Journal of Physics: Conference Series

PAPER • **OPEN ACCESS**

Conditions Database for the Belle II Experiment

To cite this article: L Wood *et al* 2017 *J. Phys.: Conf. Ser.* **898** 042060

View the article online for updates and enhancements.

Related content

- The Belle II experiment: fundamental physics at the flavor frontier
  Ivan Heredia de la Cruz
- High Level Interface to Conditions Data at Belle II
  M Ritter, T Kuhr, M Stari et al.
- Belle II distributing computing
  P Krokovny

This content was downloaded from IP address 131.169.5.251 on 29/11/2017 at 21:27

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 042060 doi:10.1088/1742-6596/898/4/042060

Conditions Database for the Belle II Experiment

L Wood, T Elsethagen, M Schram and E Stephan

Pacific Northwest National Laboratory, P.O. Box 999, Richland, WA, USA 99352
E-mail: *lynn.wood@pnnl.gov*

Abstract. The Belle II experiment at KEK is preparing for first collisions in 2017. Processing the large amounts of data that will be produced will require conditions data to be readily available to systems worldwide in a fast and efficient manner that is straightforward for both the user and maintainer. The Belle II conditions database was designed with a straightforward goal: make it as easily maintainable as possible. To this end, HEP-specific software tools were avoided as much as possible and industry standard tools used instead. HTTP REST services were selected as the application interface, which provide a high-level interface to users through the use of standard libraries such as curl. The application interface itself is written in Java and runs in an embedded Payara-Micro Java EE application server. Scalability at the application interface is provided by use of Hazelcast, an open source In-Memory Data Grid (IMDG) providing distributed in-memory computing and supporting the creation and clustering of new application interface instances as demand increases. The IMDG provides fast and efficient access to conditions data via in-memory caching.

1. Introduction

The Belle II detector [1] and the SuperKEKB accelerator are currently under construction at the KEK laboratory in Tsukuba, Japan. The aim of this next generation B factory experiment is to collect 50 times more data than its predecessor Belle [2] and to use this data to search for new physics in a variety of B meson, charm hadron, or lepton decays with unprecedented precision. This requires detailed information on varying calibration and detector conditions to be available when analyzing the data.

2. Database Design

The Belle II Conditions Database manages conditions data at run-level granularity. The time sequence of changing conditions are maintained by associating a specific payload with an "interval of validity" (IOV) during what that payload is valid. The same payload can be associated with multiple IOVs.

To define a valid set of IOVs and payloads when processing data, a structure called a "global tag" is defined. This consists of a list of IOVs and associated payloads which are considered valid for a given processing effort, as shown in Figure 1. Multiple global tags can be defined in the database, and it is expected that new global tags will be defined for existing experiment run ranges as new calibration methods are created.


Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

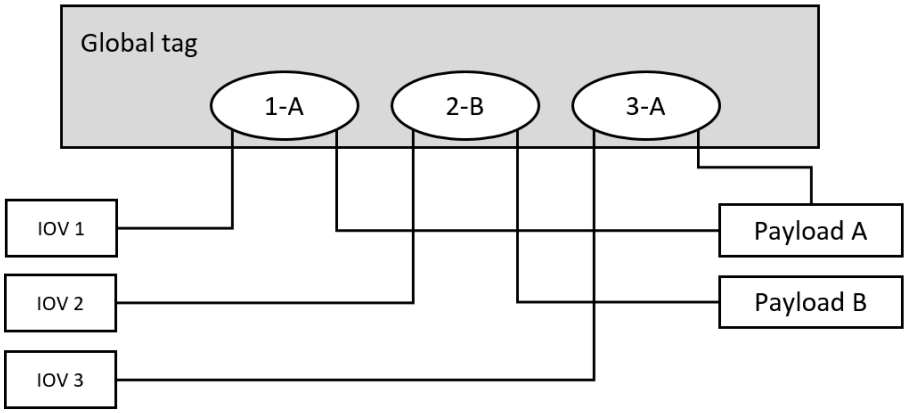

Figure 1. Relationship between payloads, IOVs, and global tags.

3. Database Server Implementation

The Belle II Conditions Database is designed as a representational state transfer (REST) service. Communication is performed by standard HTTP using XML and JSON data. The choice of a standardized REST API makes the client implementation independent of the actual database implementation details and allows for a simple and flexible implementation of clients in different programming languages. Details of the client software implemented for Belle II is available here [3]. The database itself is PostgreSQL [4], but the schema and procedures have been deliberately kept generic so other database applications could be used if necessary.

To keep the database small, the payloads only consist of references to files on a separate server. These "payload files" are currently assumed to be ROOT [5] objects by the client, although there is nothing in the database design that limits the data type. The use of ROOT objects allows for sub-run granularity to be implemented in the payload file itself if needed, and can provide support for payload type versioning.

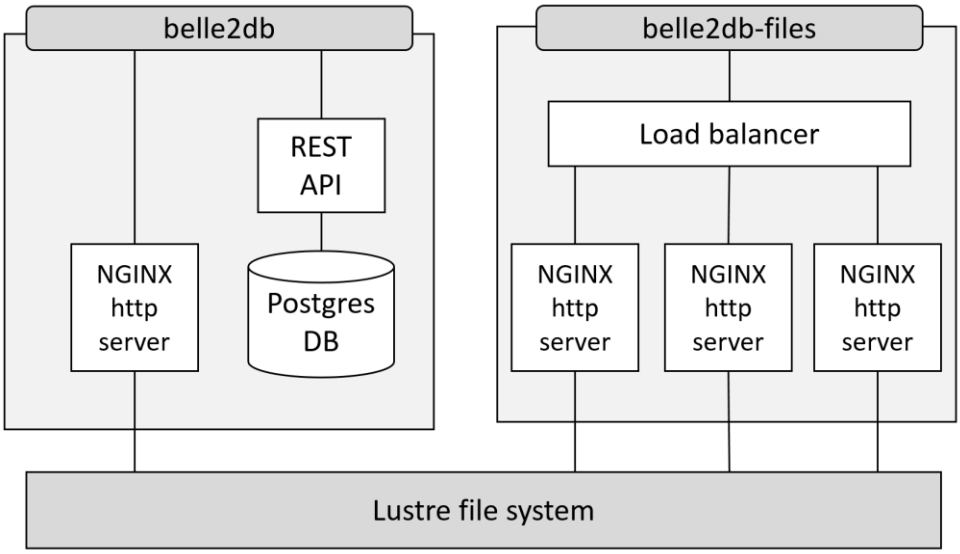

Figure 2. Current implementation of the database back-end.

The server back-end is implemented using industry-standard tools, and exists as multiple instances in the current implementation, as shown in Figure 2. One instance supports the REST API and database itself, while the other provides access to the data files. This is important since the access patterns for the two operations differ, so the two instances can be optimized separately.

The REST API interface is built from two commercially-available applications: Payara and HazelCast. Payara [6] is a Java EE application server optimized for production operations. The Belle II Conditions Database is using Payara Micro, which is a specific offering designed for operations in a modern containerized infrastructure.

HazelCast [7] is a Java-based in-memory data grid. Data is distributed evenly among separate nodes of a computer cluster, which provides horizontal scaling in both available storage space and processing power (see Figure 3). This provides multiple benefits, including caching of frequently-used data in memory, transparent scalability, and load-balancing to reduce the query load on the database.

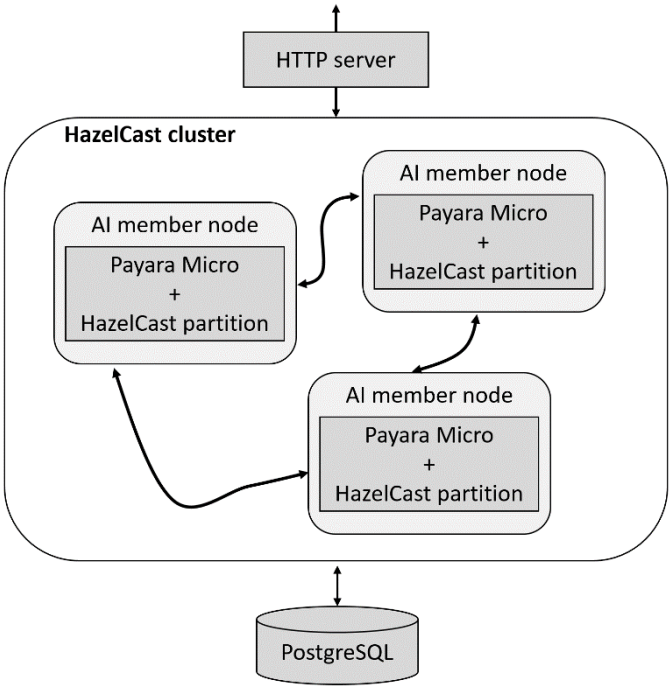

Figure 3. API implementation using Payara and HazelCast.

The database API is implemented using the Swagger [8] framework, which eases the effort of designing, building, and documenting REST APIs. Swagger provides a language-agnostic specification and editor, and can auto-generate documentation as well as API client and server stub code.

The file server currently consists of three NGINX [9] HTTP servers handled by a load balancer. NGINX provides load-balancing, session persistence, and advanced monitoring and management while delivering significantly higher performance than the Apache HTTP file server. The back-end file system is based on Lustre [10], an open-source parallel file system for HPC environments. The Lustre file system is shared between all back-end instances. This allows for payload file uploads to occur on the database instance, which helps to mitigate versioning issues between the database and payload files.

Each component in Figure 2 is implemented as a separate Docker [11] container, which reduces the resource footprint on the physical computing hardware by sharing the operating system between containers, as opposed to virtual machines which have separate operation system copies. This use of Docker containers also provides auto-restart functionality as well as making setup at other sites easier and more consistent.

4. Database Evaluation

The Belle II Conditions Database back-end has been evaluated with both direct access by grid-based Belle II Monte Carlo campaigns and directed testing by the database group. The directed testing was done with Gatling [12], an open-source load stress testing tool for HTTP servers which allows custom test design through Scala scripting and provides extensive reporting options. Testing of the database back-end was implemented by monitoring the usage patterns during Monte Carlo operations and then writing separate tests for the REST API and file server. The use of scripting allowed for much greater stress loads than was readily available from grid-based testing. The directed stress testing has been successful; an example Gating output is shown in Figure 4, demonstrating an average load of 180 payload file requests per second, with up to 10,000 open connections being supported simultaneously by the load-balanced HTTP servers. The total payload file size for a given run has been estimated to include several hundred files totaling 2-3 MB, although that may be updated as Belle II begins data-taking operations. With these estimates, this test corresponds to nearly half of the payload bandwidth for the expected full-scale Belle II grid-based analysis cloud of 100,000 nodes.

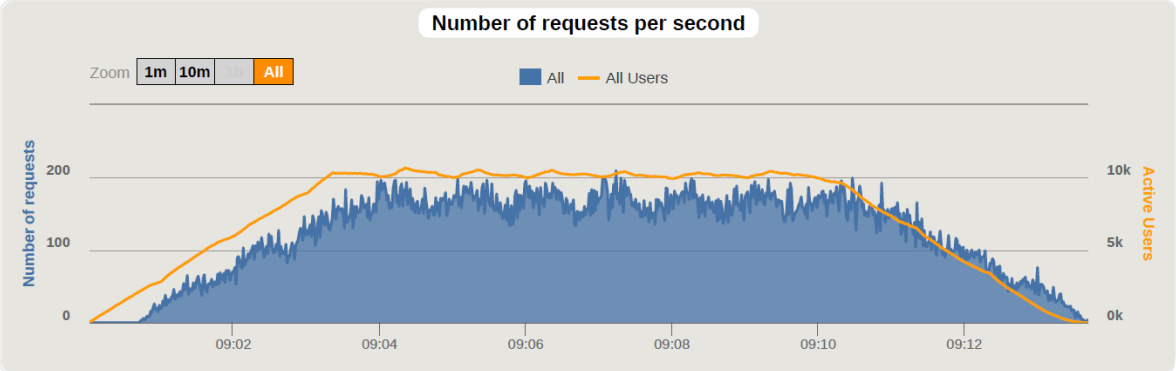

Figure 4. Example database file server evaluation, showing an effective 10,000 users.

5. Future Updates

Several areas of improvement are still planned:

- The addition of Squid [13] HTTP cache proxies before the HTTP servers would reduce the load on all other components for common accesses.

- Hazelcast supports cache clustering between remote sites, and will be evaluated as a means of supporting more localized database access worldwide.

- The PostgreSQL database is still a single instance and not currently scalable. If necessary, one option would be to migrate to the use of OpenStack Trove [14], which would provide "Cloud Database as a Service" functionality by allowing a master and multiple slave database instances. This would likely necessitate migrating from PostgreSQL to MySQL, but care has been taken to avoid PostgreSQL-specific implementation from the start.

- Authentication is currently not implemented, but is planned for database uploads at the minimum.

6. Conclusions

The Belle II Conditions Database is operational and serving the current Belle II grid-based Monte Carlo campaigns. The extensive use of industry-standard applications and tools has allowed it to be largely

supported by IT staff, as opposed to dedicated scientists, reducing cost and effort. Performance has proven more than sufficient for current loads, and planned updates are underway to bring it up to full data production requirements.

Acknowledgements
This work was carried out for the U.S. Department of Energy under Contract DE AC05 76RL01830 PNNL-SA-121968.

References
[1] Abe T et al. 2010 Belle II Technical Design Report Preprint arXiv:1011.0352
[2] Abashian A et al. 2000 The Belle Detector Nucl. Instrum. Meth. A 479 117
[3] M Ritter, T Kuhr and M Staric for the Belle II Software Group 2016 High Level Interface to Conditions Data at Belle II *also in this issue*
[4] https://www.postgresql.org/
[5] https://root.cern.ch/
[6] https://www.payara.fish
[7] https://www.hazelcast.com
[8] https://www.swagger.io
[9] https://www.nginx.com/
[10] http://lustre.org/
[11] https://www.docker.com
[12] https://www.gatling.io
[13] http://www.squid-cache.org/
[14] https://wiki.openstack.org/wiki/Trove