

ADC technical interchange meeting Tokyo, May 2013





Database aspects of ATLAS distributed computing

Gancho Dimitrov (CERN)

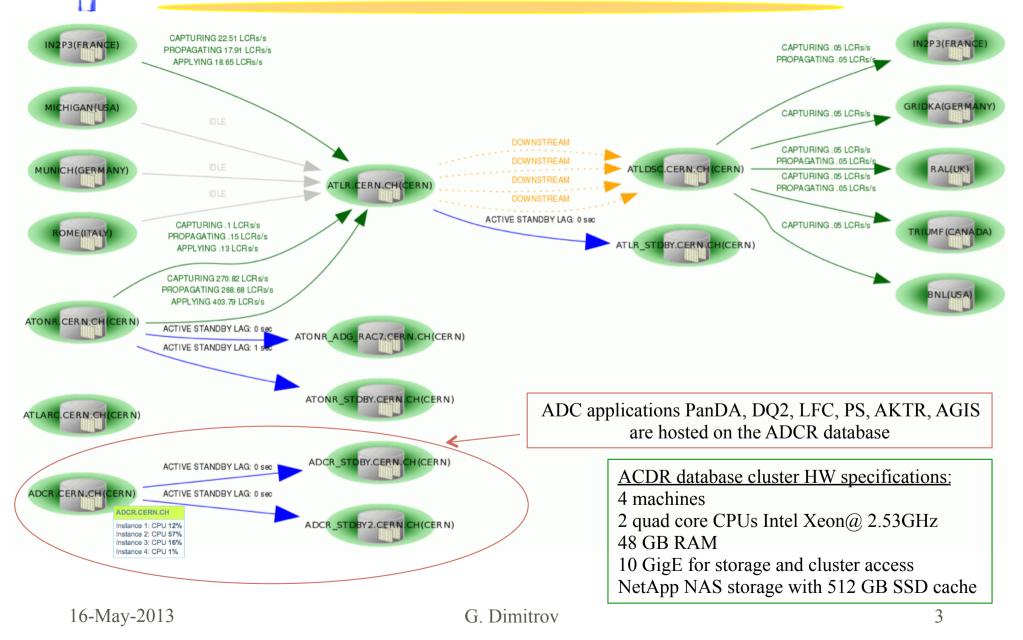




- Current ATLAS database topology, HW specifications, metrics
- PanDA accounts, roles and privileges organization
- PanDA data flow and management, volume and trends, improvements
- New developments
 - JEDI (Job Execution and Definition Interface)
 - Rucio data management system
- Newer hardware specs for the ADCR database
- Conclusions











Owner account	Write privileges granted to	Read privileges granted to
ATLAS_PANDA	ATLAS_PANDA_WRITEROLE	ATLAS_PANDA_READROLE
ATLAS_PANDAMETA	ATLAS_PANDAMETA_WRITEROLE	ATLAS_PANDA_READROLE
ATLAS_PANDAARCH	ATLAS_PANDA	ATLAS_PANDA_READROLE
ATLAS_DEFT	ATLAS_DEFT_W	ATLAS_PANDA_READROLE

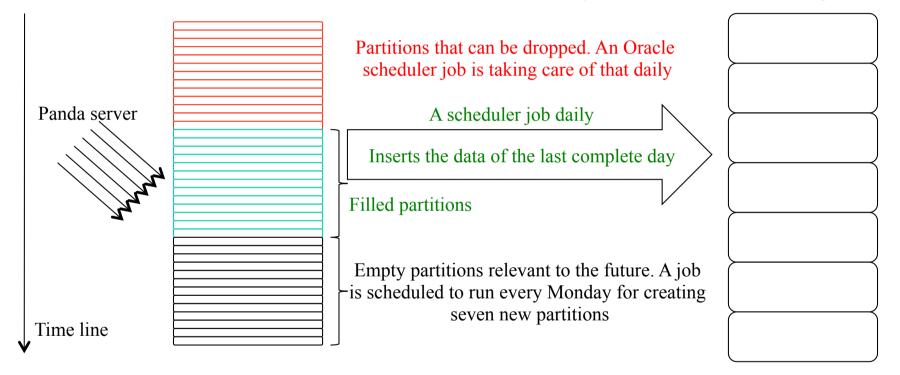
- Panda server is allowed to modify the PANDA, PANDAMETA and PANDAARCH
- PanDA monitor is allowed to modify only the PANDAMETA data
- DEfT (Database Engine for Tasks) can modify only its own tables via direct privileges (PanDA gets a direct write privilege in a single DEFT table)
- All owner accounts grant select privilege on their objects to the ATLAS_PANDA_READROLE
- All application writer and reader accounts get read privilege on all objects by granting the ATLAS_PANDA_READROLE to them



PanDA data organization



ATLAS_PANDA JOB, PARAMS, META and FILES tables: partitioned on a 'modificationtime' column. **Each partition covers a time range of a day** ATLAS_PANDAARCH archive tables partitioned on the 'modificationtime' column. Some table have defined partitions each covering three days window, others a time range of a month

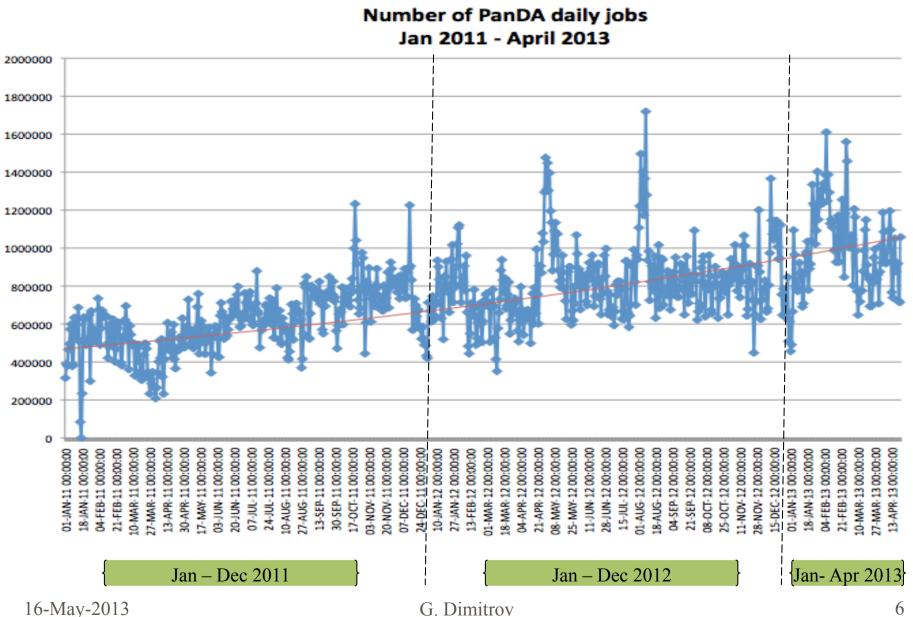


Certain PanDA tables have defined data sliding window of three days, others 30 days. Data removal on partition level happens only after data being copied to the PANDAARCH schema. This natural approach showed to be adequate and not resource demanding !



Trend in number of daily PanDA jobs



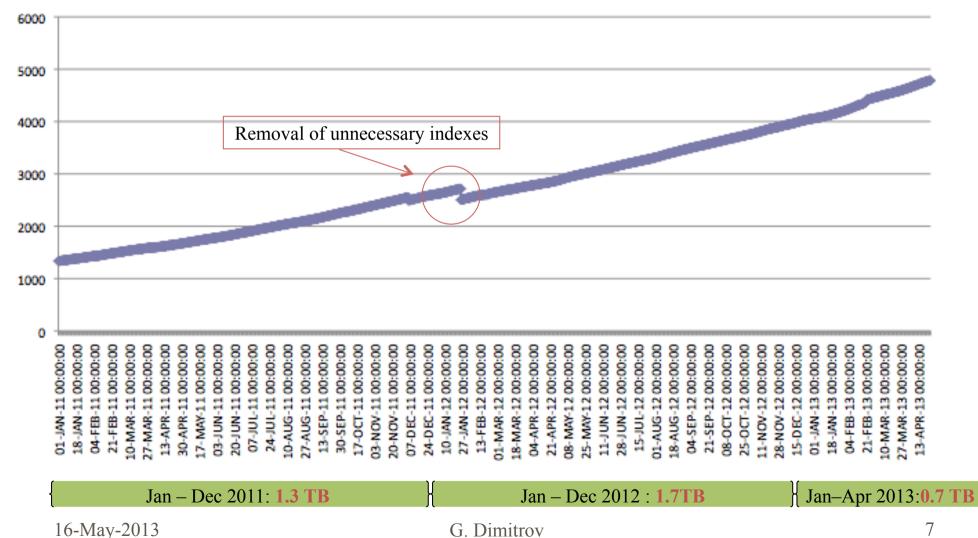




Trend in PanDA segments growth



PanDA segments growth in GB Jan 2011 - April 2013







- Despite the increased activity on the grid, using several tuning techniques the server resource usage stayed in the low range: e.g. CPU usage in the range 20-30%
- Thorough studies on the WHERE clauses of the PanDA server queries resulted in:
 - revision of the indexes: removal or replacement with more approriate multicolumn ones
 - weekly index maintenance (rebuids triggered by a scheduler job) on the tables with high transaction activity
 - OWNER
 JOB_NAME
 JOB_CLASS
 RUN_DURATION
 LAST_START_DATE
 LAST_RUN_DURATION
 LAST_STATUS
 STATE
 NEXT_RUN_DATE
 REPEAT_INTERVAL

 ATLAS_PANDA
 PANDA_TAB_INDICES_REBUILD
 PANDA_JOB_CLASS
 04-MAR-2013 11:00
 0 00:00:48.0
 SUCCEEDED
 SCHEDULED
 11-MAR-2013 11:00
 FREQ=WEEKLY; BYDAY=MON
 - queries tuning
 - an auxiliary table for the mapping Panda ID <=> Modification time
- In the light of the Rucio and the DQ2, PanDA now works with files and datasets with defined scope





- PanDA full archive now hosts information of 800 million jobs all jobs since the job system start in 2006
- Studies on the WHERE clauses of the PanDA monitor queries resulted in:

- revision of the indexes: replacement with more approriate multi-column ones so that less tables blocks get accessed

- dynamic re-definition of different time range views in order to protect the large PanDA archive tables from time range unconstrained queries and to avoid time comparison on row level.

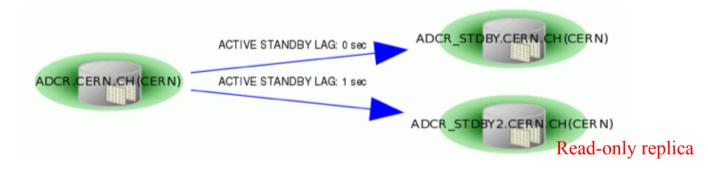
The views comprise set of partitions based on ranges of 7, 15, 30, 60, 90, 180 and 365 days. These views are shifting windows with defined ranges and the underlying partition list is updated daily automatically via a scheduler job.

VIEW_NAME	STATUS	LAST_DDL_TIME
JOBSARCHVIEW_7DAYS	VALID	26-APR-13 12:00:09
JOBSARCHVIEW_30DAYS	VALID	26-APR-13 12:00:11
JOBSARCHVIEW_15DAYS	VALID	26-APR-13 12:00:11
JOBSARCHVIEW_60DAYS	VALID	26-APR-13 12:00:11
JOBSARCHVIEW_90DAYS	VALID	26-APR-13 12:00:11
JOBSARCHVIEW_180DAYS	VALID	26-APR-13 12:00:11
JOBSARCHVIEW_365DAYS	VALID	26-APR-13 12:00:10
16-May-2013	G. Dimitrov	9





- ADCR database has two standby databases:
 - Data Guard for disaster recovery and backup offloading
 - Active Data Guard (ADCR_ADG) for read-only replica



- PanDA monitor can benefit from the Active Data Guard (ADG) resources It is planned that PanDA monitor sustains two connection pools:
 - to the primary database ADCR

- to 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 paralell slave processes per user query.





- JEDI is a new component of the PanDA server which dynamically defines jobs from a task definition. The main goal is to make PanDA taskoriented.
- Tables of initial relational model of the new JEDI schema (documented by T. Maeno) complement the existing PanDA tables on the INTR database

• Ongoing activities :

- understand the new data flow, requirements, access patterns
- studies for the best possible data organization (partitioning) from manageability and performance point of view.

- address the requirement of storing information on event level for keeping track of the active jobs' progress. That data will be transient, but the writing and reading of it must be highly optimised.

- get to the most appropriate physical implelentaion of the agreed relational model

- tests with representative data volume



JEDI database relational schema



ATLAS PANDAJEDI DATASET CONTENTS	ATLAS_PANDA.JEDI_OUTPUT_TEMPLATE				r	ATLAS_PANDA.JOBSACTIVE4
UF * TASKID NUMBER (11)	PF * TASKID NUMBER (11) PF * DATASETID NUMBER (11)				P * PANDAID	NUMBER (11)
UF * DATASETID NUMBER (11) P * FILEID NUMBER (11)	P * OUTTEMPID NUMBER (11)	ATLAS	PANDA.JEDI_DATASETS		 JOBDEFINITIONID SCHEDULERID 	NUMBER (11) VARCHAR2 (128 BYTE)
* CREATIONDATE DATE	 FILENAMETEMPLATE VARCHAR2 (256 BYTE) MAXSERIALNR NUMBER (7) 	PF * TASKID	NUMBER (11)		PILOTID	VARCHAR2 (200 BYTE)
LASTATTEMPTTIME DATE * LFN VARCHAR2 (256 BYTE)	SERIALNR NUMBER (7)	P * DATASETID U * DATASETNAME	NUMBER (11) VARCHAR2 (255 BYTE)		 CREATIONTIME CREATIONHOST 	DATE VARCHAR2 (128 BYTE)
GUID VARCHAR2 (64 BYTE)	SOURCENAME VARCHAR2 (256 BYTE) STREAMNAME VARCHAR2 (20 BYTE)	TYPE	VARCHAR2 (20 BYTE)		* MODIFICATIONTIME	DATE
* TYPE VARCHAR2 (20 BYTE) * STATUS VARCHAR2 (64 BYTE)	OUTTYPE VARCHAR2 (20 BYTE)	* CREATIONTIME * MODIFICATIONTIME	DATE DATE		MODIFICATIONHOST ATLASRELEASE	VARCHAR2 (128 BYTE) VARCHAR2 (64 BYTE)
FSIZE NUMBER (11) CHECKSUM VARCHAR2 (36 BYTE)	im JEDI_OUTPUT_TEMPL_PK (TASKID, DATASETID, OUTTEMPID)	VO CLOUD	VARCHAR2 (16 BYTE) VARCHAR2 (10 BYTE)		TRANSFORMATION HOMEPACKAGE	VARCHAR2 (250 BYTE) VARCHAR2 (80 BYTE)
SCOPE VARCHAR2 (30 BYTE)	¥	SITE	VARCHAR2 (60 BYTE)		PRODSERIESLABEL	VARCHAR2 (20 BYTE)
ATTEMPTNR NUMBER (3) MAXATTEMPT NUMBER (3)		MASTERID PROVENANCEID	NUMBER (11) NUMBER (11)		PRODSOURCELABEL PRODUSERID	VARCHAR2 (20 BYTE) VARCHAR2 (250 BYTE)
NEVENTS NUMBER (10) KEEPTRACK NUMBER (1)		CONTAINERNAME	VARCHAR2 (132 BYTE) VARCHAR2 (20 BYTE)		ASSIGNEDPRIORITY CURRENTPRIORITY	NUMBER (9) NUMBER (9)
STARTEVENT NUMBER (10)	D ATLAS_PANDA.JEDI_TASKS	STATE	VARCHAR2 (20 BYTE)		* ATTEMPTNR	NUMBER (3)
ENDEVENT NUMBER (10)	P * TASKID NUMBER (11) U * TASKNAME VARCHAR2 (128 BYTE)	STATECHECKTIME STATECHECKEXPIRAT	DATE TION DATE		 MAXATTEMPT JOBSTATUS 	NUMBER (3) VARCHAR2 (15 BYTE)
🖕 JEDI_DATASET_CONTENTS_PK (FILEID)	* STATUS VARCHAR2 (64 BYTE)	FROZENTIME	DATE NUMBER (5)		JOBNAME MAXCPUCOUNT	VARCHAR2 (256 BYTE) NUMBER (10)
JEDI_DATASET_CONTENTS_UK (TASKID, DATASETID, FILEID)	* USERNAME VARCHAR2 (128 BYTE) * CREATIONDATE DATE	NFILES NFILESTOBEUSED	NUMBER (5) NUMBER (5)		MAXCPUCOUNT	VARCHAR2 (32 BYTE)
T	* MODIFICATIONTIME DATE STARTTIME DATE	N FILESUSED NEVENTS	NUMBER (5) NUMBER (11)		 MAXDISKCOUNT MAXDISKUNIT 	NUMBER (10) CHAR (4 BYTE)
	ENDTIME DATE	NEVENTSTOBEUSED	NUMBER (10) NUMBER (10)		IPCONNECTIVITY	CHAR (5 BYTE)
	FROZENTIME DATE PRODSOURCELABEL VARCHAR2 (20 BYTE)	NEVENTSUSED LOCKEDBY	VARCHAR2 (40 BYTE)		 MINRAMCOUNT MINRAMUNIT 	NUMBER (10) CHAR (2 BYTE)
F ATLAS_PANDA.FILESTABLE4	WORKINGGROUP VARCHAR2 (32 BYTE)	LOCKEDTIME NFILESFINISHED	DATE NUMBER (5)		STARTTIME ENDTIME	DATE
P * ROW_ID NUMBER (11)	VO VARCHAR2 (16 BYTE) CORECOUNT NUMBER (9)	NFILESFAILED	NUMBER (5)		 CPUCONSUMPTIONTIME 	NUMBER (20)
* PANDAID NUMBER (11) P * MODIFICATIONTIME DATE	TASKTYPE VARCHAR2 (64 BYTE) PROCESSINGTYPE VARCHAR2 (64 BYTE)	SPLITRULE STREAMNAME	VARCHAR2 (100 BYTE) VARCHAR2 (20 BYTE)		CPUCONSUMPTIONUNIT COMMANDTOPILOT	VARCHAR2 (128 BYTE) VARCHAR2 (250 BYTE)
GUID VARCHAR2 (64 BYTE) LFN VARCHAR2 (256 BYTE)	TASKPRIORITY NUMBER (9)	STORAGETOKEN DESTINATION	VARCHAR2 (60 BYTE) VARCHAR2 (60 BYTE)		TRANSEXITCODE PILOTERRORCODE	VARCHAR2 (128 BYTE) NUMBER (7)
TYPE VARCHAR2 (20 BYTE)	CURRENTPRIORITY NUMBER (9) ARCHITECTURE VARCHAR2 (256 BYTE)	* FILEID	NUMBER (11)		PILOTERRORDIAG	VARCHAR2 (500 BYTE)
DATASET VARCHAR2 (255 CHAR) STATUS VARCHAR2 (64 BYTE)	TRANSHOME VARCHAR2 (128 BYTE) TRANSPATH VARCHAR2 (128 BYTE)	MASTERTID	NUMBER (11)		 EXEERRORCODE EXEERRORDIAG 	NUMBER (7) VARCHAR2 (500 BYTE)
PRODDBLOCK VARCHAR2 (255 CHAR) PRODDBLOCKTOKEN VARCHAR2 (250 BYTE)	LOCKEDBY VARCHAR2 (40 BYTE) LOCKEDTIME DATE		v		 SUPERRORCODE SUPERRORDIAG 	NUMBER (7) VARCHAR2 (250 BYTE)
DISPATCHDBLOCK VARCHAR2 (255 CHAR)	TERMCONDITION VARCHAR2 (100 BYTE)	+			* DDMERRORCODE	NUMBER (7)
DISPATCHDBLOCKTOKEN VARCHAR2 (250 BYTE) DESTINATIONDBLOCK VARCHAR2 (255 CHAR)	SPLITRULE VARCHAR2 (100 BYTE) WALLTIME NUMBER (9)	+			 DDMERRORDIAG BROKERAGEERRORCODE 	VARCHAR2 (500 BYTE) NUMBER (7)
DESTINATIONDBLOCKTOKEN VARCHAR2 (250 BYTE) DESTINATIONSE VARCHAR2 (250 BYTE)	WALLTIMEUNIT VARCHAR2 (32 BYTE) OUTDISKCOUNT NUMBER (9)				BROKERAGEERRORDIAG	VARCHAR2 (250 BYTE) NUMBER (7)
* FSIZE NUMBER (11)	OUTDISKUNIT VARCHAR2 (32 BYTE)				JOBDISPATCHERERRORDIAG TASKBUFFERERRORCODE	VARCHAR2 (250 BYTE) NUMBER (7)
CHECKSUM VARCHAR2 (36 BYTE)	WORKDISKCOUNT NUMBER (9) WORKDISKUNIT VARCHAR2 (32 BYTE)		EDI_JOBPARAMS_TEMPLATE		TASKBUFFERERRORDIAG	VARCHAR2 (300 BYTE)
SCOPE VARCHAR2 (30 BYTE) F FILEID NUMBER (11)	RAMCOUNT NUMBER (9) RAMUNIT VARCHAR2 (32 BYTE)	PF * TASKID JOBPARAMSTEMI	NUMBER (11) PLATE CLOB		COMPUTINGSITE COMPUTINGELEMENT	VARCHAR2 (128 BYTE) VARCHAR2 (128 BYTE)
ATTEMPTNR NUMBER (3)	IOINTENSITY NUMBER (9)	Go JEDI JOBPARAMS	TEMPLATE_PK (TASKID)		JOBPARAMETERS METADATA	CLOB
ART FILESTABLE4_PK (ROW_ID, MODIFICATIONTIME)	IOINTENSITYUNIT VARCHAR2 (32 BYTE) F WORKQUEUE ID NUMBER (5)				PRODDBLOCK	VARCHAR2 (255 CHAR)
	PROGRESS NUMBER (3) FAILURERATE NUMBER (3)	ATLAS_PANDA.JE			DISPATCHDBLOCK DESTINATIONDBLOCK	VARCHAR2 (255 CHAR) VARCHAR2 (255 CHAR)
FILESTABLE4_PANDAID_IDX (PANDAID)	TRANSUSES VARCHAR2 (64 BYTE)		NUMBER (11) CLOB		DESTINATIONSE • NEVENTS	VARCHAR2 (250 BYTE) NUMBER (10)
 FILESTABLE4_DESTDBLOCK_IDX (DESTINATIONDBLOCK) FILESTABLE4_DISPDBLOCK_IDX (DISPATCHDBLOCK) 	ERRORDIALOG VARCHAR2 (255 BYTE) COMMAND VARCHAR2 (40 BYTE)	200 JEDI_TASKPARAMS	TASKID_PK (TASKID)		GRID	VARCHAR2 (50 BYTE)
 FILESTABLE4_LFN_IDX (LFN) FILESTABLE4_DSETDESTBLPID_IDX (DATASET, TYPE, DESTINATIONDBLOCK, PANDAID) 					CLOUD CPUCONVERSION	VARCHAR2 (50 BYTE) NUMBER (9,4)
PART_FILESTABLE4_PK (ROW_ID, MODIFICATIONTIME)			ATLAS_PANDA.JEDI_WORK_QUEUE		SOURCESITE DESTINATIONSITE	VARCHAR2 (36 BYTE) VARCHAR2 (36 BYTE)
FILESTABLE4_FILEID_INDX (FILEID)	JEDI_TASKS_PK (TASKID) JEDI_TASKNAME_UK (TASKNAME)		UMBER (5) ARCHAR2 (16 BYTE)		TRANSFERTYPE	VARCHAR2 (10 BYTE)
			ARCHAR2 (16 BYTE) ARCHAR2 (16 BYTE)		TASKID CMTCONFIG	NUMBER (11) VARCHAR2 (250 BYTE)
		STATUS V	ARCHAR2 (64 BYTE)	+ - +	STATECHANGETIME PRODDBUPDATETIME	DATE
Under test is approach of TASK <	↓ JEDI_TASKS_PK (TASKID)		UMBER (5) UMBER (1)		LOCKEDBY	VARCHAR2 (128 BYTE)
table partitioning on 'status'		QUEUE_SHARE N	UMBER (3) UMBER (3)	←	RELOCATIONFLAG	NUMBER (1) NUMBER (11)
lable partitioning on status		CRITERIA V	ARCHAR2 (256 BYTE)	4	VO PILOTTIMING	VARCHAR2 (16 BYTE) VARCHAR2 (100 BYTE)
column and most of its child tables	*		ARCHAR2 (256 BYTE)		WORKINGGROUP	VARCHAR2 (20 BYTE)
	ATLAS_PANDA.JOBS_PROGRESS_TRACKING	JEDI_WORK_QUEUEID	D_PK (QUEUE_ID) NAMETYPEVO_UK (QUEUE_NAME, QUEUE_TYPE, VO)	4	PROCESSINGTYPE PRODUSERNAME	VARCHAR2 (64 BYTE) VARCHAR2 (60 CHAR)
partitioned by reference based on	PF * TASKID NUMBER (11) P * PANDAID NUMBER (11)	JEDI_WORK_QUEUEID			NINPUTFILES COUNTRYGROUP	NUMBER (5) VARCHAR2 (20 BYTE)
	P * FILEID NUMBER (11) P * JOB PROCESSID NUMBER (10)		NAMETYPEVO_UK (QUEUE_NAME, QUEUE_TYPE, VO)		BATCHID	VARCHAR2 (20 BTTE) VARCHAR2 (80 BYTE) NUMBER (11)
task ID *	DEF_MIN_EVENTID NUMBER (10)				SPECIALHANDLING	VARCHAR2 (80 BYTE)
(partitioning in uniform way)	DEF_MAX_EVENTID NUMBER (10) PROCESSED_UPTO_EVENTID NUMBER (10)				JOBSETID CORECOUNT	NUMBER (11) NUMBER (3)
(partitioning in uniform way)	JOB_PROGRESS_TRACKING_PK (PANDAID, FILEID, JOB_PROCESSID)		NDA.JOBSDEBUG		NINPUTDATAFILES	NUMBER (5) VARCHAR2 (32 BYTE)
	TASK_PROGRESS_TRACK_PK (TASKID, PANDAID, FILEID, JOB_PROCESS	D) PF * PANDAID NUM STDOUT VAR	ABER (11) CHAR2 (2048 BYTE)		INPUTFILEPROJECT	VARCHAR2 (64 BYTE)
		🛥 JOBSDEBUG_PANDA	ID_PK (PANDAID)		INPUTFILEBYTES NOUTPUTDATAFILES	NUMBER (11) NUMBER (5)
					OUTPUTFILEBYTES JOBMETRICS	NUMBER (11) VARCHAR2 (500 BYTE)
					WORKQUEUE_ID	NUMBER (5)



- The idea is the transition from the current PanDA server to the new one with the DB backend objects to be transparent to the users.
- JEDI tables are complementary to the existing PanDA tables. The current schema and PANDA => PANDAARCH data copying should not change much.
- However the relations between the existing and the new set of tables have to exist. In particular:

- Relation between JEDI's *Task* and PanDa's *Job* by having a foreign key in all JOBS* tables to the JEDI_TASKS table

 Relation between JEDI's *Work queue* (for different shares of workload) and PanDa's *Job* by having a foreign key in all JOBS* tables to the JEDI_WORK_QUEUE table

- Relation between JEDI's *Task, Dataset and File* (new seq. ID) and PanDA's *Job* processing the file (or fraction of it) by having a foreign key in the FILESTABLE4 table to the JEDI_DATASET_CONTENTS table

(when a task tries a file multiple times, there are multiple rows in PanDA's FILESTABLE4 while there is only one parent row in the JEDI's DATASET_CONTENTS table)





- Step 1: The ATLAS_PANDA schema and data exported from production and imported on the INTR testbed database – done
- Step 2: New set of JEDI tables created into the imported schema. Alter tables where necessary. Set relations between the existing tables and the new ones, enforce integrity when possible – done
- Step 3: DEfT (Database Engine for Tasks) schema validation. Test JEDI and DEfT interaction and access rights.
- Step 4: Validate access patterns to the new objects. Studies on the efficiency of the chosen partitioning and index strategy. Agreement on data archiving policy to be achieved.
- Step 5: If needed introduce changes to the PANDA => PANDAARCH data flow. Activate the scheduler jobs responsible for the data copying and maintenance of the PANDA sliding window.
- Step 6: Test with representative workload and repeat 2,3,4,5 until getting to an acceptable state.
- <u>Step 7:</u> Import the PANDA, PANDAARCH, PANDAMETA data from the production. Repeat step 2 to validate again the SQL script. Perform preproduction test with all PanDA, JEDI, DEfT components. 16-May-2013 G. Dimitrov 14





Rucio is the new ATLAS file management system meant as successor of DQ2. The performed validation on the first Rucio relational schema initially created by the DDM team resulted in :

- DB objects (tables, constraints, indices, triggers, sequences) are named based on an agreed naming convention. The purpose is to have easy way to identify object role and relation with the others.

- studies of different partitioning techniques. A promising one is the list partitioning based on a virtual column "scope" concatenated with "data type" (file, dataset, container, del_file, del_dataset, del_container)

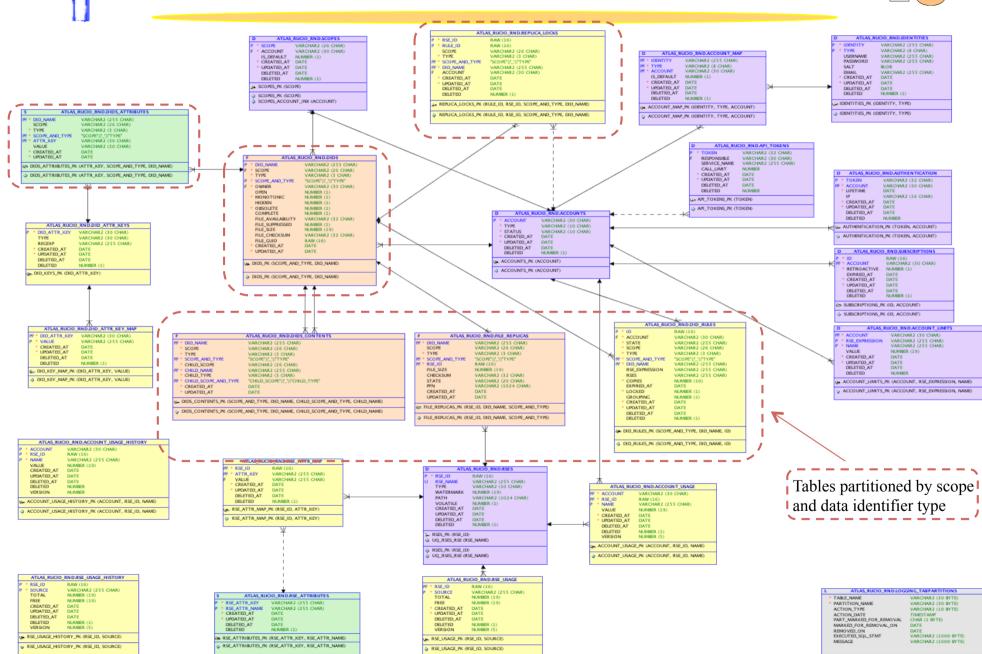
A home made logic of automatic partition creation tested and showed to be reliable.

- special check constraints logic for enforcing important policies

- explored techniques for guaranteeing any object name uniqueness within each scope for the full lifetime of the system
- test with representative data volume is needed



New development : Rucio DB relational schema







In Q4 of 2013 the ADCR hardware will be replaced by a newer one:

=> CPU 2x6 cores of newer type
 (about 2x more CPU power compared to now)

=> RAM 128 GB

(3x more than now, great for caching more data blocks)

- => 10 GigE for storage and cluster access (same as now)
- => Storage: newer generation and higher spec NetApp NAS with more SSD cache and more RAM (about 2x in performance for random reads and writes)





- The new HW will give a room for increased performance and throughput.
- However a big challenge and responsibility on many of us is to design and develop DB applications with scalability in mind to serve ever increasing workload for many years ahead.
- Much tight collaboration between SW developers, DB application admins and DB infrastructure DBAs is needed.





 The ADCR database resource usage stayed in the green zone for the whole 2012 and Q1 of 2013.

All that is due to the hard work from many experts on keeping the DB HW and SW in good state, usage of new Oracle 11g features in production, ADCR application improvements and maturity.

 On focus now are developments of few new systems. Collaborative work is much appreciated.

THANK YOU!