



Data Acquisition and Online Monitoring for Hyper-Kamiokande

Nick Latham 4th JENNIFER2 General Meeting (Pisa)



03/04/2025



Introduction

- Hyper-Kamiokande is a 260 kt water Cherenkov detector under construction in Gifu, Japan.
- ~8x fiducial mass of Super-Kamiokande.
- Physics programme: high precision neutrino oscillation measurements, supernovas, proton decay, relic neutrinos.
- •
- The **Data Acquisition (DAQ)** for the Hyper-K far detector is used to store events; it has the following requirements: Measure hits from ~50,000 photosensors of different types ulletHandle $\mathcal{O}(10^8)$ hits in ~10 s for supernovae events 71 m • High fault tolerance \bullet Real-time trigger processing and reconstruction \bullet 34 m Modularity and scalability Error-correctible and adjustable mid-run Last for > 20 years ullet





Hyper-K UK DAQ



FEE: front-end electronics

RBU: readout buffer unit



In-Water Electronics







Inner detector (ID) PMT

Outer detector (OD) PMT

2x CLK 2x counter 2x data and slow control



Multi (mPMT)





DAQ Deployment and Data Flow

- The DAQ system will be installed and operate in five electronics huts above the target volume.
- 4x huts (A, B, C, D) connect PMTs to housing RBUs.
- 1x central hut with TPUs, EBUs and brokers.
- Interconnecting cables between huts.
- 10 racks (2x per hut).
- Data flow and triggering also organised:
 - All hits received by RBUs lacksquare
 - Triggering decisions based on ~100 x 1 ms slices for standard data flow
 - Longer slices and external trigger sources studied for SN TPUs
 - Triggered data enters EBUs and is saved to disk



5

DAQ Tasks & Organisation

- Hyper-K DAQ largely under development by different UK-based universities.
- Main DAQ tasks arranged into several subgroups within the DAQ WG:
 - DAQ system design lacksquare
 - Electronics tests
 - Installation and commissioning lacksquare
 - Triggering
 - Online monitoring and slow control











UNIVERSITY OF



Technology **Facilities Council**



ToolDAQ & ToolFramework

- ToolDAQFramework is an open source DAQ framework developed by UK collaborators.
- Currently being used by ~10 particle physics experiments and will be used by Hyper-K.
- Attempts to address highly rigid, convoluted and fragmented software.
- Incorporates DAQ features while:
 - Being fast with easy modular development
 - Having dynamic service discovery and scalability
- TooIDAQ contains toolchains which hold dynamic tool objects (C++/Python) created from factories; these communicate through transient data storage classes.
- This allows for flexible complex structures and paradigms, multithreading, sub toolchains, worker farms.
- A full online front-end and command line comprises the centralised system.
- More information: <u>https://doi.org/10.1051/epjconf/201921401022</u>



- The online monitoring (OM) system will include: ullet
 - A unified web front-end \bullet
 - Databases \bullet
 - Configuration lacksquare
 - Shift checks lacksquare
 - Alarms
 - Slow control
 - Triggers
- Similar interface used at ANNIE. \bullet
- We am to learn from our experiences with Super-K and • ND280.
- Tested at recent WCTE runs. lacksquare







• Hyper-K bespoke login and arbitrary landing page:





Event display under development; example μ -like event generated using WCSim: •

	PAGEA	EVENTDISPLAY	MONITORING
3D 🗸 Ch Io new eve	arge 🗸 <	< Pause > >> 100	00 2025-02-20 19:39:23.641654+00 🗸 Last upo
white			
			2000
Fixed size			z 0
			2000 2000



1()

• Integrated SQL database:

● ● ● ① ▼ < ① ① - ⊕ 127.0.0.1/DAQ/SQL/ ···													
					2К ~ 📋 НК	,							
Exp tin	ne: 18:23:43		Us	er time: 18:	:23:43		UserName:	dev_user		Log Out			
≡ DAQ → S	SQL		Home	Shift	Alarms	DAQ	SubSystemExample	Config	Data	Users			
RUNCONT	RUNCONTROL LOGS SQL CONTROL												
SQL Command i type command here Submit Command output Select SQL table to display Inonitoring in 10													
time	device	data											
2025-03-31 middleman_1 {"demotions": 0, "reps_sent": 0, "promotions": 0, "dropped_acks": 0, "dropped_reads": 0, "log_msgs_sent": 0, "cached_queries": 0, "dropped_writes": 0, "log_send_fails": 0, "rep_send_fails": 0, "read_query_rate": 0, "replies_waiting": 0, "self_promotions": 0, "self_promotions_failed": 0, "mm_broadcasts_recvd": 0, "read_queries_waiting": 0, "write_queries_maiting": 0, "write_queries_waiting": 0, "write_queries_maiting": 0, "master_clashes_failed": 0, "mm_broadcasts_recvd": 0, "read_queries_waiting": 0, "write_queries_failed": 0, "master_clashes_failed": 0, "multicast_failed": 0, "multicast_failed": 0, "read_query_recv_fails": 0, "write_queries_waiting": 0, "dropped_monitoring_out": 0, "self_promotions_failed": 0, "self_promotions_failed": 0, "write_query_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "multicast_msg_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "mm_broadcasts_recv_fails": 0, "write_queries_waiting": 0, "dropped_monitoring_out": 0, "self_promotions_failed": 0, "standby_clashes_failed": 0, "write_query_recv_fails": 0, "mm_broadcast_recv_fails": 0, "multicast_msg_recv_fails": 0, "multicast_msg_recv_fails": 0, "multicast_msg_recv_fails": 0, "multicast_recv_fails": 0,													
2025-03-31 17:22:14.780868+00	middleman_1	{"demotions": "dropped_writ "self_promotic	0, "reps_sent": 0, "p es": 0, "log_send_fa ons": 0, "standby_cla	promotions": 0 uils": 0, "maste ashes": 0, "der), "dropped_ac er_clashes": 0, motions_failed	ks": 0, "dropp "rep_send_f ": 0, "droppe	oed_reads": 0, "log_msgs ails": 0, "read_query_rate d_logs_out": 0, "write_qu	_sent": 0, "cache e": 15123000000 ery_rate": 15123	ed_queries": (00, "replies_v 00000000,	0, waiting": 0,			

• Other examples:

● ① ~ 〈	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●
Exp time: 07:41:19 User time: 08:41:19 UserName: dev_user Log_Out	Exp time: 07:40:54 User time: 08:40:54 UserName: dev_user
Alarms Home Shift Alarms DAQ SubSystemExample Config Data Users	☰ DAQ > RunControl Home Shift Alarms DAQ SubSystemExample Config Data Users
Alarma	RUNCONTROL LOGS SQL CONTROL
Alaliiis	DAQ Service Name: WCTE_RBU Slow Control Service IP: <searching> Port: <searching></searching></searching>
nly Alarms for Device: Device Name	Status: <searching></searching>
ne device level alarm silenced	Run Control
ws)	Stop Current Run
	Start New Sub Run
NS: 50 🗘	Start New Run
bre Entries	response:
All Alarms for Device: Device Name Silence	
test alarm Back to Top A	test alarm Back to Top no current alarm Alarms page
● ● ● ① × <	● ● ● □ × <
ііі П Т2К ~ П НК ~	Image: 07:40:21 User time: 08:40:21 User Name: dev user
E SubSystemExample PlottingFunctions Home Shift Alarms DAQ SubSystemExample Config Data Users	
INTRO DATABASEQUERIES PLOTTINGFUNCTIONS REQUEST SERVICES PAGEA PAGEB	CSEIS Home Shirt Alamis DAQ SubsystemExample Comig Data Osers
getMonitoringPlot(device, options)	USERS
Description: Retrieves monitoring data from the system and returns an array of Plotly plot objects.	Add New User
Parameters: device (string): The name of the device to retrieve monitoring data for.	
Example:	
	Password
<pre>import { getMonitoringPlot } from '/includes/tooldaq.js'; const device = "test_device";</pre>	
<pre>const options = { from: '2025-02-18 12:59:13.588877+00',</pre>	Confirm Password
<pre>to: new Date(), template: { mode: 'lines+markers' } }; aetMonitoringPlot(device_options) then(plots => console_log(plots));</pre>	Permissions
gettorited the tockette, options). then prots -> console. tog(prots)),	{"permission": "test"}
Output: Logs the retrieved plot data to the console.	
You can also add as script tag directly to your HTML file: and utilise the function as shown below:	

12

Examples at WCTE: •







test alarm no current alarm Alarms page

Preliminary online monitoring schedule: •

		2025										2026												2027											
		Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun J	ul	Aug	Sep	Oct	Nov
Task	Status																																		
Discussion at HKCM	Completed																																		
WebServer ready	In progress																																		
C&P presentation	Not started																																		
Get list of contacts	Not started																																		
Get data samples	Not started																																		
Get requirements	Not started																																		
Develop subsystem pages	Not started																																		
Feedback subsytem pages	Not started																																		
Page finalisation	Not started																																		
Integrate real devices	Not started																																		
Hyper-K start-up																																			

- Integrating DAQ and OM requirements with other subsystems. \bullet
- different software.

Aim to have a unified service as part of the online interface, reducing need for expertise on multiple

Summary

- A scalable data acquisition system is under development for Hyper-K.
- An online interface will be a key feature of the DAQ; it will connect with other subsystems and provide real-time monitoring of detector parameters.
- Testing of the online interface and DAQ systems with WCTE and ANNIE.
- Integration with other Hyper-K subsystems and further tests set to occur in the next financial year.
- Hyper-K start-up ~2028, expecting lots of work in next 2.5 years.

