



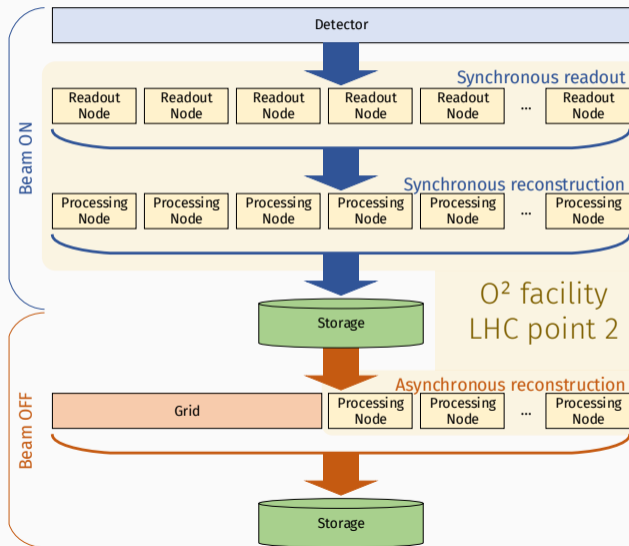
A research and
development effort
pending approval



Towards the ALICE Online-Offline (O²) Control System

Teo Mrnjavac
on behalf of the ALICE collaboration
July 9, 2018

The ALICE Online-Offline computing system



- Multiprocess data flow and **processing** framework
- 100,000s of processes, ~2000 machines
- **Synchronous and asynchronous** (grid-like) workflows

O² Control: target improvements

- Improved flexibility & latency:
 - **no workflow redeployment** when excluding/including a detector from data taking,
 - **recover** from process and server crashes,
 - **reconfigure** processes without restart,
 - **scale** EPNs during data taking (e.g. as luminosity decreases in a fill).
- Next gen web-based GUIs with SSO & **revamped design**.
- Take advantage of modern developments in computing.



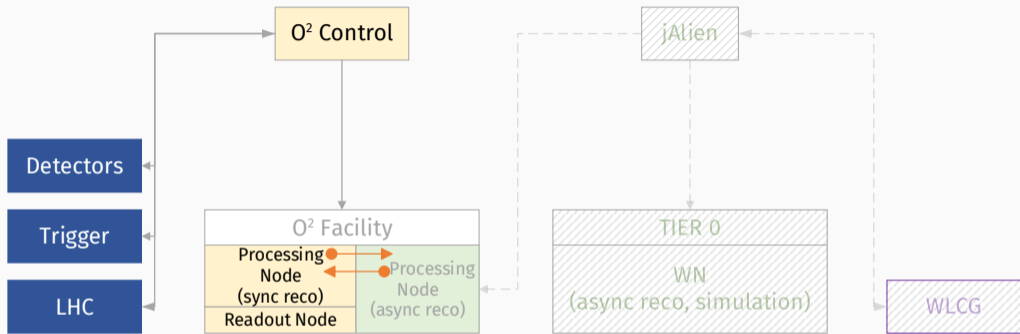
O² Control: requirements

“Just run some processes in a network...”

“Just run some processes in a network...”

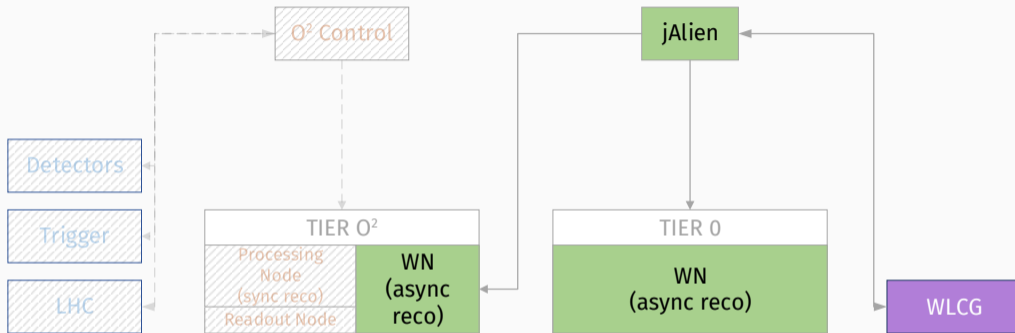
- **Manage the lifetime** of thousands of processes in the O² facility:
 - allocation of cluster resources,
 - deployment, configuration and teardown of multiple workflows,
 - high degree of autonomy.
- **Minimize waste of beam time** by reusing running processes and avoiding restarts.
- Interface with LHC, trigger, detectors and other systems.
- Ensure fair and efficient resource allocation between **synchronous and asynchronous** tasks.

O² Control: synchronous operation



O² Control can mark a node as synchronous or asynchronous.
If a node is used for **synchronous** processing, O² Control stays in charge.

O² Control: asynchronous operation



When O² Control assigns a node to **asynchronous** operation, it launches a pilot job to set up a Grid-like asynchronous execution environment. O² Control can reclaim these resources if necessary.

Managing a cluster with Apache Mesos

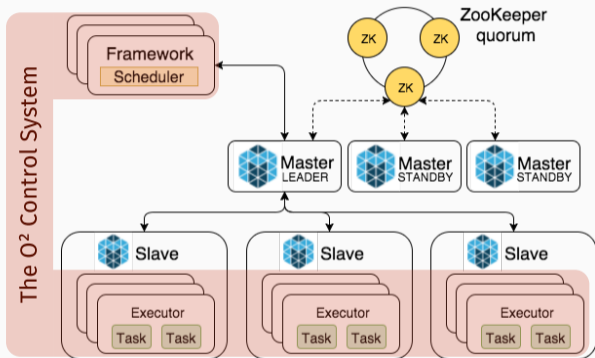
“Program against your datacenter like it’s a single pool of resources.”

Managing a cluster with Apache Mesos

“Program against your datacenter like it’s a single pool of resources.”

- We implement the **O² Control System** as a distributed application, using **Apache Mesos** as toolkit.
- Mesos acts as a unified **distributed execution environment** which streamlines how O² Control manages its components, resources and tasks inside the O² farm.

The Mesos architecture



- Mesos components on every host.
- Scales to **10,000s of nodes**.
- Open source, commercial support.
- Benefits for O² Control:
 - **knowledge** of what runs where,
 - **resource management** (ports, ...),
 - **transport** for control messages,
 - *bells and whistles...*
- Drawback:
 - one extra component in the stack.


A **framework**: a distributed application for Mesos, it has a **scheduler** and one or more **executors**.

The Mesos **master** sends **offers** to the scheduler. Mesos **slaves** then deploy executors to run **tasks**.

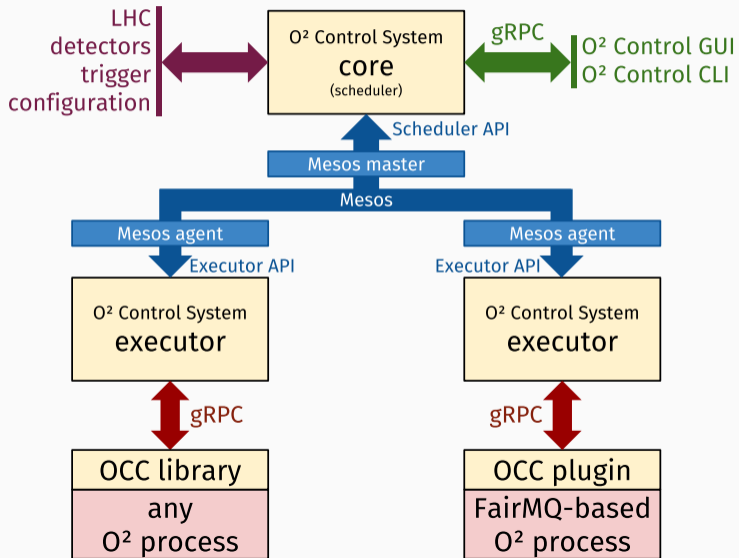
<https://github.com/AliceO2Group/Control>

- The O² Control System currently (09-07-2018) consists of:
 - the O² Control core (incl. Mesos scheduler)
 - the O² Control System executor
 - the O² Control and Configuration FairMQ plugin (FairMQPlugin_OCC)
 - the O² Control and Configuration CLI utility (coconut)
 - a deployment utility for O² development & testing (fpctl)
 - the web-based O² Control GUI

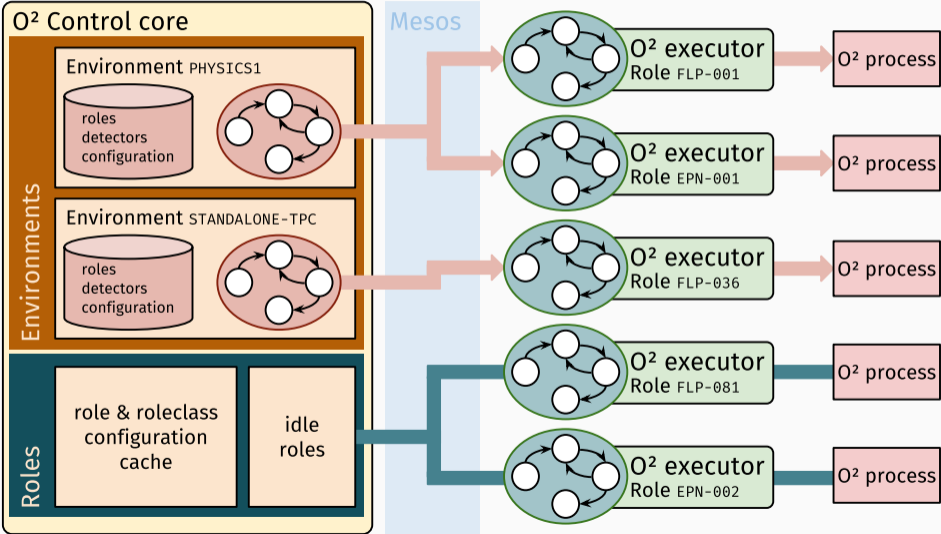


*More on Go in ALICE: *Exploring polyglot software frameworks in ALICE with FairMQ and fer* ,
Monday 14:30 track 5.

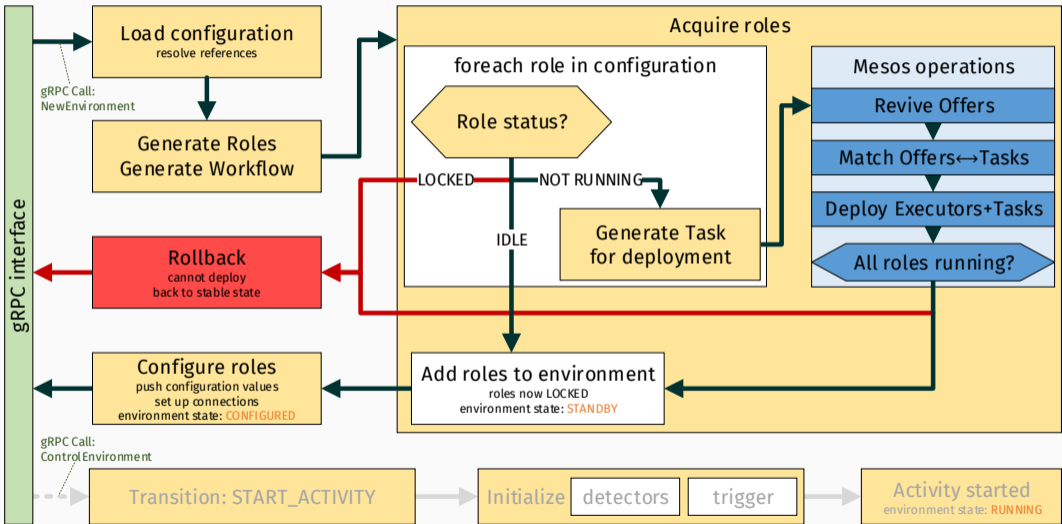
The O² Control System



O² Control role management



O² Control example: create new environment



- Next up:
 - experimenting with workflow configuration management and role operations,
 - evaluating functionality and performance of Mesos based solution,
 - evaluating DDS[†] as a complementary development addressing the deployment of ALICE workflows in unmanaged environments such as the Grid,
 - communication with trigger, detectors, LHC.
- Targets:
 - beginning of 2019** detector commissioning activities,
 - mid 2019** asynchronous environment allocation,
 - 2019+** automation, high availability, other fancy things.

[†]See *DDS - The Dynamic Deployment System* [↗](#), poster n. 407.

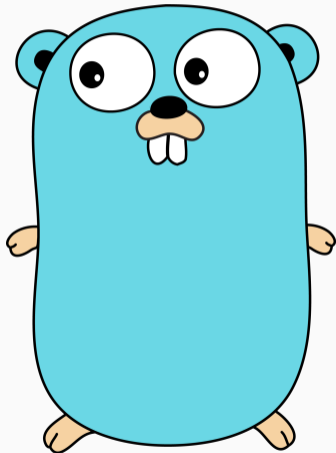
Conclusions

- The new ALICE O² computing system requires a **new Control system**.
- Opportunity to leverage technologies such as **Mesos** and Go for a high performance, low latency O² Control.
 - Mesos gives us resource management, transport and much more.
 - Improved operational flexibility.
 - Minimize waste of beam time.
 - Maximize utilization of O² facility for both sync and async workflows.

O² Control concepts

- The O² Control System interfaces with Mesos, which acts as its *cluster operating system*.
- The O² Control System also interfaces with Consul, a key-value store which acts as the system's **configuration repository**.
- The basic unit of O² Control scheduling is a *task*.
 - A task generally corresponds to a process, which implements an *O² role*.
- A collection of O² roles (along with their configuration) is an *environment*.
- An environment represents the collective state of its constituent O² roles, its associated detector components and other runtime workflow resources.
 - If an environment is in a running state (with a run number), it represents an *activity*.

Why Go?



- **Go** is a statically typed general-purpose programming language in the tradition of C
 - 100% Free and open source,
 - simple syntax and excellent readability,
 - garbage collection,
 - OOP with interface system and composition but no inheritance,
 - concurrency with lightweight processes (goroutines) and channels,
 - fast compiler with build system and remote package management included,
 - statically linked native binaries,
 - **excellent for building distributed systems.**
- Already used in some components of the O² stack, including Consul, Docker and InfluxDB.

Why Mesos?

- **Resource management:** CPU cores, memory, port allocation, ...
- Reservations and attributes: we're sure our tasks end up in the right place.
- Tracking of **knowledge** on the cluster: we know what runs where, and we're notified when it stops running for any reason.
- **Transport:** we can use Mesos to send control messages to any task.
- Seamless integration: Marathon, Chronos, Aurora, ...
- Bells & whistles:
 - high availability facilities,
 - native APIs for multiple languages,
 - overprovisioning,
 - cross-farm, ...