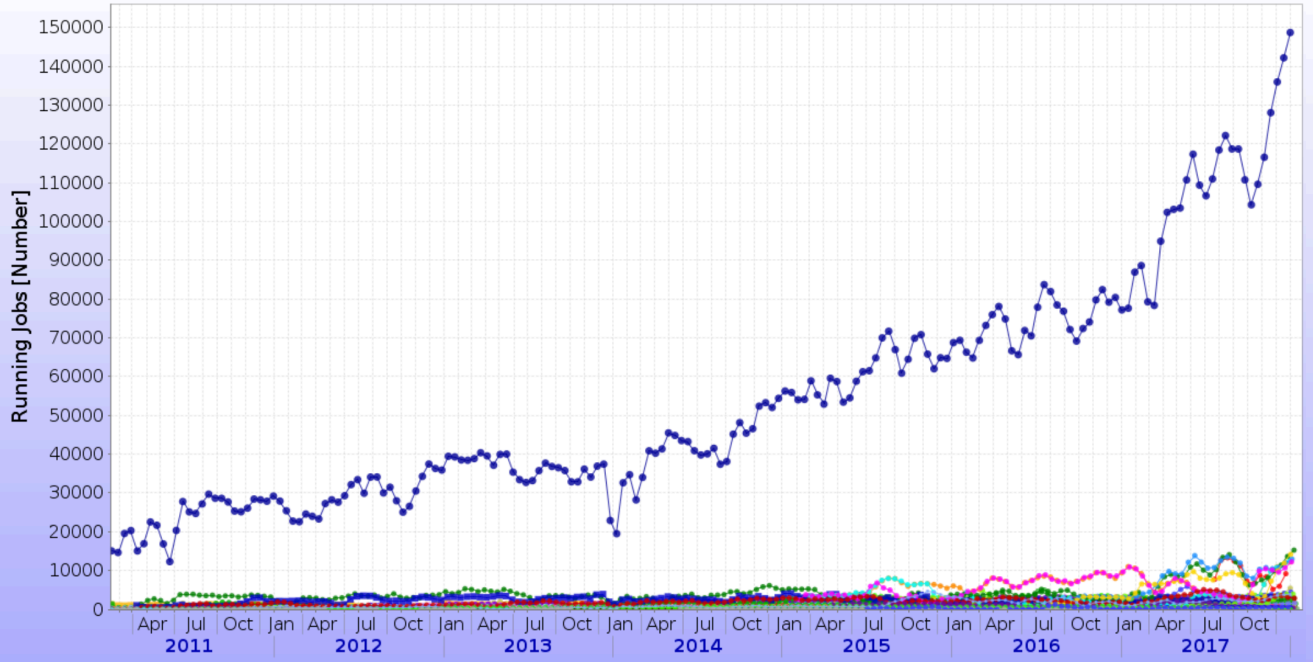
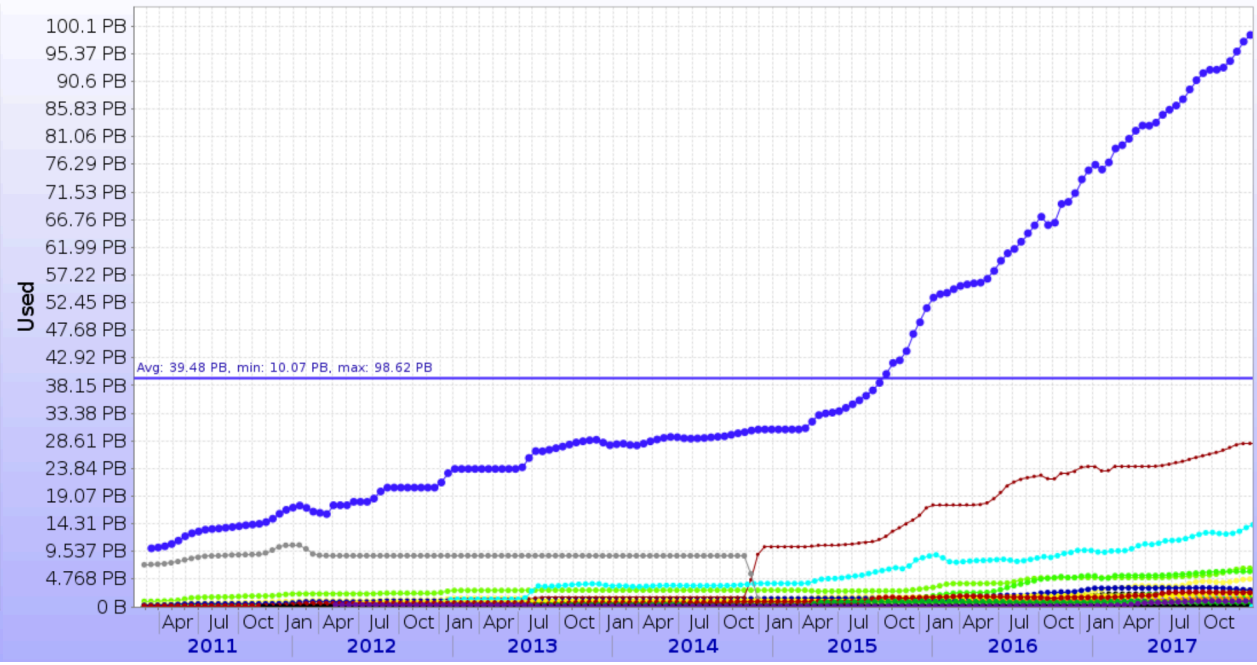
A Large Ion Collider Experiment

THE NEW ALICE HIGH-PERFORMANCE AND HIGH-SCALABILITY GRID FRAMEWORK

Miguel.Martinez.Pedreira@cern.ch | Track 3: Distributed Computing | 12/07/2018

COMPUTING CHALLENGE





- Factor 10 increase on CPU and data usage from 2011 to 2017

LOOKING TO THE FUTURE

Yearly maintained resources growth (expected to continue)

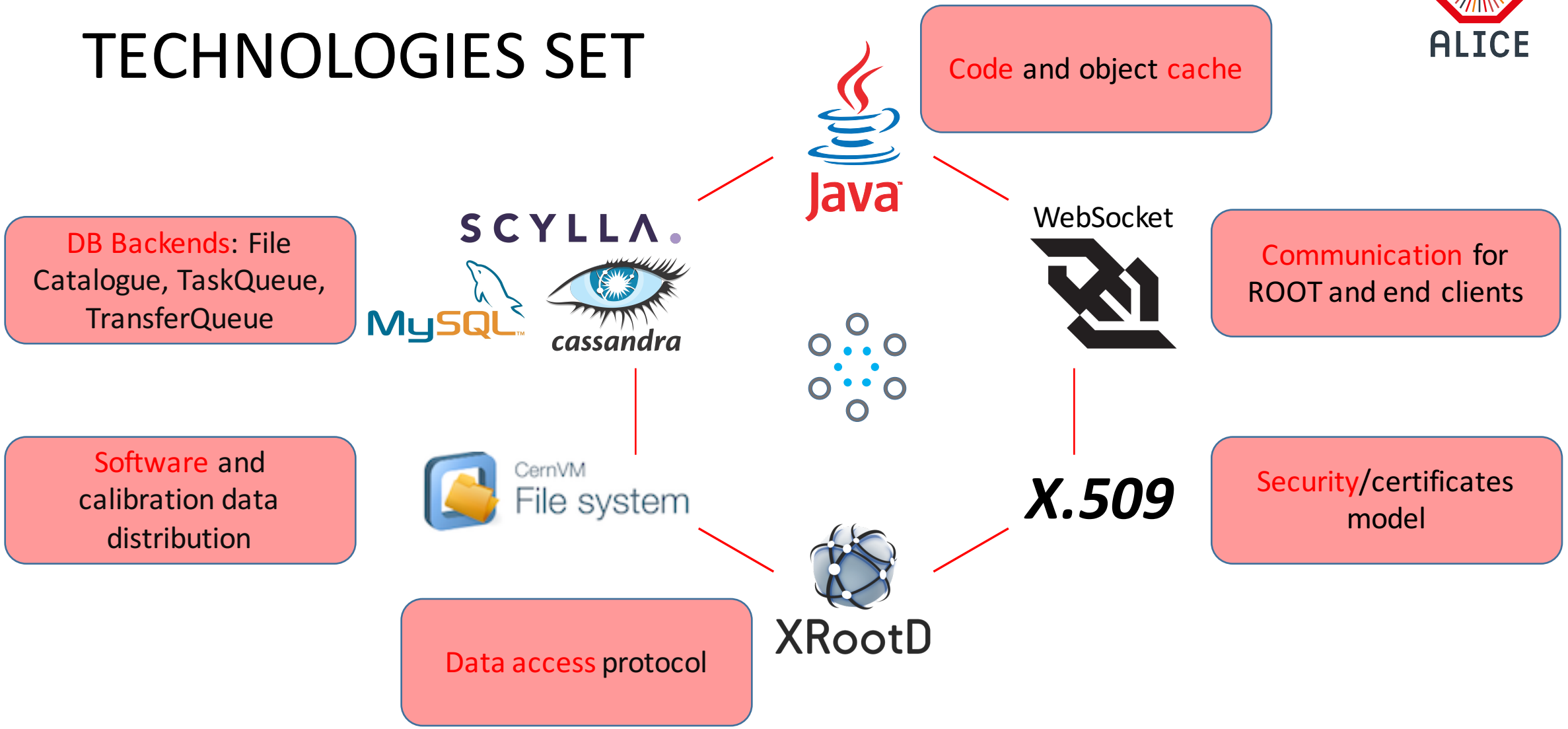
+

O^2 facility for synchronous and asynchronous data processing (60PB and 100K cores at the beginning of Run3)

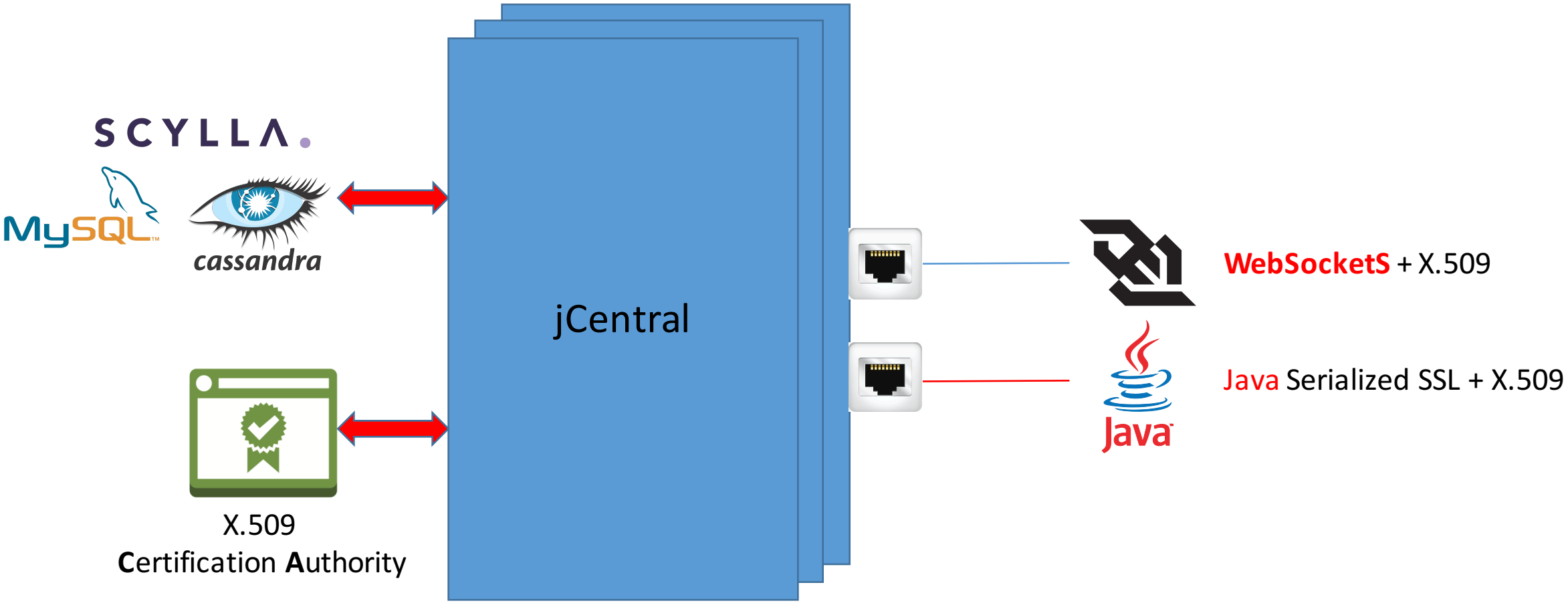
scalability of the software used in the past 10+ years is under question

decision to rewrite the entire ALICE high-level Grid services stack

TECHNOLOGIES SET



CENTRAL SERVICES



EASY TO DEPLOY AND SCALE

- Unique *jar* to deploy anywhere
- Each Central Service (*jCentral*) instance has the full functionality
- Hierarchical application configuration
  - Files, defaults, database
- Simplified dependencies
  - Java
  - Xrootd
  - Deployed on CVMFS
  - Previous framework: perl+packages, xrootd, openssl, httpd, c-perl bindings, swig, libxml2, zlib, ncurses, gsoap, classad, …!

AUTHENTICATION AND AUTHORIZATION

- Storage: keep the current model in ALICE (10+ years)
- Signed envelopes created by Central Services
- Each envelope allows for unique user-file-operation
- Central Services and Storages decoupled

- Client/server: new Token Certificates
- Full-fledged X.509 provided by the JAliEn CA and created by Central Services
- Fine-grained capabilities assigned to each token
  - Map the operations and file access allowed
  - E.g. Pilot Token can only do job matching

X.509

Following closely discussions and recommendations from the WLCG Authz WG

Full details of security model in V. Yurchenko's poster

PILOT IMPLEMENTATION

As discussed also in the WLCG Containers WG

Batch Queue -> startup script with embedded pilot token
C=ch/O=AliEn/CN=JobAgent

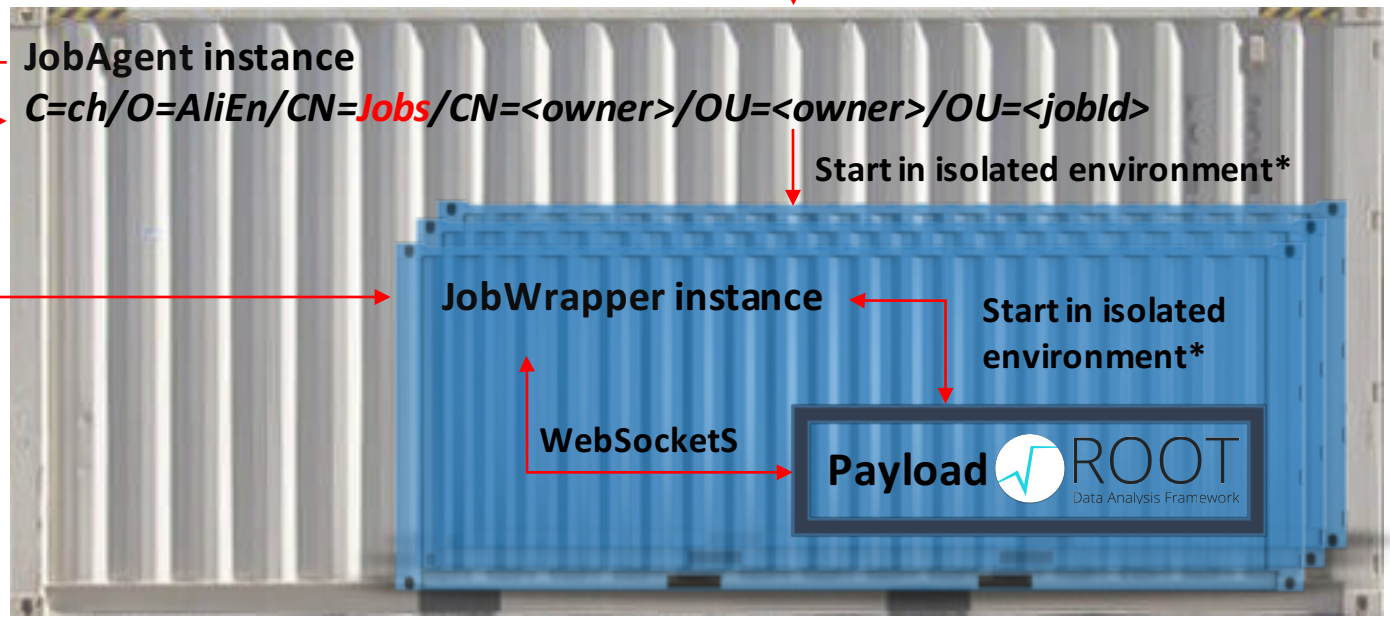
1-slot queue -> Start one pilot per slot
Full-node queue -> Start one pilot

Potentially containerized

getJob()

[jobId, token, JDL]

all job and monitoring calls

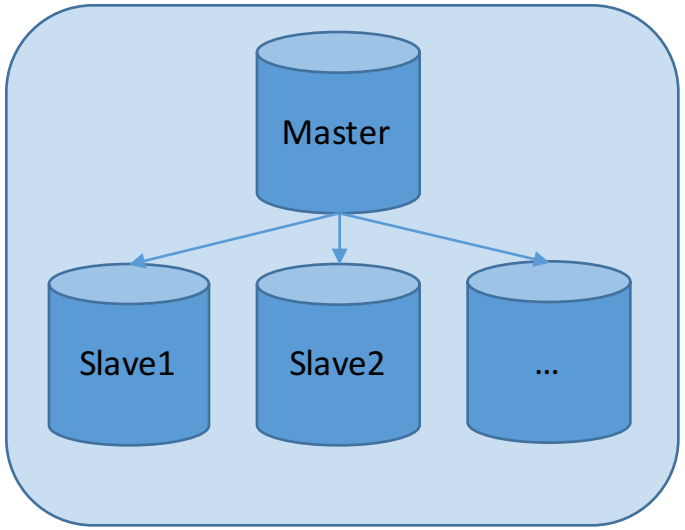


Details on container utilization in ALICE in M. Storetvedt's presentation

* Can be a simple wrapper script or container/singularity

SCALABLE AND RELIABLE BACKENDS

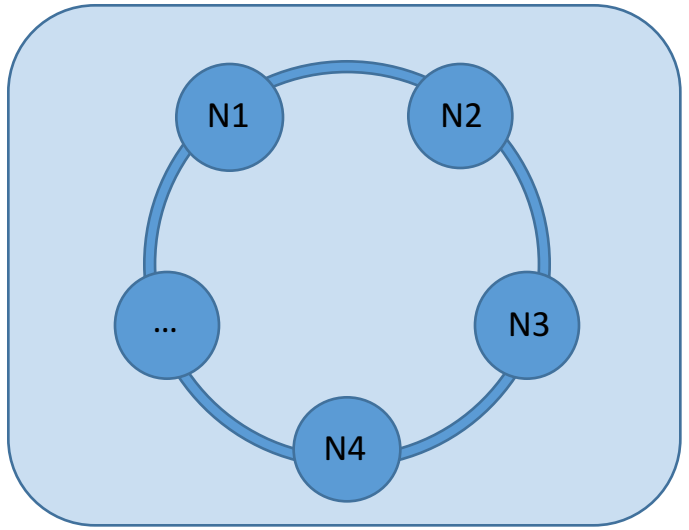
MySQL



- Manual *sharding*
  - Split file hierarchy into tables
- Single point of failure
- Rely on good hardware for performance
- Today:
  - 15B entries
  - O(10K) ops/s
  - 6TB on disk

- **Run3 challenge**
  - 50B entries
  - O(100K) ops/s

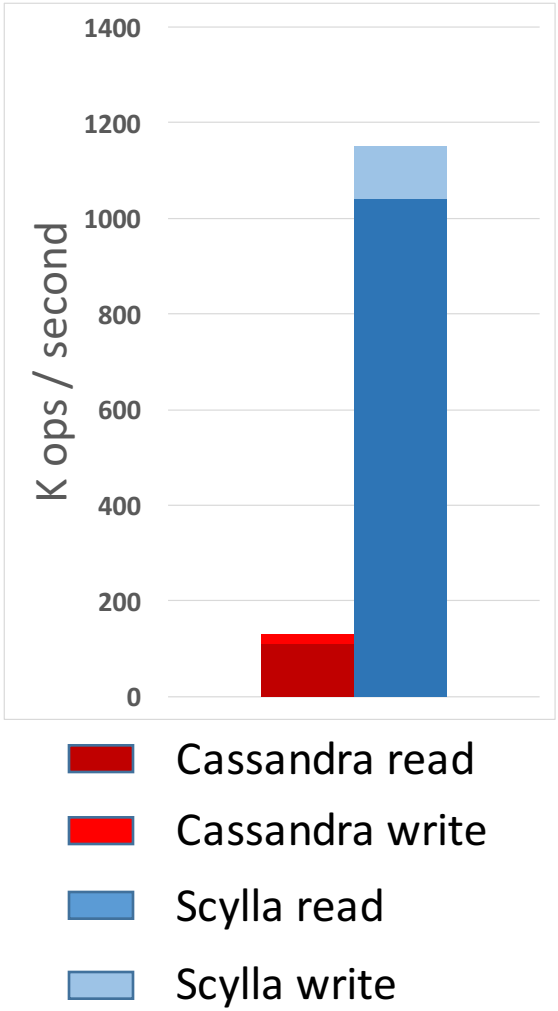
Cassandra/ScyllaDB



- Automatic *sharding*
- No single point of failure, HA
- Horizontal scaling, cheap hardware
- Consistency
- Paradigm change
  - SQL to noSQL

BENCHMARK RESULTS

Mixed (10 read : 1 write) Gauss (5B, 2.5B, 10M)



- Cassandra/ScyllaDB follow the same global architecture
- The internal implementation is very different

Cassandra	ScyllaDB
Java (JVM)	C++
Unaware of kernel/machine hardware	Kernel tuning, hardware probes
Java thread based as standard application, relies on kernel for most of resource management	Splits into 1 DB core per CPU core, splits RAM/DB cores, bypasses network from kernel (no syscalls), complex memory management
Several sync/lock points	Fully async (polling)

- Application and schema compatible with both backends

SUMMARY

- ALICE is looking forward to a major detector and software upgrade in Run3
- In addition to the standard 20-30% yearly growth, ALICE introduces the O^2 facility for synchronous and asynchronous data processing
- To cope with the increased capacity and complexity, we have decided to re-write the top level ALICE Grid services:
  - employing modern technologies
  - incorporating the best practices discussed in various WLCG WGs
- The development is well under way and will be ready in time for Run3
- For the interested: the JAliEn code repository and support list:
  - https://gitlab.cern.ch/jalien/jalien
  - jalien-support@cern.ch