Karlsruhe Institute of Technology

Dynamic & Transparent Integration and Management of Compute Resources with COBaID/TARDIS

Managing Apache Spark with COBaID/TARDIS Workshop, 19.01.2021

Manuel Giffels, René Caspart, Max Fischer, Eileen Kühn, Matthias Schnepf, Florian v. Cube

TARDIS - The Transparent Adaptive Resource Dynamic Integration System

Karlsruhe Institute of Technology

TARDIS is a resource manager …

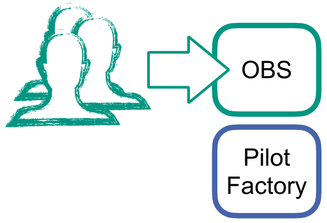
- … that enables the dynamic and transparent integration of resources of different providers into one common pool of resources
- … relies on COBalD - the Opportunistic Balancing Daemon in order to balance those resources (based on utilisation and allocation)
- … that is written in Python featuring Asynchronous I/O
- … available via PyPI

Transparent Integration: The Overlay Batch System (OBS)

Karlsruhe Institute of Technology

Or how the Grid is used today:

- Pilot factory submits placeholder jobs (pilots) to different sites
- Pilot allocates resources at the site
- Resources are integrated into the OBS
- Workload is pulled from the OBS
- Users interact only with one single-point-of-entry the OBS



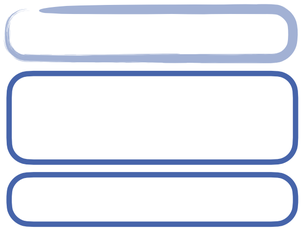


Illustration by Max Fischer (KIT)

Transparent Integration: The Overlay Batch System (OBS)

Or how the Grid is used today:

- Pilot factory submits placeholder jobs (pilots) to different sites
- Pilot allocates resources at the site
- Resources are integrated into the OBS
- Workload is pulled from the OBS
- Users interact only with one single-point-of-entry the OBS

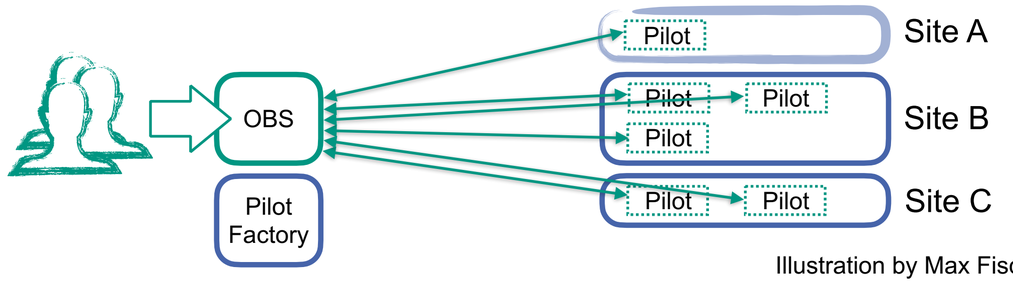


Illustration by Max Fischer (KIT)

Transparent Integration: The Overlay Batch System (OBS)

Or how the Grid is used today:

- Pilot factory submits placeholder jobs (pilots) to different sites
- Pilot allocates resources at the site
- Resources are integrated into the OBS
- Workload is pulled from the OBS
- Users interact only with one single-point-of-entry the OBS

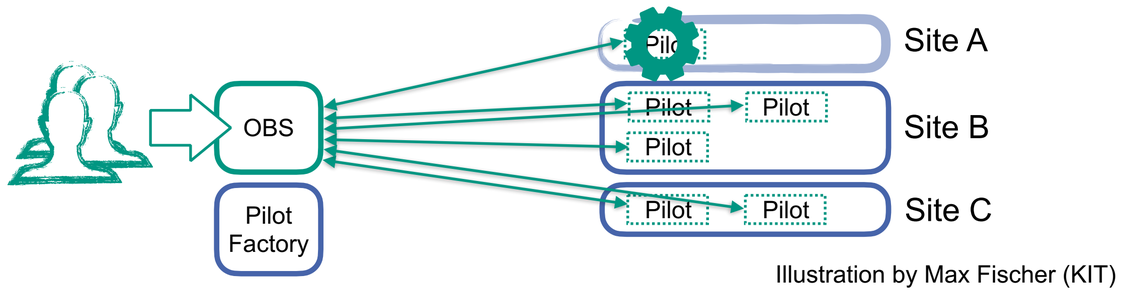


Illustration by Max Fischer (KIT)

Transparent Integration: The Overlay Batch System (OBS)

Or how the Grid is used today:

- Pilot factory submits placeholder jobs (pilots) to different sites
- Pilot allocates resources at the site
- Resources are integrated into the OBS
- Workload is pulled from the OBS
- Users interact only with one single-point-of-entry the OBS

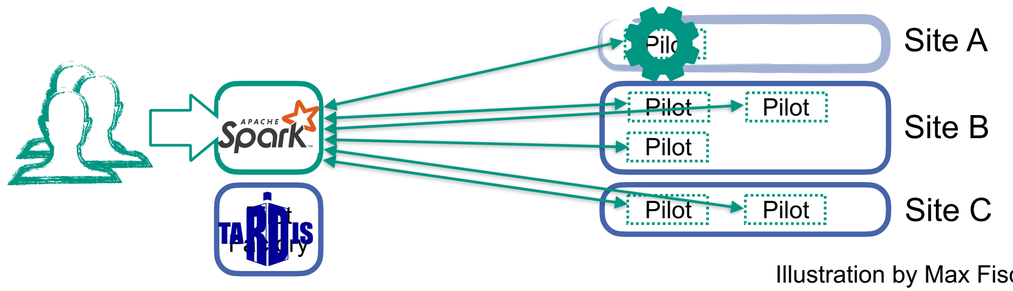


Illustration by Max Fischer (KIT)

Drone Concept

- Pilot concept
  - Allocate and integrate resources (via OBS)
  - Usually just a batch system daemon plus wrapper scripts
- Drone is a generalised pilot
  - Allocate and integrate resources (via OBS same as a pilot)
  - Provide dedicated environment (OS/Software) via virtualisation or containerisation techniques
  - Can be a script or container launching Apache Spark client daemon

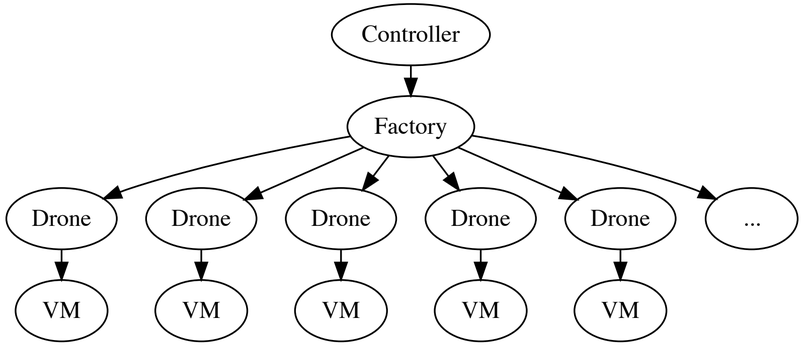
Dynamic Resource Integration with TARDIS

Transparent Adaptive Resource Dynamic Integration System (TARDIS):

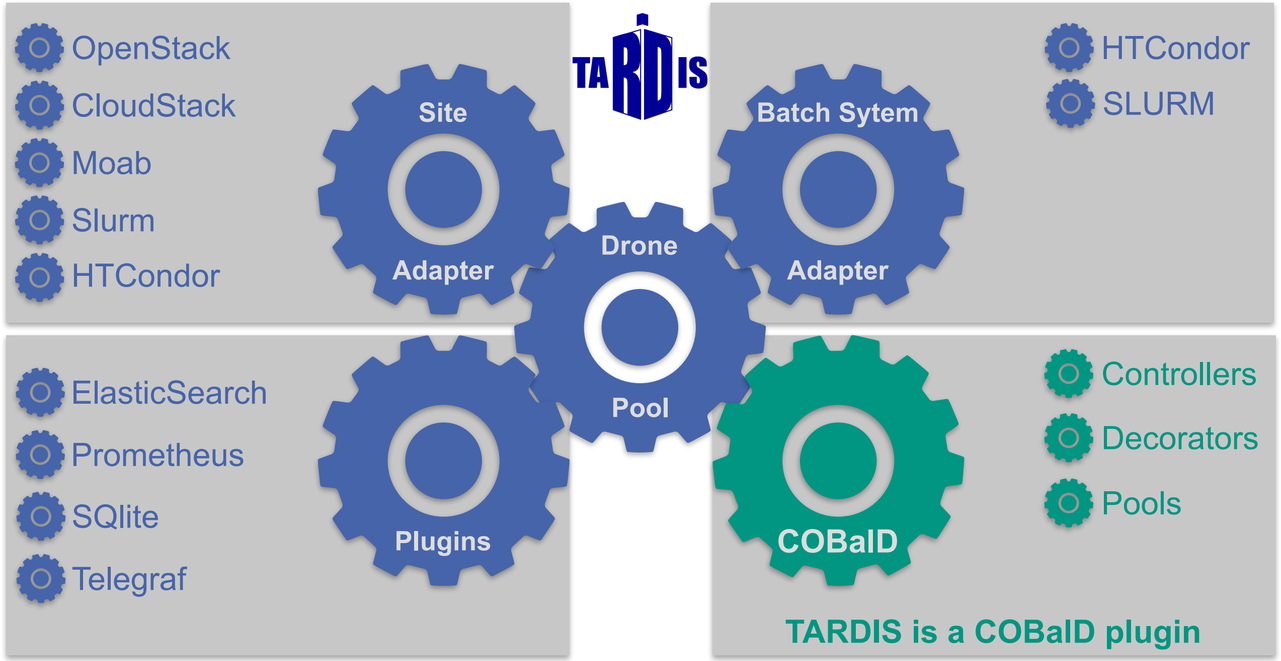
- Provides backends to various cloud providers and batch systems
- Allows dynamic orchestration of pre-built VMs and containers
- Asynchronous multi-agent COBalD plugin (ensures scalability)

Structure:

- A resource is represented by a Drone implementing the pool interface
- Drones are aggregated in dynamically sized composite pools
- Factory pool acquires and releases resources as desired by the controller

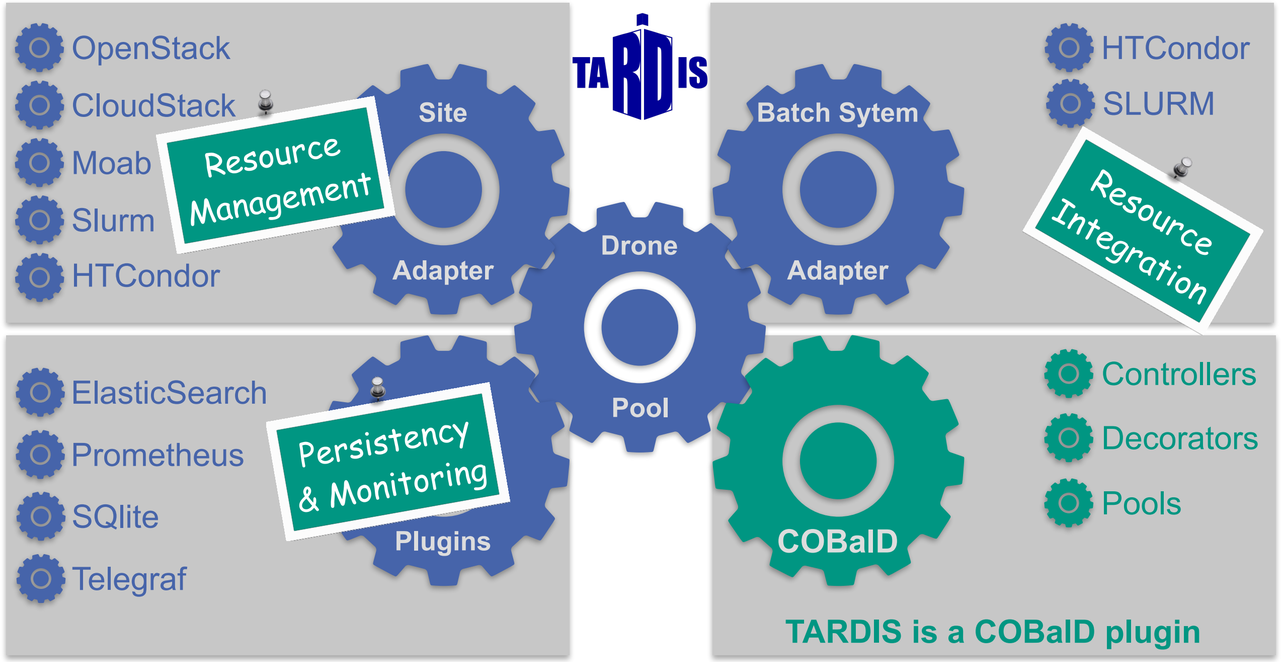


Components of COBalD/TARDIS



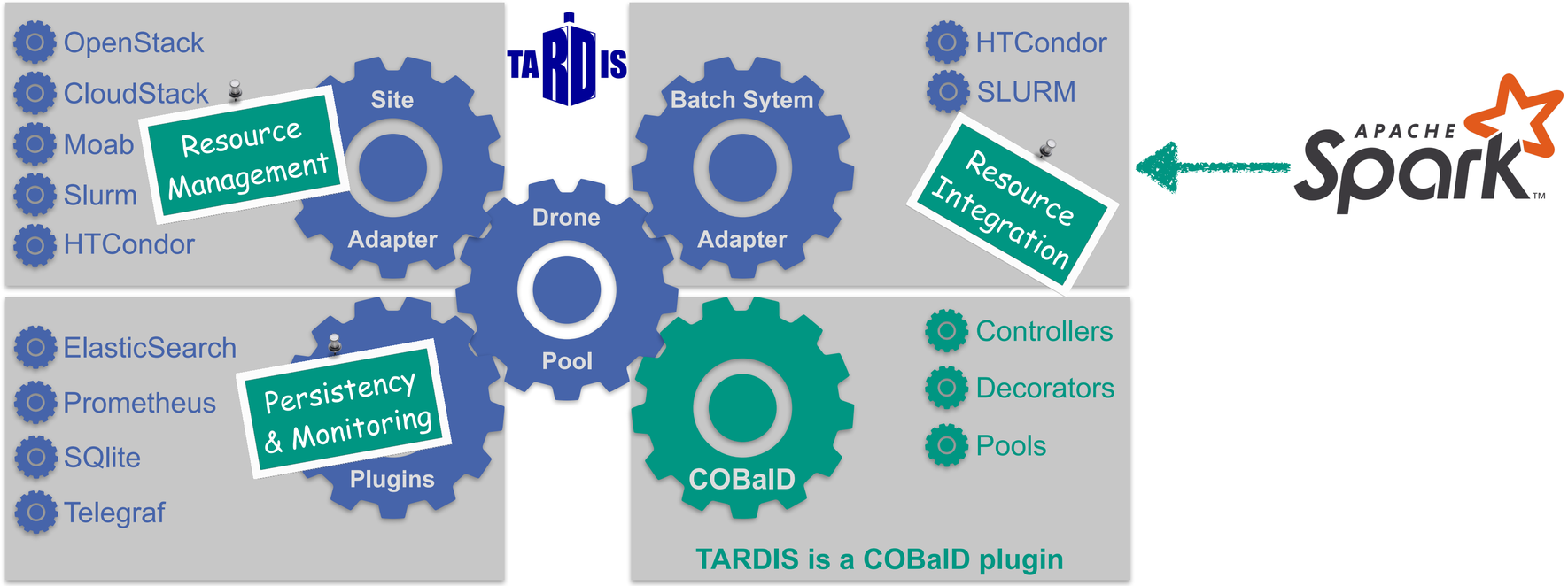
Well defined abstract base class interfaces to all components available!

Components of COBalD/TARDIS



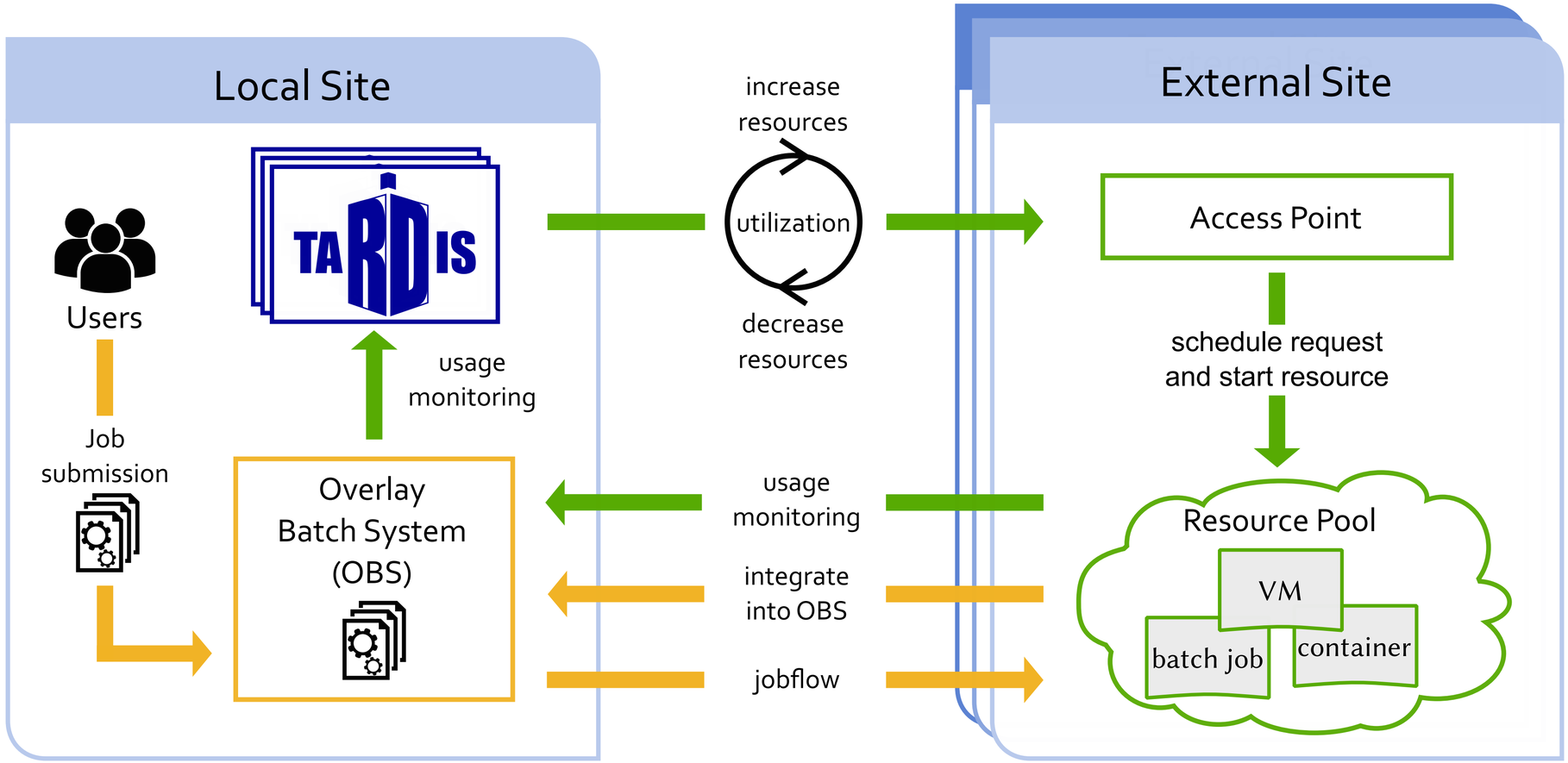
Well defined abstract base class interfaces to all components available!

Components of COBalD/TARDIS



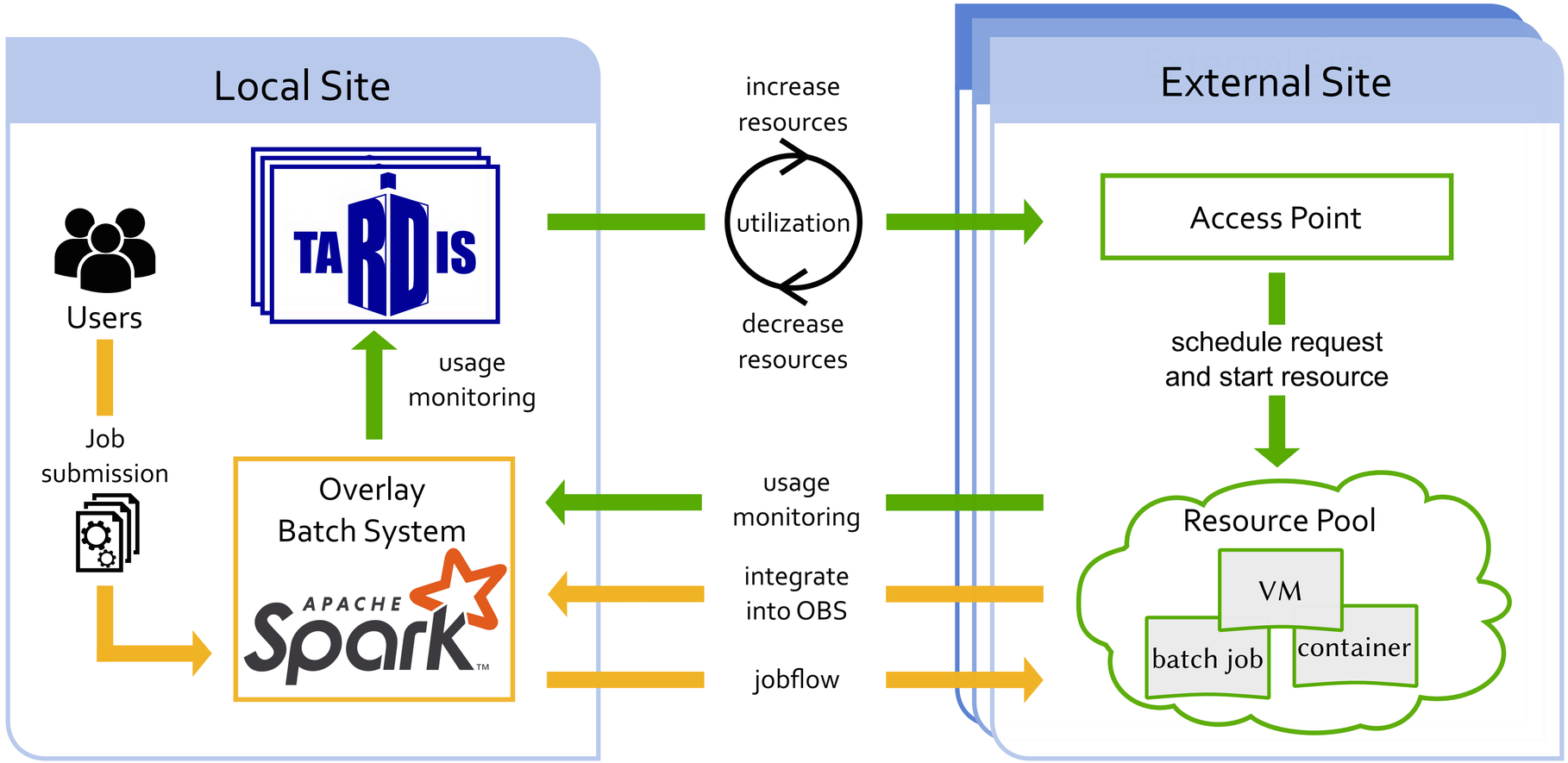
Well defined abstract base class interfaces to all components available!

The Entire Picture



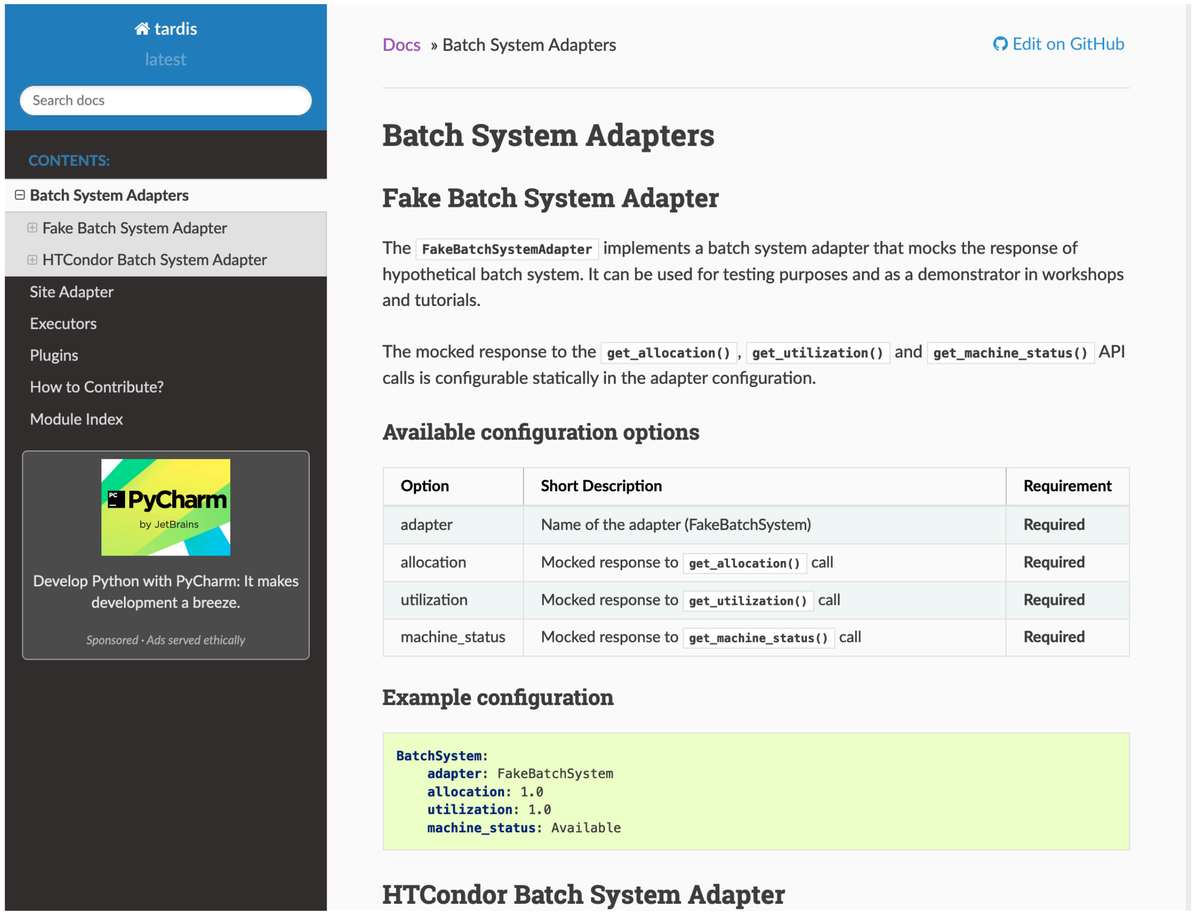
One COBalD/TARDIS instance per remote site

The Entire Picture



One COBalD/TARDIS instance per remote site

Documentation is Available



https://cobald-tardis.readthedocs.io/en/latest/index.html

Interface of BatchSystemAdapter

class **tardis.interfaces.batchsystemadapter.BatchSystemAdapter** [source]

Bases: `object`

Abstract base class defining the interface for BatchSystemAdapters which handles integration and management of resources in the overlay batch system.

abstract async **disintegrate_machine**(*drone_uuid: str*) → None [source]

Disintegrate a machine from the overlay batch system.

Parameters: **drone_uuid** –

Uuid of the worker node, for some sites corresponding

to the host name of the drone.

type drone_uuid: str

return: None

abstract async **drain_machine**(*drone_uuid: str*) → None [source]

Drain a machine in the overlay batch system, which means that no new jobs will be accepted

Parameters: **drone_uuid** (*str*) – Uuid of the worker node, for some sites corresponding to the host name of the drone.

Returns: None

Interface of BatchSystemAdapter

abstract async **get_allocation**(*drone_uuid: str*) → float [source]

Get the allocation of a worker node in the overlay batch system, which is defined as maximum of the ratios of requested over total resources (CPU, Memory, Disk, etc.).

Parameters: **drone_uuid** (*str*) – Uuid of the worker node, for some sites corresponding to the host name of the drone.

Returns: The allocation of a worker node as described above.

Return type: float

abstract async **get_machine_status**(*drone_uuid: str*) → tardis.interfaces.batchsystemadapter.MachineStatus [source]

Get the status of a worker node in the overlay batch system (Available, Draining, Drained, NotAvailable)

Parameters: **drone_uuid** (*str*) – Uuid of the worker node, for some sites corresponding to the host name of the drone.

Returns: The machine status in HTCondor (Available, Draining, Drained, NotAvailable)

Return type: MachineStatus

abstract async **get_utilisation**(*drone_uuid: str*) → float [source]

Get the utilisation of a worker node in the overlay batch system, which is defined as minimum of the ratios of requested over total resources (CPU, Memory, Disk, etc.).

Parameters: **drone_uuid** (*str*) – Uuid of the worker node, for some sites corresponding to the host name of the drone.

Returns: The utilisation of a worker node as described above.

Return type: float

Interface of BatchSystemAdapter

abstract async **integrate_machine**(*drone_uuid: str*) → None [source]

Integrate a machine into the overlay batch system.

Parameters: **drone_uuid** (*str*) – Uuid of the worker node, for some sites corresponding to the host name of the drone.

Returns: None

abstract property **machine_meta_data_translation_mapping**

The machine meta data translation mapping is used to translate units of the machine meta data in `TARDIS` as expected by the overlay batch system.

Returns: machine meta data translation mapping

Return type: AttributeDict

class **tardis.interfaces.batchsystemadapter.MachineStatus**(*value*) [source]

Bases: `enum.Enum`

An enumeration.

Available= *1*

Drained= *3*

Draining= *2*

NotAvailable= *4*

How to reach us?

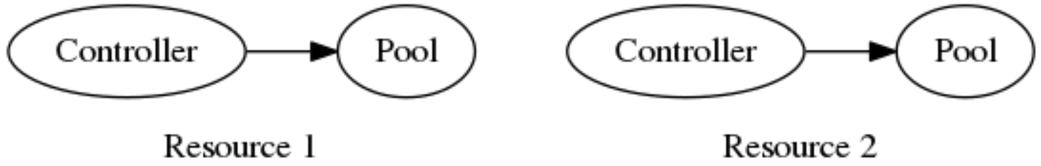
- Tutorial: https://slide-hub.github.io/
- https://chat.eudat.eu/matterminers
- https://gitter.im/MatterMiners/community

Questions?

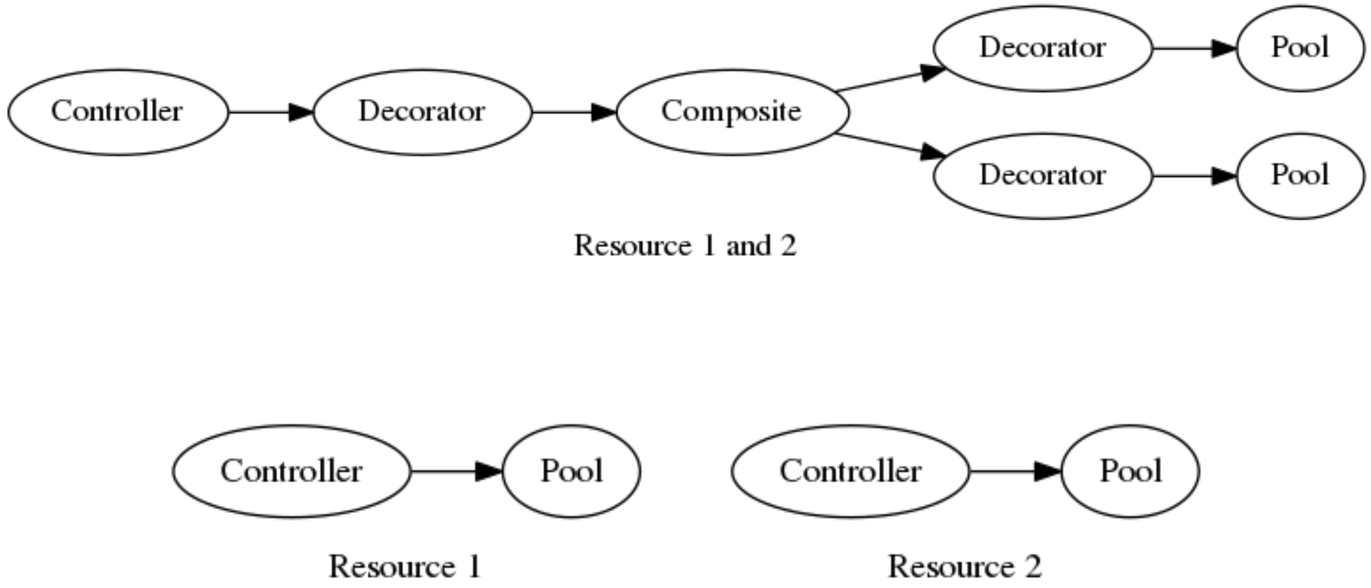
Backup

COBalD Resource Pool Model

COBalD Resource Pool Model

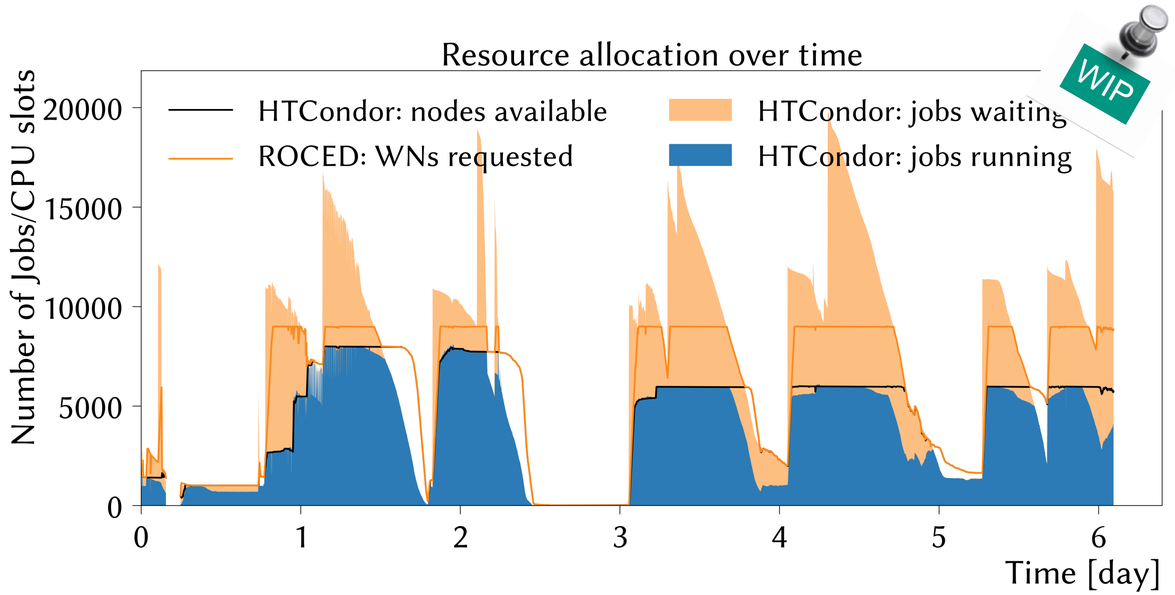


COBalD Resource Pool Model



Pragmatic View to Resource Management

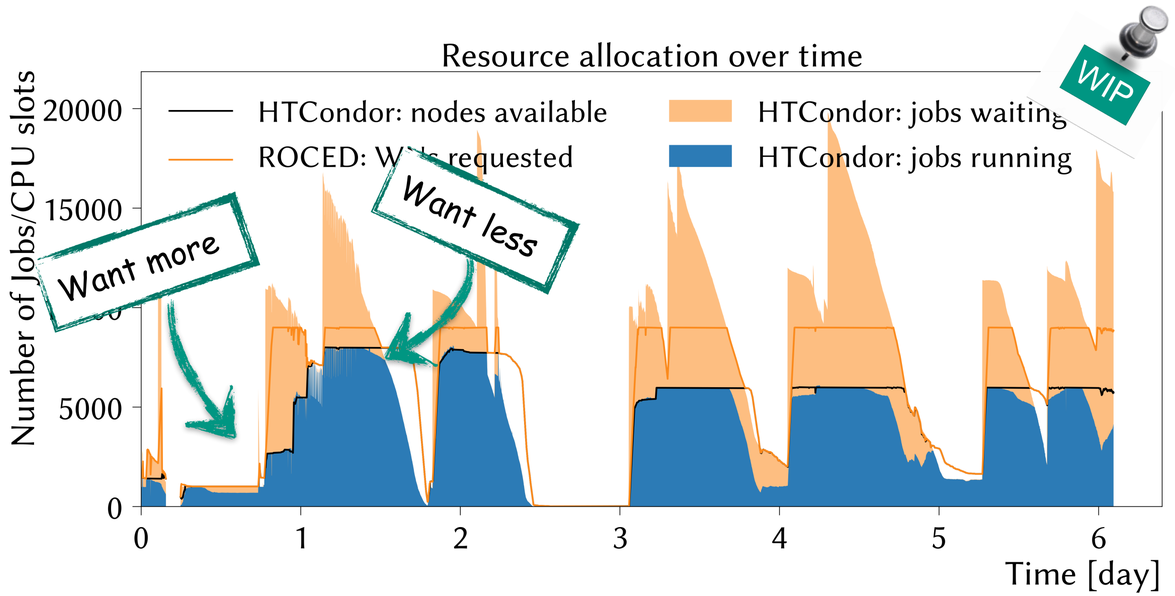
- Development for scalability and maintainability
- Simple logic: more used resources, less unused resources
  - COBalD only watches, creates and disables resources
  - Batch system scheduler selects appropriate resources



Slide by Max Fischer (KIT)

Pragmatic View to Resource Management

- Development for scalability and maintainability
- Simple logic: more used resources, less unused resources
  - COBalD only watches, creates and disables resources
  - Batch system scheduler selects appropriate resources



Slide by Max Fischer (KIT)

Scalability