Journal of Physics: Conference Series **219** (2010) 042025 doi:10.1088/1742-6596/219/4/042025

Advanced Technologies for Scalable ATLAS Conditions Database Access on the Grid

R.Basset[1], L.Canali[1], G.Dimitrov[2], M.Girone[1], R.Hawkings[1], P.Nevski[3], A.Valassi[1], A.Vaniachine[4], F.Viegas[1], R.Walker[5], A.Wong[6]

[1] European Organization for Nuclear Research, CERN CH-1211, Genve 23, Switzerland
[2] Lawrence Berkeley National Lab, 1 Cyclotron Road, Berkeley, CA 94720, USA
[3] Brookhaven National Lab, Upton, NY 11973-5000, USA
[4] Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA
[5] Ludwig-Maximilians-University, Am Coulombwall 1, Munich, Germany.
[6] TRIUMF, 4004 Wesbrook Mall, Vancouver, Canada

Abstract.

During massive data reprocessing operations an ATLAS Conditions Database application must support concurrent access from numerous ATLAS data processing jobs running on the Grid. By simulating realistic work-flow, ATLAS database scalability tests provided feedback for Conditions Db software optimization and allowed precise determination of required distributed database resources. In distributed data processing one must take into account the chaotic nature of Grid computing characterized by peak loads, which can be much higher than average access rates. To validate database performance at peak loads, we tested database scalability at very high concurrent jobs rates. This has been achieved through coordinated database stress tests performed in series of ATLAS reprocessing exercises at the Tier-1 sites. The goal of database stress tests is to detect scalability limits of the hardware deployed at the Tier-1 sites, so that the server overload conditions can be safely avoided in a production environment. Our analysis of server performance under stress tests indicates that Conditions Db data access is limited by the disk I/O throughput. An unacceptable side-effect of the disk I/O saturation is a degradation of the WLCG 3D Services that update Conditions Db data at all ten ATLAS Tier-1 sites using the technology of Oracle Streams. To avoid such bottlenecks we prototyped and tested a novel approach for database peak load avoidance in Grid computing. Our approach is based upon the proven idea of *pilot* job submission on the Grid: instead of the actual query, an ATLAS utility library sends to the database server a pilot query first.

1. Introduction

Large scale access to databases from the Grid is crucial for extracting physics results from the ATLAS detector, on the LHC at CERN. This is because *conditions* data are required in order to reconstruct the event data. Conditions data describes the time-dependent parameters of the detector. Each object stored has an associated Interval Of Validity (IOV), a start and end time between which they are valid. The objects are stored in the COOL[1] condition database, which was specifically designed to store such data with IOVs. COOL data is organized into a set of hierarchical folders, so that certain tasks will only require access to a subset of the folders. Conditions data includes voltages, temperatures, magnetic field map, as well as calibrations and alignments, which may be determined and uploaded some days later.

In the ATLAS computing model, reconstruction is done both at the Tier-0, in the case of the primary reconstruction, and at the Tier-1's for reprocessing purposes. This requires that the conditions data be accessible from several thousand worker nodes distributed across 10 sites world-wide. Hence, the access must be distributed and highly scalable. Conditions data is also accessed by calibration and alignment tasks, as well as from certain user analysis tasks. User analysis poses the particularly difficult use-case of chaotic access to the data from Tier-2s.

2. Database Deployment

Conditions data from the detector is written to the *online* Oracle database located at CERN. This is a critical part of the data taking operation, and, as such, cannot be accessed by any offline activities which may reduce its performance. Instead, CERN also hosts an *offline* replica of this database which is updated form the online one, in pseudo-realtime, by means of Oracle Streams.

The offline Database(Db) is then available for Tier-0 primary reconstruction, and also as the source for further replication. ATLAS choose to replicate the conditions database to all 10 Tier-1s, again using the Streams technology. In turn this reduces the load on the CERN offline Db, which is itself critical for the primary reconstruction.

In order that a running job, on a particular host, can locate a Db replica, there is a simple mapping of internet domain names to the closest Db replica.

A further level of replication is provided by a mechanism to extract part of the conditions data for specific use cases. The data is extracted from the CERN offline Db and stored in SQLite files, which can then be widely replicated by the existing file distribution mechanism. This is possible because ATLAS software accesses the COOL Db via an abstraction layer known as CORAL[2]. This allows the backend Db technology to be changed transparently, and SQLite is one of the supported backends. For example, current Monte Carlo productions require well-defined subsets of the conditions data which are largely time-independent. This typically results in one SQLite file for a whole production campaign. In the case of data reconstruction, it is also possible to extract run dependent conditions data into multiple SQLite files, which together cover the run period.

The advantage of the SQLite method is that each job accesses its own Db, and there are no scaling issues on the access.

Another backend supported by CORAL is FroNTier[3]. This allows access to the Db via http queries. Identical SQL queries result in identical URLs and the result can then be cached in industry standard web caches, e.g. Squid. This technology is developed for and by CMS and is under test for ATLAS. It appears to be particularly applicable to the chaotic Tier-2 user analysis use-case.

3. Scaling tests

In tests, it was important to simulate the expected access pattern. Factors include the data read by each job and the number of concurrent accesses. During preparations for data taking not all conditions data are filled yet, nor accessed from the reconstruction, so the state of the art reconstruction is only an approximation of the true access. Therefore a stand-alone test suite was developed, which contained the best guess at realistic access patterns. The test suite accessed the local Oracle Db.

Multiple instances of the test suite were run concurrently, on many Tier-1 worker nodes, in order to simulate also the concurrency. The database load was monitored, as well as the performance, in terms of time for the test jobs to complete. The Tier-0 to Tier-1 streaming for Db replication was also monitored.

It was observed that the Oracle infrastructure at some of the Tier-1s tested, was inadequate, and in particular the disk i/o was the bottleneck. There was significant interference with the

incoming streams, and indeed over the course of a few hours began to affect Streams to other Tier-1s.

A feature of Streams replication to multiple Dbs is that if one destination cannot accept updates for some time, then replication to all the other destinations must stop too. The period a Tier-1 can safely fail updates, without affecting the rest, is determined by a buffer size at the Tier-0. In order to accommodate scheduled downtimes, and failures perhaps due to load, this buffer was recently increased in size.

Since scalability was not achieved at all Tier-1s, and the disruption to Streams was not acceptable, it was decided to use the SQLite method for the first reprocessing of the 2008 cosmic and beam data. As expected, there were no Db scaling problems during this exercise.

Nevertheless, solutions to scaling problems in accessing the Oracle Db were pursued. This is because the array of use-cases for Db access within ATLAS is not yet fully understood, and the SQLite method may not be useful in all cases.

4. Solutions to scaling problems

4.1. Staggered job starts

Reconstruction of RAW data only requires access to the Conditions Db for a short time at the start of the job. This is because the data are broadly speaking in time order, and the first event triggers the reading of all conditions data for a 10 minute period. In fact, it is 1 minute before, and 9 minutes after the first event timestamp because events coming out of the DAQ can be a little out of order.

For real data reconstruction, we can estimate 5 minutes of Db access in an 8 hour job. So for a typical Tier-1 this means the average job start rate, in order to fill 2000 cores, is only 4 per minute, or around 20 concurrent Db accesses. This is a very modest average rate, but there are many effects on the Grid which tend to make jobs start in bursts.

Firstly, there is the ramp-up on a previously empty cluster, whereby jobs can start as fast as they are submitted. Since jobs are waiting for pre-stage of input files from tape, and because of the way *Panda* releases jobs for execution, this could also lead to bursts of job starts. Panda is the ATLAS production system. The fair share method, in the most popular batch scheduler on WLCG, also tends to start jobs in bursts. However, because of the use of the pull-method (or pilot method) for payload execution, Panda has full-control over when a particular job starts. So, although the rate at which pilots start cannot be controlled, the rate at which they start the actual jobs can be.

When the pilot asks for a job, it also receives a delay in seconds, and it should wait for this time before executing the given job. Panda keeps a count of such jobs in the *sent* state, and the delay given to subsequent jobs is proportional to the number already waiting. When the delay expires, then the jobs goes to *starting* state, and no longer contributes to the delay calculation. In this way, and with some tuning, one can achieve the desired N starts per minute.

4.2. Intelligent Db access throttling

Although the staggered job start method can be tuned to limit the Db load from *these* jobs, it knows nothing of the actual Db load, which may include that from other clients. On the other hand, the Db itself has very good internal monitoring of the number of sessions, connections and load. It is therefore possible to construct a function, executing some internal queries, which returns whether the Db can process more requests at this time.

The job then first connects to the Db and runs this function, and depending on the response, either starts the actual Db access, or sleeps for while before retrying. This has been implemented and tested using reprocessing jobs at CC-IN2P3 in Lyon, and it was shown that the Db load could be limited to some desired threshold.

There are two drawbacks to this approach. Firstly, the sleeping jobs are occupying and wasting cpu time. Secondly, there is no queuing mechanism, and the first job which happens to ask at a time the load has fallen below threshold will start next. This is regardless of how long it has been waiting, and gives no fair sharing between different users/tasks accessing the Db.

The cpu wastage is inevitable, when throttling at this late stage, and is mitigated by failing the job after some timeout period. On the second point, it is possible to apply some queuing. A version for the Db-side functions is under test which will track the requests, and give permission for particular ones to start. This makes it possible to operate a queue, and possibly to apply fair share between clients. This queue would be stored in the Db, with old connect requests being automatically pruned after some time.

4.3. SQLite

Monte Carlo production uses an SQLite Db packaged and distributed as a flat file. Usually, the file is copied to the worker node, from a local replica, and thus each job has exclusive access to the Db. This avoids all scalability issues on Db access, and the success of this method is strong motivation to use this for reprocessing.

The significant difference for reprocessing real data is the requirement of time-dependent conditions data. This increases the volume of conditions data such that one can no longer have a single SQLite file, but rather one per run range. This represents a bookkeeping and operational problem, but doesn't change the approach significantly. In the March 2009 reprocessing, 139 SQLite files were created, one per run, and distributed to the Tier-1s. A small amount of direct Oracle access was still required, but the bulk of the Db access was via these SQLite files, and accordingly no scalability problem was observed.

The SQLite method is particularly suited to deliver non-time dependent Conditions data, as in the case of MC production, where just a few files are required. However, when the Conditions data requirements are well-known, as for the official reprocessing passes, it is also possible to create time-dependent SQLite files. Typically one SQLite file per physics run is required, so a high level of automation is required to produce and distribute the 1000s of files required for a run period. Due to the scalability and reliability considerations, both MC production and data reprocessing make use of the SQLite access method.

5. Further Improvements

During the various tests, several areas of improvement in the ATLAS code were identified and addressed. It became clear that many of the queries made on the Db were unnecessary. Changes were made to the COOL layer to avoid these. The number of Oracle sessions opened per jobs was reduced. An aggressive read-ahead and cache strategy was adopted, in order to further reduce the number of queries made, at the minor expense of them returning more data. In order to facilitate future optimization, monitoring and statistic collection tools were added, the lack of which had allowed to go unnoticed the sub-optimal Db access patterns which we since observed.

6. Conclusions

We have performed extensive testing of the scalability of the Conditions data access from large-scale distributed reconstruction of ATLAS data. The Tier-1 Oracle infrastructure coped well when jobs started at the rate required to fill all the available cpu with reconstruction jobs. However, there are many factors leading to bursts of jobs starting at the same time and, under these conditions, several of the Tier-1s had problems.

The overload of the Db infrastructure had, in some cases, a knock-on effect that the Stream replication from Tier-0 was disrupted. Several solutions to avoid the overload were investigated. The SQLite solution for reprocessing completely avoids Db overload, but may not be applicable to all ATLAS conditions data access use-cases. In particular, some user analysis jobs may skim

over a large run range, where it is not possible to produce the required SQLite extraction. Removing the reprocessing load allows the Tier-1 Dbs to serve some of the less predictable use-cases.

References

[1] Valassi A *et al.* COOL, LCG Conditions Database for the LHC Experiments: Development and Deployment Status, Proceeding to CHEP07, Victoria, BC, Canada (2007)
[2] Molnar Z *et al.* CORAL Development Status and Plans, Proceedings to NSS 2008, Dresden, Germany (2008)
[3] Blumenfeld B, Dykstra D, Lueking L, Wicklund E 2008 *Journal of Physics: Conference Series* **119** 072007