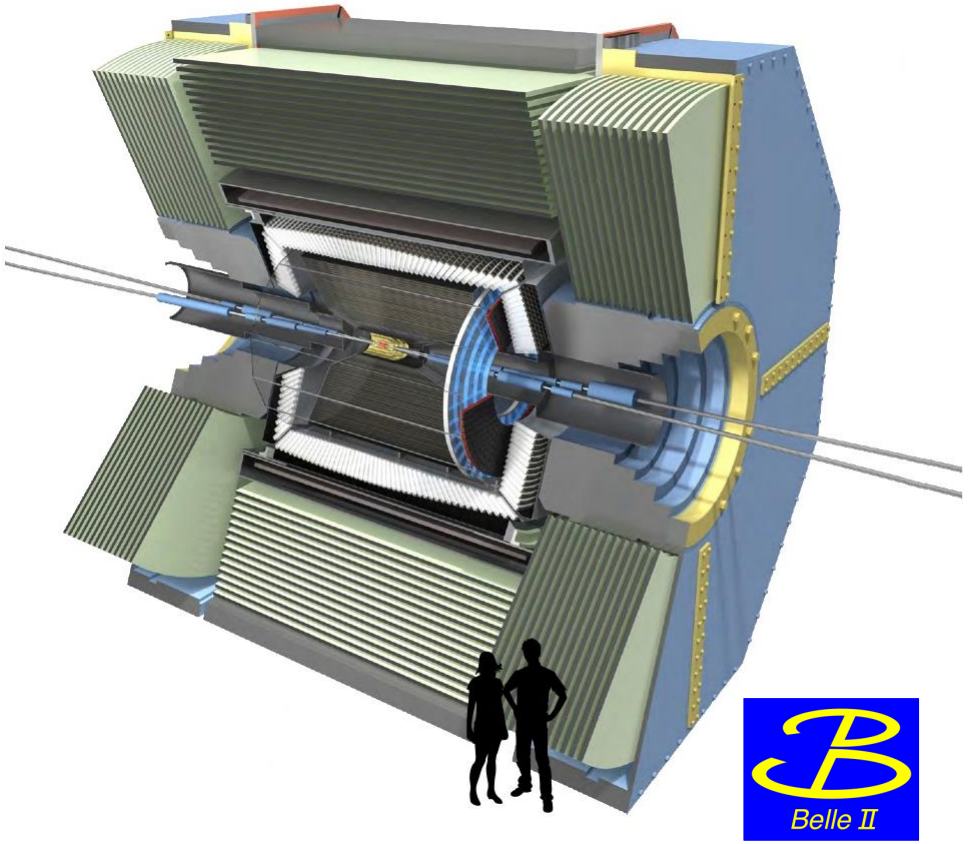
Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Conditions Database for the Belle II Experiment



LYNN WOOD, TODD ELSETHAGEN, MALACHI SCHRAM, ERIC STEPHAN

Pacific Northwest National Laboratory

22nd International Conference on Computing in High Energy Physics, October 2016

The Belle II Experiment at KEK Tsukuba, Japan

- The Belle II experiment is part of a broad-based search for new physics in the intensity frontier
  - precisely measuring and comparing with theory branching fractions, angular distributions, CP asymmetries, forward-backward asymmetries, and a host of other observables
- The SuperKEKB accelerator upgrade will provide 40x the luminosity of KEKB and 50x the data taken with Belle

Belle II Conditions Database

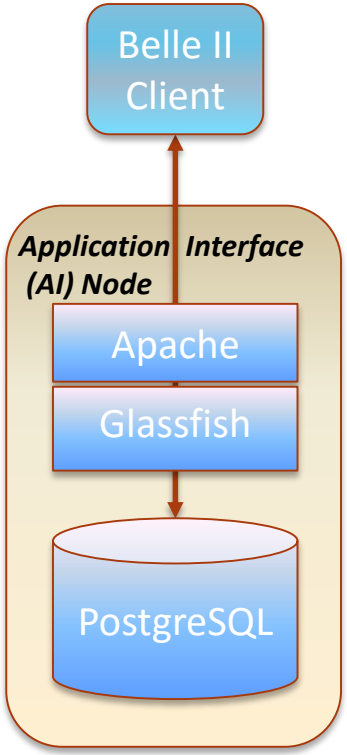
- Holds time-dependent status of detectors for data processing/reprocessing
  - Constants or time-varying parameters: calibration, geometry/alignment, etc.
- Development of current solution started in April 2014 → high-priority need
  - Focused on commercial off-the-shelf (COTS) solutions, not custom designs
- Quickly settled on **REST API** interface
  - All accesses through HTTP GET/POST/PUT
  - Limits users to allowed operations → safer
  - Much easier to implement DB design changes without affecting user code
  - Easy to scale with industry standard (HTTP) tools
- Also decided to keep actual payloads as **external files** outside of database
  - Keeps database small
  - Allows different scenarios for passing files

Overarching goal: make system supportable by **IT engineers**, not physicists

Prototype Back-End Design

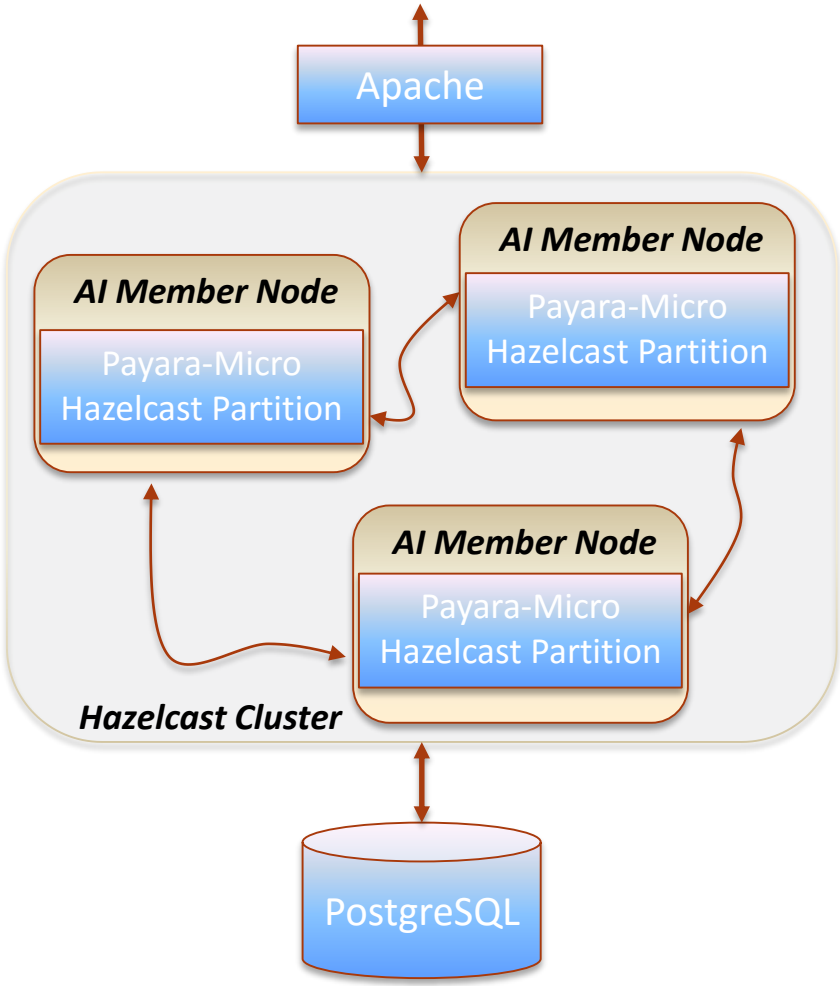
- ***Components:***
  - ***Apache HTTP server***
  - ***GlassFish*** – Java EE Application Server, runs on top of Apache
  - ***Database*** – PostgreSQL 9.3
  - ***File system*** – Used default (ext4) file system on server

- **Issues:**
  - Everything installed on a single node, which limits performance scaling to addition of CPU's, memory, etc.
  - Data access performance is constrained by disk I/O on the node
  - High availability is difficult!
    - node and its software components are all single points of failure



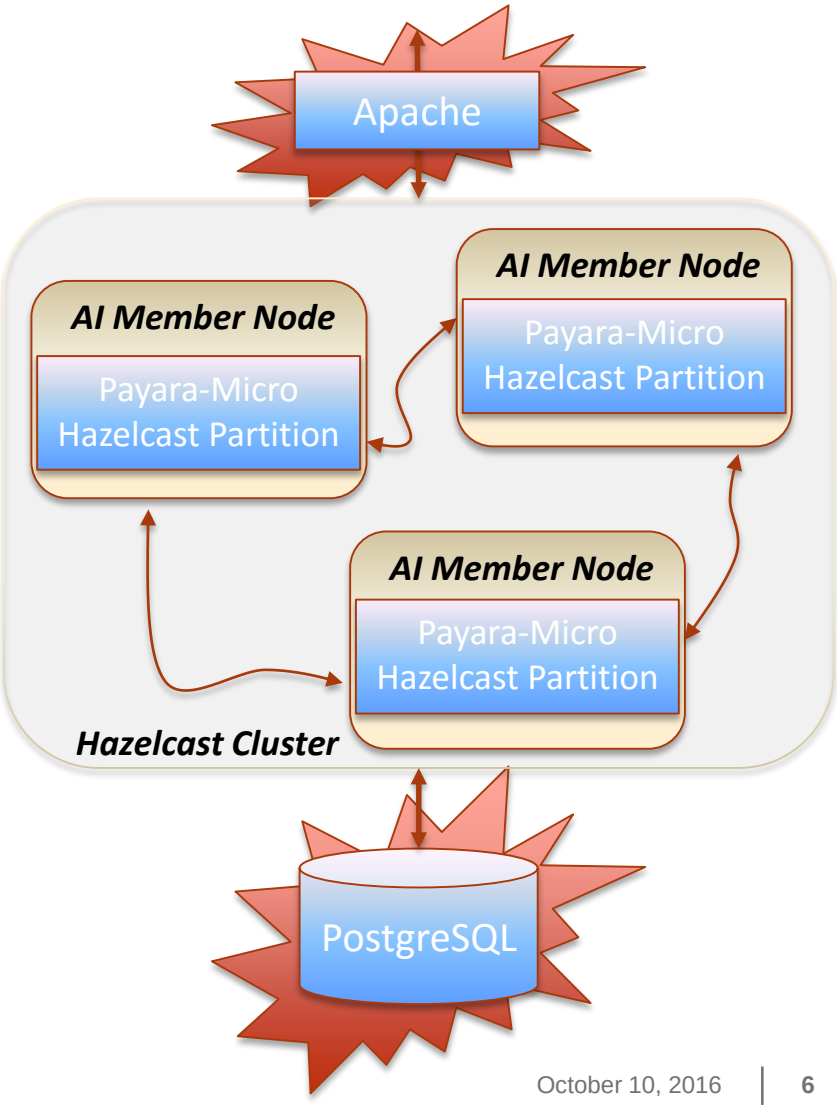
Current Implementation

- Apache HTTP server, file system still the same
- Moved from GlassFish to **Payara**
  - Essentially just GlassFish (i.e. still Java) with better support for production-level operations
- Added **HazelCast**, a Java-based in-memory data grid platform
  - Distributes data across nodes in a cluster
  - Scalable caching across multiple systems
  - Additional servers can be added/removed on demand
- Now have ability to scale API as needed
- Moved API from standalone into **Swagger** framework
  - Automatic generation of code, documentation and test cases



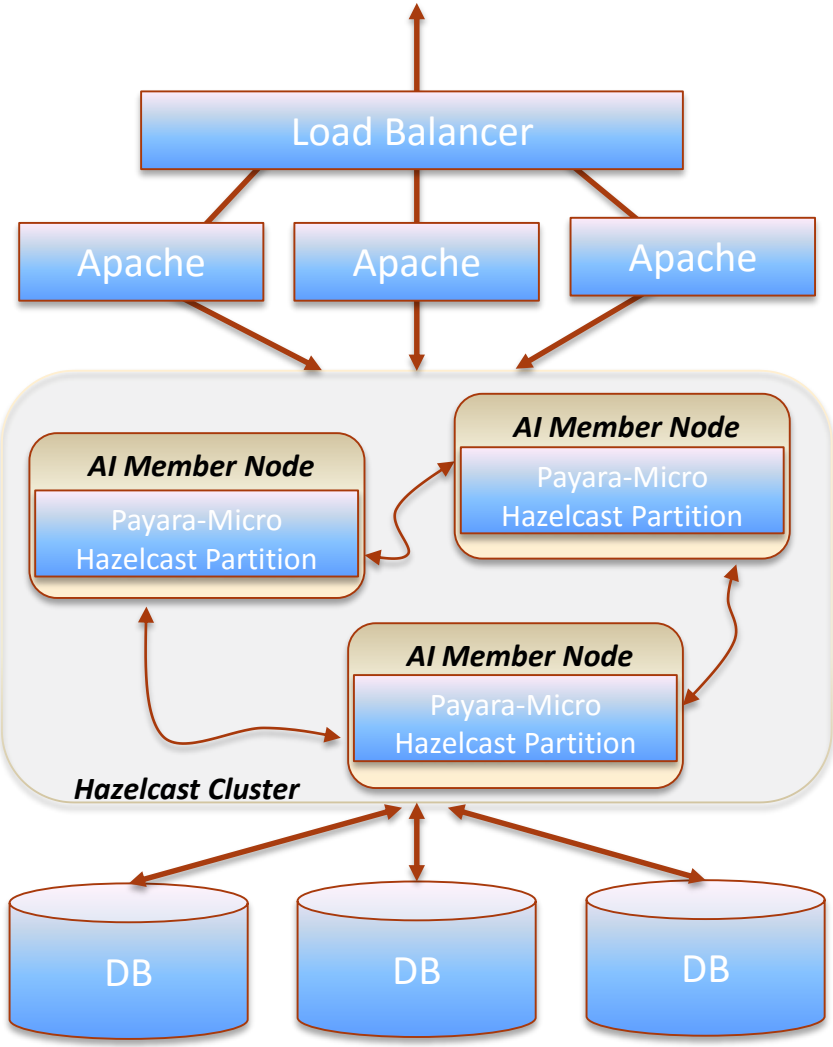
Current Implementation

- API is now scalable, but database and HTTP server are still a potential single point of failure
- Server is on a virtual machine (VM) on a physical system with other Belle II VMs
  - Have other VM's disk I/O affect database VM
- Only a single server is currently configured



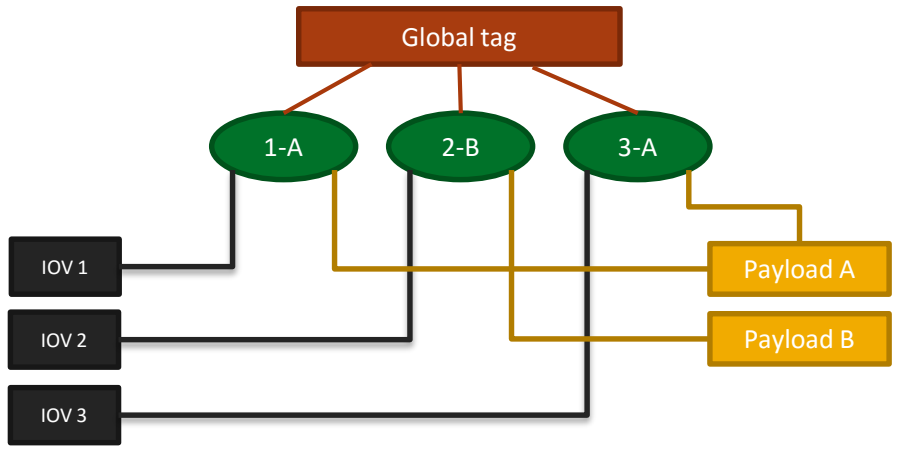
Planned Upgrades

- Add scalability to HTTP server by running load balancer and multiple servers (standard procedure)
- Implement database scalability by moving to **OpenStack Trove**:
  - "Database as a Service"
  - Load-sharing between databases
  - Replication of databases
  - May require move to DB other than PostgreSQL
- Addition of **Squid** HTTP proxy caching at the front



Global Tags, Intervals of Validity, and Payloads

- **Global tags** contain list of IOV-payload relationships
  - Global tag states: NEW, PUBLISHED, INVALID



- **Intervals of validity** designed for run-level granularity
  - Values that change more frequently are handled as a run-level object that stores finer time resolution

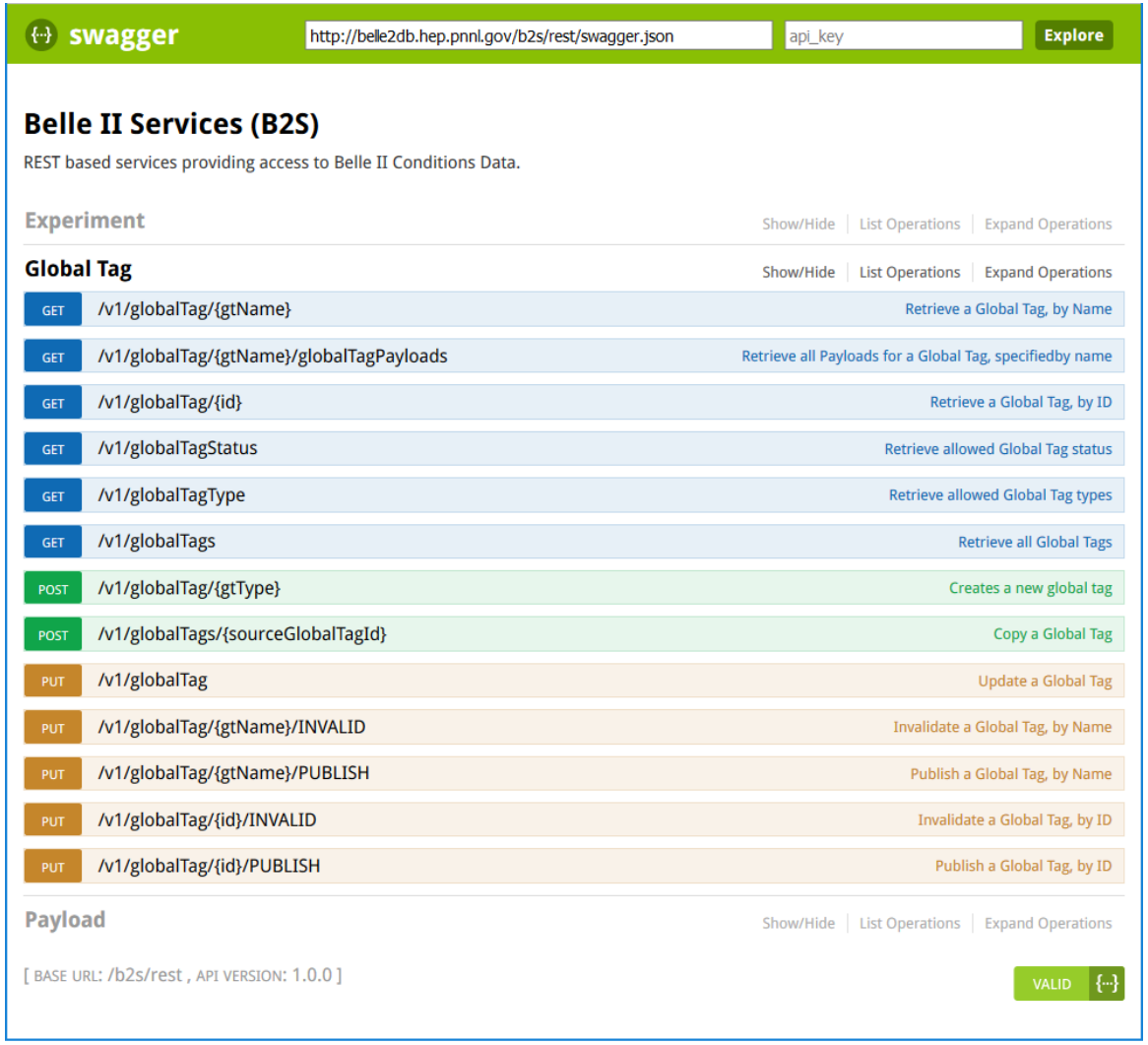
- **Payloads** in the database are referenced to a particular Basf2 software module
  - Contain URL referencing actual file
  - Higher-level software interface assumes ROOT objects, although DB itself doesn't care

- Three levels of access control are planned, but not yet implemented:
  - ***User*** Read access to database
  - ***Developer*** Can add payloads, IOVs, global tags
  - ***Coordinator*** Can change global tag status

Database API

- Simple HTTP GET, PUT, and POST operations
  - Can access via browser, command line (curl), or various software libraries
  - Returns data in JSON or XML format
- Example:

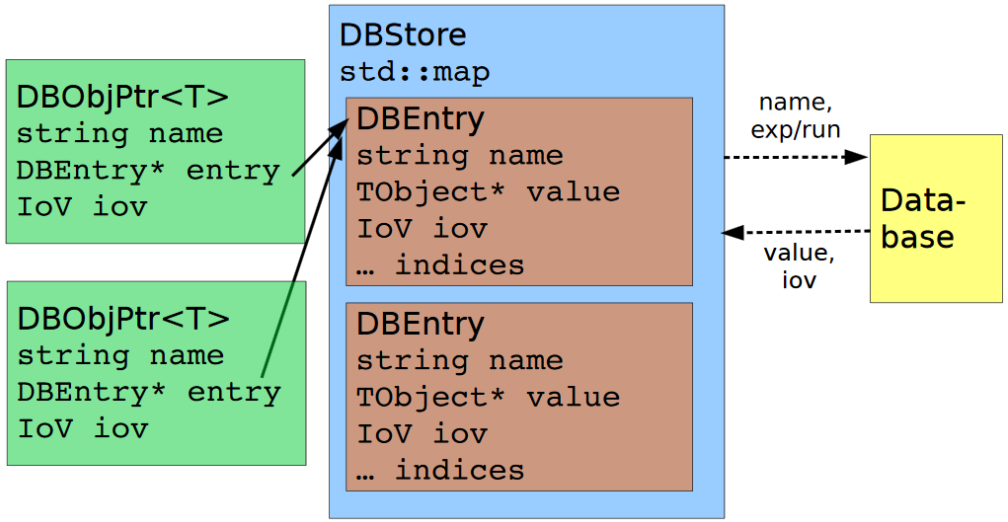
GET /v1/globalTag/{gtName}/globalTagPayloads



Analysis Framework (Basf2) Interface

- "DBStore" framework keeps database payload data similar to "DataStore" for event-level data
  - Keeps track of payloads and IOVs, updates them when necessary

→ Run dependency transparent to the user!



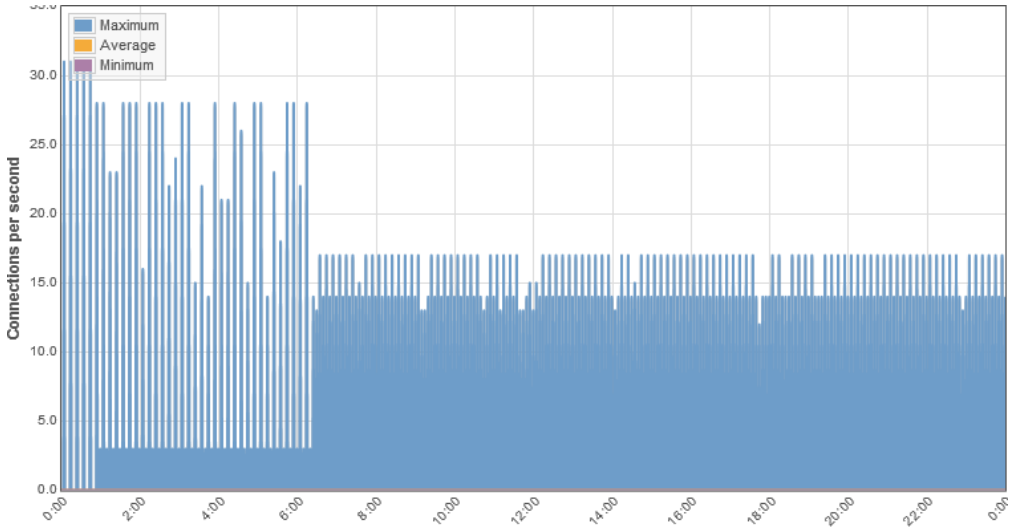
- Callback functionality for when an object changes
- Also supports intra-run dependencies for conditions that change more quickly
  - Currently based on event number
  - Could support timestamps as well
- Able to create chain of databases to access for overlays and/or fallback capability

See Martin Ritter's poster "**High Level Interface to Conditions Data at Belle II**" in Tuesday's poster session

Evaluation

- With “release candidate” database server in place, current focus is on evaluation of design
- Multiple methods being used:
  - Directed tests by database group
  - Addition of database access into MC runs
  - Connectivity/performance tests from Belle II Grid sites
  - Issues that arise during subdetector groups’ use
- Current load testing tool: **Gatling**
  - Open-source load testing framework
  - Scripted scenarios
  - Extensive reporting

- Several tools being used to monitor database health and performance
  - pgBadger: PostgreSQL log file analyzer
  - pgCluu: Active monitoring tool
  - Nagios: database availability
- Will bring other tools in as needed



Next Steps

- **Data**
  - All detector geometries in database by Nov 2016 in preparation for MC campaign
  - All detector constants in database by Nov 2016

See David Dossett's talk "**Status of the Calibration and Alignment Framework at the Belle II Experiment**" at 3:15pm

- **Integration between online/offline databases**
  - DAQ group needs conditions data for fast reconstruction at the HLT
  - Calibration group needs conditions data for real-time calibrations
  - Contents from DAQ databases (configuration, logging) needed to generate conditions data

- **Installation of OpenStack Trove, Apache load balancers**
  - Will go along with scaling up to multiple DB servers at PNNL

Conclusions

- The Belle II Conditions Database has been successfully implemented with industry-standard components
- Database is under “real-world” testing now by the Belle II collaboration
  - Directed testing by database group
  - Access from MC test jobs
  - Subdetector groups loading and accessing data
- Evaluation of current design’s performance will instruct next series of upgrades

Visit the PNNL booth for information about current research, lab strategies, and job openings!

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

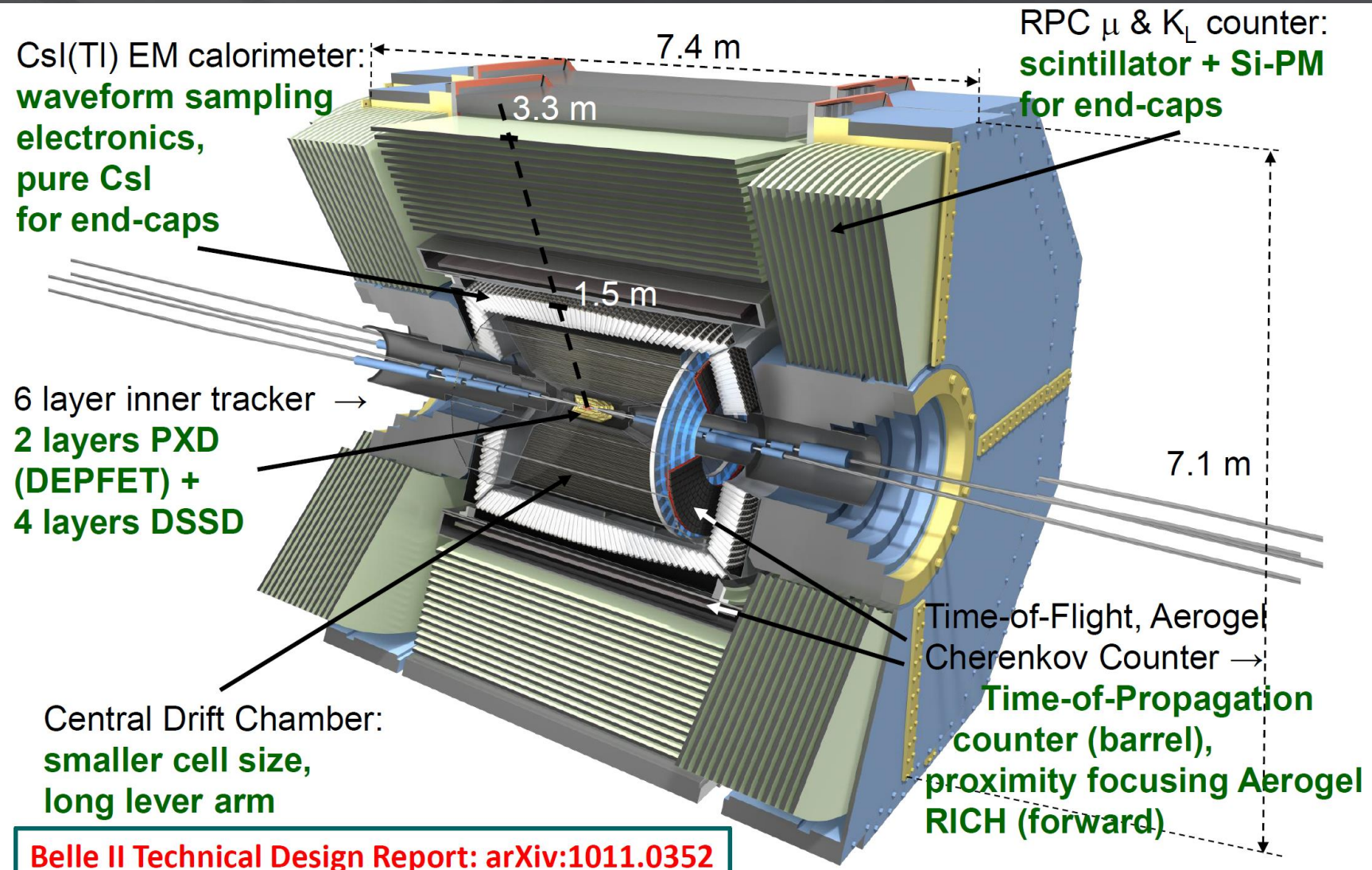
Thank you!

Belle II-related talks at CHEP 2016

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

- **Today (Monday):**
  - *Conditions Database for the Belle II Experiment* – 2:45pm
  - *Status of the Calibration and Alignment Framework at the Belle II Experiment* – 3:15pm
  - *Developing the WLCG Tier-1 center GridKa as topical center in a multidisciplinary research environment* – 3:15pm
  - *System upgrade of the KEK central computing system* – 3:30pm
- **Tuesday:**
  - *Highlights of the Belle II Computing – 11:45am*
  - *Development of stable Grid service at the next generation system of KEKCC* – 3:15pm
  - *High Level Interface to Conditions Data at Belle II* –Poster A
  - *Conversion of Belle's data to Belle II's data format* –Poster A
  - *BelleII@home: Integrate volunteer computing resources into DIRAC in a secure way* – Poster A
  - *A performance study of WebDav access to storages within the Belle II Collaboration* – Poster A
  - *A lightweight federation of the Belle II storages through dynafed* – Poster A
- **Wednesday:**
  - *Software Quality Control at Belle II* – 11:30am
  - *Integrating Prediction, Provenance, and Optimization into High Energy Workflows* – 12:30pm
  - *On-demand provisioning of HEP compute resources on cloud sites and shared HPC centers* – 12:30pm
  - *Context-aware distributed cloud computing using CloudScheduler* –12:45pm
- **Thursday:**
  - *An interactive and comprehensive working environment for high-energy physics software with Python and jupyter notebooks* – 11:15am
  - *DIRAC in Large Particle Physics Experiments – 2:00pm*
  - *Integration of grid and local batch system resources at DESY* – 2:45pm
  - Index files for Belle II - very small skim containers – *2:45pm*
  - *Migrating the Belle II Collaborative Services and Tools* – 4:30pm
  - *The Simulation Library of the Belle II Experiment* – 4:30pm

The Belle II Detector



Belle II Technical Design Report: arXiv:1011.0352

Payara is an application server for production Java EE applications

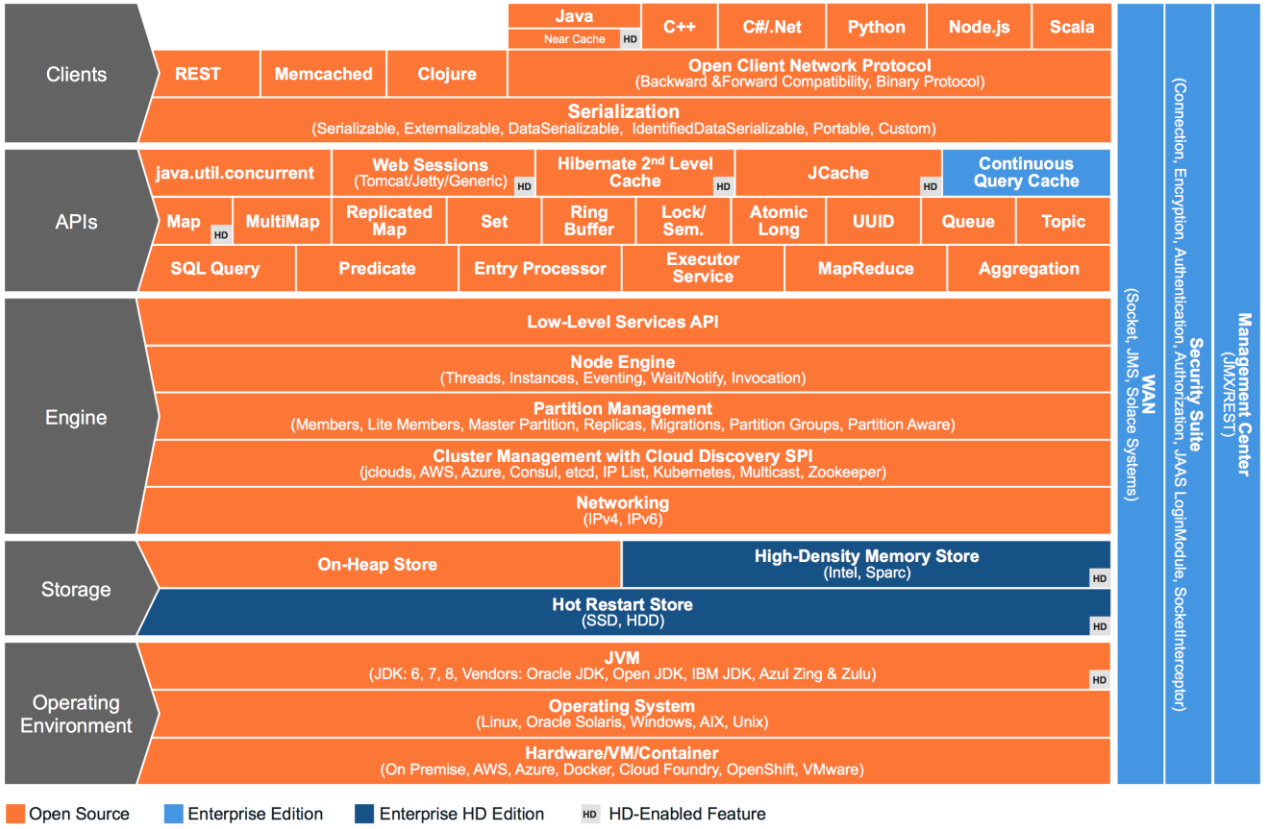
Payara Server is a drop in replacement for GlassFish Server Open Source Edition, with the peace of mind of quarterly releases containing enhancements, bug fixes and patches.

There is also Payara Micro, which enables you to run war files from the command line without any application server installation. It is small, < 70MB in size and incredibly simple to use. With its automatic and elastic clustering, Payara Micro is designed for running Java EE applications in a modern containerized/virtualized infrastructure, using automated provisioning tools like Chef, Ansible or Puppet.

Feature	GlassFish 4.x	Payara Server
License	Open Source	Open Source
Release frequency	Irregular	Quarterly
Releases in 2016	0	4
Patch releases	For versions < 4.0 only	• Monthly for support customers • Quarterly for community
Security Fixes	Infrequent	• Instant emergency & backported fixes for support customers • As soon as possible for community
Production Support	✗	✓
Developer Support	✗	✓
Component Upgrades (e.g. Tyrus, Mojarra)	Irregular	Quarterly
Supported IDEs	• Eclipse • Netbeans • Intellij IDEA	• Eclipse • Netbeans • Intellij IDEA
Caching tools	Shoal	• Shoal & JCache (open source) • Payara Scales (enterprise)
Automatic Clustering	✗	✓ via Hazelcast
Asadmin command recorder	✗	✓
Slow SQL Logging	✗	✓
Healthcheck Service	✗	✓
Request Tracing	✗	✓
Monitoring Logging	✗	✓
Microservices distribution	✗	✓ Payara Micro
Docker support	✓ Community provided	✓ Official images
IBM JDK Release	✗	✓ Payara Blue

The **Hazelcast** operational in-memory computing platform helps leading companies worldwide manage their data and distribute processing using in-memory storage and parallel execution for breakthrough application speed and scale.

Hazelcast is easy to work with and brings a highly resilient and elastic memory resource to all of your applications. At its core, Hazelcast is one of the most widely adopted open source solutions with tens of thousands of installed clusters and over 16 million server starts per month. On top of this popular open source platform, Hazelcast Enterprise HD and Hazelcast Enterprise offer licensed features for large scale deployments. Now you can free your data from slow, expensive, and hard to scale relational databases. With Hazelcast, your database remains the system of record, but bottlenecks disappear.



What is Swagger?

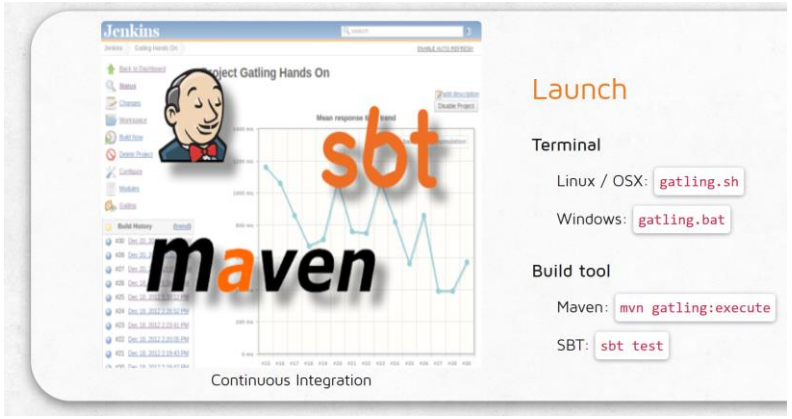
The goal of Swagger™ is to define a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Similar to what interfaces have done for lower-level programming, Swagger removes the guesswork in calling the service.

Technically speaking - Swagger is a formal specification surrounded by a large ecosystem of tools, which includes everything from front-end user interfaces, low-level code libraries and commercial API management solutions.

Gatling

Open-source load testing framework based on Scala, Akka, and Netty

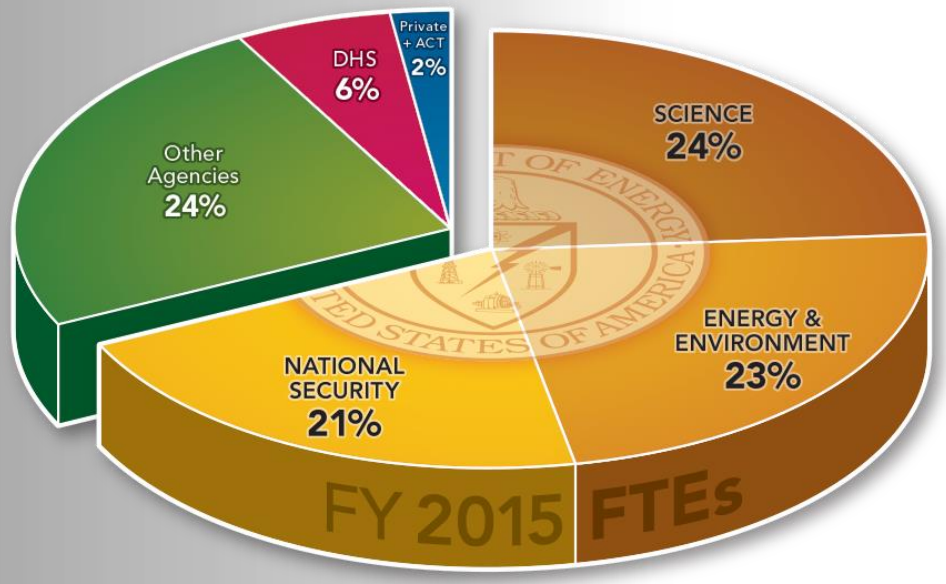
- High performance
- Ready-to-present HTML reports
- Scenario recorder and developer-friendly DSL



PNNL – FY2015 at a Glance

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

- $955 million in R&D expenditures
- 4,400 scientists, engineers and non-technical staff
- 78 U.S. & foreign patents granted
- 2 FLC Awards, 5 R&D 100
- 1,048 peer-reviewed publications



- Mission-driven collaborations with government, academia and industry
- DOE's top-performing lab for past seven years; a premier chemistry, environmental sciences and data analytics laboratory