Online Event Selection and GPU-based Waveform Compression at the High Level Trigger of the KOTO Experiment

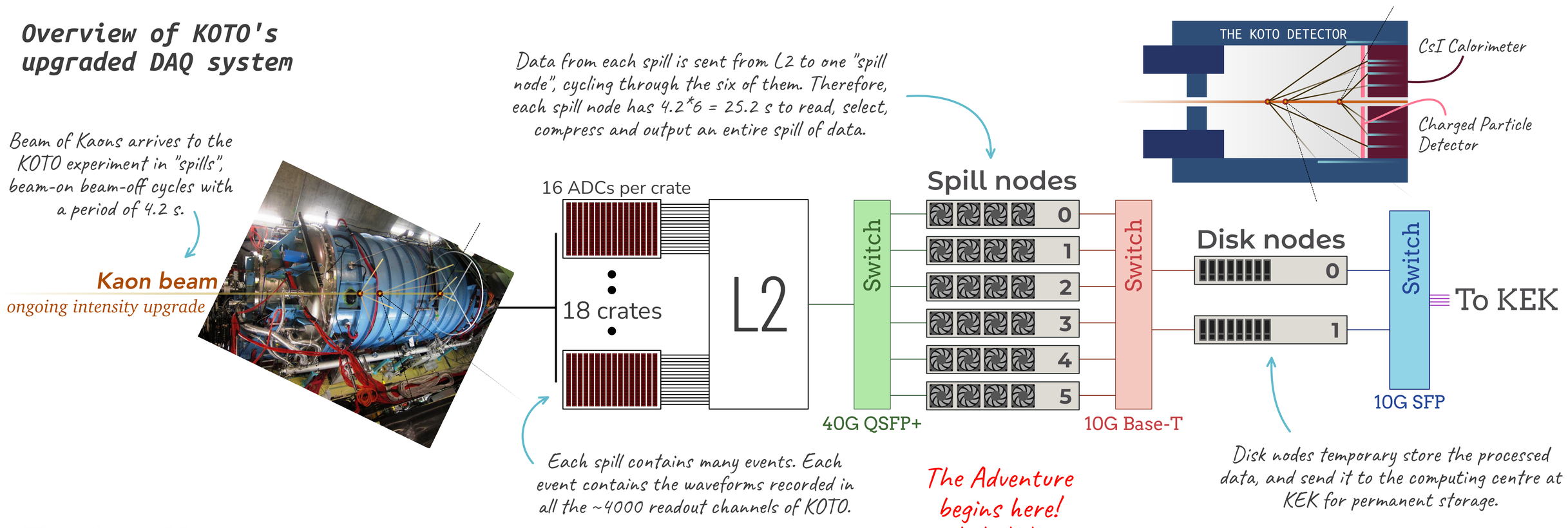
Mario Gonzalez, Osaka University, 14/09/2022

Introduction

The DAQ system of the KOTO experiment is currently undergoing a major upgrade, following an also ongoing accelerator's beam intensity upgrade. Currently, the bottleneck in KOTO's DAQ system is the hardware connection from the L2 trigger to the L3 nodes. For that reason, the data rate has to be trimmed in the upstream DAQ by applying prescalers. These prescalers do not affect KOTO's main target mode, the $K_L^0 \longrightarrow \pi^0 \nu \bar{\nu}$, but greatly limit the statistics available for other physics searches.

The new DAQ system removes this bottleneck, allowing a much larger amount of data to reach the Level 3 nodes. Furthermore, the L3 trigger has also been completely renovated, and GPUs have been added to each node to greatly increase their computing capabilities. This poster describes the design of KOTO's new L3 software, and how it aims to make the most efficient use of all its resources to expand KOTO's physics possibilities.

Overview of KOTO's upgraded DAQ system



The Spill Nodes

The Adventure begins here!

40G QSFP+ | 10G ETH

Event reconstruction

ADC Channel to Detector Channel, for detectors relevant for online selection

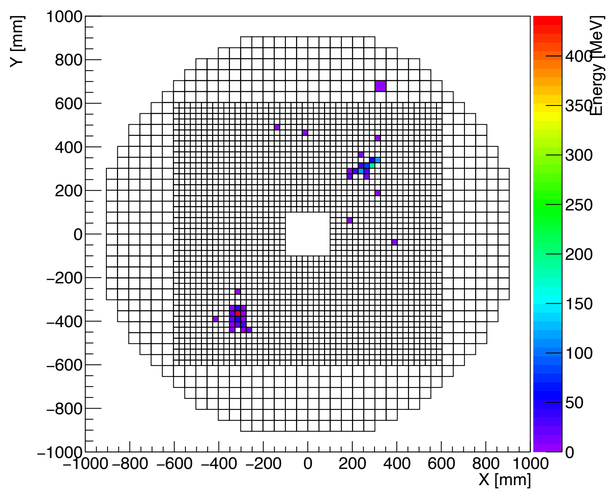
Detector CH

Fast calibration

Energy is calculated from waveform peak height, using offline calibration data (ADC Counts to MeV)

Done in parallel, using as many threads as available

CsI Calorimeter Channel Mapping. CH ID to physical coordinates.



Clustering

Event Selection

- Number of clusters on the CsI, energy
- Charge deposited in veto detectors
- Reconstructed Kl decay position,
- etc.

Event buffer

Evt No.

ADC Channel

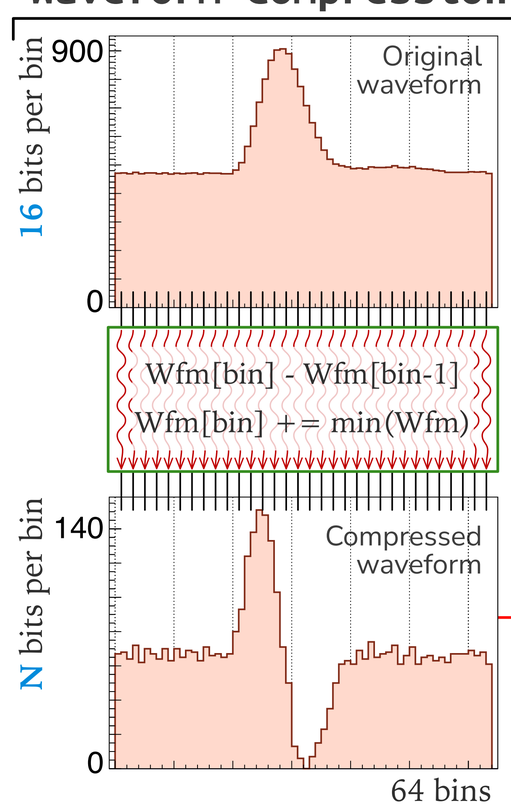
Pass the selection?

Jumbo packets. 9000 Bytes per packet, ~65 packets per event.

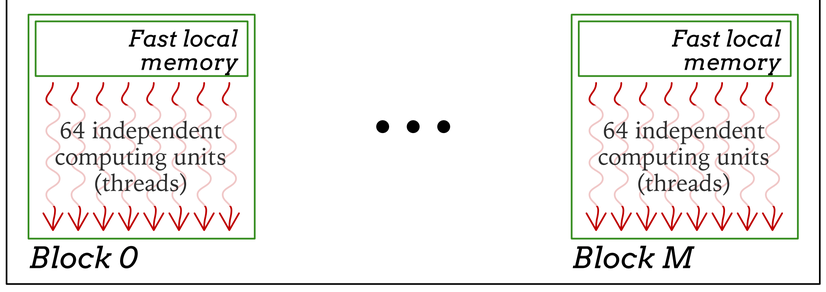
Simple checks (checksum, headers, etc.) are performed to the packets as they arrive

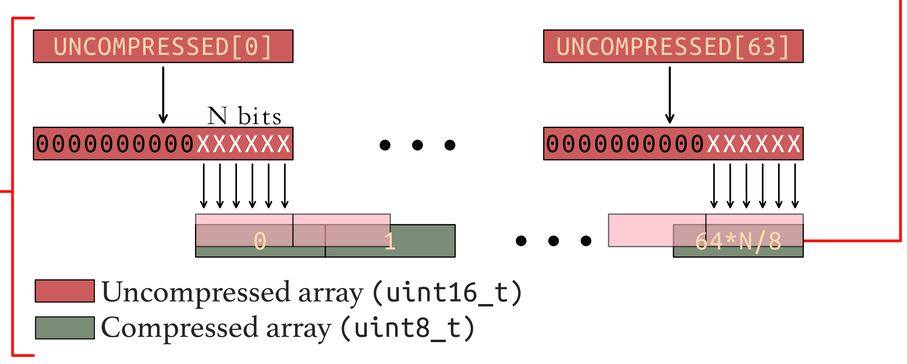
The compressed array is sent to the disk nodes using MPI (a C++ library for CPU-CPU communication). Once on the disk nodes, the array is converted to binary and multiple events are joined together as they are written to disk. An independent program sends them to KEK's computing center as soon as the bandwidth allows.

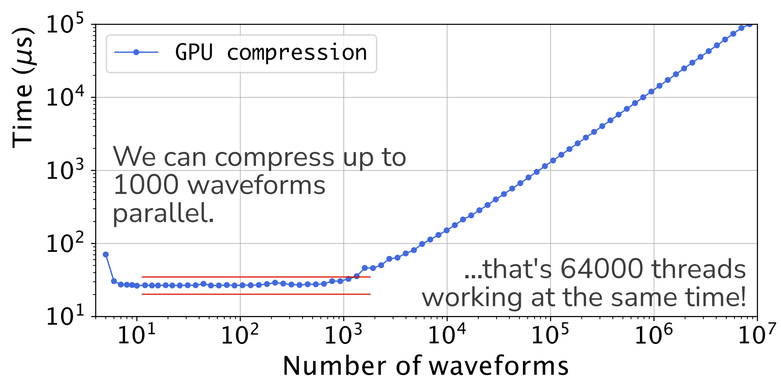
Waveform compression



GPU Threads are organized in blocks . Each block compresses one waveform, and each thread operates one bin. Threads in the same block share fast access local memory.







The GPU operates asynchronously with respect to the CPU. The CPU can then just focus on event selection.

Our CPUs are individually faster, but can only compress ~35 waveforms in parallel, depending on available number of threads.

This GPU algorithm is written in CUDA, a C++ library used to write code for Nvidia GPUs

The decay $K_L^0 \longrightarrow \pi^0 e^+ e^-$

Can happen through a direct CP violating process, involving a *s* to *d* transition similar to the one in KOTO's main target decay.

Four clusters on the CsI Calorimeter

- *Two of them leaving also signal in the charged particle detector.*
- *The other two (photons) with an invariant mass close to the one of the neutral pion*
- *All of them computing an invariant mass close to the Kaon mass*

Summary

The layout of KOTO's future software trigger has been designed to make use of all the hardware resources by implementing parallelization at three levels.

- The MPI (Message Passing Interface) library is used to parallelize over events in the event buffer across CPU cores (20 in each spill node).
- C++ Threads are used to dynamically parallelize small tasks across CPU threads (two per CPU core).
- CUDA is used to perform waveform compression in parallel, allowing the CPUs to focus on event selection.

Currently, a spill consisting in 30k events (which is the upper limit estimation for the next run), can be read and compressed in less than half of the available 25.2 seconds per spill, leaving a large room for complex event selection. In 2023, KOTO will start taking data while performing online event selection, and using GPUs to cope with part of the event processing.