AliECS

A New Experiment Control System for the ALICE Experiment

Teo Mrnjavac
CERN EP-AID-DA
on behalf of the ALICE O²/FLP project
4 November 2019
The ALICE Online-Offline computing system

- Multiprocess data flow and processing framework
- 100,000s of processes, ~1000 machines
- Synchronous and asynchronous (grid-like) workflows
- One computing system, 2 types of node arranged in 2 clusters:
  FLP - First Level Processors
  EPN - Event Processing Nodes
- Operations will start in 2021
ECS and $O^2$ cluster control

- Manage the lifetime of thousands of stateful processes in the $O^2$/FLP cluster (control of $O^2$/EPN delegated to a specialized $O^2$/EPN cluster control)
- Minimize the waste of beam time by reusing processes and avoiding time-consuming process restart operations
- Interface with the LHC, the trigger system, the Detector Control System and other systems through common APIs

**ALICE Run Control Centre Shifter**

**Experiment Control System**

**$O^2$/FLP cluster control**

**$O^2$/EPN cluster control**

**LHC detectors (DCS) trigger system**
"Program against your datacenter like it’s a single pool of resources."
“Program against your datacenter like it’s a single pool of resources.”

- Mesos acts as a **distributed execution environment** which streamlines how AliECS manages its components, resources and tasks inside the O²/FLP farm.

- Benefits:
  - **knowledge** of what runs where,
  - **resource management** (ports, CPU, RAM, ...),
  - **transport** for control messages,
  - task event **notification** (dead, failed to launch, ...),
  - node attributes, high availability, checkpointing, ...
AliECS in a nutshell

configuration

LHC detectors trigger

gRPC
DIM DIP

AliECS core
(scheduler)

gRPC

AliECS GUI
AliECS CLI

O²/EPN
cluster control
AliECS in a nutshell

![Diagram of AliECS architecture](image-url)
AliECS in a nutshell

configuration

LHC
detectors
trigger

gRPC
DIM DIP

AliECS
core
(scheduler)

Scheduler API

Mesos master

Mesos

AliECS
gRPC
GUI
CLI

O²/EPN
cluster control

AliECS
gRPC
executor

Executor API

Mesos agent

Executor API

Mesos agent

OCC library
any
O² process

OCC plugin
FairMQ-based
O² process
AliECS in a nutshell

• Components:
  • AliECS core (incl. Apache Mesos scheduler)
  • AliECS executor
  • AliECS control and configuration utility (coconut)
  • Single process state machine debug utility (peanut)
  • O² control and configuration FairMQ plugin (FairMQPlugin_OCC)
  • O² control and configuration library (libOCC)

• Also available:
  • The web-based AliECS GUI
  • AliECS deployment mechanism
AliECS in a nutshell

- Components:
  - AliECS core (incl. Apache Mesos scheduler)
  - AliECS executor
  - AliECS control and configuration utility (coconut)
  - Single process state machine debug utility (peanut)
  - $O^2$ control and configuration FairMQ plugin (FairMQPlugin_OCC)
  - $O^2$ control and configuration library (libOCC)

- Also available:
  - The web-based AliECS GUI
  - AliECS deployment mechanism
AliECS in a nutshell

• Components:
  • AliECS core (incl. Apache Mesos scheduler)
  • AliECS executor
  • AliECS control and configuration utility (coconut)
  • Single process state machine debug utility (peanut)
  • O² control and configuration FairMQ plugin (FairMQPlugin_OCC)
  • O² control and configuration library (libOCC)

• Also available:
  • The web-based AliECS GUI
  • AliECS deployment mechanism
AliECS concepts

- AliECS schedules, configures and controls tasks (stateful processes)
- Each role represents either a task, or its own child roles
- A tree of roles is a workflow
- Tasks, roles and environments have their own state machines
- An environment in **RUNNING** state is granted a unique run number which remains valid until the **RUNNING** state exits
AliECS workflow and task configuration

- Based on **Git**, multiple repositories per AliECS instance
- Task descriptors and workflow templates are **YAML** (plus template system)
- Once loaded, every task type and workflow is uniquely identified by git repository + task/workflow file name + git revision

Documentation: https://github.com/AliceO2Group/Control/blob/master/coconut/doc/coconut_repository.md
AliECS workflow and task configuration

1. Workflow template
2. Task references
3. Configuration references
4. Consul key value tree

YAML
Workflow template
Task descriptors

AliECS core
Environment
role
role
role
role
role
role

①: create environment
STANDBY state
②: control environment
CONFIGURE transition
③: launch tasks (Mesos)

Tasks

task configuration
port assignments
task-specific key-values

④: task configuration push

task-specific
cfg. query
## AliECS GUI

### Number of Tasks
7

### ID
6beaa3a0-cdb5-11ea-b369-fa163eb2197f

### Created
9/2/2019, 9:11:09 PM

### State
RUNNING

### Root Role
readout-ep-1

#### Tasks

<table>
<thead>
<tr>
<th>name</th>
<th>locked</th>
<th>taskid</th>
<th>status</th>
<th>state</th>
<th>className</th>
<th>deploymentInfo</th>
<th>actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/readout@master#606f10a3-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6df110a3-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/readout@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/internal-dpil-clock@master#6b0b16a3-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6fca37-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/internal-dpil-clock@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/readout-proxy@master#6f1a5a03-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6f6ca37-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/readout-proxy@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/Dispatcher@master#6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6f71a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/Dispatcher@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/QC-TASK-RUNNER-daqTask@master#6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/QC-TASK-RUNNER-daqTask@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/daqTask-checker@master#6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/daqTask-checker@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
<tr>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/internal-dpil-global-binary-fiesink@master#6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>true</td>
<td>6f671a37-cdb5-11ea-b369-fa163eb2197f</td>
<td>ACTIVE</td>
<td>RUNNING</td>
<td>github.com/AliceO2Group/ControlWorkflows/tasks/internal-dpil-global-binary-fiesink@master</td>
<td>teo-test19</td>
<td>More</td>
</tr>
</tbody>
</table>
• As few dependencies as possible to facilitate maintenance
• **Node.js** with Express.js as server framework
• **grpc/proto-loader** and **grpc** for communication with AliECS core
• UI and other components built from scratch and exported as **npm** module for look&feel consistency across O²/FLP interfaces
• Puppeteer for integration tests
• Kafka-node for displaying browser notifications via common Notification Service
Conclusions

- The new ALICE O² computing system requires a new control system.
- AliECS carries both ECS and O²/FLP cluster control duties.
- Opportunity to leverage technologies such as Mesos and Go for a high performance, low latency ECS.
  - Mesos gives us resource management, transport and much more.
  - Minimize waste of beam time.
  - Improved operational flexibility.

AliECS on GitHub: github.com/AliceO2Group/Control
Configuration examples: github.com/AliceO2Group/ControlWorkflows
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** workflows based on immediate needs.

- Next gen web-based GUIs with SSO & **revamped design**.

- Take advantage of modern developments in computing.
Why Go?

- **Go** is a statically typed general-purpose programming language in the tradition of C.
- 100% Free and open source.
- Prominent features include:
  - simple syntax and excellent readability,
  - garbage collection,
  - interface system and composition, but no inheritance,
  - lightweight processes (goroutines) and channels,
  - build system and remote package management included in compiler,
  - fast compilation,
  - statically linked native binaries.
- Go is already used in some components of the O² stack, including Consul, Docker and InfluxDB.
gRPC in AliECS

- **gRPC**, an RPC system based on Protobuf was chosen as the lingua franca of AliECS IPC:
  - backed by Google,
  - multi-language support,
  - already packaged for O²,
  - widely used in the microservices community.
- In AliECS, gRPC is used for
  - communication between the AliECS core and the GUI,
  - communication between the executor and the OCC plugin.
- Higher performance and better multi-language integration compared to REST (Swagger, etc.)
- Better interoperability and/or support/documentation compared to other RPC methods (JSON-RPC, MessagePack-RPC, net/rpc, Cap’n’Proto, etc.)
Why Apache Mesos? / Why not Kubernetes?

- Apache Mesos vs. Kubernetes is a false equivalence:
  - Apache Mesos is primarily a **cluster resource management system**
    - See also Marathon, Aurora, DC/OS...
  - Kubernetes is a **container orchestration platform**
    - “Opinionated software”: it enforces its own structure of Pods and Containers
- The benefits of Kubernetes + containerization are dubious at best in a heterogeneous environment such as the O²/FLP farm, which includes:
  - different configurations of FLP machines
  - custom PCIe hardware
  - physical point-to-point fiber links to detector front-end electronics.
- A **resource management system** with deployment functionality at the single process level such as **Apache Mesos** fits well with O²/FLP requirements.

A container orchestration platform wasn’t needed and still isn’t.
Workflows, roles and tasks

• Concepts:
  • **task** - the basic unit of control, generally 1 process
  • **role** - a node in the control tree, aggregates child roles and ultimately tasks
  • **workflow** - the in-memory control tree of an environment, made of roles which drive tasks

• Workflow templates generate workflows of tasks
  • Generated from DPL specs
  • Stored in $O^2$ configuration (YAML + Git)
  • Variables, iterators, internal references

```yaml
fairmq-ex-copypush:
  name: "copypush"
  vars: {}
  roles:
    - name: "sink{{ .it }}"
      for:
        begin: 0
        end: 3
        var: it
      connect:
        - name: "data"
          target: "{{ parent }}.sampler: data"
          type: "pull"
          sndBufSize: 1000
          rcvBufSize: 1000
          rateLogging: 0
      task:
        load: fairmq-ex-copypush-sink
    - name: "sampler"
      task:
        load: fairmq-ex-copypush-sampler
```
Control of FairMQ devices

- AliECS core (scheduler)
- gRPC
- AliECS GUI
- AliECS CLI

- AliECS Environment state machine
- AliECS Task state machine

- O² Control and Configuration FairMQ plugin
  - Transition(srcState, event)
  - Check conditions for transition
  - Request transition (if event is CONFIGURE, also push properties)
  - Wait for and process device state changes
  - Build event response

- FairMQ-based O² process

- AliECS ↔ FairMQ states/transitions translation and wrapping
- FairMQ Device state machine
Control of non-FairMQ O² processes (e.g. Readout)

- AliECS core (scheduler)
- gRPC
- AliECS GUI
- AliECS CLI
- library provides state machine
- AliECS Environment state machine
- AliECS Task state machine
- O² Control and Configuration library
- Transition(srcState, event)
  - Check conditions for transition
  - Request transition (if event is CONFIGURE, also push properties)
  - Wait for and process transition outcome
  - Build event response

- any O² process
- OCC state machine interface
- executor
- FairMQ transitioner
- direct transitioner
- no states/translations translation or wrapping
- user code implements state machine transitions
- Mesos transport
Debug mode for non-FairMQ O² processes

Single process debug mode:
- no AliECS core/executor
- no Mesos

**peanut**
(process execution and control utility)

- Cfg.yaml
  - task config
  - app config

- gRPC
  - AliECS transition event
  - Transition response

Interactive user input

User code implements state machine transitions

**O² Control and Configuration library**

- Transition(srcState, event)
  - Check conditions for transition
  - Request transition (if event is CONFIGURE, also push properties)
  - Wait for and process transition outcome
  - Build event response

Library provides state machine

Any O² process

OCC state machine interface