EPJ Web of Conferences **295**, 02027 (2024) https://doi.org/10.1051/epjconf/202429502027
CHEP 2023

AliECS: a New Experiment Control System for the ALICE Experiment

Teo Mrnjavac[1,*], *Konstantinos* Alexopoulos[1], *Vasco* Chibante Barroso[1,**], *Claire* Guyot[1], *Piotr* Konopka[1,***], and *George* Raduta[1,****]

[1]CERN, Geneva, Switzerland

Abstract. The ALICE Experiment at CERN's Large Hadron Collider (LHC) has undergone a major upgrade during LHC Long Shutdown 2 in 2019-2021, which includes a new computing system called O² (Online-Offline). To ensure the efficient operation of the upgraded experiment and of its newly designed computing system, a reliable, high performance, full-featured experiment control system has also been developed and deployed at LHC Point 2. The ALICE Experiment Control System (AliECS) is a microservices-oriented system based on state-of-the-art cluster management technologies that emerged recently in the distributed and high-performance computing ecosystem. It is designed, developed and maintained as a comprehensive solution and single entry point for control of experiment data acquisition (up to 3.5 TB/s) and processing. This communication describes the AliECS architecture by providing an in-depth overview of the system's components, interfaces, features, and design elements, as well as its performance. It also reports on the experience with AliECS during the first year of ALICE Run 3 data taking with LHC beam, including integration and operational challenges, and lessons learned from real-world use.

1 Introduction

1.1 Overview of the O^2 Computing System

The ALICE experiment [1] has undergone a major upgrade [2] deployed during the LHC's Long Shutdown 2 (2019-2021) in preparation for the LHC Run 3. The new and upgraded detectors are operating at a significantly increased data rate, and in order for the data processing to keep up, a new computing system called O^2 [3] has been designed, developed and deployed.

In its production instance, the O^2 computing system consists of 100,000s processes, deployed over roughly 570 heterogeneous nodes in two clusters, fulfilling roles including data readout, processing, storage and auxiliary services. The system can read out 3.5 TB/s (28 Tb/s) of raw data and record 130 GB/s (1 Tb/s) of reconstructed data.

The O^2 computing system is capable of two operation modes, depending on the data flow structure: synchronous operation, intended to be *synchronous* with the detector readout, and asynchronous operation, which can take place at any time regardless of detector or

*e-mail: teo.m@cern.ch
**e-mail: vmcb@cern.ch
***e-mail: piotr.jan.konopka@cern.ch
****e-mail: george.raduta@cern.ch

beam conditions. Most nodes run hundreds of processes of different kinds, including long-running services, asynchronous processing tasks, and data-driven process workflows. Since synchronous workflows operate on data coming from detector data links, they must run in the O^2 facility at the LHC Point 2. Asynchronous workflows do not have this constraint, so they can run at any time on WLCG (Worldwide LHC Computing Grid) nodes or on O^2 facility resources when they are not needed for synchronous operation.

1.2 The O^2/FLP Computing Cluster

The O^2 data processing workflows run on two typologies of computing nodes: FLPs (First Level Processors) and EPNs (Event Processing Nodes). Each FLP is fitted with CRU (Common Readout Unit) [4][5] or C-RORC (Common Readout Receiver Card) [6] hardware, depending on the detector. These PCI-Express cards are capable of two-way communication with detector front-end electronics. Unlike FLPs, which host the first portion of the data flow, EPNs do not have physical links to detector hardware, instead they are configured as homogeneous computing nodes, operating as a second level of data processing after FLPs.

The O^2 computing system is split into two separate computing clusters due to significant differences in requirements between FLPs and EPNs. Both clusters are deployed at the LHC Point 2.

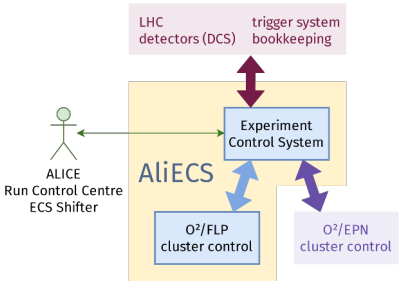

Figure 1. O^2/FLP and O^2/EPN cluster control with respect to the ALICE Run Control Centre.

The existence of direct fibre links between FLPs and detector electronics makes FLPs physically bound to a specific detector or detector component. Besides the variable number of CRU or C-RORC cards (up to 3 per FLP), FLPs may have different system specifications. Since FLPs are not interchangeable, the O^2/FLP cluster is inevitably a heterogeneous environment. EPNs, on the other hand, do not have direct links to detector front-end electronics, and are largely interchangeable.

While the two computing clusters have their own specialised control mechanisms, for synchronous operation the O^2 system as a whole is controlled via a single user interface, the AliECS GUI, operated by the ECS (Experiment Control System) shifter in the ALICE Run Control Centre (see Fig. 1).

In addition to cluster control, the O^2 project also includes a redesign of user interfaces in favor of next-generation web-based GUIs with SSO (single sign-on) and a revamped look and feel [7]. AliECS has command line and graphical user interfaces, including shifter-oriented GUIs that supersede those of the previous generation ECS.

The O^2 project is an opportunity to take advantage of modern developments in computing; AliECS is built with the best practices of a microservices distributed application paradigm, and harnessing the features of modern cluster resource management solutions.

2 Requirements of an ECS solution for ALICE Run 3

AliECS was designed and developed to improve operational flexibility and reliability compared to the system used in LHC Run 2, as well as to take advantage of new technologies and open source software that became available since the first design 15 years prior. The duties of AliECS, in its role as the control mechanism for the ALICE O^2 system, include

1. Managing the lifetime and configuration of thousands of data flow processes in the O^2/FLP cluster (while delegating control of O^2/EPN processes to a specialized control mechanism for the O^2/EPN cluster),
2. While minimizing the waste of beam time by avoiding time-consuming process restart operations whenever possible, and
3. Interfacing with the LHC, the trigger system, the DCS (Detector Control System) [8], and other systems through common APIs.

3 AliECS design overview

Due to the tight coupling required between high-level experiment control and O^2/FLP cluster control, AliECS integrates both experiment control and O^2/FLP cluster control functionality into a single system. Thus, AliECS provides in-depth control of every data-driven process running in the O^2/FLP cluster. An interface between the AliECS core and the O^2/EPN control mechanism also implements coarse-grained, high-level control of the O^2/EPN cluster.

AliECS is a distributed system in charge of the O^2 facility with full knowledge and control over the resources of the O^2/FLP cluster. It implements a distributed state machine to represent the aggregated state of the constituent O^2 processes of a data-driven workflow. Furthermore, it allows the reconfiguration of running O^2 data taking activities (called *environments*) and their processes as often as possible to avoid process restarts, as well as the simultaneous operation of multiple workflows. Finally, it reacts promptly to inputs, handling events from the user, the LHC, the trigger system, the DCS, and the cluster itself with a high degree of autonomy.

The O^2 project has chosen FairMQ [9] as the common message passing and data transport framework for its data-driven processes. It has been developed in the context of FairRoot [10, 11], a simulation, reconstruction and analysis framework for particle physics experiments. FairMQ provides the basic building blocks to implement complex data processing workflows, including a message queue, a configuration mechanism, a state machine, and a plugin system.

3.1 Resource Management in the O^2/FLP Facility

AliECS is implemented as a distributed application, using Apache Mesos [12, 13] as a toolkit. This custom solution integrates a task scheduler component, a purpose-built distributed state machine system, a multi-source stateful process configuration mechanism, and a control plugin and library compatible with any data-driven O^2 process.

Apache Mesos is a cluster resource management system, which facilitates the management of O^2/FLP components, resources and tasks inside the O^2/FLP facility, effectively enabling the developer to program against the datacenter (i.e., the O^2/FLP facility at LHC Point 2) as if it was a single pool of resources. Mesos is an open source project hosted by the Apache Software Foundation, and used in deployments of 10,000s nodes.

For AliECS, benefits of using Mesos include knowledge of what runs where, resource management (including port assignment), transport facilities for O^2-specific control messages, task status tracking (e.g. an event is raised if a task dies unexpectedly), as well as advanced features such as node attributes, cgroups-based resource isolation, containerization, resource overprovisioning, checkpointing, and others.

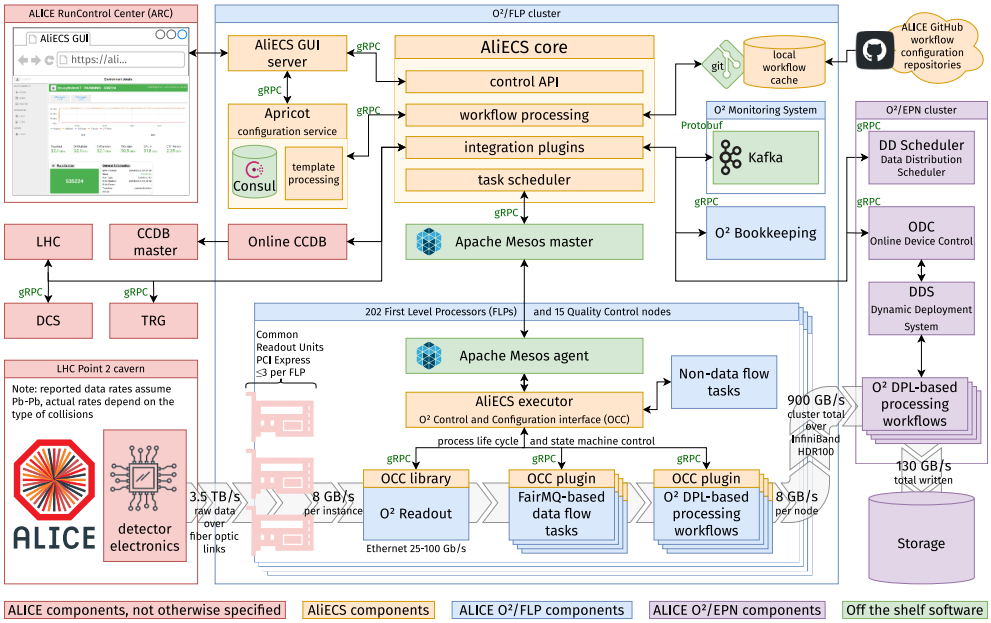

Figure 2. The role of AliECS in the ALICE O^2 system.

3.2 AliECS Components

AliECS has been used at LHC Point 2 in production since 2022, and is under ongoing maintenance and development as our solution for the problem of O^2/FLP synchronous control and ECS. The current implementation of AliECS can be found on GitHub [14]. AliECS consists of the

1. AliECS core (which includes workflow processing, task deployment and control via the Apache Mesos-facing scheduler component, and integration with other systems),
2. Apricot service (the single point of truth for ALICE O^2/FLP configuration, including AliECS configuration, hardware inventory, and task configuration templates and payloads),
3. AliECS executor (one per cluster node, manages the life cycle and controls the state machine of data flow tasks),
4. AliECS GUI (a web-based graphical user interface[7], used by the ALICE RunControl Center shift crew and experts to interact with AliECS),
5. O^2 Control and Configuration (OCC) library and plugin (loadable components that allow AliECS to interface with controllable tasks and drive their state machine),
6. AliECS control and configuration command line utility (`coconut`),
7. AliECS process execution and control utility for OCC library based O^2 processes (`peanut`).

Most components of AliECS are written in Go [15], a statically typed general-purpose programming language in the tradition of C, which is particularly suitable for distributed system development because of its advanced synchronization and threading facilities. In order to seamlessly interface with O² data flow tasks, AliECS includes a C++ library as well as a FairMQ plugin for task configuration and state machine control. The common idiom of inter-process communication in AliECS is gRPC [16], an open source, cross-language RPC (Remote Procedure Call) system widely used in the microservices community.

AliECS interfaces with Consul [17], a key-value store that acts as the system's configuration repository. The design also includes interfacing with information sources from the LHC,

the trigger system and the DCS. Once acquired by the AliECS core, configuration information is processed into an in-memory hierarchical key-value store, and fed into a template system in order to generate task deployment and configuration structures.

AliECS also interfaces with the O^2 Monitoring system, as well as with the O^2 Bookkeeping system, in order to persist data-taking activity information.

As shown in Fig. 2, AliECS oversees the full data-taking pipeline, starting from ALICE detector electronics, passing through Common Readout Units (CRUs) on FLP machines and subsequent data flow tasks, the EPN cluster and its additional processing workflows, until the processed and compressed data is finally written to storage.

3.3 AliECS Concepts

The basic unit of scheduling in AliECS is a *task*. A task generally corresponds to a process, more specifically, a process that can receive and respond to OCC-compatible control messages. All AliECS workflows are collections of tasks, which together form a coherent data processing chain.

Tasks are the leaves in a tree of *roles*. A role is a runtime subdivision of the complete system and represents a kind of operation along with its resources. Roles allow binding tasks or groups of tasks to specific host attributes, detectors and configuration values. In memory, a tree of O^2 roles, with tasks and their configuration is a *workflow*. A workflow aggregates the collective state of its constituent O^2 roles. A running workflow, along with associated detectors and other hardware and software resources required for experiment operation constitutes an *environment*, which is the highest level of state machine control (see Fig. 3).

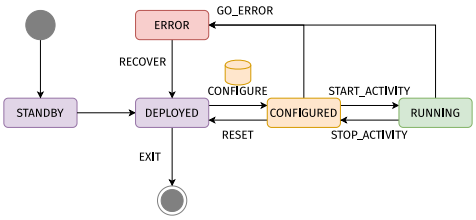

Figure 3. The state machine of an AliECS environment. The same state machine is implemented by each task. For FairMQ-based tasks the OCC plugin acts as a translation layer between the AliECS task state machine and the underlying FairMQ state machine.

3.4 Configuration Management

AliECS is both a producer and consumer of configuration data in the O^2/FLP cluster. There are three kinds of configuration information that AliECS deals with:

1. the AliECS core configuration,
2. the AliECS workflow configuration,
3. and the O^2 tasks configuration.

The AliECS core configuration is a flat list of read-only values that the AliECS core acquires on startup. Typical values that come from this configuration mechanism are the control port to use for incoming AliECS GUI or `coconut` connections and the URI of the Mesos master API.

The AliECS workflow configuration is acquired by a configuration manager component that uses Git repositories as a backend for file storage and versioning. The AliECS workflow configuration data consist of task descriptor files and workflow template files sourced from

Git repositories. A task descriptor file is a YAML document that describes how to launch and control a single task, such as an O^2 data-driven process. A workflow template file is a YAML document that describes the structure of a workflow of roles and (ultimately) tasks. This structure directly expresses the control tree, which defines the layout of the distributed state machine.

Workflow configuration is further complemented by AliECS runtime variables, sourced from persistent storage via Apricot, or from user input via the AliECS GUI, which can affect the loaded workflow and single tasks (Fig. 4).

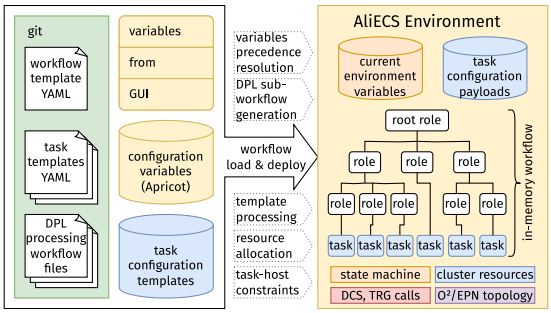

Figure 4. The AliECS core acquires workflow and task configuration information from multiple sources, and processes this information into a deployed AliECS environment.

AliECS implements task configuration as a push operation associated with the `CONFIGURE` transition. The payload of this operation is task-specific and includes communication channel configuration (i.e. hosts and ports to connect or bind) as well as an optional key-value map of application-specific configuration data. Channel configuration is generated on the fly by the AliECS core, whereas application-specific configuration data is stored, composed and processed by Apricot.

All these different sources of configuration provide inputs for the end goal of a working in-memory representation of a data-driven process workflow, and ultimately result in the same workflow deployed throughout the O^2/FLP and O^2/EPN clusters.

4 AliECS in Run 3 production

The O²/FLP and O²/EPN clusters at LHC Point 2 host two instances of the O² data-taking system: the production instance and the staging instance. The latter is a scaled-down vertical slice of the full system, for testing and release validation purposes. Consequently, the O²/FLP cluster hosts two instances of AliECS. Additional smaller-scale instances have been provisioned as needed for other objectives, such as detector commissioning.

During the first year of operation, as well as during research and development before that, we encountered the following major challenges:

1. Operating the O^2 system, including AliECS for the ECS shifter in the ALICE Run Control Centre, requires all-new training compared to Run 2. This was especially true during an initial period when operational experience with the new system at production scale was scarce. We have addressed this challenge with refreshed training materials and procedures and incremental improvements in the AliECS GUI in order to hide some complexity and streamline the shifter's choices.
2. The central position of AliECS with respect to other O^2 components results in a frequent need to change both the behavior of the system and its presentation (i.e. the

AliECS core and the AliECS GUI). AliECS is highly flexible, and its workflow template system ensures that the data flow can be modified without code changes in AliECS components, but the inputs to workflow template processing still need to come from the AliECS GUI. In order to provide the necessary flexibility and consistency, the GUI implements a dynamic input widget system, in which the AliECS core requests from the GUI a widget layout specific to the workflow template in use. This way, the environment creation page in the GUI is described entirely in the YAML-based workflow template file in a declarative way.

3. The IPC interfaces between AliECS and other ALICE systems such as the DCS, the trigger systems, and the O^2/EPN subcontrol are still evolving as the relevant teams improve and complete core functionality. AliECS has had to interact with these systems, called *integrated services* on the AliECS side, since day one of production. We have addressed this challenge by reacting promptly to integration needs, as well as by structuring this pattern of IPC into a plugin system that allows us to quickly and easily enable, disable, and change the timing of integrated service API calls on the fly within the YAML-based workflow template files.
4. The development and support responsibilities of the different components that AliECS interfaces with are handled by different teams, and the components themselves are deployed on separate hardware, sometimes even separate networks, which presents challenges to system integration. While the final round of integration testing is still performed manually on the staging instance of the O^2 system, we have put in place an automated integration pipeline that deploys and operates as much as possible of the system on a single node.
5. The deployment and teardown procedures of a data-taking environment are inevitably complex due to the inherent complexity of the system. These procedures require a carefully choreographed sequence of API calls and state transitions involving multiple systems and thousands of processes. On the one hand, some of the steps in these sequences have dependencies on one another that can't be missed; on the other hand, if all the steps are made strictly sequentially, the operation in question takes longer, leading to a waste of beam time. We have addressed this challenge by implementing a mechanism based on transition substeps and indexes in the AliECS core and finely tuning the configuration of environment deployment and teardown procedures to parallelize what we can.
6. Dozens of different kinds of data flow tasks are composed into highly complex workflows, and some of these tasks may, at times, misbehave, such as failing to comply with a transition request or allocating too much memory. We have addressed these kinds of challenges in various ways, such as defensive coding, continuous integration, and cgroups-based task isolation.

Thanks to the above, AliECS, as well as the O^2 system as a whole, are ready for the heavy ion beams in 2023 and well positioned for high-efficiency data taking in 2024 and beyond. By the beginning of 2023 AliECS with the O^2 computing system enabled ALICE to reach over 90% running efficiency in production.

5 Conclusion

AliECS is a new microservices-oriented and integrated solution for ALICE experiment control; it was designed, developed and deployed as part of the O^2 computing system in the O^2/FLP facility at LHC Point 2. AliECS is part of a broad technical refresh for ALICE, leveraging modern cluster resource management and IPC technologies for a high-performance, low-latency ECS.

By employing Apache Mesos as a platform for AliECS, we provide cluster resource management, task isolation, control message transport, events, and more to achieve improved operational flexibility. On top of this framework, we implement a distributed state machine mechanism, with an expressive configuration format, highly flexible system integration facilities, and a modular process control stack for maximum compatibility in an inevitably heterogeneous context. Open source cross-platform and cross-language technologies such as gRPC, Git and Consul are employed to maximize interoperability and minimize technical risks.

AliECS empowers the ALICE collaboration to maximize usage of LHC beam time while ensuring optimal resource allocation in the new O^2 facility for data-driven workflows. AliECS takes direct control of the O^2/FLP facility and interfaces with the O^2/EPN cluster control to gain high-level oversight of the whole data readout chain. Our design approach has achieved substantial performance improvements and operational benefits in mission-critical use cases compared to the old system.

References

[1] K. Aamodt et al. (ALICE), JINST **3.08**, S08002 (2008)
[2] B. Abelev et al. (ALICE), Journal of Physics G: Nuclear and Particle Physics **41**, 087001 (2014)
[3] J. Adam et al. (ALICE), Tech. Rep. CERN-LHCC-2015-006 / ALICE-TDR-019, CERN (2015)
[4] J. Mitra, S. Khan, S. Mukherjee, R. Paul, Journal of Instrumentation **11**, C03021 (2016)
[5] O. Bourrion, J. Bouvier, F. Costa, E. Dávid, J. Imrek, T. Nguyen, S. Mukherjee, Journal of Instrumentation **16**, P05019 (2021)
[6] A. Borga, F. Costa, G. Crone, H. Engel, D. Eschweiler, D. Francis, B. Green, M. Joos, U. Kebschull, T. Kiss et al., Journal of Instrumentation **10**, C02022 (2015)
[7] G.C. Raduta, Journal of Physics: Conference Series **2438**, 012041 (2023)
[8] P. Chochula et al., Proceedings of the 16th International Conference on Accelerator and Large Experimental Control Systems pp. 323–327 (2018)
[9] *FairMQ C++ Message Queuing Library and Framework*, https://github.com/FairRootGroup/FairMQ, accessed: 2023-08-30
[10] M. Al-Turany, D. Bertini, R. Karabowicz, D. Kresan, P. Malzacher, T. Stockmanns, F. Uhlig, Journal of Physics: Conference Series **396**, 022001 (2012)
[11] M. Al-Turany, P. Buncic, P. Hristov, T. Kollegger, C. Kouzinopoulos, A. Lebedev, V. Lindenstruth, A. Manafov, M. Richter, A. Rybalchenko et al., Journal of Physics: Conference Series **664**, 072001 (2015)
[12] *Apache Mesos*, http://mesos.apache.org/, accessed: 2023-08-30
[13] D. Berzano, G. Eulisse, C. Grigoraş, K. Napoli, Journal of Physics: Conference Series **898**, 082043 (2017)
[14] *The O^2 Control System*, https://github.com/AliceO2Group/Control, accessed: 2023-08-30
[15] *The Go Programming Language*, https://golang.org/, accessed: 2023-08-30
[16] *gRPC A high performance, open-source universal RPC framework*, https://grpc.io/, accessed: 2023-08-30
[17] *Consul by HashiCorp*, https://www.consul.io/, accessed: 2023-08-30