DataPrep Status

DUNE FD simulation and reconstruction

David Adams

BNL

July 18, 2016

Introduction

Developing new DataPrep module

- Extracts raw data from larsoft container and prepares for use in reco
- Writes prepared data (float signal for each tick) as recob::Wire
- Consolidate the existing 35t, FD and protoDUNE modules
  - CalWireDUNEXXX where XXX = 35t, 10kt, Dphase, …
- New code follow the TSI (Tool-Service-Interface) model
  - Algorithmic code resides in art services
  - Services inherit a service interface that declares all methods
    - Or at least those that are public and intended for normal use
  - Clients (such as StandardRawDigitPrepService) find the service via the interface name
    - So that alternate service implementations can be plugged in at run time
  - (Tool is a proposed art extension that will enable use of multiple named instances of a service or service interface)
- Work is tracked as a Redmine issue
  - https://cdcvs.fnal.gov/redmine/issues/12701

Data preparation

There are many steps to prepare data

- Extract from larsoft RawDigit container
  - Uncompress, int-to-float, subtract pedestals
  - Flag under/overflows and stuck bits
- Mitigation
  - E.g. interpolation for stuck bits
- Signal finding
- Noise removal
- Pedestal adjustment
- Deconvolution
- ROI building (signal finding)
- Construct output data product (recob::Wire)

Internal data representation

Struct AdcChannelData carries data between DataPrep tools

- See dunetpc/dune/DuneInterface/AdcChannelData.h
- Struct member are listed in table

Type	Name	Meaning
short	channel	Channel number
float	pedestal	Assumed pedestal
vector<short>	raw	Raw count for each tick
vector<float>	samples	Corrected count for each tick
vector<AdcFlag>	flags	Status for each tick
vector<bool>	signal	Indicates is each tick holds “signal”
const raw::RawDigit*	digit	Raw digit from which this data is derived

Container AdcChannelDataMap is used for multiple channels

- Type is map<short, AdcChannelData> (index is channel number)

Internal data representation (2)

AdcFlag specifies the state for each tick

- Type is short
- Intended for use by noise removal, calibration and monitoring tools
- Recognized values are listed in table

Variable name	Value	Meaning
AdcGood	0	OK
AdcUnderflow	1	Raw count is underflow (0)
AdcOverFlow	2	Raw count is overflow (4095)
AdcStuckOff	3	Raw low six bits are all 0
AdcStuckOn	4	Raw low six bit are all 1
AdcSetFixed	5	Corrected count set to fixed value (e.g. 0)
AdcInterpolated	6	Corrected count interpolated from other ticks
AdcExtrapolated	7	Corrected count extrapolated from other ticks

Services

Following slides describe the DataPrep services

- Provide the DataPrep flow
- Exchange and update AdcChannelData objects
- Service interfaces are in dunetpc/dune/DuneInterface/XXXService.h
  - XXXService is interface class name
- Service implementations in dunetpc/dune/DataPrep/Service
  - Service XXXService has header XXXService.h
    - Not needed because access is always via interface?
  - Source file for service is XXXService_service.cc
    - Naming convention required by use of cetbuildtools

StandardRawDigitPrepService

Interface: RawDigitPrepService

- Input: Raw digit vector
- Output: AdcChannelDataMap

This high-level service provides the data prep flow

- Calls the other data prep services (via interfaces)
- Configurable via FCL
  - Steps can be skipped
  - Choice of type and configuration for each low-level service

Data prep module will likely be a thin wrapper around service

- Extract raw digits
- Call this service
- Convert AdcChannelDataMap to recob::Wire
- Record latter in event

Service distinct from module can be used outside art FW

- E.g. a Root event display

StandardRawDigitPrepService (2)

Flow:

- Loop over Raw digits
  - Add AdcChannelData object to output map
  - Fill with RawDigitExtractService
  - Patch bad ticks with AdcMitigationService
  - Find signals with AdcSignalFindingService
    - This is “early signal finding,” , i.e. before noise removal and dconvolution
    - This information may be used during noise removal
- Do noise removal with AdcNoiseRemovalService
- Deconvolute signal
- Adjust pedestals with value obtained from PedestalEvaluationService
- ROI building are to be added
  - Issue: ROI building requires signal finding on deconvoluted data. It would be nice to have tools (rather than just services) so the same interface but different implementations can be used here and for early signal finding

StandardRawDigitPrepService (3)

Configuration:

- Note: Other services follow the same convention for LogLevel.
- Likely want to add flags for deconvolution and ROI building when those capabilities are added.
- In a worls with tools, we would probably replace the DoXXX flags with tool names.

Type	Name	Meaning
int	LogLevel	0: No log messages 1: Log messages only during initialization 2+: Log messages for every event
bool	DoMitigation	Patch bad ticks
bool	DoEarlySignalFinding	Do early signal finding
bool	DoNoiseRemoval	Do noise removal
bool	DoDeconvolution	Deconvolute signal
bool	DoPedestalAdjustment	Do pedestal adjustment

StandardRawDigitExtractService

Interface: RawDigitExtractService

- Input: const RawDigit&
- Output: Data for AdcChannelData
  - channel, pedestal, raw, samples, flags

Table gives the configuration parameters

Type	Name	Meaning
int	PedestalOption	1: Take pedestal from digit 2: Take pedestal from pedestal provider
bool	FlagStuckOff	Set flag if bits are stuck low
bool	FlagStuckOn	Set flag if bits are stuck high

InterpolatingAdcMitigationService

Interface: AdcMitigationService

- Input/output: AdcChannelData

Algorithm

- Samples with stuck bits (and optionally under/overflows) are updated with values obtained by linear interpolation between nearest good neighbors.

Configuration:

Type	Name	Meaning
int	LogLevel	Per convention
bool	SkipUnderflows	If true, underflows are not updated
bool	SkipOverflows	if true, overflows are not updated
int	MaxConsecutiveSamples	Maximum # consecutive samples to update
int	MaxConsecutiveFlag	Action for too many consecutive samples: 1: Leave sample unchanged. 2: Set sample to zero.

AdcSuppressSignalFindingService

Interface: AdcSignalFindingService

- Input/output: AdcChannelData&

Algorithm:

- Uses the AdcSuppressService interface to re-use the signal-finding suppression services developed for DetSim
  - Legacy35tZeroSuppressService – The code from the old DetSim
  - Dune35tZeroSuppressService – Simulation of the DUNE 35t algorithm

No configuration parameters

- Service is discovered by it type name
- Add name when service becomes a tool

Dune35tNoiseRemovalService

Interface: AdcNoiseRemovalService

- Input/output: AdcChannelDataMap&

Algorithm:

- Channels are organized into groups
  - Options to do this by regulator or ASIC
  - Also separate groups for the three wire orientations
- The BG is estimated as median of all signals for each tick and channel
- This value is subtracted from all corresponding signals
- Same algorithm and groups as the old FilterWF service
  - Used in 35t data production
- Service is specific to 35t because it uses 35-ton channel map interface
  - That provides regulator and ASIC grouping
  - Could be generalized to other detectors

Dune35tNoiseRemovalService (2)

Configuration:

Type	Name	Meaning
int	LogLevel	Per convention
int	GroupingFlag	1: By regulator (128 channels) 2: By asic (32 channels)
bool	SkipStuckCodes	Ignore channels with stuck bits in BG estimate
bool	SkipSignals	Channels identified as “signal” are not used in BG est.
bool	CorrectStuckCodes	If true, samples with stuck codes are corrected
int	ShowGroups	If nonzero, channel grouping is displayed in log 1: Organize by orientation and then group 2: Organize by group and then orientation
int	ShowGroupsChannel	Channel type used when groups are show: 0: none 1: online 2: offline, 3,4: offline by name (z1, z2 or z, Z)

DuneDeconvolutionService

Interface: AdcDeconvolutionService

- Input/output: AdcChannelData&

Algorithm:

- Uses SignalShapingServiceDUNE which folds and unfolds the signal with one provided in a histogram
  - Service is found with the usual service handle
  - Different signals for each view (z, u, v)

Configuration:

Type	Name	Meaning
int	LogLevel	Per convention

MedianPedestalEvaluationService

Interface: PedestalEvaluationService

- Input: const AdcChannelData&
- Output: 4 float*: pedestal, rms and error for each

Algorithm:

- Evaluates pedestal as median value in input samples
- Options to omit samples flagged (under/overflow, stuck bits, …) or identified as signal

Configuration:

Type	Name	Meaning
int	LogLevel	Per convention
bool	SkipFlaggedSamples	If true, flagged samples are omitted
bool	SkipSignals	if true, sample identified as signal are not used

Conclusions

New data prep service is under development

- Intended to replace the multiple existing CalWire modules
  - As before, input is raw::RawDigit and output is recob::Wire
- Most code moved from module to services following TSI
  - Including high-level service that calls the low-level services
  - Module will call the high-level service
  - Allowing one to run data prep outside of art
  - Easy to plug in different options for steps in data prep (e.g. deconvolution) by providing a new service and changing FCL to use it
- Transient data class AdcChannelData introduced
  - Used to exchange data between data prep services

Status

- Most low level services are in place
- Still need to deal with ROI building (somehow reuse signal finding?)
- Need to write the module wrapper
- Need to test and develop FCL for FD and protoDUNE

Conclusions (2)

Tools

- Tool = service that accessed via instance name instead of class name
  - Allows us to use multiple instances of a “service” type in a job
  - OO programming
- In many cases it would be easier if we had tools instead of services
  - E.g. different services acting on AdcChannelData& could have the same interface
  - Re-use signal finders for ROI building
- Best to do this at the art level
  - Probably small change in art code
  - But big change in POV (point of view)?