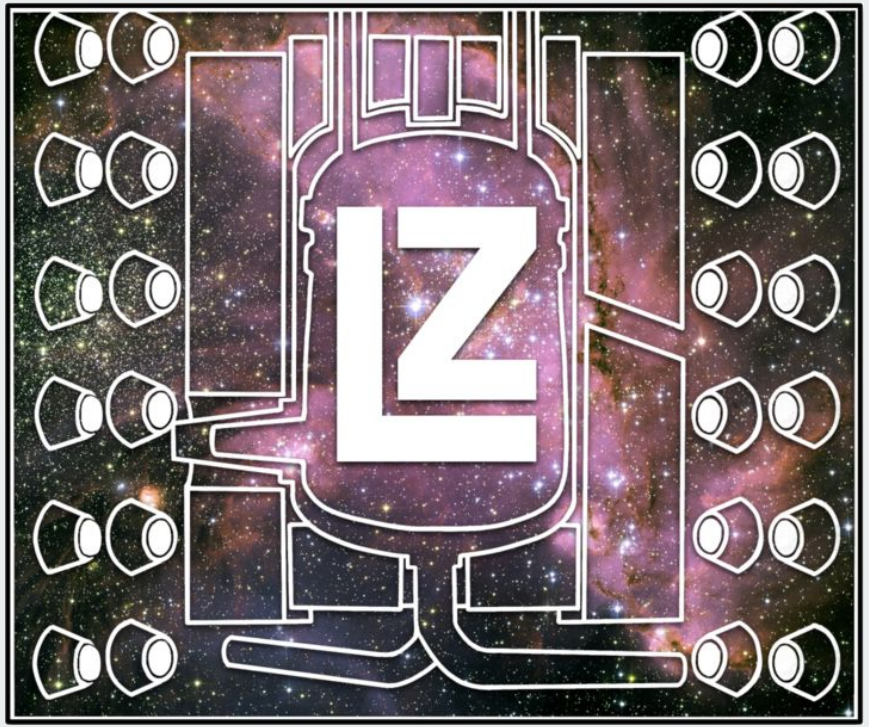
LZ Data Analysis: Software and Frameworks



Dr. Sally Shaw, UC Santa Barbara

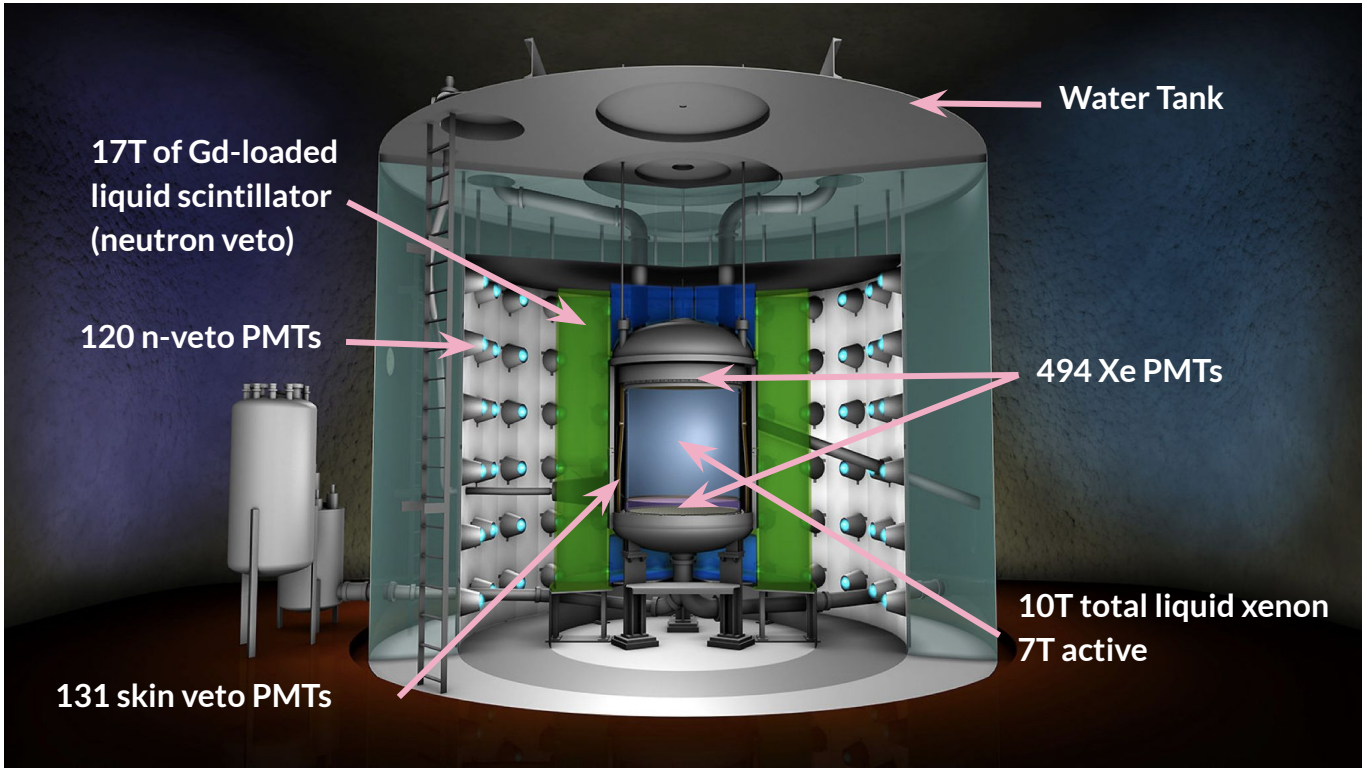
Data Analysis Convener for LZ

Dr. Will Turner, University of Liverpool

Deputy Data Analysis Convener for LZ

Introduction to LZ

LUX-ZEPLIN : a multi-tonne target dark matter detector using dual-phase xenon TPC technology

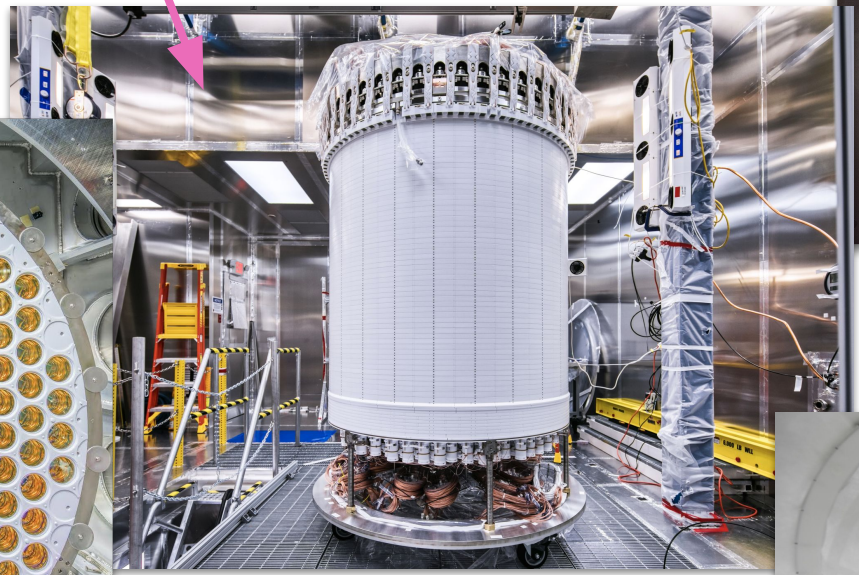


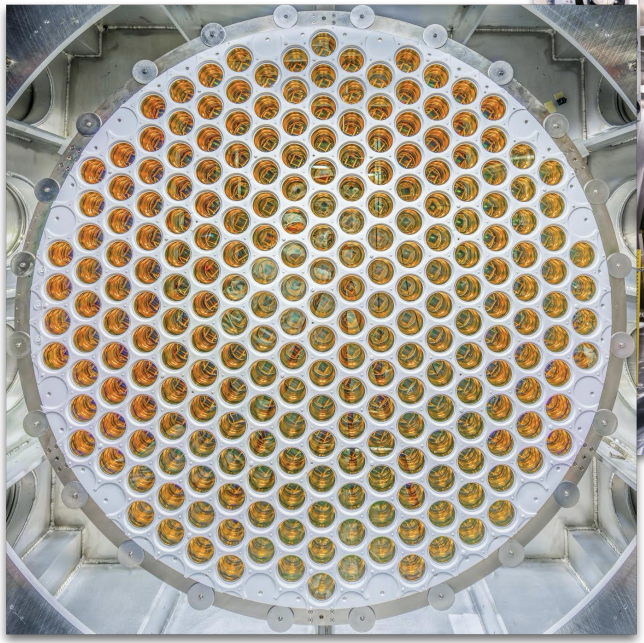
Introduction to LZ

TPC

Skin PMT Installation

Bottom PMT array



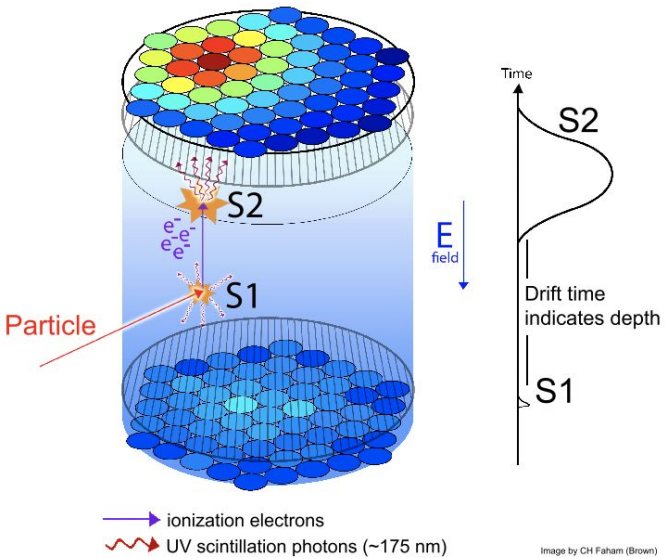


Outer Detector PMTs

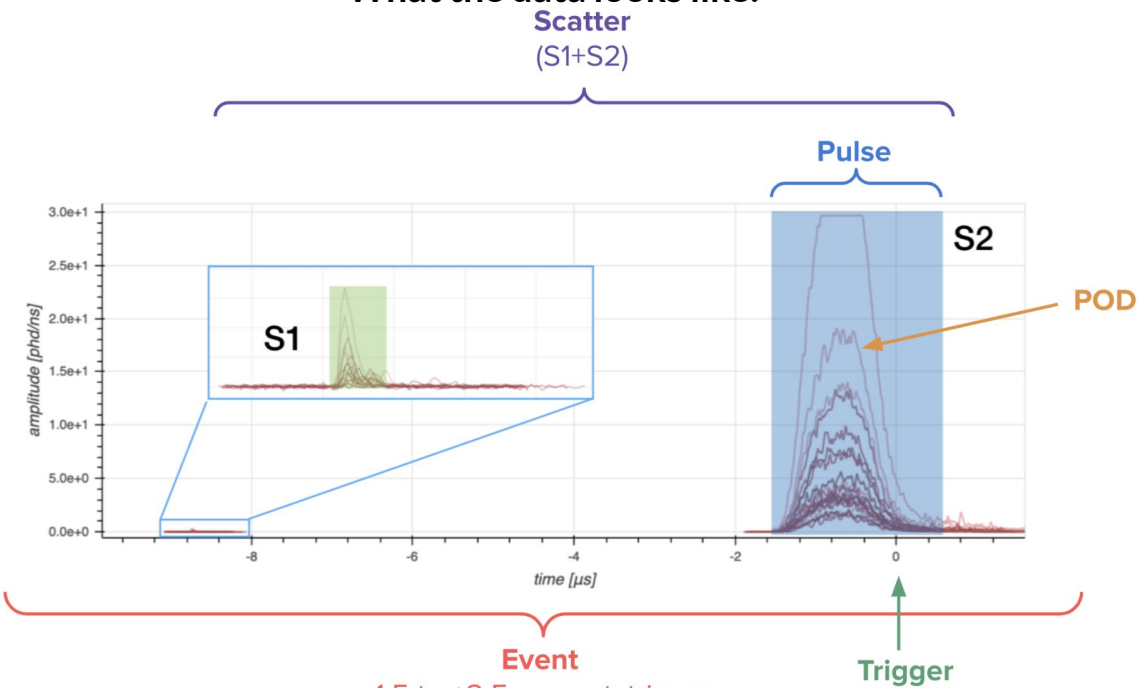
Located at the Sanford Underground Research Facility in Lead, South Dakota

LZ Data Nomenclature

What's happening:

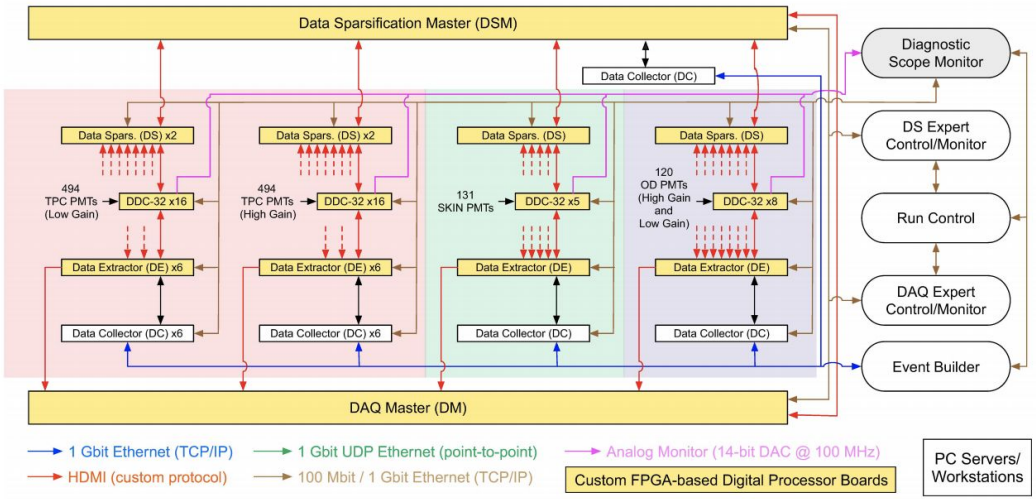


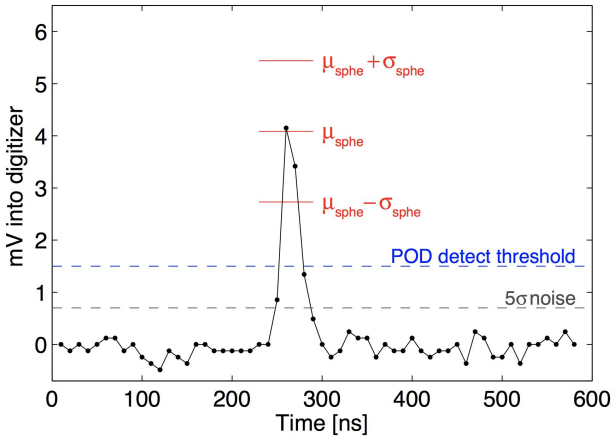
What the data looks like:



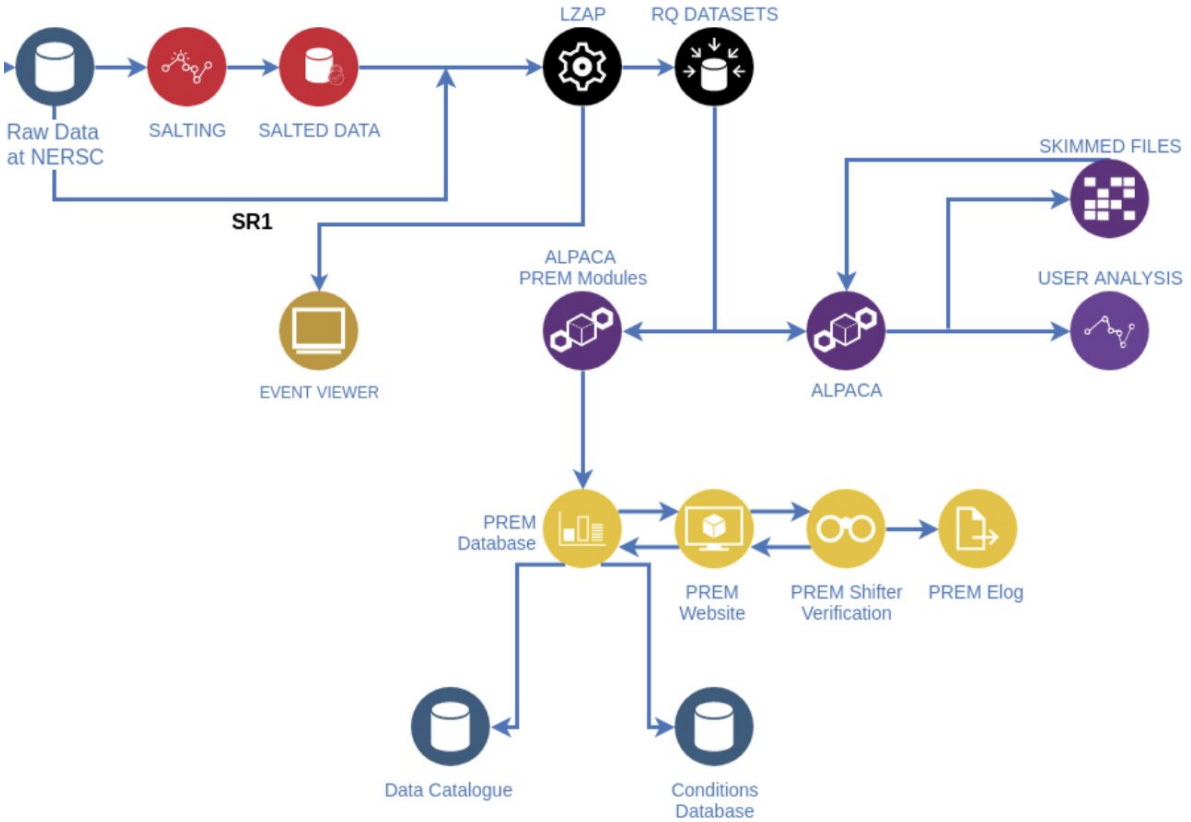
LZ Data Acquisition

- Data is collected from 494 TPC PMTs (high gain + low chain channels), 131 skin PMTs (single gain), 120 OD PMTs (high gain + low gain). Trigger modes include random, S1 and S2.
- Baseline suppression (POD-ing) around pulses happens here at the DAQ stage.
- Event building segments waveforms into 4ms windows, 1.5ms before the trigger pulse and 2.5ms after the trigger pulse. This ensures the full drift length is collected after the trigger.





Data Flow Schematic

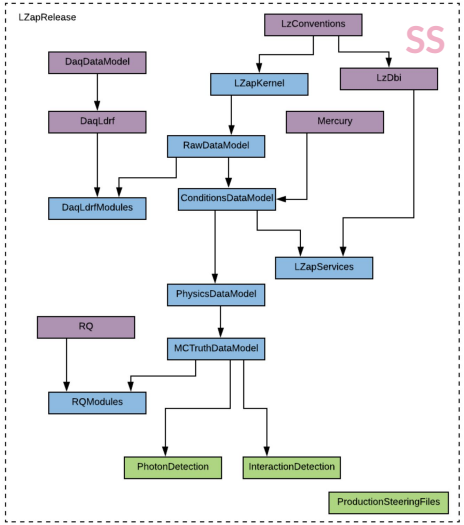


LZap:
The LZ Analysis Package*

*actually a data processing package

LZap - the LZ Analysis Package

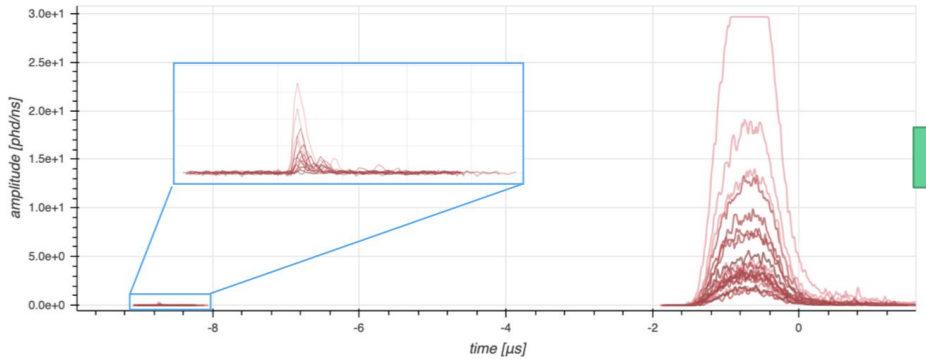
Actually, not for analysis… LZap is an event reconstruction package built in the Gaudi framework.



LZap converts raw waveform data into analysis-ready reduced quantities (RQs). It is ran automatically on raw data coming in from LZ at both the US data centre (USDC) and the UK data centre (UKDC).

Split into two main parts that actually produce Reduced Quantities (RQs):
PhotonDetection (pulse level) and **InteractionDetection** (event level)

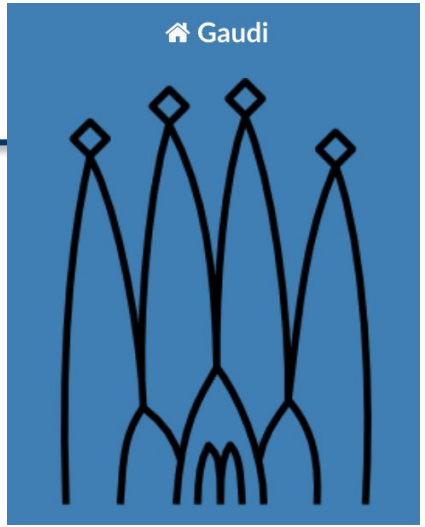
LZap performs baseline calibration, sphe calibration, channel timing offsets, pulse finding, pulse classification, pulse parameterisation, photon counting, event classification, position reconstruction, position corrections (XY and e-lifetime/Z) , veto event classification, etc...

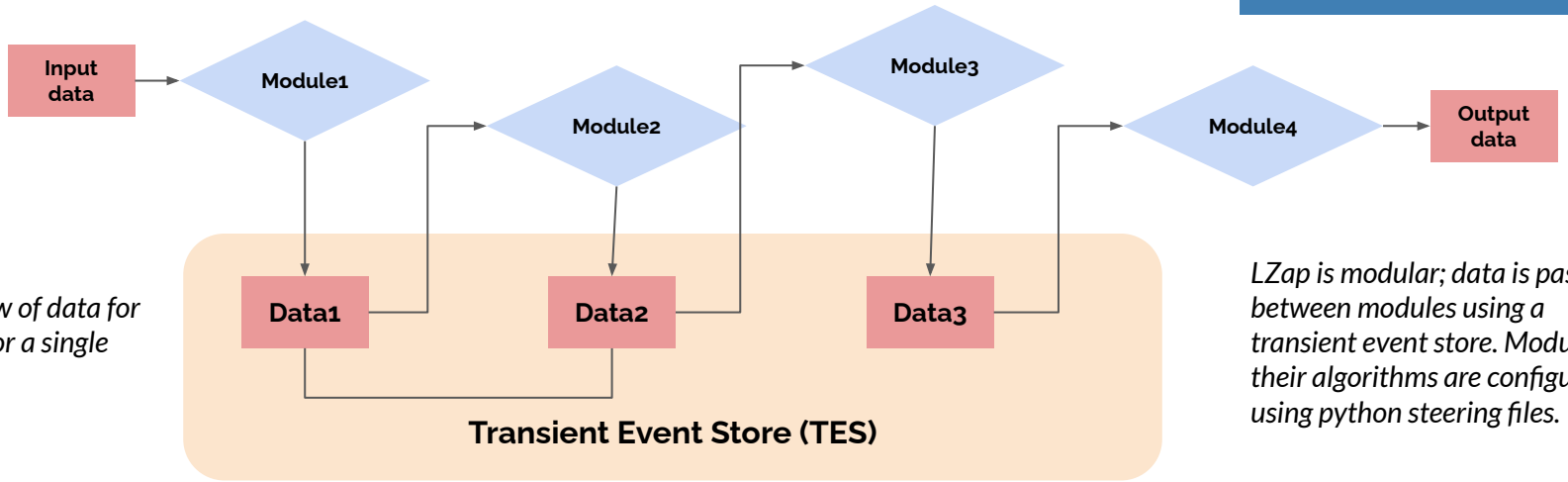



```
ss.nSingleScatters                = 1
ss.s1PulseID                      = 5
ss.s1IsHG                         = 0
ss.s2PulseID                      = 12
ss.s2IsHG                         = 0
ss.driftTime_ns                   = 290540.000000
ss.s1Area_phd                     = 627.317078
ss.s2Area_phd                     = 511225.500000
ss.s1TopArea_phd                  = 219.978912
ss.s1BottomArea_phd               = 406.813538
ss.s1TopCorrectionFactor          = 1.086862
ss.s1BottomCorrectionFactor       = 1.086862
ss.s1CorrectionFactor             = 1.086862
ss.correctedS1TopArea_phd         = 239.086746
ss.correctedS1BottomArea_phd      = 442.150208
ss.correctedS1Area_phd            = 681.236938
...
```

LZap

- Built within the **Gaudi framework** used by ATLAS & LHCb
  - Gaudi: a framework software package that is used to build data processing applications for High-Energy Physics experiments.
- Package can be cloned from GitLab and built from source, or official releases are distributed on **cvmfs**
- *Input*: raw data (or raw simulated data) in the form of ROOT TTrees
- *Output*: reduced quantity (RQ) files in the form of ROOT TTrees





Simplistic flow of data for a Gaudi job for a single event

LZap is modular; data is passed between modules using a transient event store. Modules & their algorithms are configured using python steering files.

RQ Format

eventHeader runID, eventID, raw filename, trigger info, sumPod info, run info, number of pods, buffer info

Pulse level → "Events" TTree

pulsesTPC
(pulsesTPCHG & pulsesTPCLG)

Pulse classification
Position info
(XY, errors, centroid XY)
HG LG matching info
FWHM & area in fixed windows
Top-bottom asymmetry

pulsesODHG & pulsesODLG

Reconstructed position
(theta, Z, ladder)

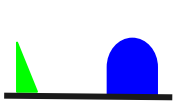
pulsesSkin

Reconstructed position
(theta, Z)
Top-bottom asymmetry
(all, dome, barrel)
Dome & barrel areas

Common to all:
pulse areas (total, channels, positive, negative), pulse height,
pulse timing & shape (start,end, area fractions, prompt fractions),
saturation flag, coincidence, RMS width

Interaction level → "Scatters" TTree

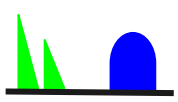
single



Corrected S1 & S2 areas
Reconstructed & corrected position (X,Y,drift)
Energy (ER & NR)
Veto info (OD prompt & delayed area, skin total barrel & dome areas)

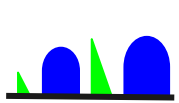
multiple

kr83m



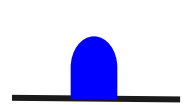
Corrected S1a,S1b, S2 areas
Reconstructed & corrected position (X,Y,drift)
Energy

pileup



Number of S1s, number of S2s & their IDs

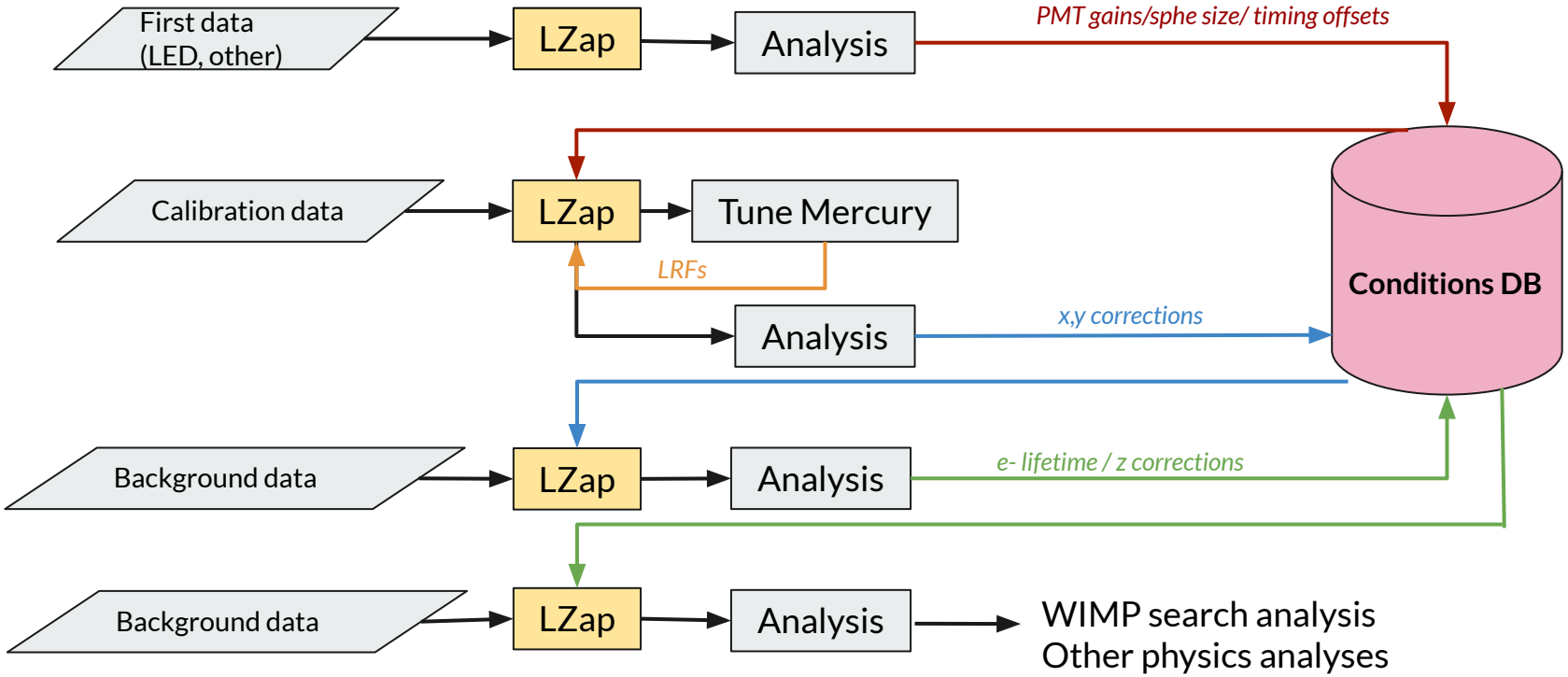
other



Counts of each pulse type & their IDs

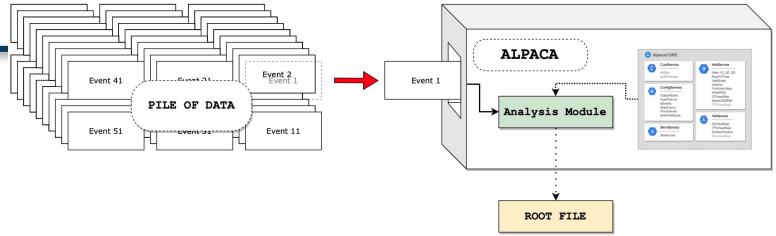
Data Reprocessing with LZap

A ***simplified*** outline of how to get data that is suitable for a physics analysis



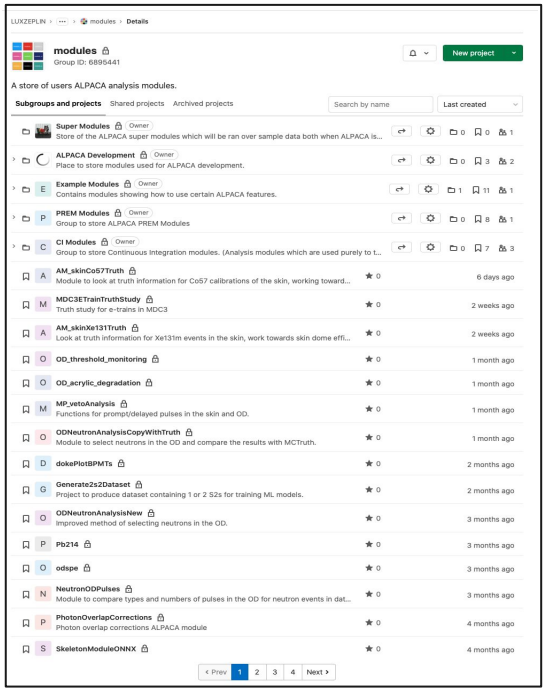
ALPACA:
Analysis Lz PACkAge

ALPACA



Analysis Frameworks aim to standardise analysis workflow and handle underlying/common features allowing users to **just write analysis**,

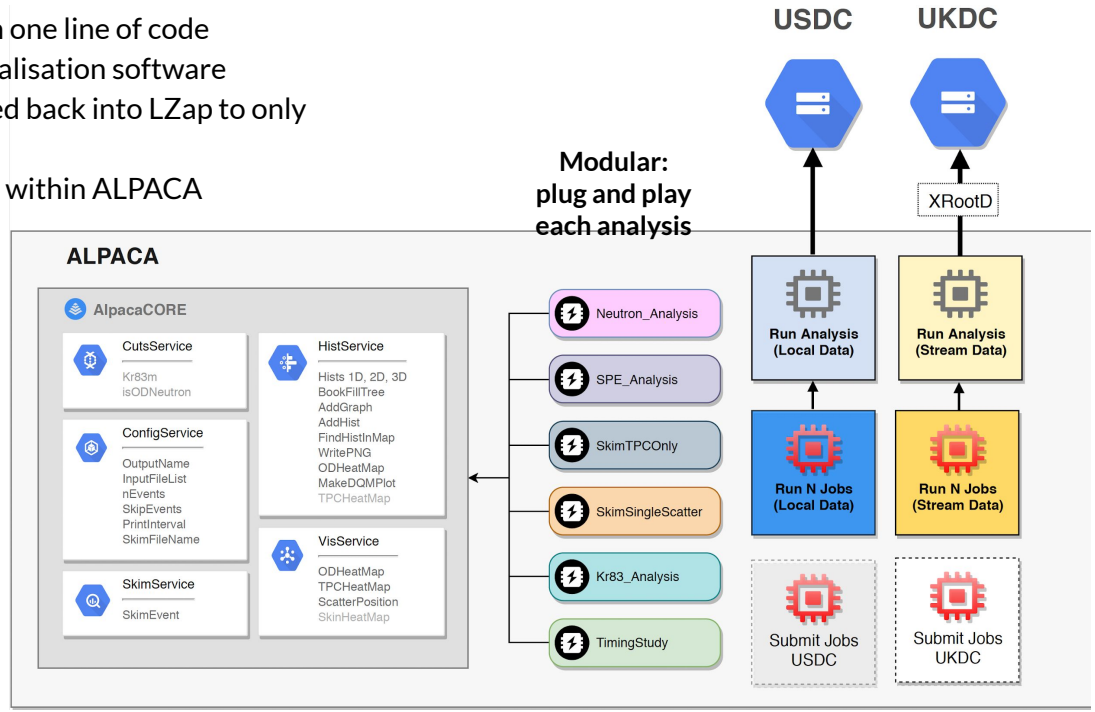
- Easy to write analysis, don't have to write your own `framework' code.
- Easy to share, you can run anyones analysis just as easily as you can run your own.
- Introduces shared knowledge with a low barrier to entry, anyone can help you.
- No duplicated code or effort, feature/cuts gets written once and introduced into shared codebase.
- Any bugs in shared codebase are found and fixed promptly.
- Minimal differences between different analyses, can contribute to a different analysis without having to learn a new tool, new workflow, new codebase.
- Keep analysis modules 'modular' and keep everything required to run it inside that module. Reduces complexity, makes it simple to run anyone elses analysis.



ALPACA Features

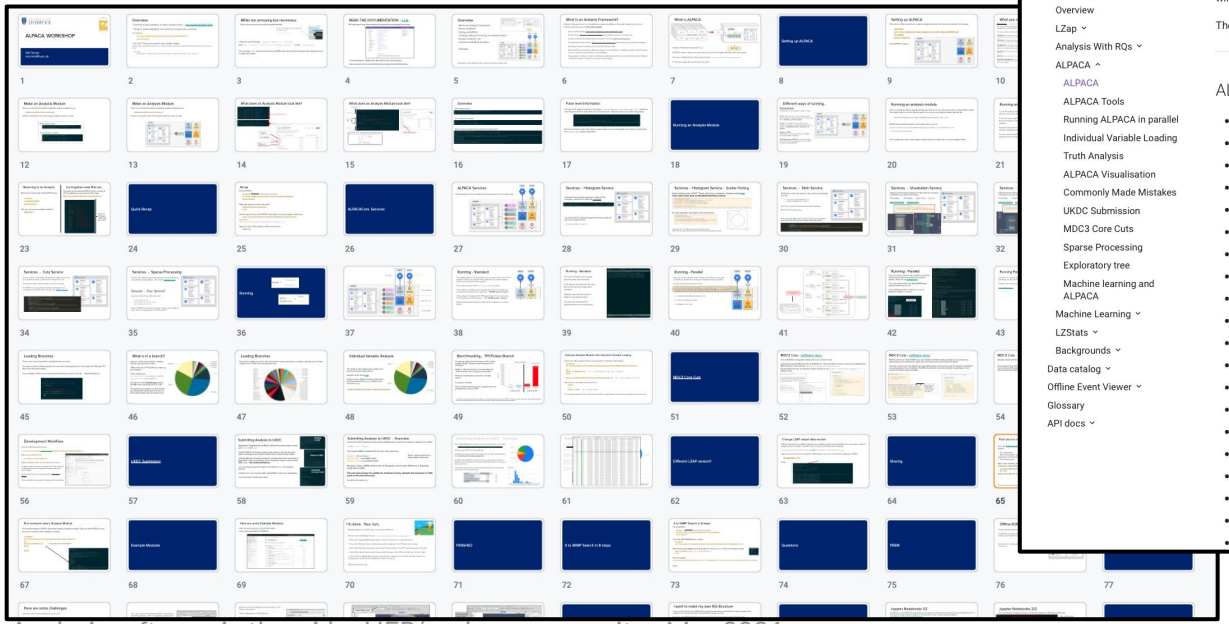
- Cuts that can be **easily** shared between modules and released as part of the package
- **Skimming service** to easily produce reduced datasets
- **Histogram service** - create, fill and save histograms with one line of code
- **Visualisation service** - export objects for ParaView visualisation software
- **Sparse processing service** - produce a file that can be fed back into LZap to only process selected events
- **ONNX Machine Learning plug-in service** - use ML tools within ALPACA

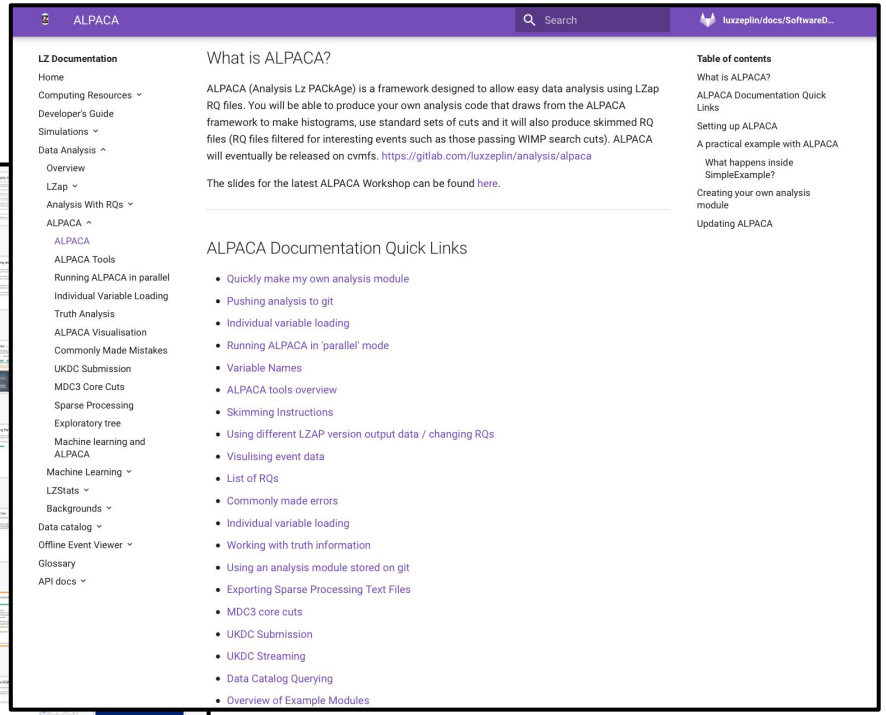
Widely adopted: currently has > 60 users and > 100 modules available on Gitlab!



ALPACA

Good documentation is worth its KLOC in ETH.
New features are only added to master branch if documentation is in place. “it’ll get done later” means it’ll never be written.





Make an Analysis Module

There is a command to produce a skeleton analysis module for you,

```
makeAnalysisModule NeutronAnalysis
```

Make Analysis module...

```
[wturner@gamma WorkshopExample]$ makeAnalysisModule NeutronAnalysis
 Making new analysis module   ->   NeutronAnalysis
[wturner@gamma WorkshopExample]$
```

See it show up as a module...

```
[wturner@gamma WorkshopExample]$ lr modules/
total 4.0K
drwxr-xr-x 6 wturner man  103 Jan  3 16:02 AlpacaCore
drwxr-xr-x 2 wturner man 4.0K Jan  7 11:57 rqlib
drwxr-xr-x 6 wturner man  134 Jan  7 15:08 NeutronAnalysis
```

What does an Analysis Module look like?

```
[wturner@gamma WorkshopExample]$ ls -lrth
total 16K
-rw-r--r-- 1 wturner man 3.2K Jan  3 16:02 README.md
-rw-r--r-- 1 wturner man  720 Jan  3 16:02 build.sh
drwxr-xr-x 4 wturner man   44 Jan  3 16:02 inputFiles
-rw-r--r-- 1 wturner man 3.5K Jan  3 16:02 setup.sh
drwxr-xr-x 3 wturner man   36 Jan  3 16:02 build
drwxr-xr-x 3 wturner man   36 Jan  3 16:02 install
drwxr-xr-x 5 wturner man   63 Jan  3 16:03 modules
-rw-r--r-- 1 wturner man  746 Jan  3 16:03 CMakeLists.txt
drwxr-xr-x 3 wturner man   51 Jan  3 16:17 run
drwxr-xr-x 2 wturner man  142 Jan  3 17:57 analysisScripts
```

Top Level ALPACA directory

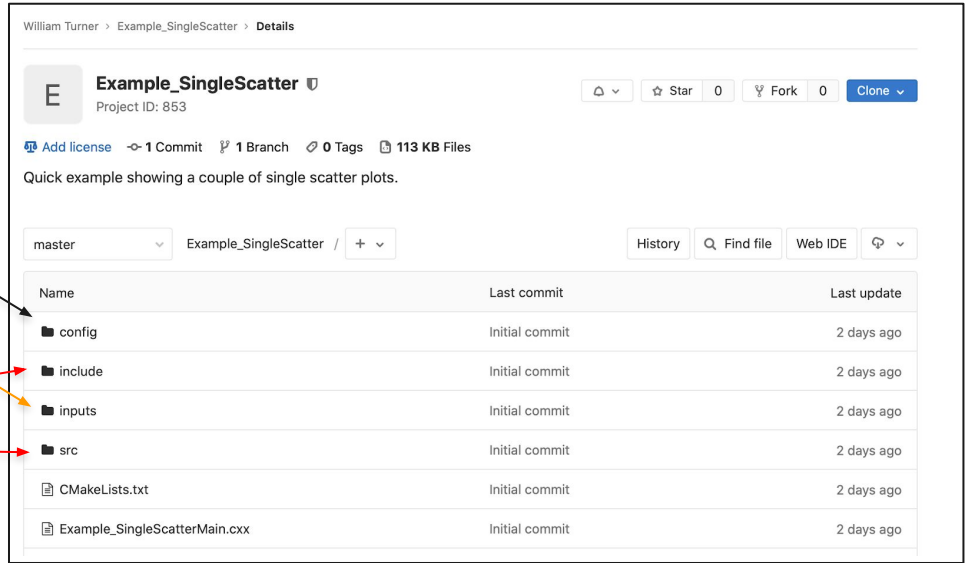
```
[wturner@gamma WorkshopExample]$ ls -lrth modules/
total 4.0K
drwxr-xr-x 6 wturner man  103 Jan  3 16:02 AlpacaCore
drwxr-xr-x 2 wturner man 4.0K Jan  3 16:12 rqlib
drwxr-xr-x 7 wturner man  151 Jan  3 18:05 Example_SingleScatter
[wturner@gamma WorkshopExample]$
```

Modules

Config file

List of datasets

Where your analysis code lives



What does an Analysis Module look like?

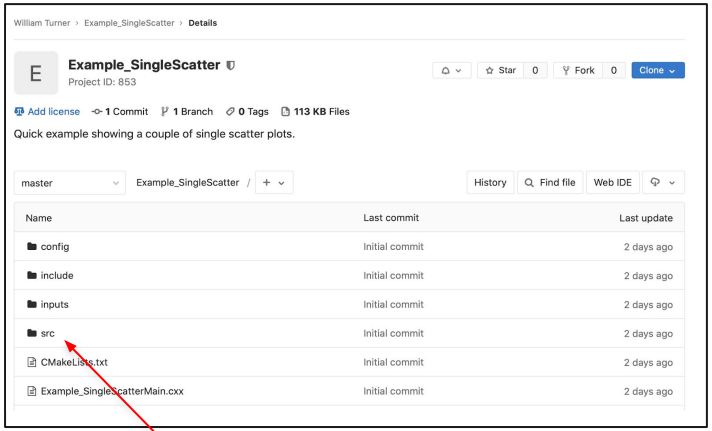
Dataset

Initialize

Execute Event Loop

Finalize

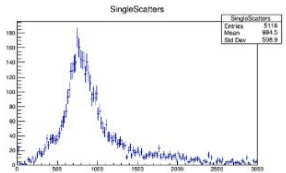
```
// Initialize() -  Called once before the event loop.
void Example_SingleScatter::Initialize() {
  INFO("Initializing Example_SingleScatter Analysis");
}


// Execute() - Called once per event.
void Example_SingleScatter::Execute() {

  // Make a variable from the contents of the data
  //  prevents having to write full ' (*event->Branch)->variableName ' structure many times in the analysis
  int numSS = (*m_event->m_singleScatter)->nSingleScatters;

  // if there is a single scatter in the event then plot the S1 pulse area
  if (numSS > 0){
    m_hists->BookFillHist("SingleScatters", 300, 0., 3000., (*m_event->m_singleScatter)->s1Area_phd);
  }
}

// Finalize() - Called once after event loop.
void Example_SingleScatter::Finalize() {
  INFO("Finalizing Example_SingleScatter Analysis");
}
```



Example_SingleScatter.cxx

Output File

ALPACA/run/Example_SingleScatter/Example_SingleScatter.root



This is all a new user has to know about

ALPACA Features

Scatter Plotting Service

Simple way to plot position information as a graph,

```
m_hists->BookFillScatterPlot("SingleScatters/R2vsZ", m_event->R2(), m_event->Z());
```

UKDC Parallel Data Streaming

Stream data via XRootD to many concurrent processes.
Using local compute and remote data lowers the barrier to entry to using UKDC resources.

PREM Service

Output files (json, root) required for PREM.
Control PREM via configuration variables.
Core Algorithms and user defined algorithms.

Machine Learning Service

ONNX (Open Neural Network Exchange) integration.
Allows use of trained models in your analysis module.
Multiple example modules, Fiducial Volume Optimization and GammaX S1 waveform classification.

Skimming Service

Cut and save selected events to a file ALPACA can read back in.
Allows much faster re-analysis.
Can carry out multiple skims to multiple output files.

Added PREM Features

Easily produce rate plots,

```
m_PREM->createRateGraphTime("Rate/ODHG_200phd_Pulses", selection, binning);
```

Add comments to plots which are shown on PREM Website,

```
m_PREM->AddCommentJSON("module", "histName", "comment");
```

Script to take PREM output and add it to the PREM website.

Continuous Integration

One build stage and 11 test stages, including,

Building and running a simple analysis module.
Streaming data from UKDC via XRootD. Truth loading. Skimming
PREM Module creation and running. Running a module with ML functionality.
Plotting, Scatter Plotting, ERNR Band plotting.

Data Catalog Integration

Introduced a script for users to easily create their own dataset list files from the data catalog storing the query metadata along with the list file.

But C++...

Here is a full WIMP Search module.

Simple, easy to read, easy to modify.

```
void MDC3WIMPSearch::Execute() {

  float logS1S2 = 0;
  double radius = 0;
  float s1_corr = 0;
  float s2_corr = 0;
  float drift = 0;
  int s1Start;
  bool skinVeto = false;
  bool odVeto = false;

  // Must be run at the beginning of every event to set up ETrain veto for the event
  m_cuts->mdc3()->InitializeETrainVeto();

  // use the MDC3 Core cuts to see if this event is salt
  bool isItSalt = m_cuts->mdc3()->IsSalt();

  //This is the main progression of cut logic
  // First see if this event is a single scatter
  if (m_cuts->mdc3()->SingleScatter()) {

    // if so, get some variables and fill some histograms
    s1_corr = (*m_event->m_singleScatter)->correctedS1Area_phd;
    s2_corr = (*m_event->m_singleScatter)->correctedS2Area_phd;
    drift = (*m_event->m_singleScatter)->driftTime_ns/1000.;
    logS1S2 = log10(s2_corr);
    radius = (((*m_event->m_singleScatter)->correctedX_cm*(*m_event->m_singleScatter)->correctedX_cm)
              +((*m_event->m_singleScatter)->correctedY_cm*(*m_event->m_singleScatter)->correctedY_cm));
    m_hists->GetHistFromMap("SS_LogS1S2")->Fill( s1_corr, logS1S2);
    m_hists->GetHistFromMap("SS_RvsDrift")->Fill(radius,drift);
    m_hists->GetHistFromMap("SS_XY")->Fill((*m_event->m_singleScatter)->correctedX_cm,(*m_event->m_singleScatter)->correctedY_cm);

    // see if the event passes the ETrainVeto cut
    if (!m_cuts->mdc3()->ETrainVeto()) {
      m_hists->GetHistFromMap("SS_ET_LogS1S2")->Fill( s1_corr, logS1S2);
      m_hists->GetHistFromMap("SS_ET_RvsDrift")->Fill(radius,drift);
      m_hists->GetHistFromMap("SS_ET_XY")->Fill((*m_event->m_singleScatter)->correctedX_cm,(*m_event->m_singleScatter)->correctedY_cm);

      // now see if the event is in the WS ROI
      if( m_cuts->mdc3()->WIMPSearchROI() ){
        m_hists->GetScatterPlot("SS_ET_ROI_LogS1S2")->Fill( s1_corr, logS1S2);
        m_hists->GetScatterPlot("SS_ET_ROI_RvsDrift")->Fill(radius,drift);
        m_hists->GetScatterPlot("SS_ET_ROI_XY")->Fill((*m_event->m_singleScatter)->correctedX_cm,(*m_event->m_singleScatter)->correctedY_cm);

        // if the event is in the Fiducial volume
        if (m_cuts->mdc3()->Fiducial()) {
          m_hists->GetScatterPlot("SS_ET_ROI_FV_LogS1S2")->Fill( s1_corr, logS1S2);
          m_hists->GetScatterPlot("SS_ET_ROI_FV_RvsDrift")->Fill(radius,drift);
          m_hists->GetScatterPlot("SS_ET_ROI_FV_XY")->Fill((*m_event->m_singleScatter)->correctedX_cm,(*m_event->m_singleScatter)->correctedY_cm);

          skinVeto = m_cuts->mdc3()->SkinVeto();
          odVeto = m_cuts->mdc3()->ODVeto();
          // if the skin veto and od veto pass, make some more plots
          if (!skinVeto && !odVeto) {
            m_hists->GetScatterPlot("SS_ET_ROI_FV_Veto_LogS1S2")->Fill( s1_corr, logS1S2);
            m_hists->GetScatterPlot("SS_ET_ROI_FV_Veto_RvsDrift")->Fill(radius,drift);
            m_hists->GetScatterPlot("SS_ET_ROI_FV_Veto_XY")->Fill((*m_event->m_singleScatter)->correctedX_cm,(*m_event->m_singleScatter)->correctedY_cm);

            // if its not a salt event then fill last plots and skim the event
            if (!isItSalt) {
              m_hists->GetScatterPlot("SS_ET_ROI_FV_Veto_LogS1S2_Unsalted")->Fill( s1_corr, logS1S2);
              m_hists->GetScatterPlot("SS_ET_ROI_FV_Veto_RvsDrift_Unsalted")->Fill(radius,drift);
              m_hists->GetScatterPlot("SS_ET_ROI_FV_Veto_XY_Unsalted")->Fill((*m_event->m_singleScatter)->correctedX_cm,(*m_event->m_singleScatter)->correctedY_cm);

              // Skim the event as all of the cuts have been passed
              m_skim->SkimEvent(m_event);
            }
          }
        }
      }
    }
  }
}
```

ALPACA Running

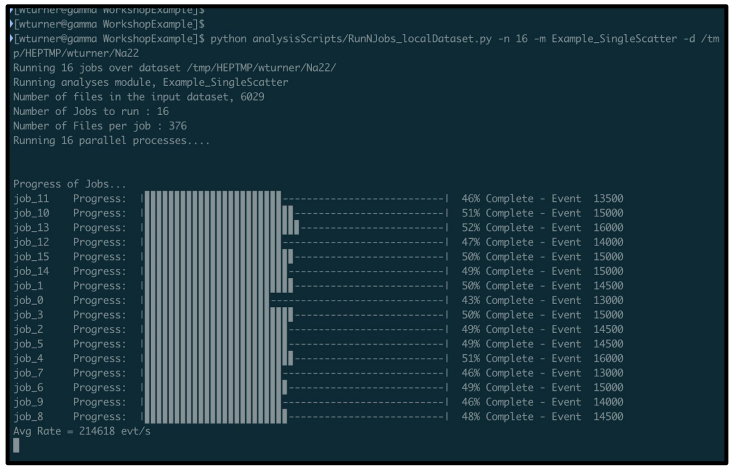
Multiple different ways to run ALPACA analysis.

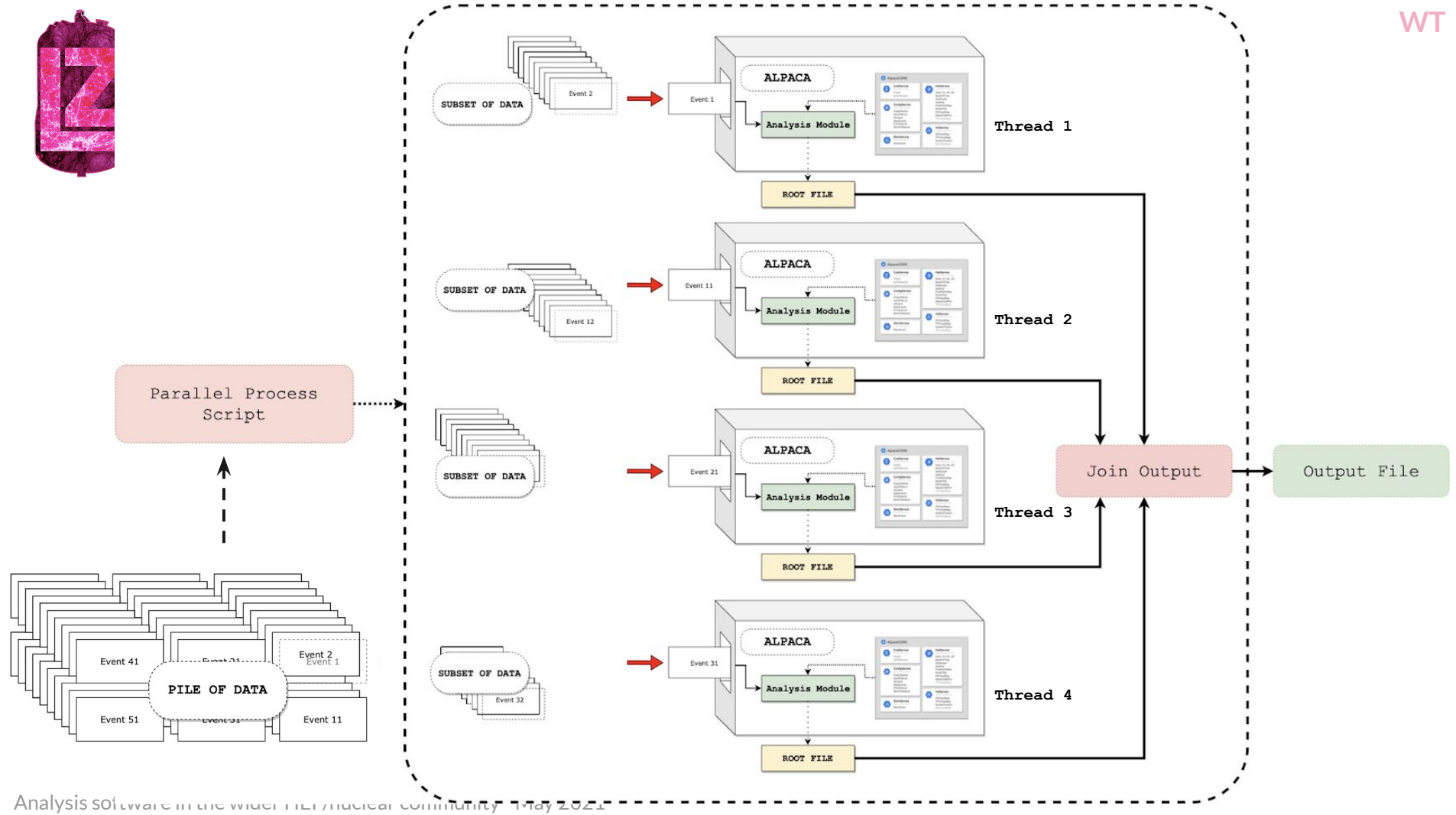
Serial mode - 1 process over 1 input file list. *Useful when you care about event to event.*

Parallel mode - splits input file list into **N** parts and runs **N** ALPACA processes over the split file lists. Then joins the output when they have all finished. *Useful when you care about speed.* 1 month WIMP Search analysis, 16 cores = 15mins.

Use python subprocess to run in parallel, check the status of each subprocess to get the % complete each process is at, print this as a nice progress bar. Works nicely up to ~50 threads then the progress bars become unwieldy.

Can use slurm/srun to spawn threads which can run over multiple nodes. When running hundreds of threads the progress bar collapses to a single progress bar for all threads. 1 month WIMP Search analysis, 96 cores = ~3mins.




Parallel Process Script
PILE OF DATA
Event 41
Event 2
Event 1
Event 51
Event 11
SUBSET OF DATA
Event 2
Event 1
ALPACA
Analysis Module
ROOT FILE
Thread 1
SUBSET OF DATA
Event 12
Event 11
ALPACA
Analysis Module
ROOT FILE
Thread 2
SUBSET OF DATA
Event 21
ALPACA
Analysis Module
ROOT FILE
Thread 3
SUBSET OF DATA
Event 32
Event 31
ALPACA
Analysis Module
ROOT FILE
Thread 4
Join Output
Output File

Submitting Analysis to UKDC

Can submit to UK GRID via DIRAC (soon ganga)

To run the MDC3WIMPSearch module over all of the April Data on the UKDC.

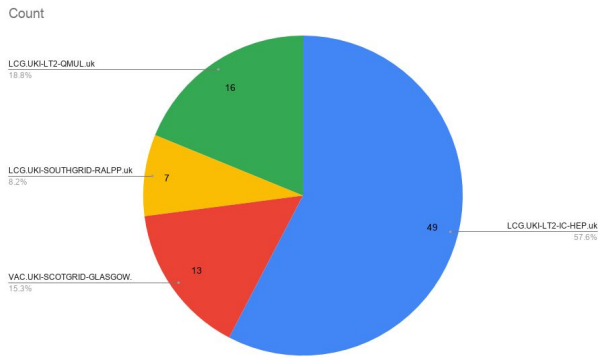
```
[wturner@gamma ALPACA]$ UKDC submitClean MDC3WIMPSearch -n 50 -s
```

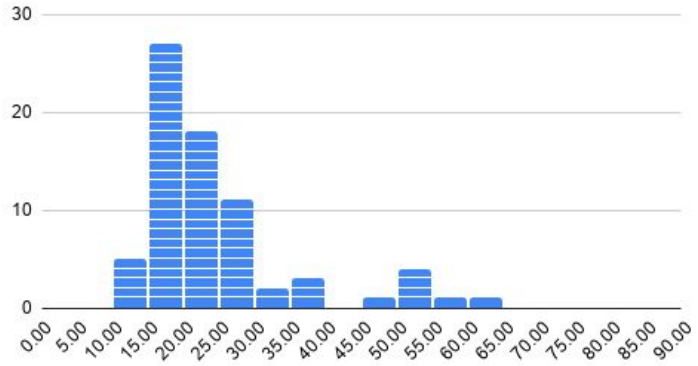
50 files per job. 4217 files total, so 85 jobs.

Average time per job is 24mins, but varies from 16 - 54mins depending on site. **All times given include queuing time.**

At CORI running ALPACA in parallel mode, using 16 cores running over 3640 files (not included the first 5 days of April) takes 15mins. Obviously no queuing time here.

This analysis also produces a skimmed file allowing future studies to be carried out in a fraction of the time.





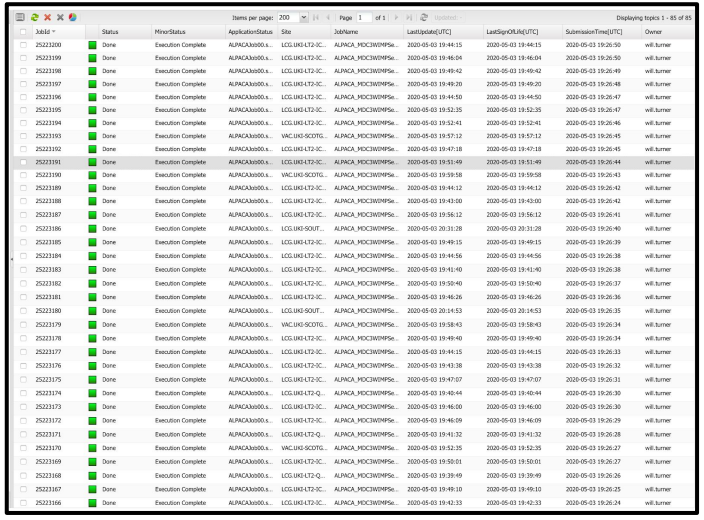
Submitting Analysis to UKDC - Behind the scenes

The `buildUKDC` command builds ALPACA and any analysis modules present using *cmake*, as usual, but also uses *cpack* to package up ALPACA and installed modules into a 'self-extracting installer' to unpack ready to use on UKDC nodes. So no compiling required.

```
(base) [wturner@gamma ALPACA]$ buildUKDC
/hepstore/wturner/LZ/ALPACA/17thSep/ALPACA /hepstore/wturner/LZ/ALPACA/17thSep/ALPACA
Setting up ALPACA...
 ALPACA top directory = /hepstore/wturner/LZ/ALPACA/17thSep/ALPACA
 Sourcing latest LZ Build setup script..
 Using LZ GCC Version: 8.2.0 and ROOT Version: 6.20.04..
 LZAP version used for RQ generation 4.8.3
 alpaca top dir = /hepstore/wturner/LZ/ALPACA/17thSep/ALPACA
 Building ALPACA and all analysis modules for UKDC submission...
 LZ Build Config:  x86_64-centos7-gcc8-opt_UKDC
```

Also, some universities compiles against libs not present on some of the UKDC nodes. So building and packaging stage is done in a Singularity container using an image which is similar to the UKDC nodes environment -> **No random job failures.**

> UKDC submit s2Area



To load whole branches or to not load whole branches...

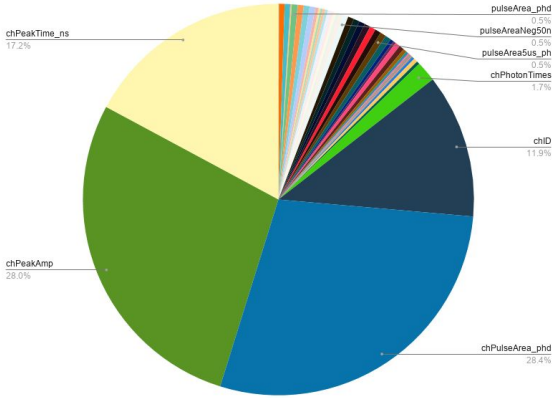
In ALPACA users can load specific branches to avoid loading all of the data.

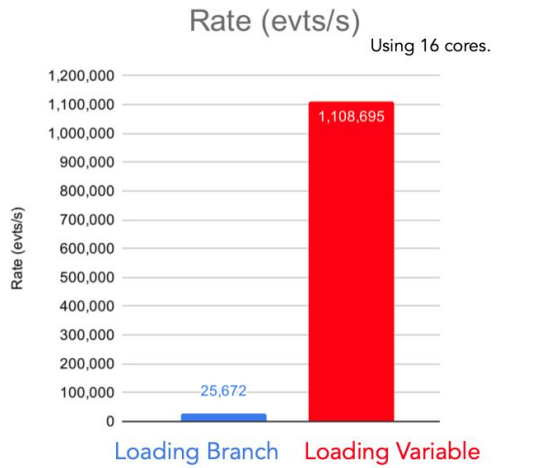
Although some branches contain vector<vector<float>>'s which take up most of the size of the branch.

If these variables are not required in the analysis it is pointless to load them. Can then use direct variable loading to avoid this.

Comparing loading the whole branch with loading a single variable using the simple example of an S1/S2 plot.

Makes it **~20** times faster for uncached data.





Development Workflow

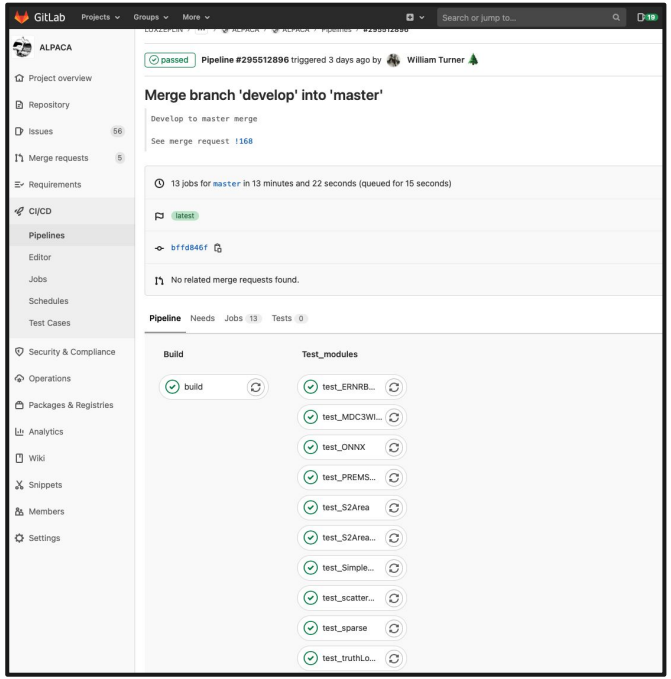
Any changes to ALPACA pass through GitLab CI.

- This checks that the core ALPACA code builds
- Runs certain test modules to ensure all the features of ALPACA still work
- Runs key analysis modules over a subset of simulated data

Output from modules are ROOT files containing plots. All plots are turned to PDFs to view in the web browser for easy comparison to previous versions.

Some python code runs at the end of each module to compare the predefined mean/std dev for each plot to detect changes in analysis results.

Any changes result in a failed stage in the CI.



Conclusion & Contributors

LZ has multiple software packages and frameworks to aid analysis, with the two key ones being **LZap** for processing (reconstruction, corrections, etc) and **ALPACA** for analysis (cuts, histograms, etc).

Our analysis framework ALPACA allows plug & play of other analysts modules, official cut sharing, fast parallel processing, interface with the Physics Readiness Monitor (PREM) and more. Uptake by users has been very high and its use fully exercised during our mock data challenge WIMP search last year where all main analyses were in ALPACA.

Contributors:

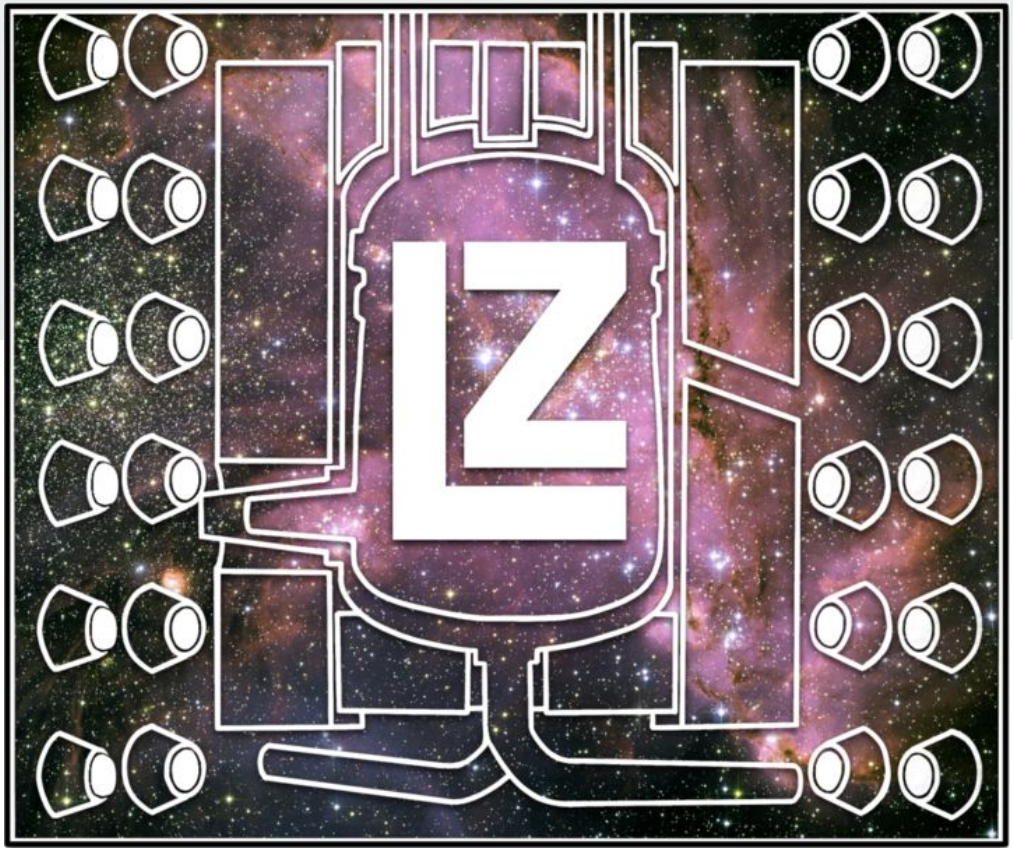
LZap

Alden Fan (SLAC), **Sally Shaw** (UCSB), **Francisco Neves** (Coimbra), **Kelly Stifter** (SLAC), **Paulo Brás** (Coimbra), **Tyler Anderson** (SLAC), **Theresa Fruth** (UCL), **Amy Cottle** (Oxford), **Vladimir Solovov** (Coimbra), **Guillerme Pereira** (Coimbra), **Alice Baxter** (Liverpool), **Reed Watson** (UC Berkeley), **Jordan Palmer** (RHUL), **Ryan Linehan** (SLAC), **Simon Patton** (LBNL), **Luke Krezcko** (Bristol), **Maria Elena Monzani** (SLAC), **Clarke Hardy** (SLAC)

ALPACA

Will Turner (Liverpool), **Alden Fan** (SLAC), **Andrew Stevens** (Oxford), **Sally Shaw** (UCSB), **Andrew Naylor** (Sheffield), **Micah Buuck** (SLAC)

Thanks!



Extra Slides

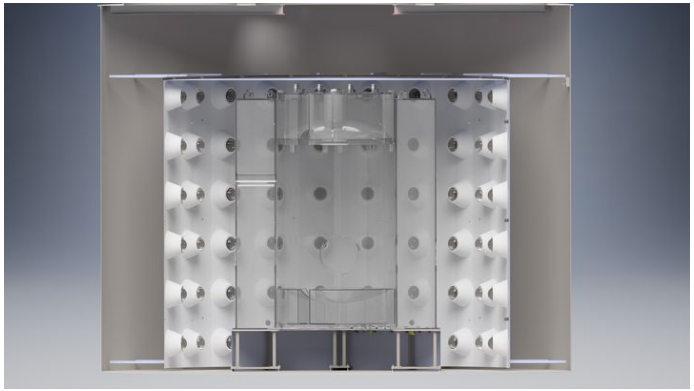
Dr. Sally Shaw, UC Santa Barbara

Data Analysis Convener for LZ

Dr. Will Turner, University of Liverpool

Deputy Data Analysis Convener for LZ

Visualisation:
Event Viewer & Geometry Viewers



Event Viewer

The LZ Offline Event Viewer implements an interactive visualization python library to display pulse and event information from LZAP output files **in a web browser.**

Now at an advanced stage and widely used.

Very versatile and easy to use.

Three ways to view waveforms:

- Preprocessed LZap file
- Process a whole raw file with the event viewer
- Process a set of events with sparse processing input

Geometry & Event Viewer

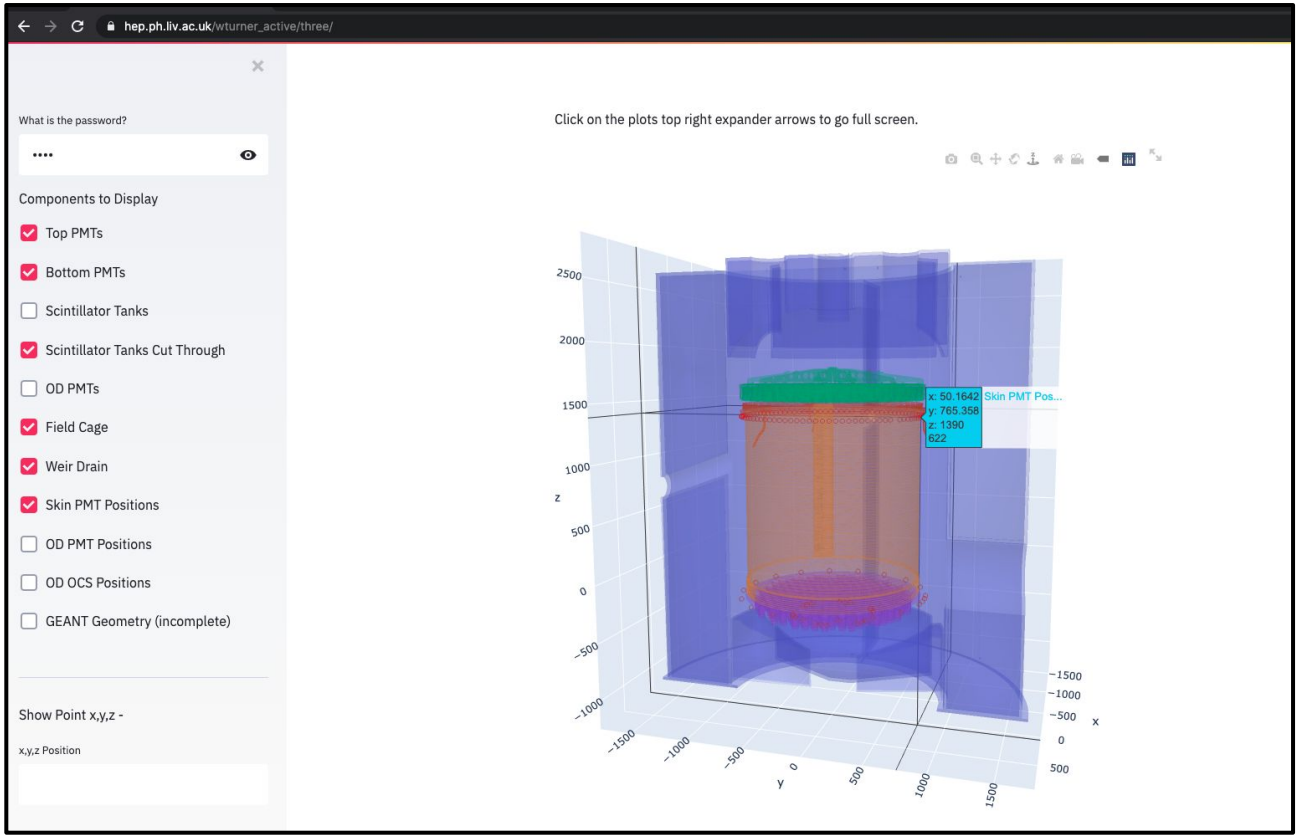
Allows easy browser based viewing of selective LZ CAD Geometry with PMT positions.

Export 3D positions from ALPACA analysis and view on top of any geometry.

Visualise simulation geometry hit positions on top of LZ CAD to verify simulation geometry.

Easily map scalar quantities to PMT positions to view versus CAD geometry.

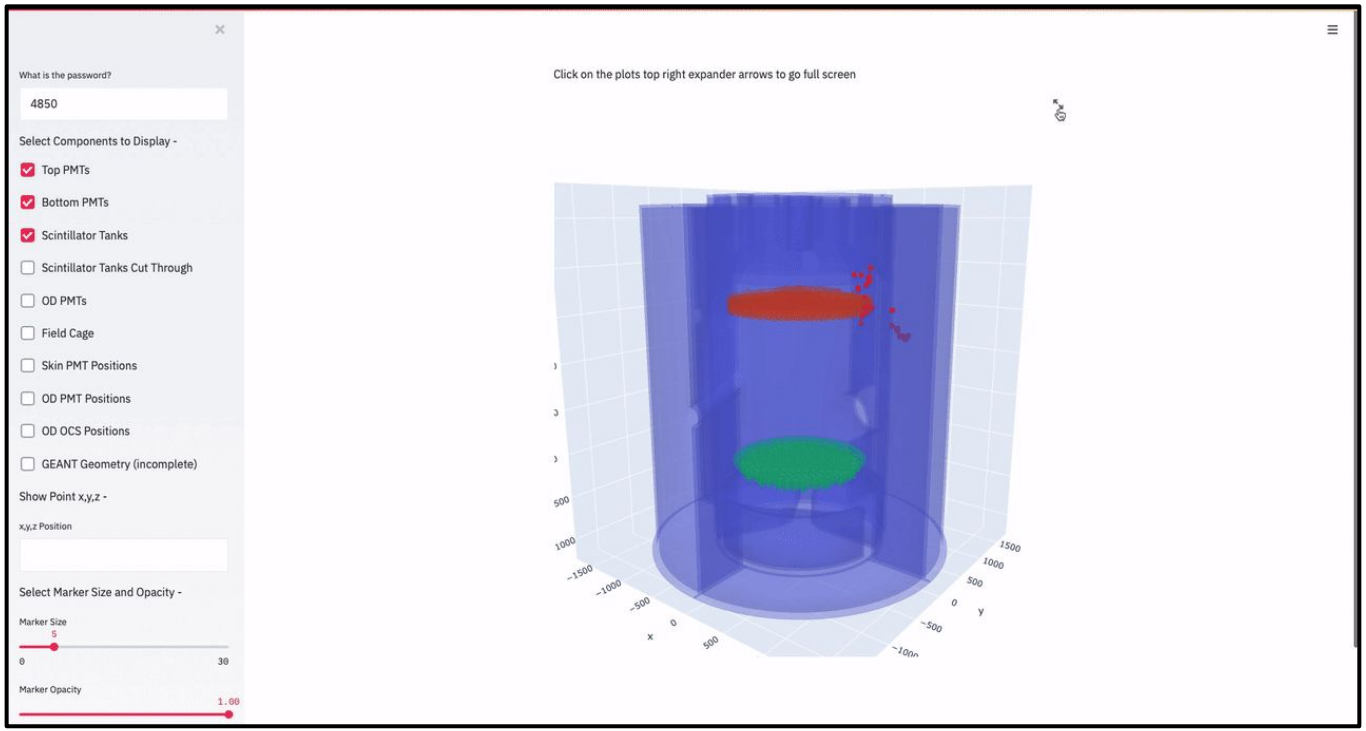
https://hep.ph.liv.ac.uk/wturner_active/three/ p/w 4850



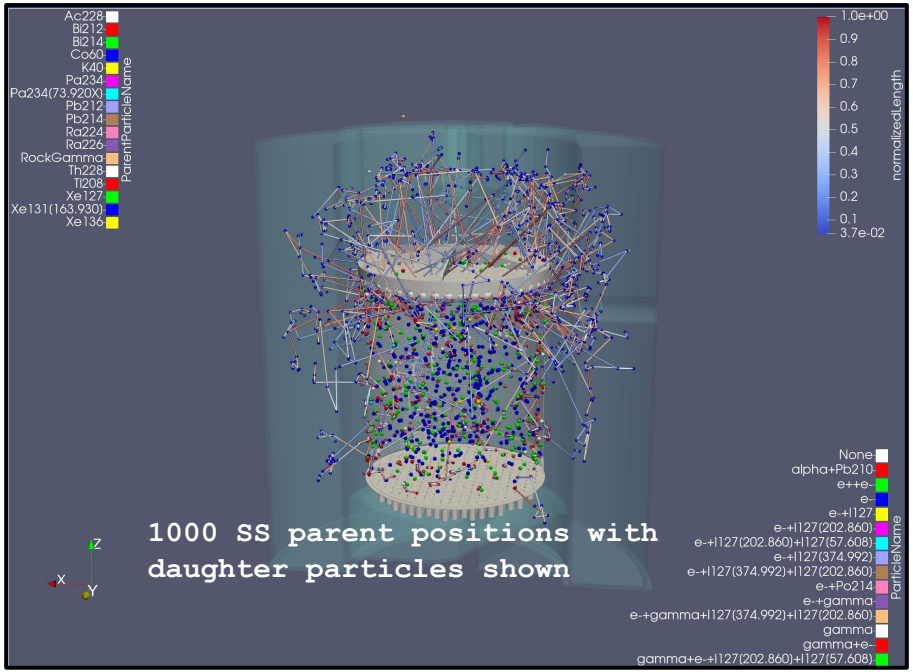
Geometry & Event Viewer

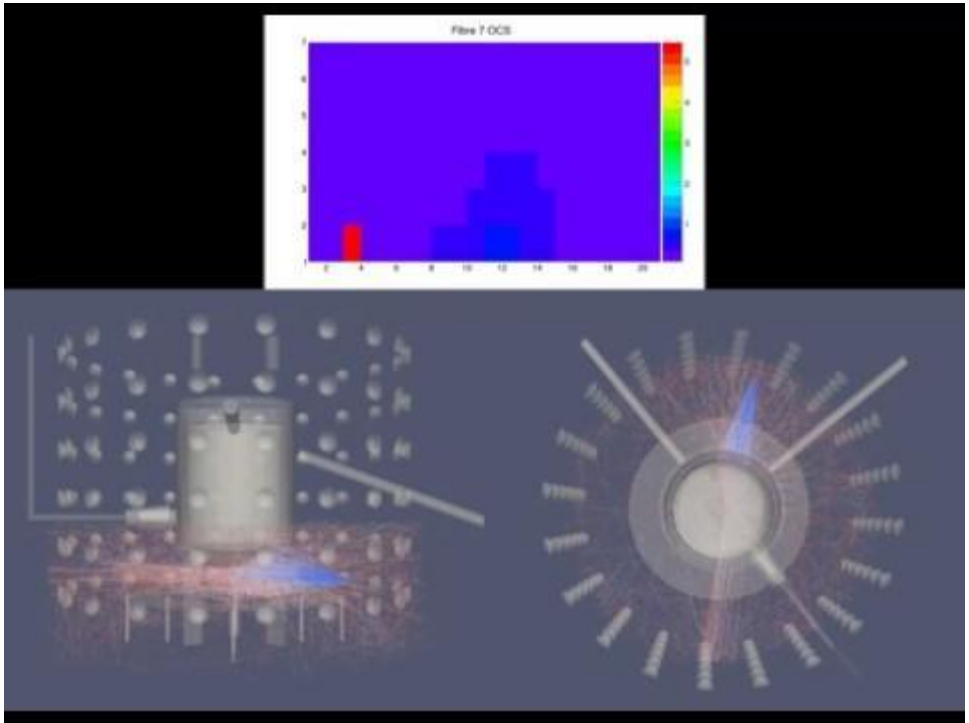
https://hep.ph.liv.ac.uk/wturner_active/three/ p/w 4850

This could be hosted on SPIN for a significant speed-up.



Paraview


Ac228
Bi212
Bi214
Co60
K40
Pa234
Pa234(73.920X)
Pb212
Pb214
Ra224
Ra226
RockGamma
Th228
Tl208
Xe127
Xe131(163.930)
Xe136
ParentParticleName
1.0e+00
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
3.7e-02
normalizedLength
1000 SS parent positions with
daughter particles shown
Z
X
Y
None
alpha+Pb210
e++e-
e-
e-+I127
e-+I127(202.860)
e-+I127(202.860)+I127(57.608)
e-+I127(374.992)
e-+I127(374.992)+I127(202.860)
e-+Po214
e-+gamma
e-+gamma+I127(374.992)+I127(202.860)
gamma
gamma+e-
gamma+e-+I127(202.860)+I127(57.608)
ParticleName

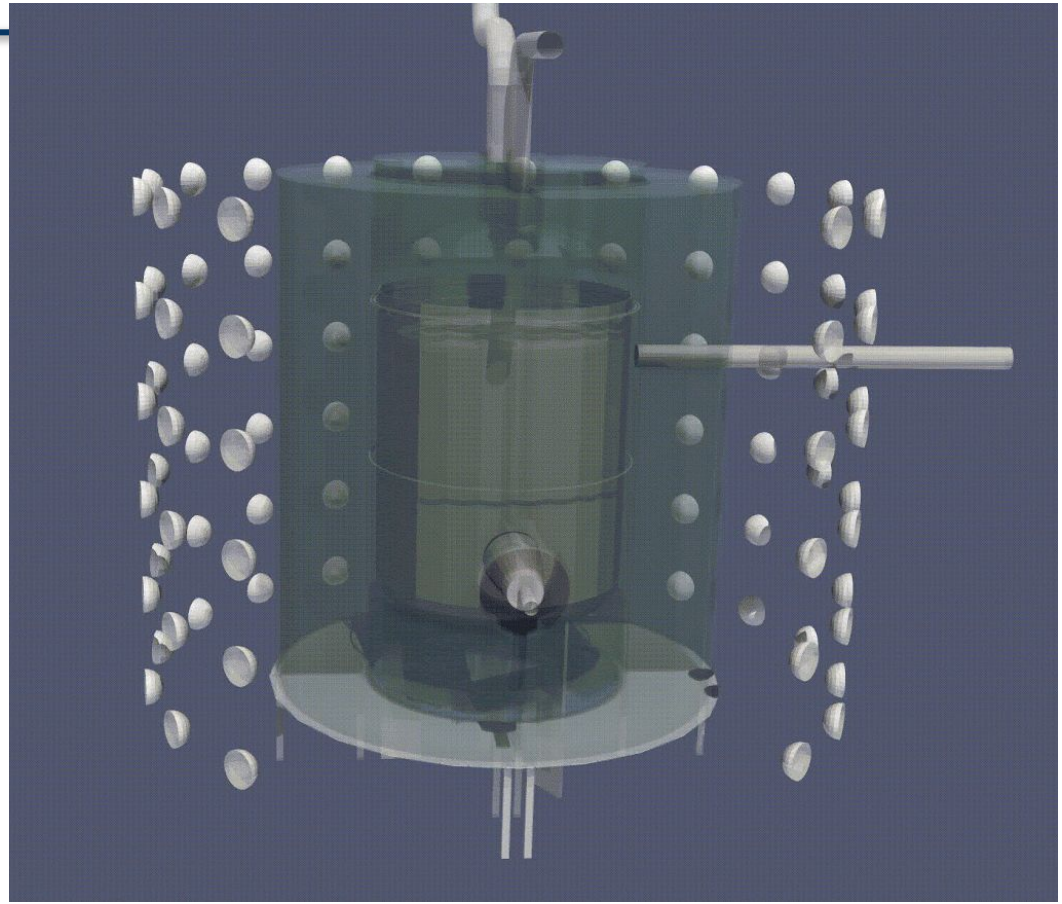

Fibre 7 OCS

Paraview

Light collection vs location in scintillator tanks.

Paraview does everything the Geometry Viewer does but also allows GEANT events to be visualised.

Can export event truth information from ALPACA to be viewed in 3D.



Paraview

Enriched Outer Detector
Background Neutrons

DD Generator