The importance of having an appropriate relational data segmentation in ATLAS

G Dimitrov*1, on behalf of the ATLAS Collaboration
1European Organization for Nuclear Research (CERN), 1211 Geneva, Switzerland
E-mail: gancho.dimitrov@cern.ch

Abstract. In this paper we describe specific technical solutions put in place in various database applications of the ATLAS experiment at LHC where we make use of several partitioning techniques available in Oracle 11g. With the broadly used range partitioning and its option of automatic interval partitioning we add our own logic in PLSQL procedures and scheduler jobs to sustain data sliding windows in order to enforce various data retention policies. We also make use of the new Oracle 11g reference partitioning in the Nightly Build System to achieve uniform data segmentation. However, the most challenging issue was to segment the data of the new ATLAS Distributed Data Management system (Rucio), which resulted in tens of thousands list type partitions and sub-partitions. Partition and sub-partition management, index strategy, statistics gathering and queries execution plan stability are important factors when choosing an appropriate physical model for the application data management. The so-far accumulated knowledge and analysis on the new Oracle 12c version features that could be beneficial will be shared with the audience.

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 SQL (Structured Query Language) [2]. It is the most widely used query language today.
ATLAS uses a distributed computing system that links together over 100 computing centers around the world into a single data storage and processing environment. In the ADC (ATLAS Distributed Computing) 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 Rucio [6].

The ATLAS production and test DB systems are currently using Oracle database version 11.2.0.4. Oracle provides many types of table partitioning:

- Range
- List
- Hash
- Interval
- Reference
- Composite

In ATLAS we use all of them except the Hash option. We use the partitioning mostly because of operations flexibility but also the appropriate data segmentation provides some key advantages to the ATLAS DB applications in terms of performance. Certain details about the PanDA and Rucio database tables are shown in table 1.

| Application / Metric | PanDA | Rucio |
|----------------------|-------|-------|
| Application description | Job submission and management system | File distribution and management system |
| # Tables | 62 | 39 |
| - Non partitioned | 56 | 10 |
| - Partitioned | | |
| Volume | | |
| - Table segments | 7.3 TB | 0.6 TB |
| - Index segments | 0.8 TB | 0.7 TB |
| - LOB segments | 1.6 TB | 0 TB |
| Average daily segments growth | 11.3 GB/day | 3.2 GB/day |
| * Including 130 GB segments of index-organized tables | | |

Table 1. Facts about the PanDA and Rucio tables (as of April 2015)

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 job status, site loads and tasks 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. An important factor in this implementation is the usage of the Oracle range partitioning based on the PanDA’s “modificationtime” column (figure 1). The PanDA => PanDA archive data flow is sustained by a set of scheduler jobs on the Oracle server that executes logic encoded in PL/SQL procedures.

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](image1)

![PanDA archive data](image2)

Figure 1. PanDA data segments organization
PanDA’s complete archive now (April 2015) hosts the information of 1.5 billion jobs (all jobs since the job system start in 2006). PanDA monitor is interested in jobs of the most recent 12 months. Set of important parameters construct PanDA’s information lifetime management (ILM). The policies are the following:

- By design, table partitions 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 by 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 without physical rows movement (figure 2).
- The new yearly-based tables are easily moved to another database cluster, as they are self-contained in dedicated tablespaces.

```
Source table partition exchange with an auxiliary table:
ALTER TABLE FILETABLE_ARCH EXCHANGE PARTITION FILETABLE_ARCH_JAN_2009 WITH TABLE AUX_FILETABLE_ARCH INCLUDING INDEXES WITHOUT VALIDATION;

Destination table partition exchange with the auxiliary table:
ALTER TABLE Y2009_FILETABLE_ARCH EXCHANGE PARTITION FILETABLE_ARCH_JAN_2009 WITH TABLE AUX_FILETABLE_ARCH INCLUDING INDEXES WITHOUT VALIDATION;
```

Figure 2. Examples of PanDA partition exchange operations

JEDI (Job Execution and Definition Interface) is a new intelligent component of the PanDA server, which dynamically defines jobs from task definitions. A task is a group of jobs that take a dataset (group of files) as input and produce one or more datasets as output. The main reason to develop JEDI are to make the PanDA system task-oriented. In terms of database implementation, JEDI data segmentation is based on range partitioning on the JEDI’s TASKID column with interval of 50000 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.

### 3. Rucio system

The Rucio project [6] is a new generation Distributed Data Management (DDM) system to allow ATLAS collaboration to manage large volumes of data (currently 143 PB in about 700 million files and adding tens of petabytes per year), both taken by the detector as well as generated in the ATLAS distributed computing system. Data are physically stored in files (files can be grouped in datasets and datasets in containers) on the Grid. In Rucio each file (or copy of it) is assigned to a user, group or ATLAS production activity.

Some of the Rucio and its back end database requirements include:
- Must be fast and accurate being a 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.

Rucio logical units (scopes) with their metadata for managing ATLAS data on the Grid turned to fit naturally to the Oracle’s data segmentation via the List partitioning feature. In Oracle’s list partitioning approach each partition is bound to an explicit column value or list of values. The number of partitions in the Rucio DB schema depends on the number of registered Rucio accounts. Currently (April 2015) there are more than 5000 accounts. As five tables are defined with list partitioning (four are heap organized one of which with sub-partitions and one is Index-organized), the total number of list partitions is 26385 and the total number of list sub-partitions is 15831. The number of new accounts (respectively partitions) is not expected to grow significantly within the next couple of years.

Evaluation of different data segmentation solutions showed that list partitioning plus list sub-partitioning is the most appropriate for most of the Rucio use cases – e.g. DIDS table, which hosts all data identifiers (files, datasets, containers), is partitioned on the SCOPE column and sub-partitioned on the DID_TYPE column (figure 3).

| PARTITION_NAME | LAST_ANALYZED | NUM_ROWS | BLOCKS | SAMPLE_SIZE | HIGH_VALUE |
|----------------|---------------|----------|--------|-------------|------------|
| DATA12_8TEV    | 10.02.15 05:35:04 | 54258400 | 1039189 | 542584 'data12_8TeV' |
| MC14_13TEV     | 10.02.15 06:15:28 | 29514200 | 516337 | 295142 'mc14_13TeV' |
| MC12_14TEV     | 10.02.15 05:54:27 | 27789500 | 558867 | 277895 'mc12_14TeV' |
| MC11_7TEV      | 10.02.15 05:51:35 | 26818300 | 1060415 | 268183 'mc11_7TeV' |

| SUBPARTITION_NAME | LAST_ANALYZED | NUM_ROWS | BLOCKS | SAMPLE_SIZE | HIGH_VALUE |
|-------------------|---------------|----------|--------|-------------|------------|
| DATA12_8TEV_C     | 10.02.15 02:46:46 | 392900   | 8074   | 3929 'C'    |
| DATA12_8TEV_D     | 10.02.15 02:46:46 | 243337   | 8074   | 5095 'D'    |
| DATA12_8TEV_F     | 10.02.15 02:53:43 | 53431500 | 101041 | 534315 'F'  |

**Figure 3.** Snapshot on some partitions and their sub-partitions of the DIDS table

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 of the order 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 data identifiers table called DIDS with the following WHERE clause "WHERE SCOPE='data12_8TeV' AND did_type='D' AND NAME LIKE 'data12_8TeV.%AOD.%' " showed about 30 times more data block reads (248K compared to 8085 buffers, an Oracle block = 8KB) when using the index compared to a single sub-partition scan (figure 4).
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.

However, the described fine-grained data segmentation (partitioning) does not provide only benefits but also sets challenges to the database system itself. Oracle statistics gathering on the three levels, sub-partition, partition and table, takes considerable time (> 24 hours to all Rucio tables). Oracle uses one of three levels of statistics depending on whether the requested data in each query is within a single sub-partition, single partition or in case it spans more partitions, then table global stats are used. There are ongoing studies on how to speed up this process of statistics gathering. Another and related challenge is stability of the queries execution plans. Sometimes Oracle decides on using full sub-partition scan instead of using index access even when it matters for retrieving only a single row via a primary key global index. Currently we address this anomaly by having explicit instructions (hints) within the concerned queries to get the desired data access path.

We believe that we achieved a Rucio DB relational schema that gives the flexibility and scalability we need for smooth operation of the Rucio system for many years ahead. Moreover the appropriate table partitioning provides 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 DB systems. Keeping data compact within the blocks contributes to the overall performance as more rows can be cached into the Oracle server buffer pool or in the SSD cache on storage level.

4. NICOS system

The ATLAS Nightly Build system (NICOS) [7] database schema hosts package tags, nightly jobs timestamps, status information and other important parameters plus results from ATLAS software tests. The data volume is not huge (few 10s GB/year), but the application and the availability of the data in the database are considered critical. The retention of the data is agreed to be 12 months. Having these requirements, a logical choice was the Oracle’s reference partitioning technique for the child tables that host the nightly job attributes. The parent table is configured to use range partitioning.
In more detail, a parent table called JOBS is created with range partitioning based on a column of type NUMBER (the table’s primary key is partitioned as the table itself). These number values are with a specific format YYYYMMDDNNNNIII and encode the logic described in Figure 5.

![Diagram of JOBS table]

**Figure 5.** NICOS jobs ID format and encoded logic on which is based the partitioning

Any children tables to the JOBS parent table are configured with reference partitioning for achieving uniform data segmentation. All indices are locally partitioned as shown on Figure 6.

![Diagram of child table creation]

**Figure 6.** Example of NICOS child table creation with reference partitioning

Some of the major advantages in having this particular partitioning in NICOS include:

- Tables are partitioned in uniform way.
- On event of partition creation in the parent table, Oracle automatically creates relevant partitions into the children tables.
- Provides flexibility for maintaining any data-sliding window. Dropping the parent partition triggers automatic removal of the relevant child partitions.
- Partition pruning takes place when retrieving data (Oracle reads data only from the relevant partition).
5. Conclusions

With the usage of the Oracle database partitioning features in the physical implementation of the ATLAS database schemes, we bring flexibility, manageability and scalability to the ATLAS systems. Moreover, in certain use cases this directly contributes to faster data retrieval. Looking ahead in the new Oracle 12c version, we plan to explore and hopefully gain performance from a new DB feature called In-memory columnar store. Oracle 12c supports transparently for the applications row and columnar formats for the same table (or partition) simultaneously active and transactional consistent. By getting many orders of magnitude compression from the in-memory columnar store we would buffer larger volumes of data in the server memory and thus increase queries performance.

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. on behalf of the ATLAS Collaboration 2012 J. Phys.: Conf. Ser. 396 032071 doi:10.1088/1742-6596/396/3/032071
[5] Maeno T et al on behalf of the ATLAS Collaboration 2014 J. Phys.: Conf. Ser. 513 032062 doi:10.1088/1742-6596/513/3/032062
[6] Garonne V et al on behalf of the ATLAS Collaboration 2014 J. Phys.: Conf. Ser. 513 042021 doi:10.1088/1742-6596/513/4/042021
[7] Undrus A et al on behalf of the ATLAS Collaboration 2014 J. Phys.: Conf. Ser. 513 052034 doi:10.1088/1742-6596/513/5/052034