# Database on Demand: insight how to build your own DBaaS

**Ruben Gaspar Aparicio, Ignacio Coterillo Coz**

European Organization for Nuclear Research (CERN), CH-1211, Geneve 23, Switzerland

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

**Abstract**. At CERN, a number of key database applications are running on user-managed MySQL, PostgreSQL and Oracle database services. The Database on Demand (DBoD) project was born out of an idea to provide CERN user community with an environment to develop and run database services as a complement to the central Oracle based database service. The Database on Demand empowers the user to perform certain actions that had been traditionally done by database administrators, providing an enterprise platform for database applications. It also allows the CERN user community to run different database engines, e.g. presently three major RDBMS (relational database management system) vendors are offered. In this article we show the actual status of the service after almost three years of operations, some insight of our new redesign software engineering and near future evolution.

## 1. Introduction

Today's constellation of DBaaS (database as a service) broadens to not only provide Relational Database Management Systems but other type of databases like NoSQL (not only SQL) or newSQL, the latter keeping the ACID approach of traditional relational databases but with a share-nothing architecture to scale out. These technologies are getting their momentum with some technologies getting more popular than others e.g. MongoDB, a document oriented database, or TokuDB in the newSQL arena recently bought by Percona [1]. Major RDBMS vendors also offer more and more integration with NoSQL like systems, which dictates that NoSQL, more than a replacement of RDBMS systems, will be complementary to those bringing schema flexibility and scalability to a transactional engine, so developers will be interacting with both system types in a given application. In the High Energy Physics community there is a significant interest and several experiments are testing these frameworks as an alternative to RDBMS. Time is required to see the real use cases that could be better addressed by those technologies.

PostgreSQL is gaining major presence also in cloud environments [2] and it presents as a strong competitor of MySQL and Oracle databases, with more and more open source projects choosing PostgreSQL as their backend. About a year and a half ago, the Database on Demand (DBoD) service decided to add PostgreSQL to the platform, validating the flexibility of the architecture to add new RDBMS suites always as single instance databases.

DBoD has been running for the last three years with an increasing number of instances being created every month, covering CERN community needs in terms of RDBMS databases as a complement to the traditional Oracle service. DBaaS shows to be a valid model for our clients. For the time being, DBoD instances are mostly granted to IT or HEP community services, therefore the lifespan of an instance is strongly bound to its application/service. Nevertheless the platform is getting ready to deal with a more dynamic turnover, a consequence of our migration to an Agile infrastructure being able now to use OpenStack VM (virtual machines), bare metal servers and likely containers in the near future. The whole DBoD infrastructure is managed following IT department configuration management tools and guidelines.

## 2. DBoD status

DBoD has consolidated as a CERN IT database service. Actual status in terms of the number of instances and the type of RDBMS can be seen in figures 1 and 2.

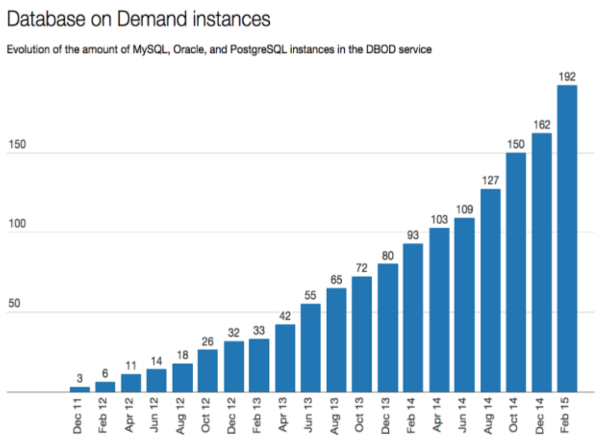


**Figure 1.** Total number of DBoD instances per month.

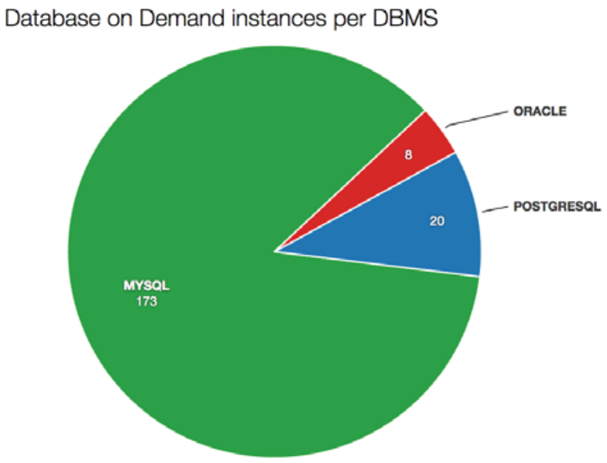


**Figure 2.** Distribution of DBoD instances per type.

The service hosts more than 200 instances nowadays, mainly MySQL followed by PostgreSQL and Oracle. The CERN Information Technologies (IT) department followed by the Physics and Beams departments are the major consumers of DBoD instances at the moment. The service runs MySQL Community Edition (CE) although different trends in MySQL (e.g. MariaDB [3], Webscale [4] and Percona server[5]) are being followed, and if the need arises for change to a different MySQL flavor, all technical requirements are already in place. A consultation on that topic was conducted last year and had as result to stay with the MySQL CE.

Except for Oracle, DBoD does not run with vendor support. This is well explained in DBoD Manifesto [6] and the DBoD user community has not indicated that it has concerns about it. Nevertheless, the DBoD service would certainly benefit from having some vendor support for all options offered that would help our user community in case of difficult problems either in data manipulation, corruption or hitting possible bugs of the software.

Several major changes have been performed last year: the migration to the new CERN Agile Infrastructure, extending our monitoring to use AppDynamics for databases [7] and upgrading our hardware to new servers and storage. These major changes, together with new features being implemented, are described in the next sections.

## 3. Service architecture

This section presents the current architecture of the Database on Demand service, and comments on the future plans and evolution of some of its components.

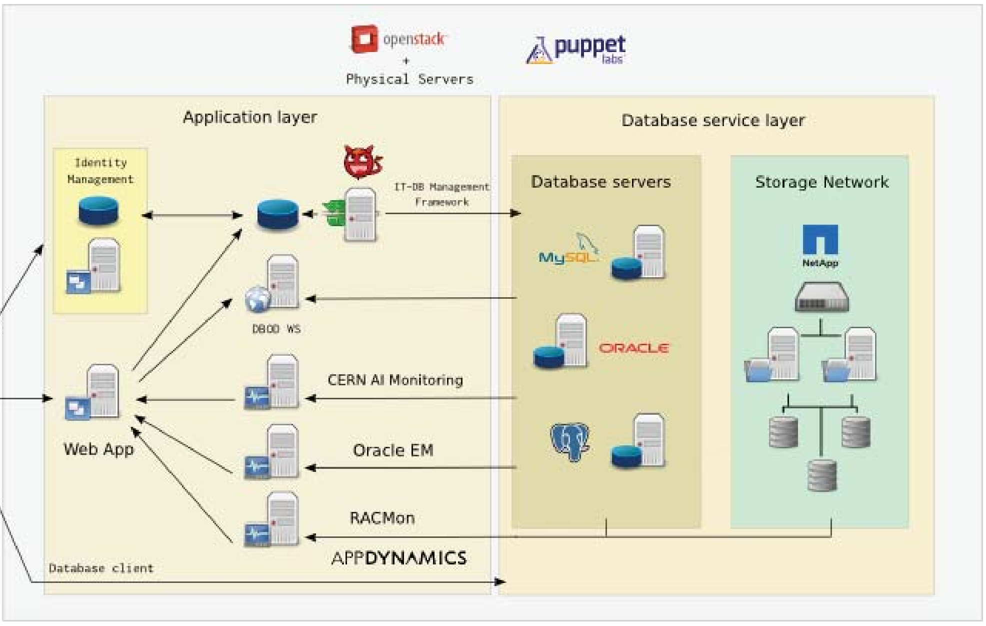


**Figure 3.** Simplified representation of the current architecture of the CERN Database On Demand Service.

Figure 3 shows a simplified view of the service architecture as of the moment of writing this document. The so called database service layer of the service contains both the database servers (either physical or virtual), and the storage systems attached to them, holding separate storage volumes for data and log files. The application layer of the service provides users with access to automatic managing operations to their database (automatic backup and recovery, configuration management, etc. [8]) as well as to data extracted from the different monitoring solutions supervising the relevant components of the infrastructure such as the RDBMS or the hosting server. All the components in the infrastructure are configured using Puppet according to CERN configuration service guidelines [9].

A Web Services API exposes certain information fetched from the hosting server for each instance (e.g. the list of valid snapshots) which is required on the service web application to enable some the functionality.

The web application presents the user with a general performance monitoring view, integrating a selection of metrics from the several tools employed to monitor each of the infrastructure different components, as well as providing links to each of the specific monitoring tools interfaces. Our approach to monitoring is to use the right tool for each case, and following it we employ several tools with different purposes. The CERN AI monitoring platform takes care of registering the health status of servers (both physical and virtual), while database specific solutions like Oracle Enterprise Manager and AppDynamics are employed to monitor the different database solutions offered by the service, providing access to our users to SQL level monitoring, profiling and analysis. RACmon is a CERN database services group in-house developed monitoring framework which focuses mainly on service availability monitoring and alarming, while complementing the functionalities of the previously mentioned tools.

With this current architecture, a user requesting a new instance has to first request the creation of a new resource under the CERN Identity Management system (FIM) and then request the new instance to the DBoD service which will be afterwards linked to her FIM request. The FIM service interacts with the DBoD service at the database level accessing directly the DBoD application database. This database is also accessed by the DBoD web application and the job dispatching daemon that manages the execution of user initiated activities on the database instances using a custom management framework from the CERN database services group.

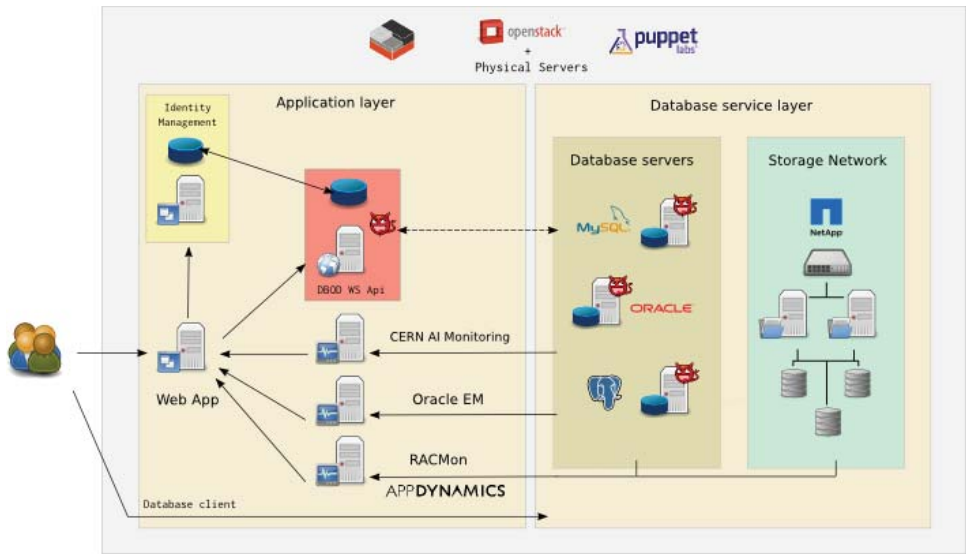


**Figure 4.** Simplified representation of the planned future architecture of the CERN Database On Demand Service.

Figure 4 shows a representation of the planned (may still be subject to change) architecture for the application layer of the service, which aims to simplify some of the relationship between the components and also facilitate the procedure of instance creations for the user.

In order to simplify the process of instance creation from the point of view of the user, the plan is to integrate the required resource request on the DBoD web application workflow, and use the FIM API to create the request on behalf of the user.

The second main objective is to consolidate the DBoD application components data access layers towards a single entry-point, by having a Web Services API which will be accessed from all components (web application, Job dispatching daemon, configuration tools, etc.) and will present an homogeneous access point to the different data back ends the system relies on (application database, LDAP directories, DBOD WS, etc.). The reason to do this change is ensure a better future evolution. In the current implementation data access code for each data backend in each of the components (generally this means also different programming languages) needs to be maintained separately, and thus the planned solution intends to remarkably reduce the code base footprint and complexity.

Another change, (this one still on the design phase) which would contribute to further simplify the elements of our infrastructure is to re-implement the job dispatching daemon as a multi agent system. The basic idea here is that once the part of the code which interacts with the database (which is how the daemon gets the list of commands to execute from the web application) is taken out of the daemon and moved to the WS API; what remains in functionality is basically the implementation of a task queue which serves as a middleware component to interface with the IT-DB Management framework mentioned in Figure 3. In fact, this daemon-management framework pair can be substituted by a message broker (i.e. ActiveMQ, RabbitMQ) and consumer agents running locally on the database servers, contributing additionally to simplify configuration requirements on the infrastructure servers while also improving security (as this local agents would need less access rights than the current implementation requires).

Finally, at the lower levels of the infrastructure, the use of Linux containers is looking very promising. They will be introduced to impose CPU and memory quotas for running multiple RDBMs in highly consolidated servers. The objective is to achieve isolation between the instances running on the same server imposing resource quotas to avoid resource hoarding by high activity databases.

The effort in redesigning and re-implementing the different components is being carried under a Free Software License (GPLv3) and can be followed in the project OSS repositories [10][11].

## 4. New hardware: servers and storage

In the last six months the DBoD service has undertaken a smooth migration towards new servers and a new storage. On the server side we have moved from Dell PowerEdge M610 with 48GB RAM and 8 cores Intel(R) Xeon(R) CPU E5630 running at 2.53GHz to Transtec servers with 128GB RAM and 16 cores Intel(R) Xeon(R) CPU E5-2650 running at 2.00GHz. DBoD instances intended for test and development are deployed using OpenStack virtual machines from our private cloud running Scientific Linux CERN, and production instances run on bare metal shared servers running Red Hat Enterprise Linux 6. For a very limited set of highly critical instances a cluster solution based on Oracle Clusterware [12] is in place.

Servers have undergone a scale-up. New machines are much more powerful and can accommodate many more instances. To protect resources among instances on the same server, the DBoD service is studying the introduction of Linux containers [13] and related technologies.

On the storage layer we have moved from isolated high availability pairs to a clustered storage network provided by NetApp clustered ONTAP [14] which scales out our storage infrastructure. DBoD instances are running in two major clusters of 14 and 8 controllers respectively. On top of the stability of the Network File System (NFS) protocol, NetApp server implementation, the clustered Data ONTAP adds advantages such as a global namespace, transparent file system moves and advanced networking. These allow to perform many interventions transparently (e.g. moving a file system from a SATA storage to a SSD one to face some more demanding database workload).

As access to our databases data files is done via NFS protocol, especial care must be taken in setting the right mount options. There is extensive documentation on this topic, especially for MySQL and Oracle [15], but there is almost none on PostgreSQL as they are normally not used together. These are the mount options we use on our single instances for MySQL and PostgreSQL:

```
--For data, binary logs and WALs
controller:/ORA/dbs0X/CPGSQL on /ORA/dbs0X/CPGSQL type nfs
(rw,noatime,bg,hard,nointr,tcp,vers=3,timeo=600,rsize=65536,wsize=65536)
```

Backup and recovery and (soon to be introduced) cloning functionalities are based on NetApp snapshot technology [16]. Snapshots are located in the same storage pool as the data. Initially, they do not consume any space and just after block changes are applied to the active file system, their space consumption accounts on a private snapshot area. Triggers are set to automatically clean old snapshots if certain size threshold is overpassed. The number of snapshots is monitored to ensure that at least one valid point of recovery for the instance is always available.

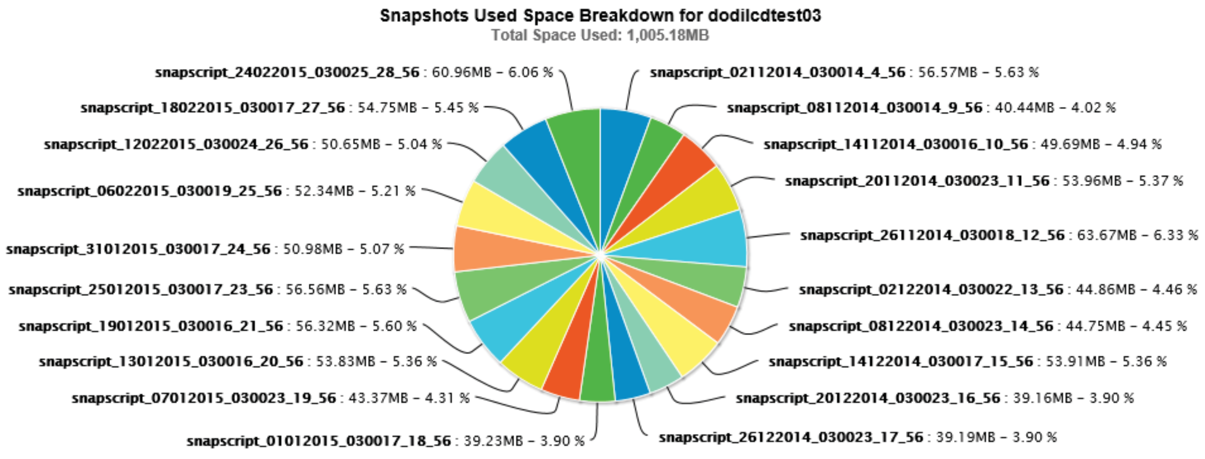


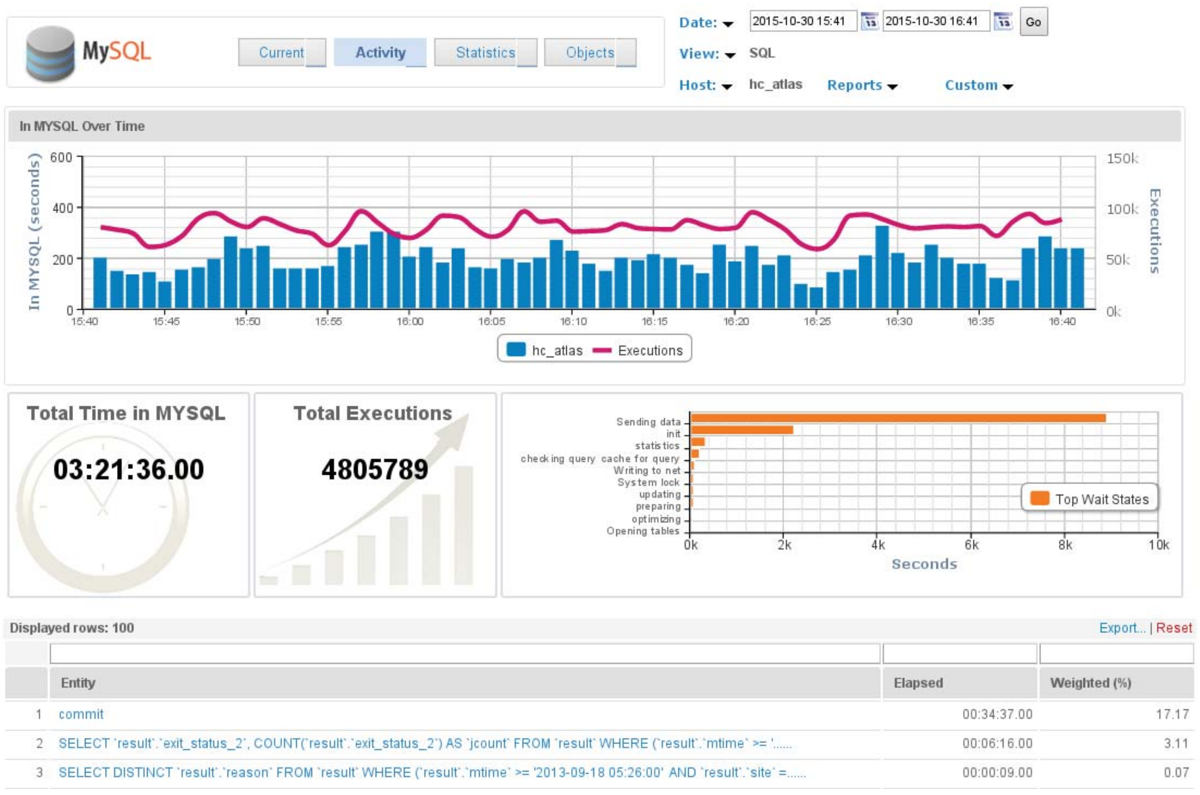
**Figure 5.** AppDynamics view of backups of a MySQL instance and their space occupancy.

## 5. New monitoring tool: AppDynamics

Effective monitoring is critical for a DBaaS where users have the obligation to analyze and solve performance issues within their instances. It is also expensive for the DBaaS provider to implement such a tool that requires deep knowledge of the different database vendors and adapts monitoring to their evolution. After a few months researching all possible tools, AppDynamics was selected as the best tool min the market at that time.

AppDynamics is being used since October 2014 by DBoD administrators. Performance analysis of MySQL and PostgreSQL instances and basic monitoring is done with this tool. Access to the user community was introduced around February 2015. Feedback has been very positive. It has allowed DBoD service to outsource most of its monitoring functionality.

Nevertheless one major inconvenient has been spotted. AppDynamics does not provide authorization and access is based on a single active directory e-group. That means that all owners of DBoD instances can look to performance analysis of any DBoD instance. The current conflictive case is the following: due to its nature, long running transactions (defined by the application as those taking longer than one second in a period of a minute), are tracked by AppDynamics providing the user the ability to inspect execution plans or weight a particular SQL statement against total DB time for a given interval. This is of extreme importance when tackling performance analysis and AppDynamics provides access to this view with a single click, (see figure 6). The caveat is that, if the application has not been carefully designed, some SQL statements could contain delicate information that should not be visible e.g. certificates or passwords, which, due to lack of fine-grained access control to the monitoring views, allows users to look into the activity of databases they do not own. This issue has been reported to the software vendor and it is expected it will be addressed in a future release. From the developer point of view this security concern can be solved by means of using bind variables, where those sensitive values are replaced by a meta-character.



Displayed rows: 100 Export... | Reset

| | Entity | Elapsed | Weighted (%) |
|---|---|---|---|
| 1 | commit | 00:34:37.00 | 17.17 |
| 2 | SELECT \`result\`.\`exit_status_2\`, COUNT(\`result\`.\`exit_status_2\`) AS \`jcount\` FROM \`result\` WHERE (\`result\`.\`mtime\` >= '...... | 00:06:16.00 | 3.11 |
| 3 | SELECT DISTINCT \`result\`.\`reason\` FROM \`result\` WHERE (\`result\`.\`mtime\` >= '2013-09-18 05:26:00' AND \`result\`.\`site\` =...... | 00:00:09.00 | 0.07 |

**Figure 6**. AppDynamics view of SQL activity

## 6. New features: cloning and replication

In this section, it is briefly described two new features being developed in our platform. Within the last years, our user community has expressed several possible use cases in which clone technologies are particularly well suited. For example, a DBoD owner willing to know before executing a recovery the contents of a particular snapshot, or testing on a clone a possible schema upgrade or some changes in a query access path like a new index. All these use cases can be addressed satisfactory with a clone. Our implementation is relying on the storage feature NetApp FlexClone [17]. I a few words a FlexClone is just a snapshot with a read write layer. It is particularly efficient, because it reuses the same space as its parent file system, and just the difference will be accounted for.

Performance wise there is no big difference between a clone and its parent file system, as it can be assessed in the next figure, where the results of performing the same workload on a PostgreSQL database running in both a parent and a cloned filesystem are presented.

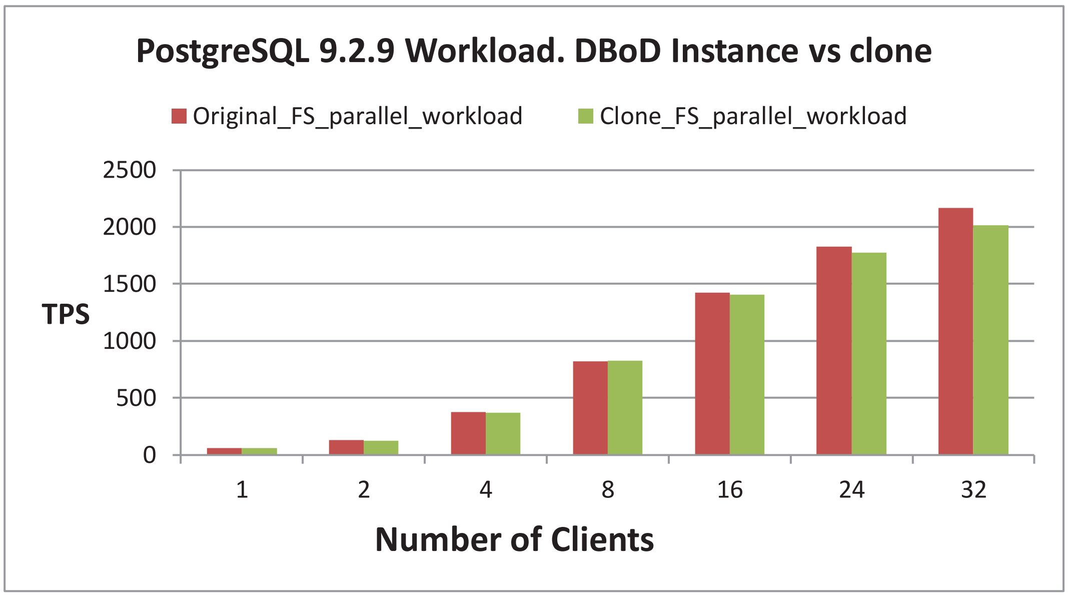


**Figure 7**. Small impact running simultaneous workload on PostgreSQL, on both clone and parent file systems. Bars represent Transactions per second (TPS).

The second major feature DBoD service has envisaged is replication. Replication has been used a number of times, especially on critical database service migrations. At this time, as with other data protection techniques it is intended to base this functionality on our storage infrastructure. This will abstract the functionality from any particular application layer. Nevertheless the functionality is still not there and changes on our platform and website are planned to provide replication in a traditional master/slave [18] format at the application layer, MySQL or PostgreSQL replication. To keep in mind that DBoD replication is planned to happen between clusters at CERN site, it is not a total disaster recovery solution using an external data center in Wigner. This is mainly due to high latencies between CERN and Wigner data centers.

## 7. Conclusion

DBoD has had an incredible acceptance by CERN community since early beginning which is reflected in growth in terms of instances. Many critical services are running on DBoD instances despite the limitations in user and vendor support. The service has evolved providing very attractive features [8] that have been enhanced by new monitoring tool this year. CERN community continuous feedback helps to define new functionalities to be added to the system. It is expected that cloning and replication will have a big acceptance.

Nowadays the service is undergoing a big transformation on its basic API and architecture to adapt better to dynamic cloud environments and possibly have an easier integration on PaaS (Platform as a Service) type of services.

**Acknowledgements**
To CERN Database services colleagues, especially to our actual core DBoD team that includes David Collados Polidura and Jose Andres Cordero Benitez and former colleagues Daniel Gomez Blanco and Dawid Wojcik and our encouraging section leader Eva Dafonte.

## References

[1] TokuDB & Percona, http://www.tokutek.com/2015/04/percona-acquired-tokutek-peter-zaitsev/ , accessed 30th October 2015
[2] Enterprise, DB http://www.enterprisedb.com/ , accessed 30th October 2015
[3] MariaDB, http://mariadb.org/ , accessed 30th October 2015
[4] WebScale, http://webscalesql.org/ , accessed 30th October 2015
[5] Percona server, http://www.percona.com/ , accessed 30th October 2015
[6] DBoD Manifesto, https://twiki.cern.ch/twiki/bin/view/DB/DBOnDemandManifesto , accessed 30th October 2015
[7] AppDynamics for databases, http://www.appdynamics.com/solutions/database-monitoring/ , accessed 30th October 2015
[8] Gaspar Aparicio R, Coterillo Coz I, et al. J. Phys.: Conf. Ser. 396 052034 2012, accessed 30th October 2015
[9] CERN Configuration Management System, https://configdocs.web.cern.ch/configdocs/ , accessed 30th October 2015
[10] BOD-core library repository, https://github.com/cerndb/DBOD-core , accessed 30th October 2015
[11] dbod-api repository: https://github.com/cerndb/dbod-api , accessed 30th October 2015
[12] Oracle clusterware, http://docs.oracle.com/database/121/CWADD/toc.htm , accessed 30th October 2015
[13] Linux containers, https://linuxcontainers.org/ , accessed 30th October 2015
[14] NetApp clustered ONTAP, http://www.netapp.com/us/system/pdf-reader.aspx?m=tr-3982.pdf , accessed 30th October 2015
[15] Jeffrey Steiner, Best Practices for Oracle Databases on NetApp Storage, TR-3633, http://www.netapp.com/us/system/pdf-reader.aspx?m=tr-3633.pdf&cc=us , accessed 30th October 2015
[16] NetApp snapshot technology, http://www.netapp.com/ca/products/platform-os/snapshot.aspx , accessed 30th October 2015
[17] NetApp FlexClone technology, http://www.netapp.com/as/products/platform-os/flexclone.aspx , accessed 30th October 2015
[18] Master/Slave replication, https://dev.mysql.com/doc/refman/5.6/en/replication-howto.html , http://www.postgresql.org/docs/9.2/static/runtime-config-replication.html , accessed 30th October 2015