Context switching tools (and services)

DUNE SW Architecture

David Adams

BNL

February 17, 2023

Motivation (1)

DUNE code makes use of conditions data

- Detector conditions change with time
  - Different CRPs tested in CERN coldbox in 2021-2022
  - TPC readout channels went and bad and recovered during ProtoDUNE SP
  - Varying temperature, voltage, purity, …
- Conditions DB intended to provide access to these data
  - But this has been slow in providing all the data we want
  - Robustness and scaling have not yet been tested
  - For slowly-varying data like channel status, it may be preferable to read conditions data from files
    - Those files might be plain text, fcl or SQLite
  - May even want a mix with conditions taken from DUNE DB for one run period and from files for another
    - E.g. recent vs. old data
    - Or development vs. production
- Conditions-dependent data is also useful for analysis
  - And so should not require user to be inside art event loop

Motivation (2)

May want choice of algorithms to vary with data conditions

- For example
  - Beam on or off
  - Trigger
  - Pulsers or lasers on
  - Missing detector elements e.g., TPC plane

Like to avoid proliferation of top-level fcl configs

- Confusion about which to use or was used for a particular event
- Fixes to one may not be propagated to others

Definition

I use the label context

- To refer to the metadata used to select appropriate conditions data or algorithms
- Classic examples are run and event number
  - But these are not unique for all DUNE prototype data
  - And data can be taken without these labels
- Others that help distinguish are
  - Location: np02/4, Iceberg, CERN coldbox
  - Detector configuration: CRP2/3, verical/horizontal readout, …
  - Time
- Some context might be derived from the above
  - Beam conditions, trigger, …

Proposed solution(s)

1. Make tools and services explicitly context dependent
   - Require caller to pass context or obtain it from a context manager
   - In both cases, we would like to have a generic interface for context
     - If we add to context, we don't have to change all the existing clients
2. Add a context-redirecting tool
   - I.e., a tool that returns the name of the tool appropriate to the current context when it is dereferenced
     - The switching tool does not have the interface of the returned tool
     - One class: do not have to replicate the code for each tool type
   - Context-switching services can be provided as wrappers around such tools
     - We *will* need a separate class for each service interface

Highlighted items above are the topics of this talk

Old stuff

Tools and services (review of concepts)

Art service

- Dynamic lookup
  - Caller is compiled only against the service interface
- Configured from fcl
  - Fcl reference is by interface type name so effectively singleton
    - Only one service of a given interface can be used in a job
- Offers callbacks for change of run or event (documented in header)

Art tool

- Dynamic lookup
- Configured from fcl
  - Art model is to include the tool fcl configuration inside the configuration of the object using the tool
    - Tool is referenced by the name assigned within the object
  - DUNE/dataprep adds a tool manager so tools can be referenced by a global name: named tool
    - And so can be shared by multiple objects and across events

Overview of dataprep (1)

Dataprep module(s)

- See DataPrep or DataPrepByApa in dunedataprep/DataPrep/Module
- Called once per event from the art event loop
- Module uses RDP (RawDigitPrepService) to process data
  - RDP holds a list of dataprep tools to be run in sequence
  - Begin and end of event notifications are forwarded to the tools
  - Method prepare is used to process TPC data
    - Passed data includes
      - » ADCM = map of AdcChannelData.h objects
      - » recob::Wire container to hold output data
    - ADCM is passed to each tool in turn which updates these objects
    - Optionally the wire container is filled from the ADCM

Overview of dataprep (2)

Each event, the the dataprep module

- Reads from the event store: event info, trigger info, beam data
- Creates DEI object holding run ID, event ID, time and trigger info
  - DEI class: DuneEventInfo.h
- Optionally creates the output recob::Wire container
- Notifies the RDP of the new event (passes the DEI)
- Loops over detector regions (e.g. APAs)
  - Reads the raw data for the region
  - Constructs ADCM from DEI and raw data
  - Calls RDP to prepare the TPC data (see preceding page)
- Notifies RDP of end of event
- Writes the recob::Wire container to the event

New stuff

Context

Context interface

- An abstract in interface for context is added:
  dunecore/DuneInterface/Data/DuneContext.h
- Virtual methods to return run and event IDs
  - Add time?, subrun?, run period name?

Existing implementation

- Above interface was added to the dataprep DEI
  dunecore/DuneInterface/Data/DuneEventInfo.h

Context manager

Context manager

- Manager is a singleton where one can discover the current context
  - Manager class → instance → current context
  - Might also provide callback to notify registrant when context changes
- Implementation at dunecore/DuneCommon/Utility/DuneContextManager.h
  - Callback is not (yet) provided
  - Not (yet) thread-safe. A plan for multithread managers is discussed later

Tool redirection

An interface for a redirecting tool has been added

- See dunecore/ArtSupport/ToolRedirector.h
- The class provides a static method *isRedirecting(fcl)* that returns true if the fcl config includes the parameter name *tool_redirector*
- In that case, the tool can be retrieved as this type and the method *getName()* returns the name of the redirected tool

Concrete redirecting tool

- To be redirecting, a tool must be a subclass of ToolRedirector and override the method *getName().*
- To create a context-redirecting tool, call the context manager in that method and return the corresponding tool name
  - Or use the helper class described on the following page

The tool manager has been modified accordingly

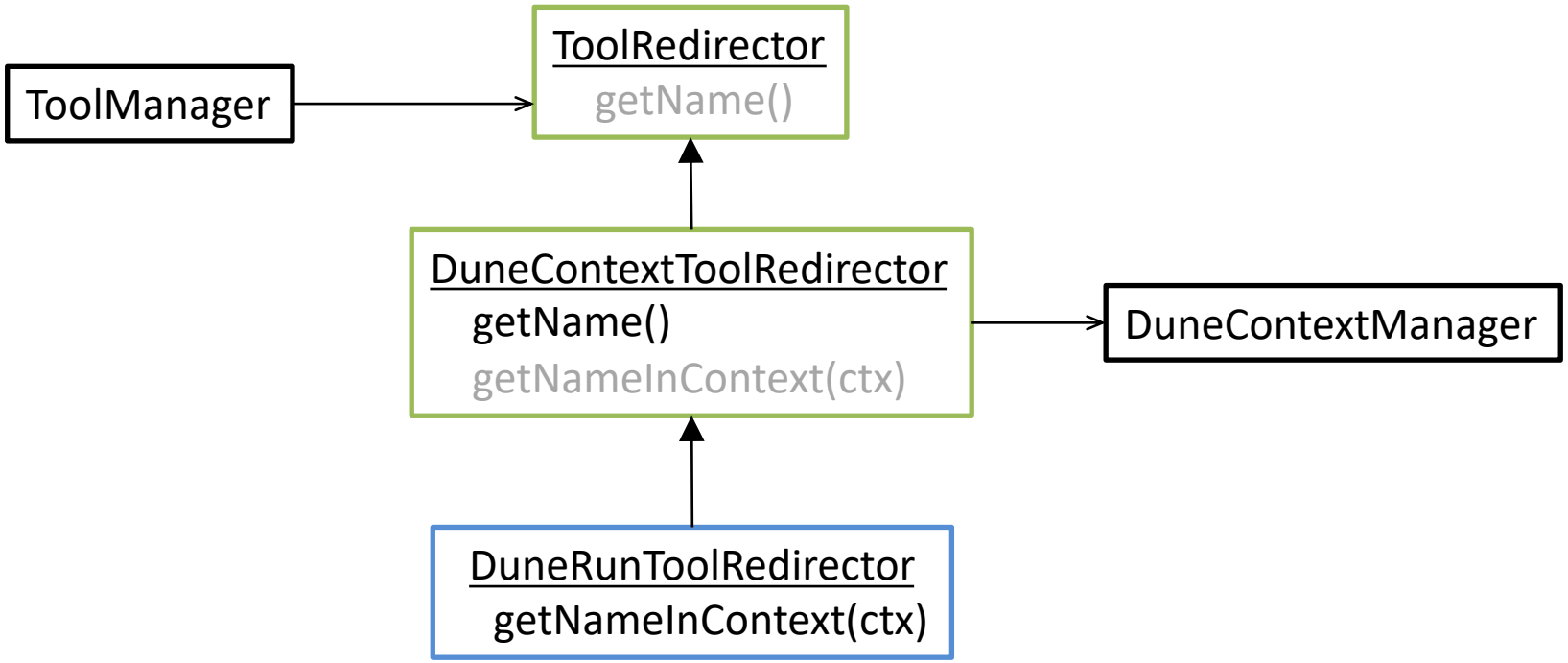
- When presented the redirecting tool name, it returns the redirected tool

Context redirection helper

There is a context redirection helper

dunecore/DuneCommon/Utility/DuneContextToolRedirector.h

- Inherit from this and implement *getNameInContext(ctx)* to provide a tool that redirects according to context *ctx*
- Diagram below shows how these classes and the one on the following interact



Run redirection

A tool redirecting based on run number is implemented

dunecore/DuneCommon/Tool/DuneRunToolRedirector.h

- As indicated in the figure, it inherits from *DuneContextToolRedirector*
- An example configuration is shown below.
  - For example, any run in the range [1373, 1623) will return the name "chanstat_vdcrp2_jul2022"

```
# Tool for any vdct (vertical-drift top-electronics coldbox) run.
tools.chanstat_vdct: {
  tool_type: DuneRunToolRedirector
  tool_redirector: true
  LogLevel: 1
  DefaultName: chanstat_allbad
  Ranges: [
     "401:1037:chanstat_vdcrp1t_dec2021",
    "1037:1373:chanstat_vdcrp1t_apr2022",
    "1373:1623:chanstat_vdcrp2_jul2022",
    "1623:1798:chanstat_vdcrp3_oct2022",
    "1857:1899:chanstat_vdcrp2_nov2022"
  ]
}
```

Context-switching service

A context-switching service has been implemented:

dunecore/DuneCommon/Service/ToolBasedChannelStatus

- This is a service implementing the *ChannelStatusService* interface that provides a wrapper around a tool of type *IndexMapTool*
- An example configuration using the *chanstat_vdct* tool on the preceding page is below.
  - Because the tool is redirecting based on run, the service switches tools based on run

```
data.ChannelStatusService_vdct: {
  service_provider: "ToolBasedChannelStatusService"
  LogLevel: 1
  NChannel: 3200
  ToolName: chanstat_vdct
}

services.ChannelStatusService: @local::data.ChannelStatusService_vdct
```

Changes to come (or not)

Set context in DUNE jobs

The context manager is in the DUNE release

- But we need a mechanism to set the context
  - Set the the run and event number, etc. each time a new event is read in

Options for setting the context

- Add event context service to services
  - Run and event only

```
services.EventContextService: {
  service_provider: EventContextService
  LogLevel: 1 }
```

- Update dataprep modules to set context
  - They already build context each event
  - Includes run, subrun, event, time and trigger clock
- Create and schedule new dedicated module
- Set context in the data input service

Add DUNE DB tools

We should create tools using DUNE conditions DB

- Try out that system
- Is there any data available now?
  - Maybe run conditions was discussed at the collab meeting
  - Instruction for access?
- Like to compare performance with file-based data
  - Tools should have the same interface
- Design
  - Use context manager to get run and event
  - Get data from conditions DB or cache
    - Cache results until context switches
    - Same MT issues a for any stateful tools
  - Return piece of data requested by user

Context switch notification

Redirected tools are not context switching

- If a tool pointer is cached after retrieval with a context-redirecting config, the cached reference will not change with context change
  - In general, a tool of any type could be context-redirecting and so have to worry about this for any named tool.
- We can use ToolRedirector::isRedirecting(name) to check a tool name and safely cache if not.
  - It would be nicer to get this information directly from the tool manager.
  - Maybe do not allow redirecting unless user indicates they can handle it

If a tool was obtained by redirection:

- Safest is to not cache. Go back to the tool manager and retrieve the tool with the the tool name each time a tool is used.
  - This can be expensive, e.g., *ChannelStatusService* is often called one channel at a time
- Safe to cache within a function where we know the context will not change. This is generally the case.
- We can record the context and check on each call but this is now difficult and, in any case, should be automatic

Running in python

Want to get tools and maybe services working in python

- Root dictionaries exist for most dataprep tools
- Need to add those for the new components here
- And demonstrate use

Context-dependent detector properties

Field strength can change during CRP coldbox runs

- Changes in max drift distance
- Tests with varying HV
- We would like to have context-dependent field strength
- But reco obtains this from DetectorPropertiesService
  - Fixed fcl config of values
  - Many other properties also
  - Used throughout reco and probably do not want to use another interface to fetch the field strength. Right?

Somehow add context dependence to this service

- Addresed in dunesw issues 53

Multithreading

State and MT

State

- Most C++ objects carry data
- If the data in an object (or anywhere else) are modified in one thread, then their values seen in other asynchronous threads can be unpredictable and likely lead to incorrect behavior
- Use *state* to denote data that may be modified after an object is constructed
  - An object which has such data is *stateful*
  - And one that does not is *stateless*
- In an MT environment, stateful objects can cause problems
  - Easy solution is to avoid these but we often want state
    - Counters, merging data from multiple events, …
  - Another solution is to synchronize: make sure state is set before accessing threads are created and does not change during their lifetime
    - We use this to handle state for managers
  - Another is to create copies of the objects for each thread
    - This is the proposal discussed later for stateful tools

Managers and MT

Singleton managers with state will have problems with MT

- Context manager carries state (it exists only for that purpose)
- Tool manager has state for caching shared tools
  - And tools themselves can carry state (see following page)

Plan for addressing this

- Manager instance() returns object for the current thread
- Class holds fixed-length arrays mapping thread ID to instance
  - This likely has to be in the main thread
- Need to know when a thread exits so we can free up that slot
- Want to merge state info into main or parent instance when a manager instance terminates
- May need/want a thread manager for the above
  - Also useful for tracking the total number of threads

Named tools and MT

Stateless tools have no issues in MT

- If they are fully constructed before being accessed
- Many or most tools fall in this category

However we would like to have stateful tools

- Counters: number of calls, errors, …
- Histograms or plots with data from multiple events
- Caching of conditions data
  - Though much of this can be avoided with context-redirected tools

Proposal for handling state

- Thread-local tool manager returns thread-local tools
- Add mechanism to merge the states from all tools with a given name

Summary

Summary

We have too many top-level fcl configs

- Same algorithm sequence but different conditions data

One solution is to make context-aware tools and services

- Context is the keys used to select condition data
  - E.g. run and event numbers
- Use context manager so we don't have to pass context

Another is context-redirecting tools

- Delivered tool covers a limited context range
- Redirecting tool specifies the tool for current context
- Example: channel status for vdtcb (vertical-drift top-readout coldbox)

Many possible directions for future development

- But useable code in place now
- Need to add and test fcl for vdtcb
  - See my talk at the vertical-drift analysis meeting

Thank you