

Hall A Analysis Software & Computing Update

Ole Hansen

Jefferson Lab

Hall A Collaboration Meeting
February 10, 2022

Podd Status

- Current release: **1.7.0** (16 Nov 2021)
 - ▶ Many new features (already presented at previous meetings)
 - ▶ Additional improvements and bugfixes based on early SBS data taking
 - ▶ Significant speedup, primarily in decoder and database
 - ▶ Improved CODA 3 support
 - ▶ Dynamic raw data event buffer size
 - ▶ PID calculation based on Bayesian likelihoods
 - ▶ Requires **C++11** compiler and ROOT 6. Installed in counting house and on the farm.
- Priority development: **2.0-devel** (Summer 2022, delayed because of SBS work)
 - ▶ **Multithreading**
 - ▶ Will benefit SBS and Hall C, primarily for online replay
 - ▶ Requires **C++17** (e.g. gcc 9+, available on ifarm)
 - ▶ Existing code will need minor modifications
- Auxiliary development: **1.8-devel** (if time permits)
 - ▶ Add small new features missed in 1.7
 - ▶ Maintain system requirements and API of version 1.7 as much as possible

Podd: Profile-Based Code Optimization

```
callgrind-analysis.post-APVMAP.27173 - emacs@aon14.jlab.org (on aon14.jlab.org)
File Edit Options Buffers Tools Help
[Icons] Save Undo [Icons]

Profile data file 'callgrind.out.27173' (creator: callgrind-3.15.0)
-----
LL cache:
D1 cache:
LL cache:
Time range: Basic block 0 - 73993361480
Trigger: Program termination
Profiled target: analyzer -x replay_gnn_ole.C111615,10000,0,"e1209029",0,1,0 (PID 27173, part 1)
Events recorded: Ir
Events shown: Ir
Event sort order: Ir
Thresholds: 99
Include dirs:
User annotated:
Auto-annotation: off

-----
Ir
374,728,900,610 PROGRAM TOTALS

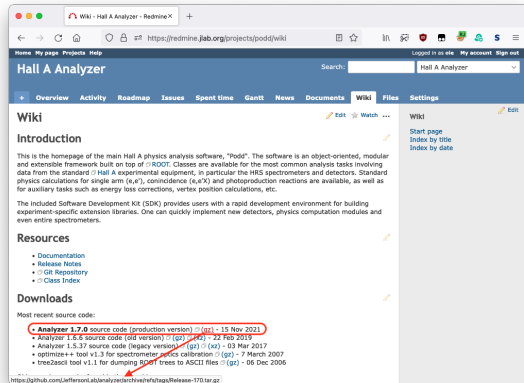
-----
Ir
files:Function
-----
373,348,393,338 < /adaqfs/apps/R00T/6.24-06/src/core/rint/src/TrInt.cxx:TrInt::Run(bool) (14x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libRint.so.6.24.06]
11,916,498,830 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THSpectrometer.cxx:THSpectrometer::Init(TDtime const&) (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
1,290,990,209 < /adaqfs/apps/analyzer/1.7.0/src/apps/analyzer_cxx/main (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/bin/analyzer]
1,290,963,864 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaInterface.cxx:THaInterface::HaInterface(char const*, int*, char**, void*, int, bool) (15x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
1,111,709,780 < /adaqfs/apps/R00T/6.24-06/src/core/rint/src/TrInt.cxx:TrInt::TrInt(char const*, int*, char**, void*, int, bool) (17x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libRint.so.6.24.06]
656,393,019 < /adaqfs/apps/R00T/6.24-06/src/core/base/src/TR00T.cxx:TR00T::Internal::Run(R00T2i) (2x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libCore.so.6.24.06]
494,945,441 < /adaqfs/apps/R00T/6.24-06/src/tree/tree/src/Tree::Init(char const*, int, int) const (2x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libTree.so.6.24.06]
442,929,895 < /adaqfs/apps/R00T/6.24-06/src/core/retacling/src/TCling.cxx:TCling::CreateInterpreter (1x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libCling.so.6.24.06]
357,690,961 < /SBS-offline/src/SBSOEMSpectrometerTracker.cxx:SBSOEMSpectrometerTracker::Init(TDtime const&) (1x) [/adaqfs/home/a-onl/gnn/ole/SBS-offline/gcc48/RelWithDebInfo/lib64/libbsbs.so]
357,633,503 < /SBS-offline/src/SBSOEMTrackerBase.cxx:SBSOEMTrackerBase::CompleteInitialization() (3x) [/adaqfs/home/a-onl/gnn/ole/SBS-offline/gcc48/RelWithDebInfo/lib64/libbsbs.so]

--[X]-- callgrind-analysis.post-APVMAP.27173 Top (92.0) (Fundamental)
357,752,479,568 * /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::Process(THaRunBase*) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
202,704,796,508 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::MainAnalysis() (10,010x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
93,154,735,474 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::ReadOneEvent() (10,010x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
47,643,722,917 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::EndAnalysis() (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
12,617,786,395 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::Init(THaRunBase*) (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
11,414,741,281 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaOutput.cxx:THaOutput::End() (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
222,364,470 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::BeginAnalysis() (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
12,683,081 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaCutList.cxx:THaCutList::ClearAll(char const*) (10,010x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
4,095,213 < /adaqfs/apps/R00T/6.24-06/src/core/base/src/TBenchmark.cxx:TBenchmark::Stop(char const*) (10,014x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib64/libCore.so.6.24.06]
3,656,714 < /adaqfs/apps/analyzer/1.7.0/src/hana_decode/THaBenchmark.h:THaBenchmark::Begin(char const*) (10,014x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
1,273,580 < /adaqfs/apps/R00T/6.24-06/src/io/lo/src/TDirectoryFile.cxx:TDirectoryFile::Fungel(short) (1x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libFIO.so.6.24.06]
901,760 < /adaqfs/apps/R00T/6.24-06/src/core/base/src/TObject.cxx:TObject::Write(char const*, int, int) (2x) [/adaqfs/apps/R00T/6.24-06/gcc48/RelWithDebInfo/lib/libCore.so.6.24.06]
617,338 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaAnalyzer.cxx:THaAnalyzer::PrintSummary(THaAnalyzer::EE::tStatus) const (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
302,451 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaRunBase.cxx:THaRunBase::Update(THaVData const*) (10,010x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
11,674 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaRun.cxx:THaRun::Open() (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
11,060 < /adaqfs/apps/analyzer/1.7.0/src/Podd/THaRun.cxx:THaRun::operator=(THaRunBase const&) (1x) [/adaqfs/apps/analyzer/1.7.0/gcc48/RelWithDebInfo/lib64/libPodd.so.1.7.0]
4,313 < /usr/src/debug/llvm-11.0.0-rc1/lib64/2.17-c758a68e1f1f/llvm-11.0.0-rc1/lib64/2.17.so

--[X]-- callgrind-analysis.post-APVMAP.27173 298 (4010,162) (Fundamental)
```

Podd Source Code & Documentation

JLab Redmine



The screenshot shows the JLab Redmine Wiki page for the Hall A Analyzer. The page has a blue header with navigation links: Home, My page, Projects, Help. Below the header, there's a search bar and a dropdown menu. The main content area is titled "Hall A Analyzer" and includes a "Wiki" tab. The "Introduction" section describes the software as an object-oriented, modular framework for Hall A physics analysis. The "Resources" section lists links to Documentation, Release Notes, Git Repository, and Class Index. The "Downloads" section lists recent source code releases, with the latest release, "Analyzer 1.7.0 source code (production version)", highlighted in a red box. A red arrow points to the download link for this release.

Wiki

Introduction

This is the homepage of the main Hall A physics analysis software, "Podd". The software is an object-oriented, modular and extensible framework built on top of ROOT. Classes are available for the most common analysis tasks involving data from the standard Hall A experimental equipment, in particular the HRS spectrometers and detectors. Standard physics calculations for single arm (e,e'), coincidence ($e,e'X$) and photoproduction reactions are available, as well as for auxiliary tasks such as energy loss corrections, vertex position calculations, etc.

The included Software Development Kit (SDK) provides users with a rapid development environment for building experiment-specific extension libraries. One can quickly implement new detectors, physics computation modules and even entire spectrometers.

Resources

- Documentation
- Release Notes
- Git Repository
- Class Index

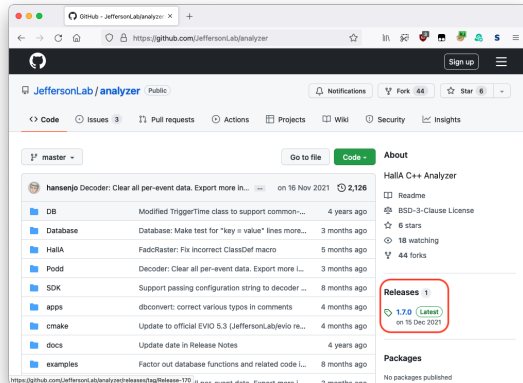
Downloads

Most recent source code:

- Analyzer 1.7.0 source code (production version) (gz) - 15 Nov 2021**
- Analyzer 1.6.6 source code (old version) (gz) (xz) - 22 Feb 2019
- Analyzer 1.5.37 source code (legacy version) (gz) (xz) - 03 Mar 2017
- optimize++ tool v1.3 for spectrometer peaks calibration (gz) - 7 March 2007
- tree2ascii tool v1.1 for dumping ROOT trees to ASCII files (gz) - 06 Dec 2006

<https://github.com/JeffersonLab/analyzer/archive/refs/tags/Release-170.tar.gz>

GitHub



The screenshot shows the GitHub repository page for JeffersonLab/analyzer. The page has a dark header with navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights. The main content area shows the repository details, including the name "JeffersonLab/analyzer", the license "BSD-3-Clause License", and the number of stars (6) and forks (44). The "Releases" section is highlighted with a red box, showing the latest release, "1.7.0 (Latest)", dated 15 Dec 2021. The "Packages" section shows that no packages have been published.

JeffersonLab/analyzer

Code Issues Pull requests Actions Projects Wiki Security Insights

master

Go to file Code About

hansenjo Decoder: Clear all per-event data. Export more in... on 16 Nov 2021 2,126

File	Description	Time
DB	Modified TriggerTime class to support common...	4 years ago
Database	Database: Make test for "key = value" lines more...	3 months ago
HallA	FadcRaster: Fix incorrect ClassDef macro	5 months ago
Podd	Decoder: Clear all per-event data. Export more l...	3 months ago
SDK	Support passing configuration string to decoder ...	8 months ago
apps	dbconvert: correct various typos in comments	4 months ago
cmake	Update to official EVIO 5.3 (JeffersonLab/evio re...	4 months ago
docs	Update date in Release Notes	4 years ago
examples	Factor out database functions and related code l...	8 months ago

Releases 1

1.7.0 (Latest) on 15 Dec 2021

Packages

No packages published

Podd: Building with CMake

Prerequisites:

- Install ROOT (ensure root-config is in PATH, or set \$ROOTSYS)
 - ▶ Farm: run setroot_CUE.csh. RHEL: install from EPEL. macOS: install from Homebrew.
 - ▶ See also https://redmine.jlab.org/projects/podd/wiki/ROOT_Installation_Guide
- Ensure you have CMake ≥ 3.5 (cmake --version. cmake3 on RedHat)

Building & Installing Podd with CMake ≥ 3.15

```
$ git clone https://github.com/JeffersonLab/analyzer.git
$ cmake -S analyzer -B analyzer-build [-DCMAKE_INSTALL_PREFIX=/some/dir]
$ cmake --build analyzer-build [-j4]
$ ./analyzer-build/apps/analyzer
$ [cmake --install analyzer-build]
$ [/some/dir/bin/analyzer]
```

Notes:

- Installing recommended (cmake --install): Set CMAKE_INSTALL_PREFIX
- Will phase out aging SCons build system (too many limitations)

Pre-Installed Podd

farm/ifarm (works in Counting House, too)

```
$ module use /group/halla/modulefiles
$ module load analyzer
$ analyzer --version
Podd 1.7.0 Linux-3.10.0-1160.31.1.el7.x86_64-x86_64 git @e26c21d ROOT 6.22/06
```

Counting House (local installation, faster, safer)

```
$ module use /adaqfs/apps/modulefiles
$ module load analyzer
$ analyzer --version
Podd 1.7.0 Linux-3.10.0-1160.31.1.el7.x86_64-x86_64 git @e26c21d ROOT 6.24/06
```

The SDK is located in `$ANALYZER/./src/SDK/`

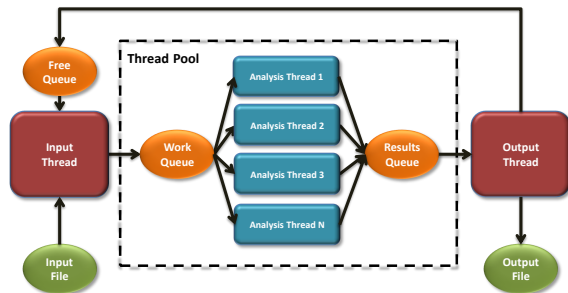
Podd 2.0

- Event-based parallelization/**multithreading**
 - ▶ Important for online replay
 - ▶ Reduced memory footprint compared to multiple individual jobs
 - ▶ Requires **thread safe** user code (→ only const or protected globals, statics)
- I/O improvements
 - ▶ Output system upgrade (full set of data types, object variables) — largely complete
 - ▶ TBD: **HIPO** or **PODIO** output file format support
 - ▶ TBD: **EVIO 6** input format support (HIPO-like raw data files)
 - ▶ Goal: Make output easily usable with Python and Julia tools (e.g. [uproot](#), [UnROOT](#))

ETA: This summer. Delayed because of work on SBS.

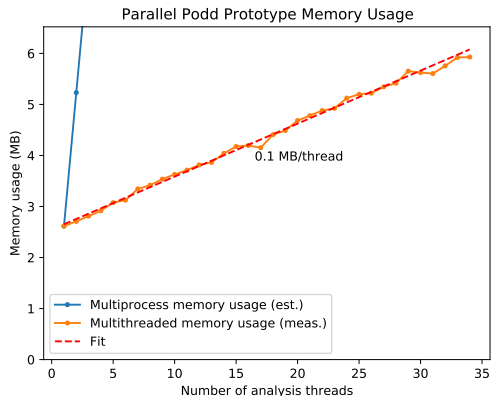
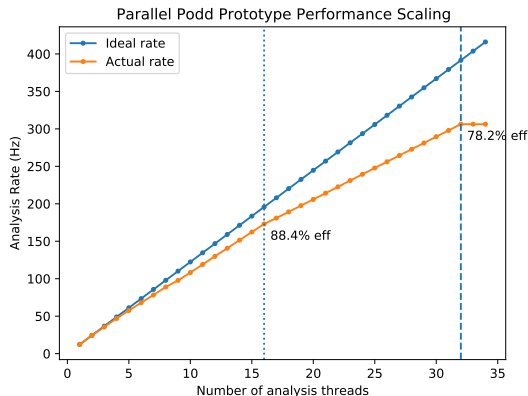
Podd Parallel Processing Prototype

- <https://github.com/hansenjo/parallel>
- Small standalone toy analyzer with **hand-crafted multithreading** (`std::thread`)
- Mimics main components of Podd (e.g. decoder, analysis variables, output)
- A few example “detectors” included whose processing is intended to burn CPU cycles
- Exploring migration to **TBB** (Intel Thread Building Blocks)



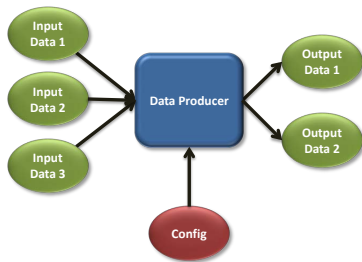
Parallel Podd Performance Scaling Benchmark

- Benchmark processing rate as function of number of analysis threads
- Run on aon11 (16 hyperthreaded cores, Intel Xeon E5-2650 v2 @ 2.60GHz), RHEL 7.9, idle
- Admittedly extreme example: maximally CPU-bound (negligible I/O & memory use)

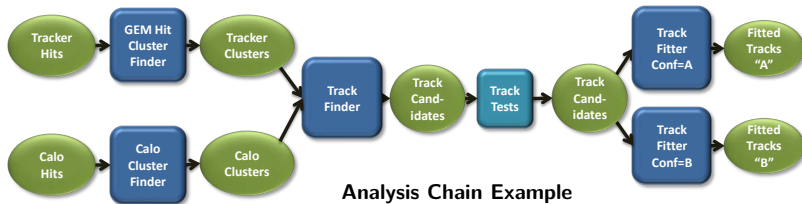


Remaining Podd Limitations

Separated Data and Algorithms



- Algorithms and Data are closely coupled
 - More work to add new algorithms
 - Difficult to stream event data only
- No native event data I/O and API
 - Podd cannot take its own output as input
 - One-pass analysis only:**
EVIO raw data → ROOT trees + histograms
 - Major limitation with large data sets
- Addressing these would require complete re-write**



Analysis Chain Example

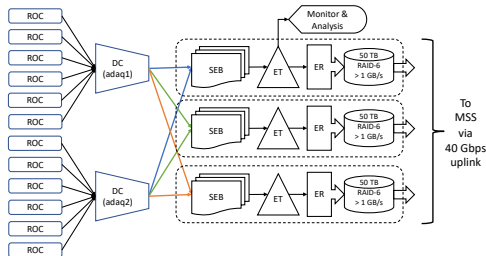
SBS Online Computing

- Traditional CODA3 DAQ for GMn: Single Event Builder host (new high-performance server), demonstrated ≥ 1 GB/s peak raw data rate
- Plan to use CODA's scalable **event stream parallelization** to achieve up to ≈ 3 GB/s
- **Online replay** on aonIX systems (128 threads), 2014-vintage servers (to be upgraded)
- SBS is the first experiment to take full advantage of these systems, running **100 automated parallel analysis jobs**.
- Online replay typically able to keep up with incoming data.

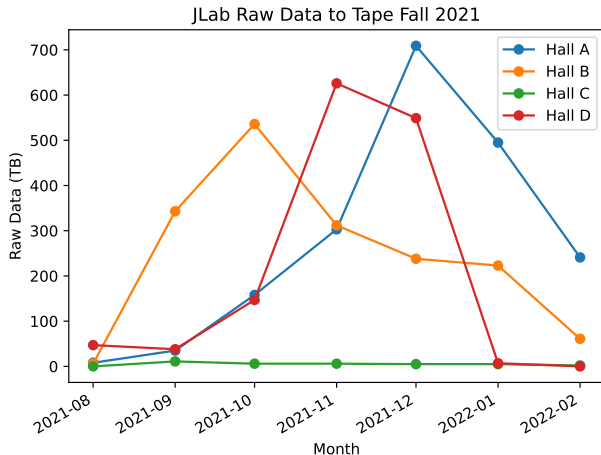
New CODA Event Builder Machines



Full SBS DAQ Configuration



SBS-GMn Data Volume in Comparison



Hall	Total (TB)
A	1,949
B	1,719
C	35
D	1,414
Sum	5,117

Fun fact #1: SBS-GMn took ≈ 5 times more data in these 6 months than all prior Hall A experiments combined in 25 years

Fun fact #2: The entire SBS program expects to accumulate ≈ 25 PB raw data through 2024

Scientific Computing Resources

- Farm/ifarm upgraded to **CentOS 7.9**. RHEL 8 clones being evaluated.
- Farm batch system has been transitioned to **slurm** and **swif2**. Legacy Auger/swif commands will stop working March 1, 2022. See <https://scicomp.jlab.org/docs/FarmUsersGuide>.
- Current farm resources
 - ▶ Disk: **Lustre: 4.1 PB**, Work: 1.4 PB (recent upgrade).
 - ▶ CPU: **14192 cores / 28384 threads**. Total capacity **249 M-core-hours/year**
 - ▶ Almost half the capacity is on **AMD EPYC 7502** 64C/128T systems (speed demons!)
 - ▶ 6 nodes with Nvidia TitanRTX GPUs dedicated for ML applications
- Mass storage system (as of Feb 2022)
 - ▶ Throughput \approx **8 GB/s** (20 LTO-8 drives, uncompressed, theoretical)
 - ▶ \approx 150 PB capacity (LTO-8, uncompressed), \approx 85 PB used (23.4 raw, 26.7 rawdup).
 - ▶ Significant capacity headroom (more frames, LTO-9) with current silo, up to \approx 325 PB.

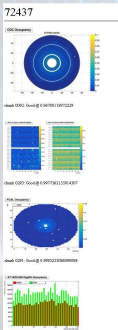
New Counting House Desktop Systems

- Clean **separation of desktops and servers**. → increased reliability and stability.
- Platform for browsers, editors, slow controls (some), remote logins
- Extensive use of **VNC** servers/clients, very successful
- No significant issues. Small updates planned (EPICS etc.)
- **Feedback welcome** (ole@jlab.org)



AI-Assisted Online Monitoring (Hydra)

- EPSCI group has offered support to deploy the Hall D Hydra system in Hall A for **automated data quality monitoring**.
- Will tap into online histograms generated by panguin.
- Currently being set up. Test version expected \approx March–April.
- One-time human review (“labeling”) required. Volunteers welcome.





72437

A portion of the real-time web dashboard of Hydra inference

Hydra

A.I. Data Quality monitoring

- Traditionally, scientists working shifts must frequently scan dozens of plots to ensure the quality of incoming data
- Plots are themselves just pictures. A.I.'s are now very good at classifying pictures.
- This is applied A.I. since it uses models already designed for image classification such as Google's Inception_v3 network
- Between 93 and 99% accurate when compared to expert labeling
 - Has found mislabeling by human experts indicating an irreducible error that is expert dependent
- Currently capable of analyzing an image in under 200ms
 - This equates to a throughput in excess of 10,000 images a day when running. (far more than a human)



(slide from David Lawrence, Jan 2021)

Summary

- “Podd” analysis software continues to be **actively maintained** and used by current experiments in Halls A & C.
- Significant **modernization** work (multithreading etc.) underway.
- The **large data volumes** from SBS are putting Hall A in the same league as Halls B & D in terms of computing resource needs. This will require **careful planning** going forward.
- Experience with the upcoming SBS mass replays on the farm will inform **future direction** of the Hall A software.