data is transient and has to be managed properly.

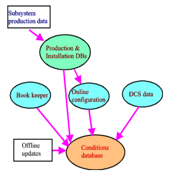
  - Fast free pattern searches based on file and dataset names have to be supported.

  - Each Rucio action on the Grid has to be recorded for traceability and analysis purposes.



Rucio development

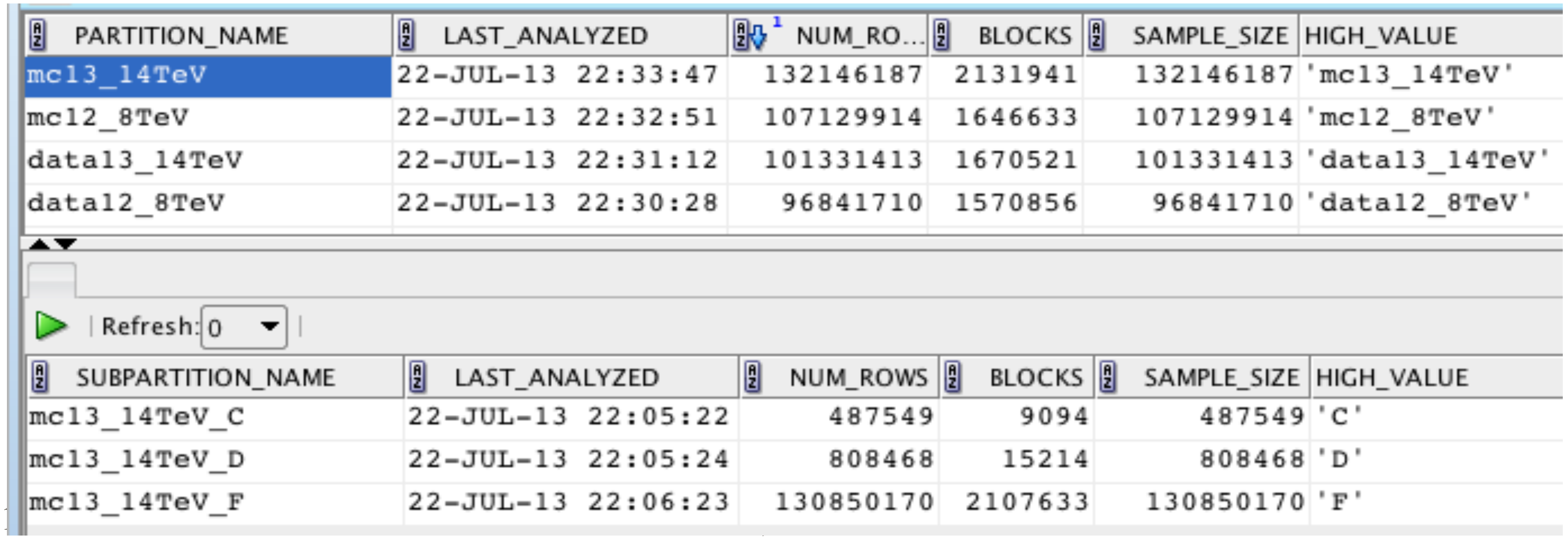
- During development of the Rucio database schema we **focused on simplicity, manageability and scalability** aspects.
- **DB objects (tables, constraints, indices, triggers, sequences, etc) are named based on an agreed naming convention** for an easy identification of the object role and relation to the others.
- **Tables are divided in four main categories and spread on several Oracle tablespaces**: attribute data, fact data, highly transient data and historical data.
- Special **check constraints logic enforce important policies**.
- Following the application logic, **the most natural choice for organizing Rucio data is by using the Oracle's list partitioning approach** (each partition is bound to an explicit column value).



Rucio’s physical data organization

- Evaluation of different data segmentation solutions **showed that list partitioning plus list sub-partitioning is the most appropriate one** for most of the Rucio use cases – e.g. DIDS table (Data Identifiers) being partitioned on the ‘scope’ and sub-partitioned on the ‘did_type’ columns.
- **Sub-partitioning the data on three pieces (‘D’ - datasets, ‘F’- files, ‘C’ - containers) provides an advantage of storing the data physically in separate segments**.

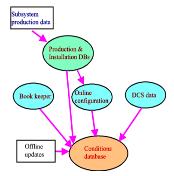
  That proved to be beneficial for one of the most wanted features – *free pattern searches on data identifier names plus retrieval of any relevant attributes.*



PARTITION_NAME	LAST_ANALYZED	NUM_RO...	BLOCKS	SAMPLE_SIZE	HIGH_VALUE
mc13_14TeV	22-JUL-13 22:33:47	132146187	2131941	132146187	'mc13_14TeV'
mc12_8TeV	22-JUL-13 22:32:51	107129914	1646633	107129914	'mc12_8TeV'
data13_14TeV	22-JUL-13 22:31:12	101331413	1670521	101331413	'data13_14TeV'
data12_8TeV	22-JUL-13 22:30:28	96841710	1570856	96841710	'data12_8TeV'

Refresh: 0

SUBPARTITION_NAME	LAST_ANALYZED	NUM_ROWS	BLOCKS	SAMPLE_SIZE	HIGH_VALUE
mc13_14TeV_C	22-JUL-13 22:05:22	487549	9094	487549	'C'
mc13_14TeV_D	22-JUL-13 22:05:24	808468	15214	808468	'D'
mc13_14TeV_F	22-JUL-13 22:06:23	130850170	2107633	130850170	'F'



Oracle list partitioning: an operational challenge

List partitioning is appropriate for Rucio, but it is rather static as partitions must be manually pre-created for each new partition key value.

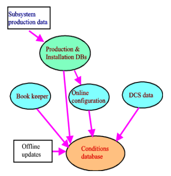
This is an operational burden as such values (Rucio's scope attribute) are created dynamically.

Our solution:

Automatic partition creation in all relevant tables takes place whenever a new partition key value is inserted into a dedicated dimensional table.

- ✓ An 'after insert' row level trigger gets fired and executes a PL/SQL procedure responsible for ALTER TABLE … ADD PARTITION action (the procedure has measures against SQL injection)
- ✓ Logic for handling ORA-00054 "resource busy" error (happens when there are uncommitted transactions on the table)
- ✓ Each partition creation action is logged into a dedicated logging table

Conclusions



- **Versatile database features and performance tuning techniques applied for the success of the ADC large scale applications PanDA, DQ2 and Rucio.**

- **JEDI's dynamic job definition capabilities are essential to support ATLAS production and analysis use cases in LHC Run2**. The database is expected to scale in line with the increasing PanDA user workload.

- **Rucio system is due to be fully deployed in production in 2014 bringing new capabilities and power to the end users.**