Kisti hack generalisation (Scheduler refactor)

Usage:

The machine and platform values will be automatically loaded by the modules qcore.config and shared_workflow.platform_config. Each has a variable for the machine/platform name and a dictionary of configured values, loaded from the relevant configuration file. Use on a local machine may require configuring the machine_local.json and platform_local.json files. The HPC enumeration is also available from shared_workflow.platform_config and is dynamically generated from the platform configuration file.

The scheduler to be used is determined by the platform configuration. The scheduler must be initialised by any script that wishes to submit, cancel or check the status of jobs. Once the scheduler has been initialised by providing the username and optionally account of the user the scheduler may be accessed.

The scheduler functions can be accessed from scripts.schedulers.scheduler_factory in the slurm_gm_workflow repository.

The available top level functions are intialise_scheduler and get_scheduler.

Calling get_scheduler without first calling initialise_scheduler will result in an exception, calling initialise_scheduler again will also result in an exception.

Kisti hack integration, or how I learned to stop worrying and love the scheduler.

In order to integrate the changes to the workflow for Tacc and Kisti it was necessary to generalise and abstract out job scheduler operations and platform /machine specific configurations.

Machine configs

The simplest of these was to create a machine config for each supported machine in qcore. This worked in the same way as the existing Maui and Mahuika configuration files.

The supported machines now include Maui, Mahuika, Stampede2, Nurion. Users can also set values in a local machine config if they wish.

The machine configuration to load is determined by the host name using pre-set known host name prefixes.

Platform configs

Some settings were required on a platform level, such as the scheduler used.

This required the implementation of platform configurations in the slurm_gm_workflow repository in the same manner as the machine configuration files.

The platform to load is selected by the machine that was detected. A local machine platform is available if a user would like to run the workflow locally.

The content in the platform config is primarily the content of the previous workflow_config.json file. However some other notable inclusions are the task to machine mapping formerly in auto_submit.py, and the HPCs available on the platform. The platform config is checked on loading to ensure it contains exactly the expected keys, which are the elements of the PLATFORM_CONFIG enumeration found in qcore.constants.

At present the following machine to platform mappings exist:

Machine	Platform
Maui	NeSI
Mahuika	NeSI
Stampede2	Tacc
Nurion	Kisti
Local	Local

Scheduler

The most difficult part of the integration is the abstraction of the scheduler operations.

The platforms NeSI and Tacc both use the Slurm job scheduler, while Kisti uses the Torque PBS scheduler. In addition the implementations used by NeSI and Tacc have some key differences requiring slightly different job script content.

In order to abstract the Scheduler behaviour an AbstractScheduler was made, with concrete implementations of Slurm and PBS. A limited Bash implementation is available for the local machine.

The primary scheduler interactions currently implemented in the workflow are job submission, job cancellation and queue checking.

The scheduler to load is chosen by setting the scheduler value in the platform configuration.

The current platform to scheduler mapping is below:

Platform	Scheduler
NeSI	Slurm
Tacc	Slurm
Kisti	PBS
Local	Bash

Table of machines/platforms/schedulers

Machine	Platform	Scheduler
Maui	NeSI	Slurm
Mahuika	NeSI	Slurm
Stampede2	Tacc	Slurm
Nurion	Kisti	PBS
Local	Local	Bash