Experience with ATLAS MySQL PanDA Database Service

Y Smirnov¹, T Wlodek¹, K De², J Hover¹, N Ozturk², J Smith¹, T Wenaus¹ and D Yu¹

¹ Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000, USA
² Department of Physics, University of Texas at Arlington, Arlington, TX, 76019, USA

Abstract. The PanDA distributed production and analysis system has been in production use for ATLAS data processing and analysis since late 2005 in the US, and globally throughout ATLAS since early 2008. Its core architecture is based on a set of stateless web services served by Apache and backed by a suite of MySQL databases that are the repository for all PanDA information: active and archival job queues, dataset and file catalogs, site configuration information, monitoring information, system control parameters, and so on. This database system is one of the most critical components of PanDA, and has successfully delivered the functional and scaling performance required by PanDA, currently operating at a scale of half a million jobs per week, with much growth still to come. In this paper we describe the design and implementation of the PanDA database system, its architecture of MySQL servers deployed at BNL and CERN, backup strategy and monitoring tools. The system has been developed, thoroughly tested, and brought to production to provide highly reliable, scalable, flexible and available database services for ATLAS Monte Carlo production, reconstruction and physics analysis.

1. Introduction

The MySQL database [1] forms the backbone of ATLAS [2] PanDA [3] (Production and Distributed Analysis) system. This system including corresponding database backend has been developed in Brookhaven National Laboratory (USA) in 2005, deployed since late 2005 to support the distributed Monte Carlo (MC) simulation, reprocessing production and user analysis on the Grid originally within US ATLAS. Since early 2008 it is globally used as the worldwide distributed production system in ATLAS collaboration.

MySQL PanDA databases were configured, created, tuned, debugged and used first at BNL, then since June 2008 the second PanDA DB instance has been installed and brought to production in CERN. PanDA databases are the repository for all PanDA-related information, namely they contain the following data:

- Description of MC-production, reprocessing, validation, test and user analysis jobs (active and archived) and tasks
- Input, output files and dataset catalogues
- Short extracts from logfiles and some metadata
- Site, queue and pilot configuration data
- Monitoring information
2. **PanDA DataBase servers’ structure and performance**

In this section we discuss PanDA DataBase backend. PanDA databases at BNL are installed on several independent servers. Two of them (primary and spare) contain production information, four additional ones (two primary and two spare) contain archive information, metadata, logs and dataset data. In addition to the production servers there are two MySQL servers which support PanDA development testbed, We use this development database testbed for debugging and testing new schema and configuration changes, performing some scalability and performance investigations, etc.

In terms of hardware PanDA database production servers represent two Dell PowerEdge 2950 machines with 4-core Intel Xeon(R) CPU 5150@2.66GHz, 16GB memory using 64-bit RHEL4 OS and the hardware Raid-10 over 6 15krpm 145GB SAS disk drives. The four archive, metadata, log and dataset servers are Penguin Computing Relion 2600SA machines with 8-core Intel Xeon CPU E5335@2.00GHz and 16GB memory using 64-bit RHEL4 OS and six 750GB SATA drives with the software raid 10 implementation.

PanDA production DB servers dbpro and dbpro02 run MySQL 5.0.X and host the main PanDA database PandaDB which acts as a “fast buffer” and stores the information about active Monte Carlo production, validation, reprocessing and data analysis jobs. Completed job records are stored in the database for up to two weeks, following this time period they are moved to the archive database. The database consists of 31 InnoDB tables containing up to 17M of rows. The hardware allows for 380-440 threads accessing the database simultaneously (although maximum number of connections about 800 has been reached). The average performance is 360 queries per second. Queries are approximately 35% select type, 35% update and 25% insert, the remaining ones are of other types (delete, etc.)

PanDA archive production MySQL server dbarch3 and the spare node dbarch5 support the following databases: PandaArchiveDB, PandaMetaDB, and PandaLogDB. The largest archive database PandaArchiveDB keeps the full archive of PanDA managed Monte-Carlo production, reprocessing, validation, test and user analysis jobs since the second half of 2005. In the end of 2007 these servers were migrated to a new more powerful hardware and run MySQL 5.0.X. Tables in PandaArchiveDB use MyISAM engine, doesn’t have autoincrement-counter, and allow us to move the data from PandaDB through cron-jobs running on the servers. We have developed and successfully implemented some kind of partitioning in this DB: a new bi-monthly structure of job/file archive tables provide the better performance in terms of acceleration of the data mining and getting access to the data required. The database consists of 44 tables; the maximal number of rows per table is about 33,000,000.

Other 2 databases running on the same server are DataBases PandaLogDB, and PandaMetaDB. They store the archive of log-extract files for jobs, some monitoring information about pilots, autopilot and scheduler-configuration support (schedconfig table). The same engine MyISAM and bi-monthly partitioning schema is used as well, These DB have ~52-54 tables with maximum number of rows ~4,600,000 per table. Access pattern to this server: ~400-450 parallel threads open (max ~740) simultaneously. Average query performance is ~1300-1600 queries per second (maximum ~2800). The query distribution on that server is the following: select ~80%, insert ~20% (usually we don’t have delete or update queries in the archive).

The Dataset, Metadata-prime and Dataset browser databases occupy the servers dbarch4 and the spare node dbarch6. They store metadata and dataset information as well as some statistics about pilots, service information and monitoring. The 55 database tables are implemented using MyISAM engine and allowing for 100-200 parallel threads to access the database (250 threads during peak load periods). The databases serve 3-5 queries per second (up to 12 during peak periods). About 40% of the queries are select type, 30% are update, the rest are delete, insert etc.
PanDA databases at CERN have been recently migrated to Oracle. Since January 2008 till March 2009 they were based on MySQL using similar architecture to the the BNL PandaDB. There were 3 primary and 3 fail-over servers. PandaDB used machine pandadb1 and fail over machine pandadb4. It could handle 130-15 threads (max 250 threads) and 10-20 queries per second. The Archive, Meta and LogDBs were installed on pandadb2,3 machines with pandadb5,6 backup-servers. They could handle 10-25 parallel threads (maximum 40 was reached) and 50-60 queries per second.

The database performance was monitored using the dedicated PanDA web-interface called PanDA-monitor, which forms an integral part of PanDA production software package. Another very helpful package we used for MySQL statistics monitoring is an open-source tool Mysqlstat.

3. Database backup and replication
At BNL we’ve developed several database backup policies and implemented them as part of the PanDA database service.

1. Full monthly backup - once a month a full text dump of all databases is performed. The dump is then loaded into fail over databases. An additional copy is stored on tape.
2. Daily backup. Text dump of selected databases is performed daily. The dump is then transferred to fail over node and loaded into fail over servers. Extra copy is stored on tape.
3. Combined text and binary backup. Text backup procedure is slow. For large databases may take considerable time and even create some unexpected issues on the production servers. For that reason it cannot be done more often than once per day. In between the text backups binary (incremental) backup is used. The MySQL server writes record of all write operations performed to database to special log file. This log file is then periodically transferred to failover server and the failover database is then updated. This procedure reduces the amount of data which can be lost in the event of catastrophic database failure.

Full text backup procedure uses extensive table locking which affects the database performance, since queries which happen during the backup cannot be executed, are held waiting and frequently time out. To avoid this problem some databases are backed up using so called "hot backup". Hot backup is performed using commercially available software called "innobackup"[4] which performs backup of InnoDB tables while locking rows only. This allows us to avoid an interruption in the production service. The resulting hot backup file is then transferred to fail over machine and loaded into fail over server and a copy is sent to tape storage. Since hot backup is less disruptive from the production point of view it can be performed “on-line” more frequently than the text backup.

Hot backup is not used in combination with incremental backup due to complications with synchronisation of those two backups. In principle it is possible to synchronize them, however we have found out that in practice a similar reduction of data loss risk can be achieved simply by increasing hot backup frequency.

4. Slow query monitoring
We also implemented slow query monitoring for Panda database servers.

Slow or badly constructed queries can cause severe performance degradation. A slow query can lock MyISAM tables for extended period of time thus holding execution of other queries. It is therefore necessary to perform monitoring of queries and eliminate bad ones. A custom program has been written in Python which monitors queries which are being executed and identifies slow ones. Those which execute longer than allowed time are then killed and logged into report which is periodically submitted to database administrators for analysis. This report allows identifying users and sites which often submit long, inefficient of badly constructed queries. Responsible persons are then contacted and ways to improve their queries are then suggested.
MySQL server keeps log of slow queries, however the format of this log makes it very hard to read it. A dedicated program has been written which provides GUI to analyze the slow query logs. This program allows following up MySQL queries on minute by minute basis and helps to identify which exactly query cases slowdowns of the system - even if the query executes fast enough to avoid being killed by slow query killer. The users can be then contacted and suggested changes to their MySQL usage patterns on the client side which could improve the query performance.

5. MySQL monitoring
A vary important part of MySQL database monitoring is performed by Nagios[5] - a general purpose monitoring program, de facto community standard. Monitoring is done via nagios plugins - a set of custom written probes which are executed to check the status of monitored services. For the purpose of monitoring MySQL a dedicated set of probes has been written[6]. Examples of tasks performed by nagios plugins include:

- Periodic verification whether MySQL server is alive and available
- Measure time required to perform a benchmark query, raise alarm if it is too slow,
- Check disk space available on server, raise alarm if it is to low,
- Check the status of backup, raise alarm if the last backup failed,
- Perform backup validation and quality assurance. Compare the databases on production and failover node and raise alarm if their contents diverge - which can be a sign of failed backup or other data corruption,
- Check load on database server, raise alarm if it is to high,
- Check number of MySQL threads and processes.

In addition to service monitoring with Nagios a separate monitoring system has been deployed to monitor the database performance using the Ganglia software [7]. It is an open source product for monitoring MySQL databases. Among others it collects information on the system load on servers, memory usage, and available disk space. The performance data is presented on operator’s dashboard. In addition to Ganglia MySQL specific performance information (like for example number of running connection threads, average connection time) is collected by mysqlstat – a free open source monitoring tool.

6. Conclusions
MySQL PanDA DataBase system is one of the most critical components of PanDA, and has successfully delivered the functional and scaling performance required by PanDA. It currently operates at a scale of half a million jobs per week with much growth still to come. Archive catalogues keep track of almost 30,000,000 Monte-Carlo production, reprocessing, validation, test and user analysis jobs, and more than 150,000,000 files produced on the grid during last 4 years. Partitioning (bi-monthly tables design) gives an advantage of a quick searching through the archive and the flexibility in terms of schema evolution. The system provides highly reliable, scalable, flexible and available MySQL database services. Multilevel incremental backup strategy designed and implemented for PanDA DataBases allows quick and easy recovery in case of failure, minimizes the probability of data loss and can be easily implemented for support of other MySQL services (GUMS/VOMS, LRC/LFC, user databases, etc.)

References
[1] www.mysql.com
[2] ATLAS Collaboration (G. Aad et al.). 2008. JINST 3:S08003,2008.
[3] ATLAS Collaboration 2008 J.Phys.Conf.Ser.119:062036,2008.
[4] http://www.innodb.com/
[5] http://nagios.org
[6] Tomasz Wlodek, Carlos Gamboa, Zhenping Liu, Shigeki Misawa, Robert Petkus, Jason Smith, Thomas Throwe, Yingzi Wu and Dantong Yu. 2008
J.Phys.Conf.Ser.119:052031,2008.
[7] http://ganglia.info/