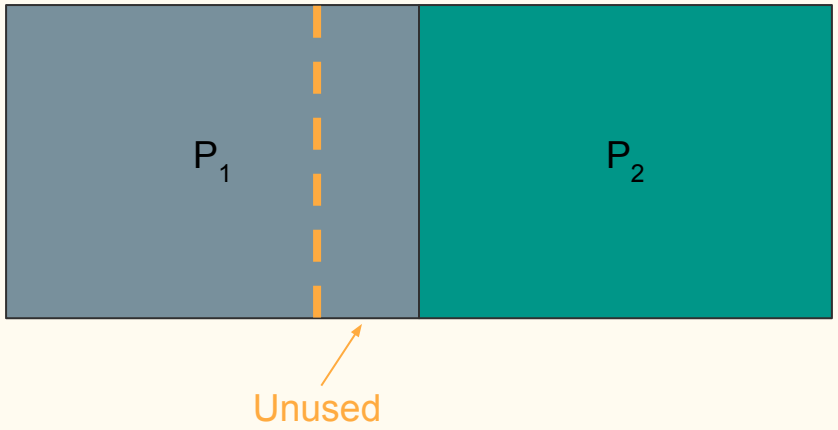
Microservices Scheduling for ALICE O^2 Facility

—

Kevin Napoli
Supervisor: Giulio Eulisse

Architectural Shift

- Applications cannot easily run together on the same cluster
- Statically partitioning the cluster is not always effective
- Implies Architectural shift from clusters running a single application to ones that can run multiple applications and reassign resources dynamically



Microservices Architecture

- Microservices:
  - Small in scope. Each application has one single purpose.
  - Self contained. Usually (but not necessarily) deployed using containers.
  - Designed for horizontal scaling. Extra capacity by adding new instances.
  - Disposable. Distributed nature of the system makes each application instance non-fundamental.
- Advantages:
  - Designed to handle dynamic workloads and ultimately improve resource utilisation
  - Reduce / remove single points of failure.
  - Smaller scope simplifies replacing one component with another.

ALICE O^2

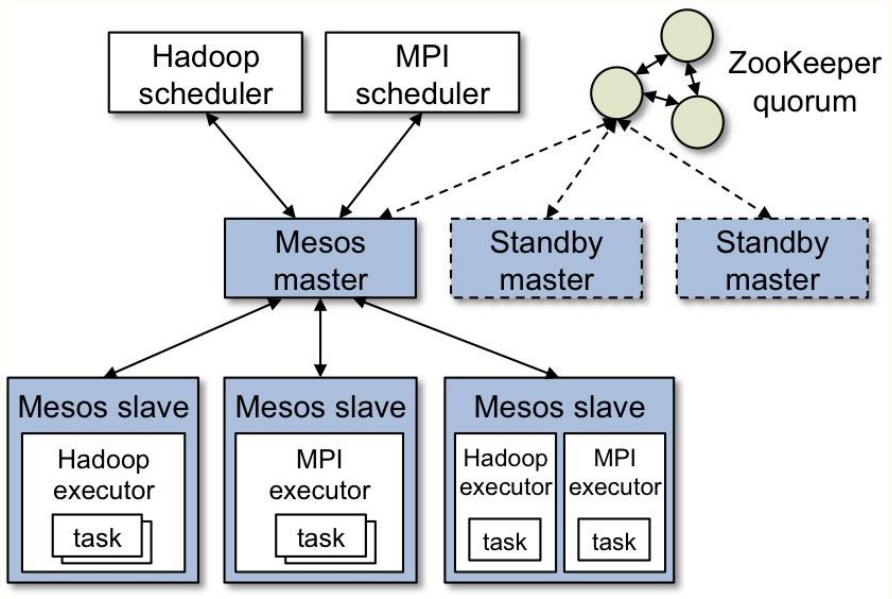
- Currently, dedicated clusters are responsible for the online tasks of data acquisition and storage
- In order to meet the elevated data requirements of Runs 3 and 4, ALICE computing will upgrade to O^2
- ALICE O^2 is a new concurrency framework that will be used for the next run
- With the O^2 software framework, a common ALICE computing facility will share data acquisition and processing responsibilities
- O^2 will need to dynamically partition the cluster among a broad variety of online and offline jobs with a high degree of flexibility and resilience

Dynamic Deployment System

- Dynamic Deployment System (DDS) is a toolset that simplifies the process of deploying tasks on a cluster
- Can be used by the O^2 software framework to deploy microservices
- DDS deploys ‘agents’ on cluster nodes
  - The agents connect back to DDS Commander
  - The agents are instructed to execute processes by the DDS Commander
- DDS features pluggable Resource Management Systems
- DDS resources could potentially occupy the whole cluster
  - A list of nodes where agents will be deployed must be known ahead of time

Apache Mesos

- Apache Mesos is to a cluster what a Kernel is to a PC
- Offers fine-grained resource allocation
- Allows different applications to run on the same cluster
- Allows applications to scale elastically
- Offers a two-level scheduling mechanism and allocates resources by negotiation
- Redundancy makes Mesos fault tolerant
- Used by industry giants such as Google, eBay, Twitter and more
- Many applications like Hadoop can already run on Mesos



Solutions and Status

- Implement a plugin for DDS and a Mesos framework so that DDS can run on a Mesos governed cluster alongside other frameworks
  - Existing DDS users can use the normal interface - no changes and no learning curve
  - Elastic frameworks will benefit
- Try Mantl.io and analyse the complexities of deploying DDS agents from the Mantl GUI interface
- Implement a network topology detector to help Mesos and its framework take better scheduling decisions
  - Minor testing performed
  - Implement this as a Mesos Allocator - In progress

Future Work

- Extend DDS to better exploit Mesos resource allocation capabilities
- Integrate and evaluate network topology awareness in Mesos
- Evaluate on larger scale deployments

Thank You