Kubernetes for the Deep Underground Neutrino Experiment Data Acquisition

Pierre Lasorak[1,*], *Tiago* Alves[1], *Gordon* Crone[2], *Enrico* Gamberini[3], *Jonathan* Hancock[4], *Bonnie* King[5], *Patrick* Riehecky[5], *Alexander* Tapper[1], and *Alessandro* Thea[6] for the DUNE Collaboration

[1]Imperial College of Science Technology and Medicine, London SW7 2BZ, United Kingdom
[2]University College London, London WC1E 6BT, United Kingdom
[3]CERN, The European Organization for Nuclear Research, 1211 Meyrin, Switzerland
[4]University of Birmingham, Birmingham B15 2TT, United Kingdom
[5]Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA
[6]STFC Rutherford Appleton Laboratory, Didcot OX11 0QX, United Kingdom

Abstract. The Deep Underground Neutrino Experiment (DUNE) is a next-generation long-baseline neutrino experiment based in the USA which is expected to start taking data in 2029. DUNE aims to precisely measure neutrino oscillation parameters by detecting neutrinos from the LBNF beamline (Fermilab) at the Far Detector, 1,300 kilometres away, in South Dakota at the Sanford Underground Research Facility. The Far Detector will consist of four cryogenic Liquid Argon Time Projection Chamber detectors of 17 kT, each producing more than 1 TB/sec of data. The main requirements for the data acquisition system are the ability to run continuously for extended periods of time, with a 99% up-time requirement, and the functionality to record both beam neutrinos and low energy neutrinos from the explosion of a neighbouring supernova, should one occur during the lifetime of the experiment. The key challenges are the high data rates that the detectors generate and the deep underground environment, which places constraints on power and space. To overcome these challenges, DUNE plans to use a highly optimised C++ software suite and a server farm of about 110 nodes continuously running about two hundred multi-core processes located close to the detector, 1.5 kilometres underground. Thirty nodes will be at the surface and will run around two hundred processes simultaneously. DUNE is studying the use of the Kubernetes framework to manage containerised workloads and take advantage of its resource definitions and high up-time services to run the DAQ system. Progress in deploying these systems at the CERN neutrino platform on the prototype DUNE experiments is reported.

1 Introduction

The Deep Underground Neutrino Experiment (DUNE) is a next generation long-baseline neutrino experiment based in the USA [1]. It uses a high-intensity, 2 MW neutrino beam produced at Fermilab. Neutrinos are detected in a Near Detector complex, 0.5 km from the neutrino creation point, and at a Far Detector, in the Sanford Underground Research Facility,

[*]Presenter, e-mail: plasorak@imperial.ac.uk

1,300 km away. Figure 1 illustrates DUNE. The main physics goal of DUNE is to precisely measure neutrino oscillations. Doing so will allow DUNE to detect a non-zero leptonic violation of Charge Parity (δ_{CP}), which would have tremendous importance to understand the matter/anti-matter asymmetry of the universe. DUNE will also be able to measure the neutrino mass ordering. Besides neutrino oscillations, DUNE will be able to detect supernova burst neutrinos, and to shed light on various beyond the standard model processes.

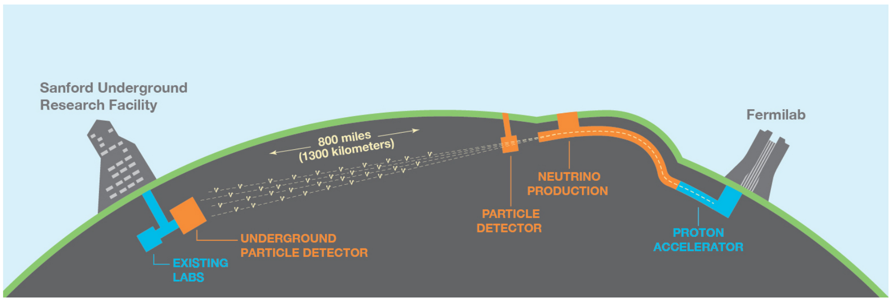


Figure 1. The Deep Underground Neutrino Experiment (DUNE). Neutrinos are produced by impinging an intense proton beam from Fermilab's Proton Accelerator to a target. The neutrinos flux and cross sections are characterised at the Near Detector and will travel 1,300 km to the Sanford Underground Research Facility where the Far Detector will detect the oscillated flux.

The DUNE Far Detector (FD) will be located 1.5 km underground. It will be composed of 4 cryogenic modules of 17 kT of liquid argon each. The location of the cavern and underground environment constrain the access to the experiment and the power and the cooling available to the data acquisition (DAQ).

The DUNE FD will use the Liquid Argon Time Projection Chamber (LArTPC) technology to detect neutrinos. An illustration of the concept is given in Figure 2. With this concept, for the first DUNE FD module, the readout wires continuously sample and record data digitised to 14 bits at a rate of 1.95 MHz. For this module, there will be 2560 channels for each Time Projection Chamber (TPC) unit and 150 TPC units. The total data rate of this module will therefore be 1.2 TB/sec. Similar rates are expected for other FD modules. The DAQ needs to handle varying event sizes, corresponding to the geometrical extent and the recording time of the physical events. The event's sizes will range from $O(100)$ MB to more than 100 TB. Furthermore, the entirety of the data is processed online by high-end CPUs and the data is buffered for a couple of seconds, pending trigger decisions. This buffer enables DUNE to both identify supernova neutrino burst events, and to record their precursors interactions. Finally, DUNE is expected to be active for 99% of the time, as supernova neutrino burst events can happen at any time, but are very rare. Only one supernova (1987A) ever produced observable neutrinos to date.

2 DUNE DAQ

The DUNE DAQ [2] is organised in six entities:

- Readout
- Trigger
- Dataflow
- Timing

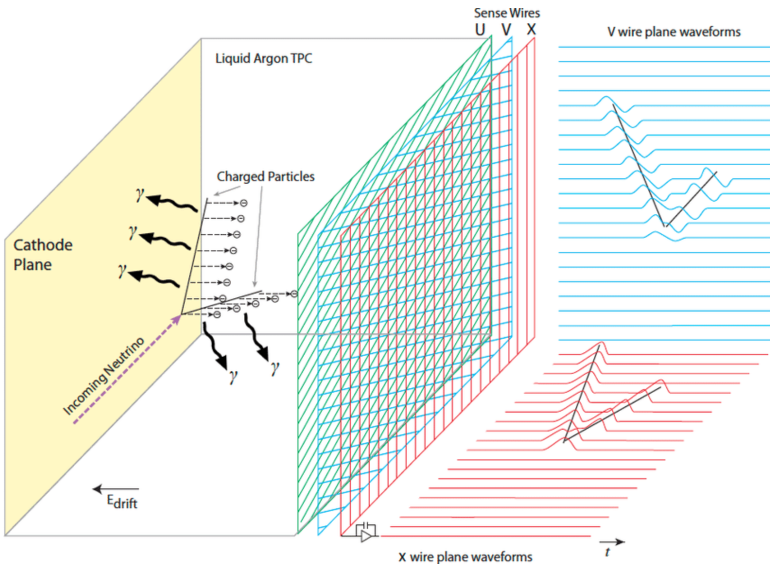


Figure 2. The Liquid Argon Time Projection Chamber (LArTPC) concept. Ionisation electrons are produced in the liquid argon. They are drifted by an electric field to the anode which is composed of sensing wires. Photons are detected by dedicated detectors.

- Data Quality Monitoring (DQM)
- Control, Configuration and Monitoring (CCM)

Each subsystem has different hardware provisioning. The DAQ structure is represented in Figure 3. The detector electronics produce raw data that gets processed in around 80 Readout servers located 1.5 km underground in a cavern near the FD. The Readout applications buffer the data and produce Trigger Primitives (TPs). TPs are small-size data objects containing information about charge depositions in the readout detector, or photon hits. The TPs are passed on to the Trigger applications, which are distributed over 20 servers underground. The Trigger uses the TPs or the beam synchronisation infrastructure from the Timing system and forms Trigger Decisions (TDs). These TDs are forwarded to the Dataflow subsystem (10 servers, on the surface). The Dataflow requests data from the Readout applications which fetch it from their buffers and send it to the Dataflow. It is the Dataflow's responsibility to write the triggered data to disk. An additional process called the data filter is run on the written files to further exclude background events. In parallel, continuous monitoring of the data at different stages (Readout, Trigger and Dataflow) is done by the DQM subsystem. The whole DAQ relies on the CCM subsystem, which provides a set of standardised tools and orchestrate the DAQ. Thirty servers, located underground, are expected to host the DQM and CCM services.

3 Kubernetes for DAQ

Given that the DAQ will employ multiple types of server with varying specifications and run around two hundred processes on them, it is interesting to investigate advanced techniques for scheduling processes and managing the resources available. Kubernetes [3] is a modern, industry-standard container orchestration tool. It is used in data centres, which have high up-

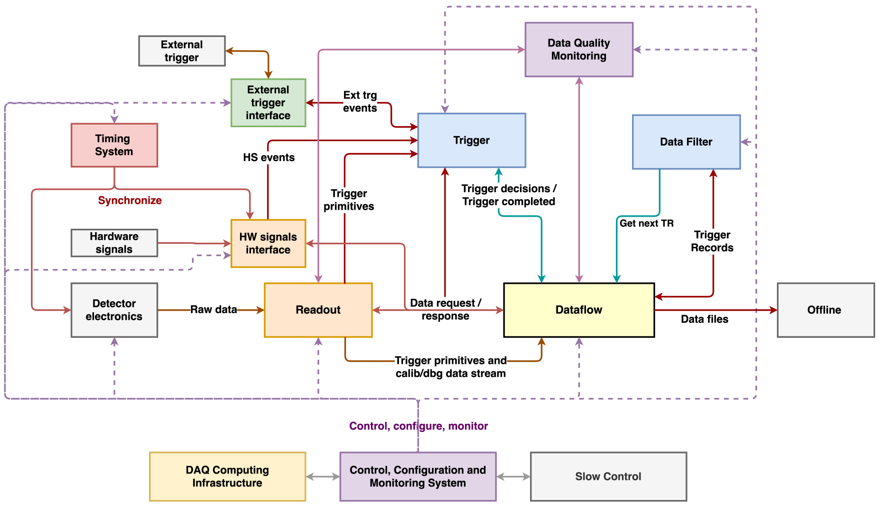


Figure 3. The organisation the data acquisition for the Deep Underground Neutrino Experiment.

time requirements. It handles the scheduling of containers, resource management, networking and automated recovery actions.

To facilitate the discussion, we separate the DAQ components in two categories: the DAQ services and the DAQ applications.

3.1 DAQ services

DAQ services are DAQ processes that need to run while the DAQ is taking data or preparing to take data. These processes are not directly involved in the data readout, triggering, and recording. The services are generally third-party open-source tools (for example, Grafana [4], PostgreSQL [5]), or simple microservices (generally written in python) that the DUNE-DAQ group has developed. These services enable the archiving of the run time conditions, monitoring, alarms and specialised messaging. As an example, the monitoring chain used for the prototype DUNE detector (ProtoDUNE), at CERN, is displayed in Figure 4. Monitoring relies on services for user interfaces and the archiving of log messages and metrics. In the current scheme, DAQ applications send metrics and logging information to a Kafka broker. Two microservices are connected to the broker and consume the metrics and logs and store them in dedicated long-term databases. Both the microservices also provide data to the user interface (Grafana). Grafana also directly consumes data related to server monitoring via Prometheus.

Deploying services with Kubernetes has already proven to be efficient, as it enable monitoring of the services through the Kubernetes dashboard (see Figure 5) and a single point-of-entry to manage the services via the `kubectl` command. It also enables packaging of the services such that they can be used at different test-stands (at Fermilab, CERN and different universities) with a minimal set up using the Kind tool [6].

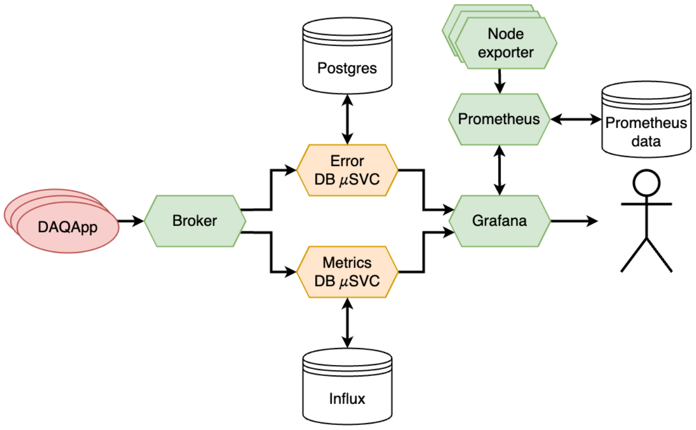
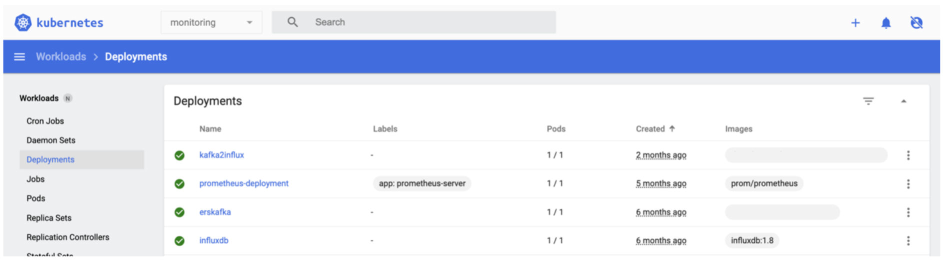


Figure 4. DUNE DAQ monitoring, used for ProtoDUNE data taking. Red circles are the source of the data for monitoring; green boxes are third-party tools; orange boxes are DAQ-specific are microservices; white cylinders represent databases. Most of the databases and services are handled by Kubernetes.



Deployments

	Name	Labels	Pods	Created ↑	Images
✓	kafka2influx	-	1 / 1	2 months ago	
✓	prometheus-deployment	app: prometheus-server	1 / 1	5 months ago	prom/prometheus
✓	erskafka	-	1 / 1	6 months ago	
✓	influxdb	-	1 / 1	6 months ago	influxdb:1.8

Figure 5. An excerpt of the Kubernetes dashboard, enabling monitoring of the DAQ services.

3.2 DAQ applications

Cluster architecture

DAQ applications provide the core functionality of the DAQ. They are written in a specialised C++ framework [7] and use a highly efficient and optimised modular code base. DAQ applications are spawned at the start of each DAQ session. A DAQ session represents a coherent data taking system composed of multiple DAQ applications interacting with each other to record detector data. Depending on the environment, DAQ sessions can be long or short lived. To run DAQ applications in Kubernetes, we decided to create a different cluster from the services cluster described above, so that potentially CPU-intensive tasks from the DAQ applications would run on different servers and not interact with the DAQ services. A depiction of the current, prototype DAQ application integration inside the Kubernetes cluster is shown in Figure 6. Process management is handled via the run control application, which directly interacts with the control plane of the cluster via Kubernetes' python bindings. Each application runs in a Kubernetes *Deployment* and the DAQ application binary is executed inside the container. To avoid name clashes, the Kubernetes *Namespace* is the DAQ session name. Thus, one can run many DAQ sessions at the same time in the same Kubernetes cluster. Raw data storage is handled outside of the Kubernetes cluster, simply by mounting the disk on which the data is written in the container. The Kubernetes Domain Name System

(DNS) is used by the applications to resolve their own IP addresses and those of the other DAQ applications that they communicate with.

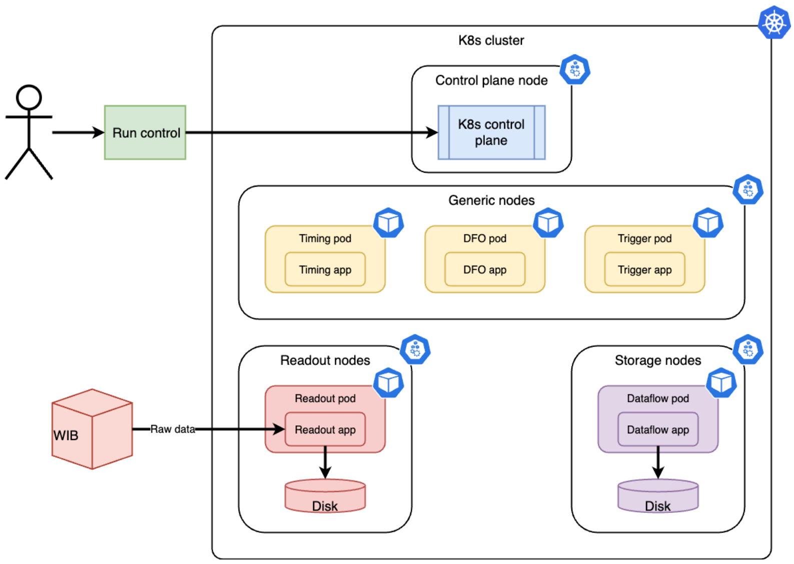


Figure 6. Run control and DAQ applications interactions with the Kubernetes cluster

Readout application

The Readout applications require special attention since they require at the same time:

- Fast network I/O to handle the throughput from the detector;
- High-end CPU to process the data and create TPs in quasi-real time;
- Fast and large disk access to enable the recording of the raw data in case of a supernova neutrino burst trigger.

To overcome these challenges, the DAQ group decided to use high-end Commercial Off The Shelf (COTS) Network Interface Cards (NICs) to interface the detector electronics. Powerful multi-socket servers will be used to process the data online, along with Solid State Disks (SSDs) for the raw data recording. Each of these hardware components require careful tuning to achieve the required performance. For example, the NIC places the raw detector data on a specific Non Uniform Memory Access (NUMA) RAM region which needs to be accessed by the corresponding CPU for processing. Part of our investigation is whether Kubernetes can provide the tools and configuration required to realise this.

Furthermore, unlike other parts of the DAQ system, the Readout applications are not relocatable. One Readout server will be connected to exactly two TPC units in the first and second FD modules. Hence, this is an area where the Kubernetes resource management overlaps with the physical detector. To explicitly match detector topology to Kubernetes resources, we created a hardware discovery tool based on Kubernetes *Device Plugin* framework [8]. This tool is a *DaemonSet* that runs on all the servers in the Kubernetes cluster. The tool polls the node interfaces when started, and creates a Kubernetes resource representing the detector if the node is connected to it. This enables, when scheduling a Readout application to run on the cluster, to request a Readout resource and be sure that it is connected to the correct part of the detector.

CVMFS, image building and distribution

One of the common problems when using a containerised environment relates to software development. Some of the issues we have encountered are related to the size of the container image. The DAQ libraries and dependencies are large: in our tests, a complete standalone DAQ image was roughly 5 GB. Such image sizes constrain the container registry that we can use (noting that we will not have access to commercial infrastructure) and the possibility of building an image every time the software is built. One solution is to use a CVMFS or a Network File System (NFS) mount in the DAQ application container to share the software. We still need to investigate whether this solution is acceptable for the final FD DAQ.

4 Conclusion

Initial tests integrating some components of the DUNE-DAQ within Kubernetes were successful. The DUNE-DAQ group plans to adopt Kubernetes and use a containerised environment for its services (monitoring, run archiving, etc.). More investigation is needed to decide whether DAQ applications will run in a dedicated Kubernetes cluster. Kubernetes solves many problems related to resource and process management, and networking becomes much simpler at the application level due to the Kubernetes DNS. There are, however, still many challenges to be resolved before a decision is made to adopt it: we have not fully quantified overheads related to the containerisation of the applications, we need to streamline the development procedure with containers and finally, some details and optimisation of the Readout process need to be investigated further.

References

[1] B. Abi et al., Journal of Instrumentation **15**, T08008 (2020)
[2] A. Abed Abud et al., Tech. Rep. EDMS 2812882 (2023)
[3] *Kubernetes*, `https://kubernetes.io/`
[4] Grafana Labs, *Grafana*, `https://grafana.com/`
[5] The PostgreSQL Global Development Group, *Postgresql*, `https://www.postgresql.org/`
[6] *Kind*, `https://kind.sigs.k8s.io/`
[7] DUNE DAQ group, *Application framework*, `https://github.com/DUNE-DAQ/appfwk`
[8] *Kubernetes device plugins*, `https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/device-plugins/`