DataBase on Demand

R Gaspar Aparicio¹, D Gomez², I Coterillo Coz³, D Wojcik⁴

¹ IT-DB department, CERN
² IT-DB department, CERN
³ IT-DB department, CSIC/CERN
⁴ IT-DB department, CERN

E-mail: ruben.gaspar.aparicio@cern.ch

Abstract. At CERN a number of key database applications are running on user-managed MySQL database services. The database on demand project was born out of an idea to provide the CERN user community with an environment to develop and run database services outside of the actual centralised Oracle based database services. The Database on Demand (DBoD) empowers the user to perform certain actions that had been traditionally done by database administrators, DBA’s, providing an enterprise platform for database applications. It also allows the CERN user community to run different database engines, e.g. presently open community version of MySQL and single instance Oracle database server. This article describes a technology approach to face this challenge, a service level agreement, the SLA that the project provides, and an evolution of possible scenarios.

1. Introduction

Database services play a critical role in all organizations. They are usually the backend of many multi-tier applications. They should not block normal workflow of applications relying on them as in a transactional world; an application cannot work further, unless a transaction commit has been assured. They must be reliable in order to assure that information is safe on a storage media, and in case of major problem they should be recoverable with minimal, preferably zero, data loss. Consistency is critical for most of the applications. These characteristics are often referred to as atomicity, consistency, isolation, and durability [1].

Nowadays, a number of cloud providers offer not only computing, storage resources and infrastructure as Service (IaaS), but also database as service (DBaaS). The major difference among them lies in the type of database software they support, i.e. Rackspace (http://www.rackspace.com) expands MySQL, Oracle and SQL Server, whereas Xeround (http://xeround.com) with its specialized storage engine is based on MySQL and Amazon Relational Database Services (Amazon RDS - http://aws.amazon.com/rds) offering Oracle, Microsoft SQL server and MySQL. The kind of support and high availability (HA) solution also varies. Amazon RDS offers replication for disaster recovery purpose and for load balancing reads workload for MySQL, Xeround is based on replication of its built-in storage engine and supports running on different cloud providers allowing users to move database services among them, while Rackspace offers storage and application replication at the database level, supporting offsite replication.
At CERN, a need for MySQL service has emerged with a number of applications either only available with MySQL as a back-end database or for which MySQL is the preferred and best supported back-end database.

Cloud providers, to a certain extent, have inspired the approach of DataBase on Demand (DBoD), as we wanted to create a platform that would allow to grow and to add new functionality, always keeping in mind the lowest total cost of ownership (TCO) of the whole product. We do not use the cloud as a platform because we believe we can provide service more efficiently and at lower cost, being able to adapt and serve better our user community.

An initial user interface of DBoD is based on a web interface. Extending it to implement a REST interface following OpenStack REST interface for MySQL should be easy.

In this paper, we describe the architecture, storage requirements the user interface and functionality provided to the user community.

2. Architecture

The DBoD service technological choice allows us achieving the principles that guide the service. It has been designed to fully integrate with CERN’s wider XaaS (anything as a service). Although MySQL was our first offering, DBoD flexible architecture allows integrating easily other RDBMS. Oracle single instance database server is our second vendor. We intend to migrate as far as possible any Oracle service that fits DBoD principles, for example any kind of canned applications that require DBA privileges. As we want to count on the support from the database vendors. OracleVM is our virtualization platform for the time being, because it is the only virtualization platform with the Oracle official support and it provides technical features required for possible service evolution, like Oracle real application cluster (RAC). OracleVM[1] is based on Xen[3]. We do not focus on provisioning of virtual machines in this article, although we would like to mention a few words to complete the idea. The actual virtualization platform may change in the future. Nowadays our virtual machines comply with the CERN IT standards; in fact, they could be considered physical hosts, as from all perspectives, they do not differ from a physical machine in the way they are monitored, using Lemon[4], installed or configured via Quattor[5]. We expect to keep this model, following the CERN IT guidelines as long as it fits our service mandate. The way we handle virtual machines differs from the way cloud vendors or different products like Enterprise Manager 12c[6] usually work, as those are based on images. The benefit of our solution is that the virtual images can be evolved in small and most of the time transparent steps with the software being maintained in a declarative way and evolving in time while operating system images usually have to be replaced for upgrades.

A general view of the lifecycle of a DBoD instance can be seen in the figure 1.
The workflow would be as follows: first, a user requests a relation database management system (RDBMS) system indicating a few key parameters, such as storage requirements, RDBMS implementation and the expected number of connections. We also register the purpose of the database and status of the intended service, whether it is official (in production) or in development. This action triggers a record on the CERN IT account management system that can be treated by a DBoD admin. Once the request is approved, the user receives an email with an IP alias, username and temporary password for the full privileges database administrator account, and the user’s CERN identity credentials are given access to the DBA operations WEB interface to manage specific operations on its own instance. From now on, the user is responsible for the use of this CERN IT resource and, therefore, under CERN rules has to maintain it and respect the security rules. This is made reasonably easy and guaranteed, if it complies with DBoD service guidelines, for example not disabling the system and RDBMS patch upgrades on its RDBMS instance. To help with the management of an instance, an electronic group (e-group) and a list of physical user principals can be indicated under the DBoD user web interface.

The DBoD user web interface is used in any administrative action that a DBoD DataBase Administrator needs to perform, such as a backup, restore, change in instance configuration, monitoring, downloading logging files and startup or shutdown of an instance. Some of these DBA actions will be described later.

A general overview of the whole architecture can be seen in figure 2.
Each component will be described in more detail in the following paragraphs.

2.1 User interface
DBoD users are provided with a set of DBA actions that they can execute on their instances via a web interface: start up, shut down, manage backups, dispatch restores, manage files and access monitoring, among others. Following this approach, the service allows an easy management of database instances. For instance, performing an upgrade of the database version, which involves several steps such as creating a backup, stopping the database, installing software, etc. can be achieved with a single click. At the same time, this approach eliminates the need of giving users direct access to the virtual machines where their databases are running, abstracting them from the virtualization layer and minimizing the security issues that may arise when providing users with system administrator privileges.

This web application is implemented using an open source J2EE, Java to Enterprise Edition, Ajax-based framework called ZK Framework[7] its open source version, running on CERN central servers. ZK takes the so-called server-centric approach. The content synchronization of components and the event pipelining between clients and servers are automatically done by the engine and Ajax plumbing codes are completely transparent to web application developers. Therefore, end users get an engaged interactivity and responsiveness similar to a desktop application, while development retains a similar simplicity to that of desktop applications.

The web application is integrated with several technologies at CERN for authentication, authorization and managing of e-groups in order to allow several users to manage one single instance. By making use of the CERN Single Sign-On (SSO) system, Active Directory identity information can be retrieved for the user that is logged into DBoD. Information such as username, email, or the e-groups that the user belongs to, can be obtained from the SSL headers present in every request made to the server. The web application uses this information to allow users to manage the instances they own, or the instances that the e-groups they belong to are authorized to manage. DBoD also uses Apache
Axis[8] in order to use CERN Single Object Access Protocol (SOAP) web services to create and manage e-groups, as well as the web services implemented specifically for the DBoD project.

The connection to the DBoD management database is implemented using JDBC with DBCP[9] connection pooling via JNDI[10], to improve performance on database connections. Connection pooling is transparently done by the container using Apache Jakarta Commons DBCP, provided by the CERN J2EE public service.

The wrapper around the core implementation of DBoD commands could be expanded and indeed it may be expanded to offer DBoD DBA operations via different application interface (API). For example a kind of REST, representational state transfer, OpenStack based wrapper could be developed if in the future there is demand for such.

2.2 Software architecture and job execution

Operations carried by the user through the WEB interface are registered in the management database, whose status is periodically monitored by a job execution daemon running in the management node (see Figure 2).

Every management operation is currently implemented using the CERN IT-DB framework command called Syscontrol. Syscontrol is a management infrastructure developed and used inside the Databases Group at CERN which allows executing administrative tasks against a target system. Target systems (entities in the Syscontrol nomenclature) are registered together with a number of related parameters and configuration information in a LDAP directory.

The job execution daemon works basically as a job dispatcher. Every certain configurable number of seconds, currently five, it polls a database table where commands triggered by user action in the WEB interface are placed. Every command is related to a DBoD instance, and carries a certain number of parameters (configuration files, snapshot file names, etc.) when necessary. If the command requires it, the daemon executes pre-processing tasks regarding the job parameters (as when uploading a new configuration file, see following sections), and proceeds to launch a child process which executes the command and logs the output. After the command finishes, the daemon populates the management database with the job result, command execution log, completion date and any other relevant information.

The life cycle of a user operation job is shown in Figure 3. Once a new job is created, the instance state is set to PENDING, indicating that an action has been launched and a result is expected, and locking the instance against the execution of further operations until completion. If no jobs are found to be in RUNNING state after 30 seconds, an email alarm is generated to warn the service administrators of a problem existing in the management daemon. During normal operation, once the job is fetched, its status is set to RUNNING in the database; and after execution, depending of the result, the job final state is set to either FINISHED_OK or FINISHED_FAIL, in which case an email notification is sent to the service administrators. All command execution outputs are logged in both cases. If a job is found to be running for more than six hours (the longest time it takes for any of the commands to complete – backup to tape) a warning email is also sent to the service administrators.
The daemon is developed in Perl, for better integration and code reutilization with the current toolset of the DB Group. It follows a modular structure where adding functionality for supporting new RDBMS can be easily done by way of using modules. RDBMS specific code is dynamically selected based on both the instance type and the type of command to execute.

2.3 Monitoring

One of the important aspects of running any service is performance and availability monitoring. Both users and administrators require monitoring data to assess proper functioning of DBoD instances, additionally DBoD administrators rely on email alerts (SMSes for critical systems) in case availability is impacted. Monitoring can also warn about potential problems that can affect availability (machine starts swapping, local disk is getting full, etc.).

We have evaluated several monitoring solutions for MySQL databases, including Oracle’s MySQL Enterprise Monitor and MySQL plugin for Oracle Enterprise Manager[5]. We have found that they provide comprehensive set of features for DBoD administrators, but they do not deliver functionality that would allow us to open those systems to our users (no role-based access in Enterprise Monitor, a bug in former version of MySQL plug-in for Oracle Enterprise Manager). Those tools require supplementary installation and/or configuration steps that would increase DBoD complexity. This is why we have decided to use home-built tools.

The basic host-level monitoring of servers (virtual machines) is implemented with LEMON[3] and additional monitoring of DBoD instances is done by an in-house developed tool called RACMon (see figure 2), which is used for monitoring of production Oracle databases since 2006. RACMon is a three-tier, agentless monitoring system, written in python, which connects via secure shell, ssh, to monitored machines and executes specific host and database checks. It currently runs every six minutes and uses local xml configuration files (synchronized from LDAP, when a DBoD instance is added) to prepare necessary monitoring scripts. Then it connects to all specified targets, opening one thread per target. This design together with implemented timeouts allows RACMon to monitor many

![Diagram of life cycle of a user initiated administrative action.](image-url)
targets within short period of time. Different metrics are additionally uploaded to a monitoring database that is used later for plotting (see Figure 2). In case of the database’s unreachability the main RACMon’s availability monitoring features are not affected.

Monitoring scripts are prepared on a central server and are executed remotely (copied via secure copy, scp, and executed via ssh). RACMon supports Oracle and MySQL database targets and can be extended to monitor any other type of targets (we have added support for NAS storage and Oracle VM instances recently). Additionally to checking the instances on the hosts they are running, an external connectivity test is conducted in order to detect problems that may affect external connectivity or database log-on issues.

RACMon’s pluggable architecture allows adding new checks and tests to existing target types as well as adding new target types like MySQL master-slave replication. Correspondingly the metrics that are uploaded to the monitoring database can be changed and the DBoD DBA could request new ones to be added, if necessary for performance monitoring (see 3.6 for the interface description). This flexibility is an important feature of the DBoD monitoring solution and together with easy access to monitoring data provides added benefit to users compared to other monitoring solutions. The drawback of this implementation is that there are limitations as to the frequency of the monitoring runs (as this system also monitors over hundred production Oracle servers) and that it needs to be maintained as any other in-house built solution.

2.4 Virtual Machine provisioning

Currently Virtual Machine provisioning is a semi-manual process that requires interaction with: Oracle VM Manager to request a virtual machine, Network Database to populate Media Access Control, MAC, address and IP information, NAS storage system in order to create new volumes, Lightweight Directory Access Protocol (LDAP) in order to register the instance and Quattor[4] in order to prepare machine’s profile and finally install it. These operations are becoming more and more automatic and one day will be replaced with kiosk-like approach (fully automatic VM creation and installation). One of the solutions that CERN IT is investigating at the moment is puppet and OpenStack integration.

The most important feature the Oracle VM provides is “Secure Live Migration”. This allows moving running virtual machines to another host in case of a scheduled intervention. The active memory of a virtual machine is copied to another server and once this is done, the virtual machine is paused and only the remaining dirty pages are copied and the machine is resumed on the new physical server. This final move takes fraction of a second and services’ availability is not affected. Moreover during unexpected host machine’s failure the VM is restarted on another server in the same cluster and pool. All these HA features are handled by Oracle VM clustering solutions which include disk and network heart-beats and split-brain cluster protections.

2.5 Storage

Access to data is done via 10GbE links using NFS, network file system protocol, version 3. RDBMS instances usually have two different file systems. One dedicated to binary or redo logs archive and another one dedicated to data, see figure 4. Each file system is located in a different storage cluster.
Figure 4. Storage overview. Each DBoD instance has two active file systems: data and binary logs.

The storage is based on NetApp appliances. Storage clusters are made of a pair of NetApp controllers that work in “active/active” mode. Almost all components in a storage cluster are duplicated, which makes the storage infrastructure resilient to hardware failures. Almost 99% of the storage operations, that is software updates, disk firmware, disk shelf firmware, etc. can be done without downtime for DBoD instances relying on them.

DBoD takes advantage of many features of the underlying storage like snapshots, thin provisioning, defragmentation, a performing RAID6, Solid-State Drive (SSD) cache, etc.

DBoD relies on already existing storage infrastructure in use in CERN IT Databases group since 2006.

3. What a DBoD DBA can do.
Part of the mandate of DBoD is to empower traditional RDBMS users to act as DBA, a function with high responsibility as all database operations depend on their decisions. The system allows a DBoD instance owner to have its own DBA account, therefore entitling the user to do any operation on the database, such as changing auditing, creating a database or schema, managing users, etc.

There are certain DBA operations that need to be at the operating system level or require knowledge of the existing hardware and storage. These operations can be done via the DBoD web interface, described previously. We will describe now some of these operations, referring mainly to MySQL, although equivalent operations are provided for Oracle DBoD instances.

3.1 Backup via snapshots or to tape.
When a DBoD instance is created, automatically a part of the file system dedicated to data is reserved for snapshots. By default, the amount reserved corresponds to 20% of the file system size. That is, on a file system of 100Gbytes it will be 20Gbytes. Snapshots track block changes on a copy-on-write file system making rollback to a previous system state almost instantaneously, see figure 5.
At $t_0$ a snapshot is taken. Active file system and snapshot are the same. At $t_1 > t_0$, snapshot at $t_0$ contains previous block version of those blocks that have changed since $t_0$.

**Figure 5.** Snapshots at data file system at different points in time.

The more snapshots we take, the more possible snapshots at a point in time we have in order to start a restore. We can have up to 256 simultaneous snapshots per file system. The minimum internal for schedule snapshots is six hours. As soon as we get to about 98% of the snap reservation space, 19.60 Gbytes in our example, snapshots are deleted until a certain threshold of freed space is reached. This guarantees that snapshots do not consume active file system space.

Using the web DBoD interface (see figure 6) the user can set up a schedule to take snapshots periodically. She can also take a punctual snapshot.

**Figure 6.** Backup configuration dialog window on DBoD web site.

Taking a snapshot is always done by setting the database in consistent mode; otherwise the snapshot may be unusable for restore, if not all the committed transactions have been flushed to disk. This is achieved differently depending on the RDBMS vendor, for MySQL we use: flash tables with read lock and for Oracle alter database begin backup, removing this status once the snapshot has been taken. Creating a snapshot takes usually few seconds. Therefore, freeze of the database is minimum. This plays an important role especially in MySQL where the above statement blocks any transaction.

It is also possible to perform backup to tape. In that case a snapshot is sent to tape. Sending a snapshot to tape has an extra benefit of not blocking the database to produce a consistent copy on tape. This affects especially MySQL databases that do not use transactional storage engines like MyISAM.

Backup to tape was introduced to allow users to send a copy of the database to a secondary storage completely distinct from where the data is stored. Snapshots are located on the same storage as the file system. In case of disaster scenario, for example, a fire in the computer centre, the data would be lost.
Backup to tape requires contacting DBoD admins, as tape accounts are not automatically generated. Backup to tape can only be done once a week. Binary logs are also sent to tape in that case.

The snapshot technology allows to perform backup and restore operations, which are completely independent from the database vendor. Recovery operations remain vendor specific.

3.2 Restore and recovery till a certain point in time.
Two basic restores can be done via DBoD interface (see figure 7).

![Figure 7. Restore and recovery using snapshots and binary logs.](image)

A basic restore can be done just by setting back the database at the time a snapshot was taken. When we need to replay some binary logs till a certain point in time, this is what we have called Point In Time Recovery (PITR). This is translated on a few input boxes in DBoD interface (see picture 8).

First of all, a restore to a certain snapshot can be done only if binary log files exist; otherwise the snapshot is not selectable.

![Figure 8. Restore via DBoD interface, towards a snapshot with or without replaying binary logs. The latter case would be a point in time recovery.](image)

When taking a snapshot, the system hardcodes time and binary log index, from which transactions will be needed, for example `snapscript_<date>_<time>_<index>`. If for a given snapshot the binary
log does not exist anymore, - i.e. it was deleted by instance expiration policy - the snapshot cannot be selected, because the necessary binary log(s) to perform crash recovery do not exist.

The Point In Time Recovery is always done by restoring to the closest snapshot and then replaying binary logs until the required point in time.

3.3 Start/Stop an instance
Starting and stopping the RDBMS are the two most basic operations a user can perform on his/her instance, see figure 9. Both operations extract all the information required for their execution from the IT-DB management infrastructure and the corresponding instance definitions.

![Image of user interface for start/stop instance]

**Figure 9.** User view after executing an instance start-up command. The view shows user action controls and the log window showing the results of the command execution.

3.4 File management
In order to allow DBoD users to manage important files related to their instances, file management functionality is provided from the web interface. Using the window in Figure 8, which pops up when the user clicks on the file management button, users are able to upload/download configuration files for providing long standing changes to their system configurations, and also to download database logs.
Figure 10. File management window.

Downloading a configuration file is a simple process. The user selects the file to download and clicks on the download button. A DBoD web service is called using Apache Axis [7], providing the configuration file to the server, which is then provided to the user.

When a configuration file is uploaded to the server, a new job is created in the management database, and the content of the file is stored in a CLOB, character large object, column within a table containing job parameters. The instance is set to a job pending state. The job execution daemon fetches the job parameters, included the candidate configuration file and executes a parsing and template matching step. In this step, every configuration parameter for the server is checked to ensure that its value is within proper valid limits and won’t affect negatively the behaviour and performance of the system. The set of parameters resulting after enforcing restriction on the user provided configuration file is then passed to the job execution sub-process which will overwrite the configuration file in the target system, making the changes available after the next restart.

The file management window also offers the possibility to download slow queries logs for an instance. The slow query log consists of SQL statements that took more than long_query_time (a MySQL configuration variable) seconds to execute.

In order to make the download of a file faster, the log files are rotated when the size of the current file reaches 20 MB. When this happens, the file is compressed and the log is rotated, keeping a maximum of 30 compressed log files nowadays.

3.5 System Upgrades

When a new release of the supported RDBMS is available and the service administrators consider that it provides improvements considered necessary, for example for security reasons, the upgrade procedure between the currently supported version and the new one is tested and automatized as a component integrated in the CERN IT DB toolset management architecture. Once the upgrade procedure is integrated this way, it is made available as an action to the instance owner through the WEB interface (see figure 11). Instance owners are notified via email of the existing upgrade and the time window during which they can freely apply said operation when is more convenient. If by the end of the upgrade time window the upgrade operation has not been applied, the upgrade will be applied by the service administrators. The idea is to minimize the number of coexisting RDBMS versions supported by the service as a way to reduce operation complexity.
3.6 Monitoring

DBoD uses an in-house developed system called RACMon to provide users with significant metrics of the performance of the instances they are authorized to manage. The metrics’ list can be easily extended, if users request to add additional ones. The data is sampled every 6 minutes and stored in RACMon’s database. The detailed monitoring architecture has been described under point 2.3.

RACMon provides raw data to the web application, which is then rendered to users using the Google Visualization API, see figure 12. The use of this API allows the development of a more intuitive interface, which allows users to filter the range of dates they want to display, and also the specific value at a given point in time.

Figure 12. Metric plot for a MySQL instance.

A link to LEMON[3]monitoring is also provided to users in order to allow them to monitor the virtual machines in which their instances are running, this can help them identify performance characteristics of their workload.

4. Conclusion

The DBoD attempts to satisfy new database needs from the CERN user community. It aims to provide an enterprise level service to a new set of user cases, which cannot naturally be attended by the IT-DB Oracle support service. The DBoD provides a solution to applications that require a MySQL backend, or to applications that require a standalone database instance, as they will be managed by a third company, for example.

The DBoD empowers instance owners to do some DBA operations like tuning their instance, backup, recovery, monitoring, etc. By relying on our user community we expect to be able to scale and
evolve the service to satisfy mainstream needs. Along those lines, the service is researching high availability solutions, such as replication or clustering. This is a new degree of complexity and a step that needs to be done with care, as it also imposes new demands on the underlying technology.

This technology choice is based on actual resources and extensive experience running our systems. Those evolve naturally following IT trends. As much as possible, the service intends to integrate with the CERN IT infrastructure following its standards, like those for server configuration or network infrastructure. It has been designed in a way to offer API to integrate with new Openstack initiative.

After a few months running, we have already counted 15 MySQL production instances. This has allowed us to gain experience on MySQL and get feedback in a number of aspects, like performance monitoring and tuning, interface usability and desired new features, to which the DBoD service has given reply. Further work on the virtualization platform will allow us to accomplish automation of the DBoD instance creation.

We expect to be able to keep serving the database demand of the CERN user community!

References
[1] Proceedings of the 7th International Conference on Very Large Databases. 19333 Vallco Parkway, Cupertino CA 95014: Tandem Computers. pp. 144–154
[2] OracleVM, http://www.oracle.com/us/technologies/virtualization/oraclevm/index.html
[3] Xen, http://www.xen.org/
[4] Lemon – LHC Era monitoring, http://lemon-monitoring.web.cern.ch/
[5] Quattor - fabric management for grids and clouds, http://quattor.sourceforge.net/
[6] Oracle Enterprise Manager 12c, http://www.oracle.com/technetwork/oem/grid-control/overview/index.html
[7] ZK Framework, http://www.zkoss.org/
[8] Apache Axis, http://axis.apache.org/axis/
[9] Database connection pools, http://commons.apache.org/dbcp/
[10] Java Naming and Directory Interface, http://www.oracle.com/technetwork/java/jndi/index.html