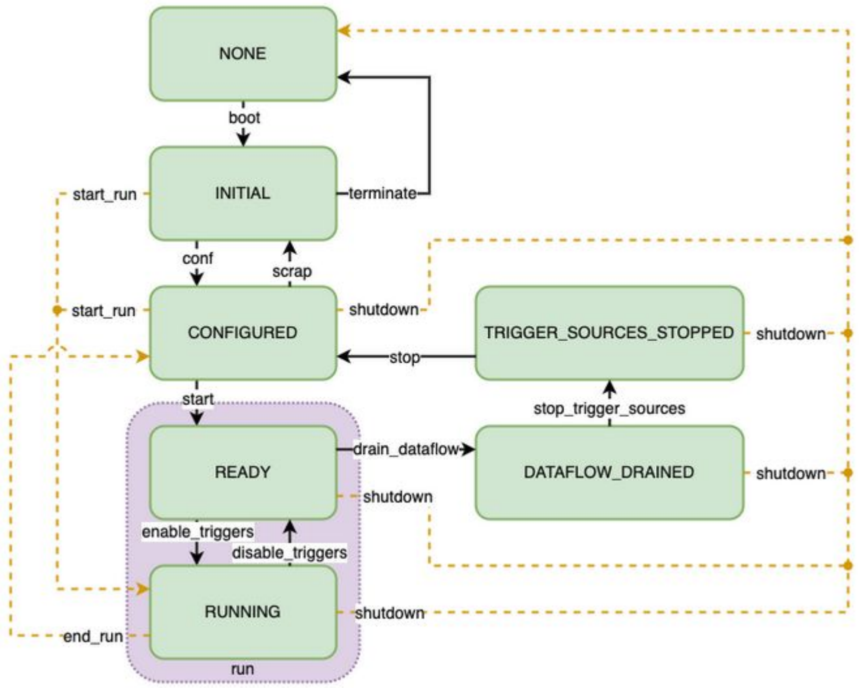
Recent work on run control

Jonathan Hancock
University of Birmingham
On behalf of the DUNE CCM group

Run Control Overview

- The prototype run control currently in use for testing is called NanoRC (Not ANOther Run Control).
- It is responsible for generating a set of DAQ apps given in a configuration file.
- The state of these apps can be represented by a Finite State Machine, and the user can enter commands in order to make them undergo transitions.



The Kubernetes Cluster

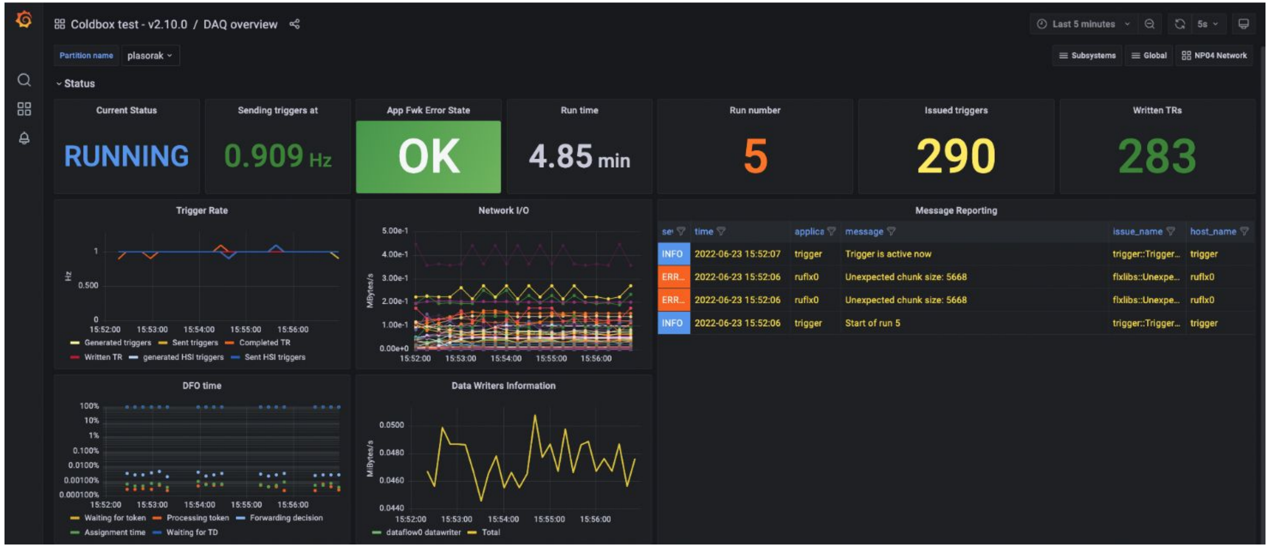
- NanoRC can use an SSH process manager, but the intended usage pattern is to host apps with Kubernetes.
- Kubernetes is a container orchestration package that can run many containerised applications concurrently, and is used to automate processes such as creating and restarting instances of applications, and managing IP addresses for services.
- The cluster used for testing at NP04 consists of 15 nodes with different assigned purposes.
  - Control Plane
  - DAQ host
  - Storage
  - FELIX host

Using NanoRC

- Configurations are generated by writing a .json file and passing it to the daqconf_multiru_gen tool.
- The tool generates a set of jsons that serve as the configuration.
- This config is then uploaded to the config service (mentioned again later) which saves it to a MongoDB database.
- NanoRC is then run from the command line by passing it the addresses of the Kubernetes cluster, a partition name, and the name of the config stored in the DB.
- The desired DAQ applications are initialised in pods on the K8s cluster.
- The user sends commands to NanoRC by using the command line, or though a UI. NanoRC will then execute these commands on the applications.

Monitoring

- Metrics relating to cluster nodes are extracted as time series data using Prometheus.
- This data is passed to Grafana, and visualised in the web browser.
- Another Grafana dashboard is used to keep track of the state of NanoRC.



Current Status	Sending triggers at	App Fwk Error State	Run time	Run number	Issued triggers	Written TRs
RUNNING	0.909 Hz	OK	4.85 min	5	290	283

Message Reporting

se	time	applica	message	issue_name	host_name
INFO	2022-06-23 15:52:07	trigger	Trigger is active now	trigger::Trigger...	trigger
ERR...	2022-06-23 15:52:06	ruflx0	Unexpected chunk size: 5668	flxlibs::Unexpe...	ruflx0
ERR...	2022-06-23 15:52:06	ruflx0	Unexpected chunk size: 5668	flxlibs::Unexpe...	ruflx0
INFO	2022-06-23 15:52:06	trigger	Start of run 5	trigger::Trigger...	trigger

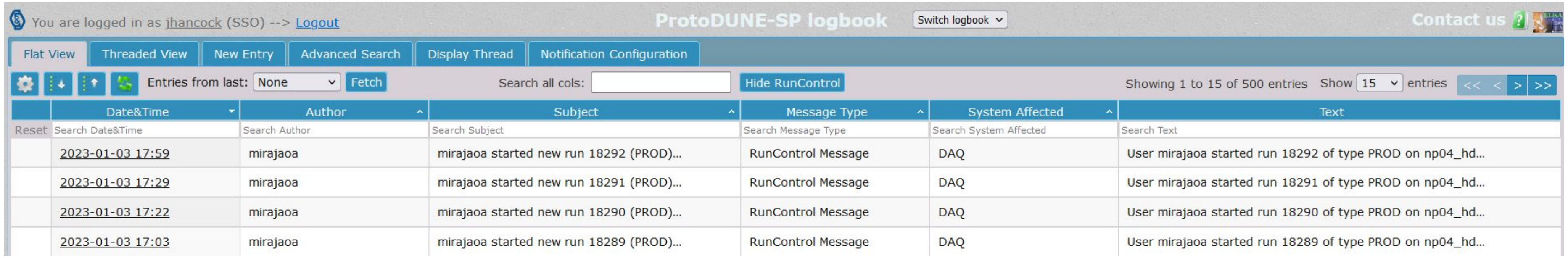
Microservice Architecture

It is desirable to split the functionality of NanoRC into several microservices that operate independently and communicate via REST when needed. Currently 7 have been developed.

- Monitoring
  - ERS database writer
  - Opmon database writer
- Run Control
  - Config service
  - Connection service
  - Logbook service
  - Run Number service
  - Run Registry service

ELisA Logbook Service

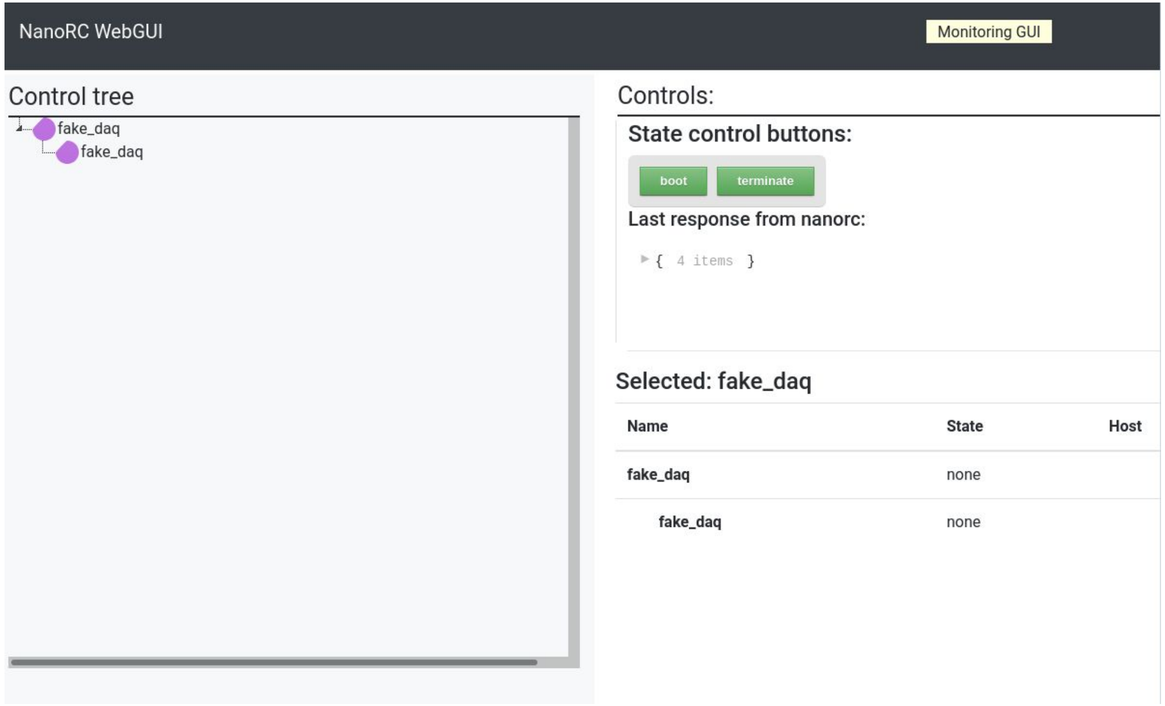
- The service is a Flask API, with routes for starting runs, stopping runs, and creating an informational message.
- Requests are sent via NanoRC (WIP) or manually with cURL. Messages can also be added via the website.
- The service then handles the communication with the ELisA logbook, and sends it the appropriate message.
- Once NanoRC is modified to communicate with this microservice, an instance of it will be set up to be permanently available in the Kubernetes cluster.



Date&Time	Author	Subject	Message Type	System Affected	Text
2023-01-03 17:59	mirajaoa	mirajaoa started new run 18292 (PROD)...	RunControl Message	DAQ	User mirajaoa started run 18292 of type PROD on np04_hd...
2023-01-03 17:29	mirajaoa	mirajaoa started new run 18291 (PROD)...	RunControl Message	DAQ	User mirajaoa started run 18291 of type PROD on np04_hd...
2023-01-03 17:22	mirajaoa	mirajaoa started new run 18290 (PROD)...	RunControl Message	DAQ	User mirajaoa started run 18290 of type PROD on np04_hd...
2023-01-03 17:03	mirajaoa	mirajaoa started new run 18289 (PROD)...	RunControl Message	DAQ	User mirajaoa started run 18289 of type PROD on np04_hd...

UI Development

- In addition to the standard command line interface, NanoRC has a web GUI, and a TUI which is still being developed. To use them, a command option should be provided when starting NanoRC.
- Selecting the GUI spawns a Flask server, and provides an address that can be connected to in the browser using a SOCKS5 proxy.
- Selecting the TUI creates a Textualize app, which is interacted with directly from the terminal.
- The foundation of these interfaces is the API that was built into NanoRC for this purpose.
- By communicating with the API, external applications can send commands to the RC, and extract a list of all apps and their FSM state.
- Consequently, UI development is decoupled from run control development, meaning changes in one will not break the other.

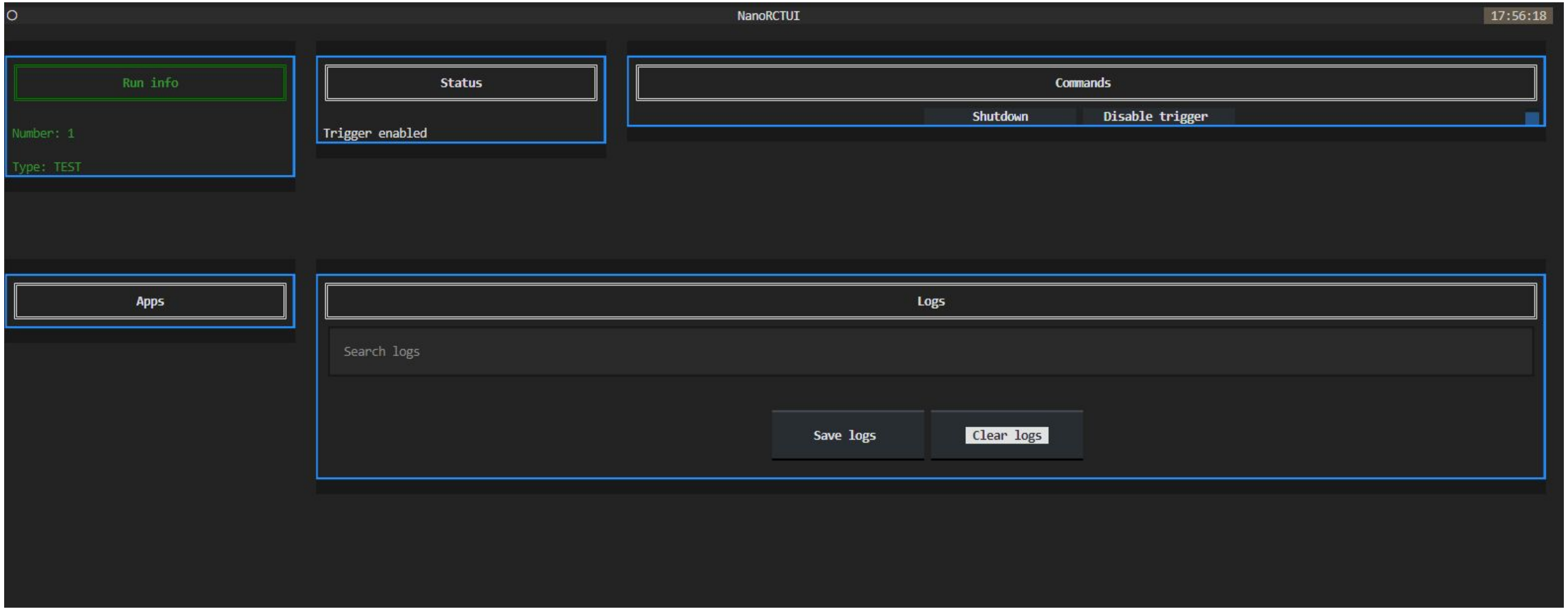


Last response from nanorc:

```
▶ { 4 items }
```

Selected: fake_daq

Name	State	Host
fake_daq	none	
fake_daq	none	



```
Number: 1
Type: TEST
Trigger enabled
```

The future of Run Control

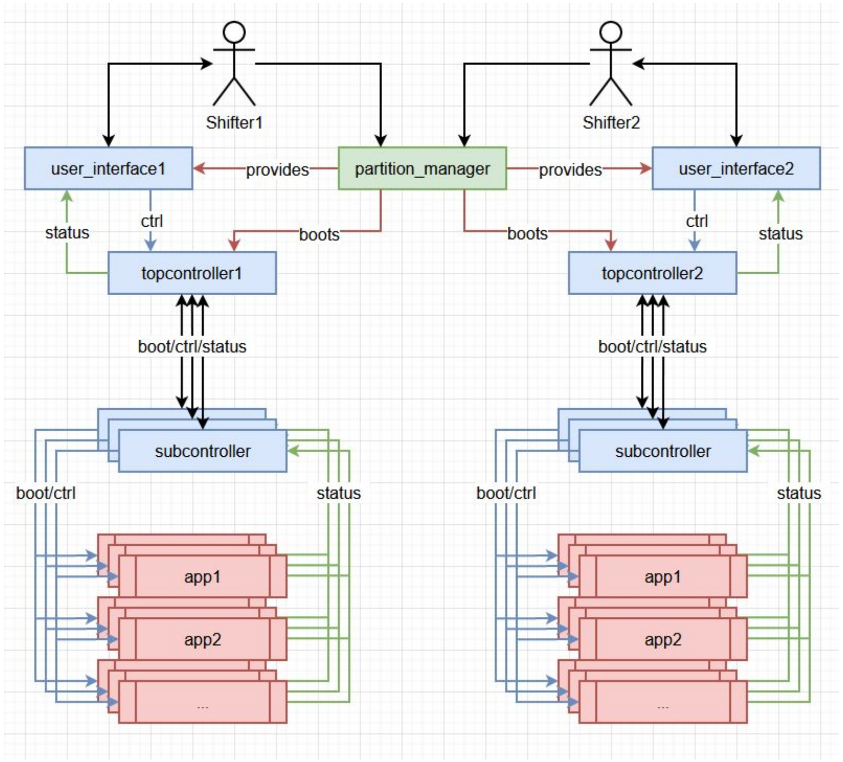
It is expected that the centre of this year's development work will be the new (as of yet unnamed) Run Control. This should hopefully be the RC that is used for the actual DUNE experiment, and will make use of all the lessons that were learned in the development of NanoRC.

Currently Pierre Lasorak is the sole developer of the Run Control code, since it is in a foundational stage.

Features

- Different parts of the control tree can be controlled independently.
- gRPC will be used instead of REST.
- Proper user authentication.
- Handling of partitions/sessions

Currently, a working prototype is expected to be complete by the April 2023 release, minus the authentification. Full functionality should be achieved by September, and NanoRC should be replaced as the default by the end of the year.


Shifter1
Shifter2
user_interface1
partition_manager
user_interface2
provides
provides
status
ctrl
boots
boots
ctrl
status
topcontroller1
topcontroller2
boot/ctrl/status
boot/ctrl/status
subcontroller
subcontroller
boot/ctrl
status
boot/ctrl
status
app1
app2
...
app1
app2
...

Conclusion

- NanoRC is nearing the end of its development (it was never planned to take it this far), and we are in a good position to begin with the final Run Control.
- A robust framework for hosting DAQ applications has been created in the Kubernetes cluster, with good options for observability of relevant metrics.
- UI development is proceeding with flexibility in mind, leaving the door open for alternate UI designs, as well interactions by other entities such as supervisors.