Next generation database relational solutions for ATLAS distributed computing

G Dimitrov1*, T Maeno 2, V Garonne 1, on behalf of the ATLAS Collaboration

1European Organization for Nuclear Research (CERN), 1211 Geneva, Switzerland
2Brookhaven National Laboratory (BNL), 2 Center St, Upton, NY 11973, USA

* Corresponding author: gancho.dimitrov@cern.ch

Abstract. The ATLAS Distributed Computing (ADC) project delivers production tools and services for ATLAS offline activities such as data placement and data processing on the Grid. The system has been capable of sustaining with high efficiency the needed computing activities during the first run of LHC data taking, and has demonstrated flexibility in reacting promptly to new challenges. Databases are a vital part of the whole ADC system. The Oracle Relational Database Management System (RDBMS) has been addressing a majority of the ADC database requirements for many years. Much expertise was gained through the years and without a doubt will be used as a good foundation for the next generation PanDA (Production ANd Distributed Analysis) and DDM (Distributed Data Management) systems. In this paper we present the current production ADC database solutions and notably the planned changes on the PanDA system, and the next generation ATLAS DDM system called Rucio. Significant work was performed on studying different solutions to arrive at the best relational and physical database model for performance and scalability in order to be ready for deployment and operation in 2014.

1. Introduction

The ATLAS experiment [1] is one of the biggest multi-purpose particle detectors in the history of particle physics. It is one of the four big experiments built at the Large Hadron Collider (LHC) at CERN, and is designed to record and study proton-proton collisions at a final center-of-mass energy of up to 14 TeV. The ATLAS detector is located in Geneva, Switzerland at the European Organization for Nuclear Research (CERN). The data that are recorded by the detector are transferred to globally distributed computing centers for analysis. Additionally, the computing centers produce simulated particle collisions for comparison with detector data. The aggregate data accumulate to 20 Petabytes per year not including temporary and transient data. This rate is likely to grow with the energy and luminosity increases of the LHC in the coming years.

Apart from the notable volumes of event data, there are also impressive numbers for the management data stored in the relational databases not only in terms of volume but also in number of versatile applications, each with its own requirements. Relational databases are the most popular form of database scheme and have been commonly used since 1970. In relational databases, data are organized in the form of related tables: each table can have many rows and these rows can have many
columns. The software interface used to build and access the data structures is the Structured Query Language (SQL) [2]. It is the most widely known query language used today.

In the ADC scope the Oracle RDBMS (Relational Database Management System) [3] is used as a central database backend for the job submission and management system called PanDA (Production ANd Distributed Analysis) [4][5] and file distribution and management system DDM (Distributed Data Management) system called DQ2 [6] (its future successor is named Rucio). Some database facts on these two systems are shown in Table 1.

| System/Metric         | PanDA                                   | DQ2                                 |
|-----------------------|-----------------------------------------|-------------------------------------|
| System description    | Job submission and management system    | File distribution and management system |
| # Tables              | 58                                      | 77                                  |
|                       | 22                                      | 7                                   |
| Volume                |                                         |                                     |
| # Table segments      | 5.0 TB                                  | 2.7 TB                              |
|                       | 0.8 TB                                  | 4.1 TB *                            |
|                       | 0.4 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

Table 1. Facts about the ADC systems PanDA and DQ2 (as of Sept 10th 2013)

2. PanDA system

The PanDA system is the ATLAS workload management system for production and user analysis jobs. Originally the database backend was a MySQL database at BNL (Brookhaven National Laboratory, NY). In 2008 it was migrated to an Oracle database at CERN.

Some of the PanDA and its database back end requirements include:

- PanDA must operate with high reliability and robustness.
- It has to manage millions of grid jobs daily.
- Changes of jobs’ status, site loads and task progress have to be reflected on the database instantaneously. Fast data retrievals by the PanDA server and monitor are essential.
- DB system must cope with spikes of user workload.

Those requirements set a challenge on the database system for being able to deal efficiently with two different workloads: transactional from PanDA server and (to some extent) data warehouse load from PanDA monitor. Moreover the whole system including the database structures has to be able to evolve adequately meeting the ever-changing requirements of the analysis use cases.

Logically PanDA data is separated into 'operational' and 'archive' types. Information relevant to a single job is stored in four major tables. The most important attributes are kept separately from the
other space consuming attributes like job parameters, input and output files. The ‘operational’ data is kept in a separate schema that hosts active jobs plus finished ones within the most recent 3 days. Jobs that get status ‘finished’, ‘failed’ or ‘cancelled’ are moved to an archive PanDA schema. The PanDA archive data flow is sustained by a set of scheduler jobs on the Oracle server that executes a logic encoded in PL/SQL procedures (Figure 1).

The benefits from such data segmentation organisation are:

- High scalability: the PanDA jobs data copy and deletion is done on table partition level instead of on a row level.
- 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 segments (alter table … drop partition).
- Fragmentation in the table segments is avoided. Additional gain is better space utilization and caching in the buffer pool.
- No need for index rebuild or coalesce operations for these partitioned tables.

**PanDA operational data**

| PanDA jobs, attributes, input and output metadata are partitioned on column named ’modificationtime’. | The archive data is partitioned based on the ’modificationtime’. Certain tables have partitions of three days interval, others a time range of a month. |
|---|---|
| Each partition covers a time range of a day. | Partitions that can be dropped. An Oracle scheduler job is taking care of that daily. |

**PanDA archive data**

- Insert the data of the last complete day
- Filled partitions
- 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.

Figure 1. PanDA data segments organization

Scalability, reliability and adequate level of performance have always been the most important requirements for the data management and processing. Databases are a core component of the ATLAS computing infrastructure and are expected to be highly available and reliable. In the scope of PanDA we have validated and used in production many advanced database techniques:

- **Automatic interval partitioning**: partition is created automatically when a user transaction imposes a need for it.
- **Result set caching**: Oracle sends back to the client a cached result if the result has not been changed meanwhile by any transaction.
- **Data aggregation for fast result delivery**: A table with aggregated stats is re-populated by a PLSQL procedure on an interval of two minutes by an Oracle scheduler job.
- **Customized table settings for the Oracle stats gathering**: Oracle spends time and resources on collecting statistics only on partitions that are transactional active.
PanDA’s complete archive now hosts the information of 900 million jobs (all jobs since the job
system start in 2006). For offloading the main database from the PanDA monitor queries that span
their searches on large time periods (months and years) we have the option of using the data replica.
ADCR database has two standby databases: a Data Guard for disaster recovery and backup
offloading and an Active Data Guard (ADG) as a read-only replica (Figure 2).

![Active Data Guard deployment models](image)

**Figure 2.** Active Data Guard deployment models

The PanDA monitor can benefit from the 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 completely rely on it.

JEDI (Job Execution and Definition Interface) is a new intelligent component of the PanDA server
which dynamically defines jobs from task definitions. The main reasons to develop JEDI are to make
the PanDA system task-oriented. JEDI works with work queues. Tasks are mapped to work queues
based on their attributes, such as processing type, working group, production source label and core
count. Each task belongs to only one work queue and thus all jobs in the task belong to that particular
work queue.

JEDI was put in production in the first week of July 2013 and gradually more and more systems
started using its new functionalities. To address and support the new features, extension of the DB
implementation took place. Its data segmentation 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). 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 while the application does
row inserts or updates. According to the current analysis and production submission task rates of 10K
to 15K per day, the estimate for the JEDI needed disk space is in the range 2 to 3 TB per year.

JEDI tables (as total 9) were added to the existing PanDA database schema and some of the
existing PanDA tables were altered by adding new columns and foreign keys relevant to the JEDI new
objects. In order to achieve transparency during the DB schema changes for the PanDA applications,
‘not null’ constraints on these columns were not explicitly set and a confidence (for a transition period
of several months) for providing meaningful values was given to the PanDA server.
3. Rucio system

The Rucio project [7][8] is the next generation Distributed Data Management (DDM) system for allowing the ATLAS collaboration to manage large volumes of data (currently about 140 PB and adding tens of petabytes per year), both taken by the detector as well as generated in the ATLAS distributed computing system.

Some of the Rucio and its back end database requirements include:

- 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, a 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.

Following the application logic and nature of the data we concluded that the most natural choice for organizing Rucio’s data is by using Oracle’s list partitioning approach where each partition is bound to an explicit column value or list of values. 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 being partitioned on the SCOPE column and sub-partitioned on the DID_TYPE column (Figure 3).

![Figure 3. Snapshot on some partitions and their sub-partitions of the DIDS table](image)

Sub-partitioning the data into 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 user features – the free pattern searches on data identifier names plus retrieval of any relevant attributes (metadata). Since Oracle scans only the relevant sub-partition, the elapsed time is in the scope of seconds (the user’s focus is usually on the dataset sub-partition). The studies showed that this kind of data segmentation is the most efficient one for the free pattern searches. Indexing the column itself does not provide a performance gain and in many cases is just the opposite due to the larger number of index and table single block reads. Oracle has to perform these random block reads for getting the ROWIDs (address of the row in the data block) from the index leaf blocks and then getting each data row from the table segments for each of the ROWIDs. As an example, studies using a query on Rucio’s data identifiers table called DIDS with the following WHERE clause “WHERE SCOPE='data13_14TeV' AND did_type='D' AND NAME like 'data13_14TeV.%' AND NAME like 'data13_14TeV.AOD.%'” showed...
four times more data block reads (69551 compare to 15005 buffers, an Oracle block = 8KB) when using the index compare to sub-partition scan (Figure 4).

![Table showing data block reads comparison](image)

**Figure 4.** Comparison on two access paths of a query based on the buffers metric

List partitioning is appropriate but is also an operational challenge, due to its static nature - 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. A homemade solution is building automatic partition creation in all relevant tables whenever a new partition key value is inserted into a dedicated dimensional table

- An ‘after insert’ row level trigger gets fired and executes a PLSQL procedure responsible for ALTER TABLE … ADD PARTITION (procedure has measures against SQL injection).
- There is a logic for handling ORA-00054 “resource_busy” error (happens when there are uncommitted transactions).
- Each partition creation action is logged into a dedicated logging table.

Another appropriate technique used for the Rucio rows representing historical data is segmenting the data by time (daily partitions). Tests showed that Oracle’s range partitioning plus automatic interval partition creation best serves that data organization. In addition Oracle’s OLTP (Online Transaction Processing) compression is activated (which is effectively column values de-duplication within each Oracle data block of 8KB). Apart from the obvious disk space savings, additional benefit is the faster data mining for analysis and decision tasks. The exact values in terms of data volume savings and query performance are still to be studied more thoroughly.

As usual when building a new system a challenge comes with the decision of the right index strategy: global or local (partitioned as the table segments), long term data preservation and maintenance, ways to handle the most significant access patterns in the most efficient way, etc. For the Rucio system we plan to use the resources of the database Active Data Guard (which is a read-only replica of the primary database, Figure 2) for some specific use cases like complex analytic queries or regular data exports.

Recently (September 2013) we arrived to a relational schema which we believe will give us the flexibility and scalability we need for smooth operation of the Rucio system for many years ahead. Performed tests with four times the current production DDM workload brought confidence in that.
Moreover the appropriate table partitioning provides us a freedom in maintenance operations of the data and index segments when fragmentation gets to high level or in cases we need to move partitions to data files of other file systems. Keeping data compact within the blocks (each block is of 8KB size) contributes to the overall performance as more rows can be cached into the Oracle server buffer pool or on the SSD cache on storage level.

4. Conclusions

The new PanDA component called JEDI was successfully deployed in production in the beginning of July 2013. 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. The new Rucio system is currently in validation phase. Having the database knowledge and expertise of the last years we designed the Rucio database schema according to basic rules of simplicity, manageability and scalability. The Rucio system is due to be fully deployed in production in 2014, bringing new capabilities and power to the end users.

References

[1] The ATLAS Collaboration 2008 JINST 3 S08003 doi:10.1088/1748-0221/3/08/S08003
[2] Structured Query Language, http://en.wikipedia.org/wiki/SQL
[3] Oracle RDBMS, http://www.oracle.com/us/products/database/index.html
[4] Maeno T et al. 2012 J. Phys.: Conf. Ser. 396 032071 doi:10.1088/1742-6596/396/3/032071
[5] Maeno T for the ATLAS Collaboration 2013 Evolution of the ATLAS PanDA Workload Management System for Exascale Computational Science These proceedings (CHEP2013)
[6] Garonne V et al. 2012 J. Phys.: Conf. Ser. 396 032045 doi:10.1088/1742-6596/396/3/032045
[7] Garonne V for the ATLAS Collaboration 2013 Rucio - The next generation of large scale distributed system for ATLAS Data Management These proceedings (CHEP2013)
[8] Rucio, http://rucio.cern.ch/