Adaptable data-processing for HEP computing frameworks

https://indico.fnal.gov/event/52666/contributions/231769/attachments/153101/198580/SCDProjects_LDRD_Knoepfel.pdf

Kyle J. Knoepfel

DUNE/LDRD monthly meeting (Kick-off)

13 April 2022

Establishing expectations

- **Goal of LDRD is to explore solutions for DUNE’s framework needs**
  - Monthly meeting to ensure that LDRD directions are headed in the right direction
  - Hoping to keep this meeting “small” in attendance

Establishing expectations

- **Goal of LDRD is to explore solutions for DUNE's framework needs**
  - Monthly meeting to ensure that LDRD directions are headed in the right direction
  - Hoping to keep this meeting "small" in attendance
- ***However…***
  - The scope is broader than just DUNE.
  - I request your ideas/guidance, but the project is still self-directed.
  - The deliverable is not a new framework.
  - The deliverable is not to adjust an existing framework.
    - This is desirable and I'll keep it in mind, but it is not the immediate goal.
  - The outcome will inform how to adjust existing frameworks.

My approach

- Use community-provided (i.e. non-HEP specific) software

External software	Provided by Spack or system
Building and testing	CMake
Configuration	Boost JSON (probably Jsonnet later; not FHiCL or Python)
Plugin handling	Boost DLL
Task scheduling	Intel TBB (yes, MPI is on the table)

Fermilab

My approach

- Use community-provided (i.e. non-HEP specific) software

External software	Provided by Spack or system
Building and testing	CMake
Configuration	Boost JSON (probably Jsonnet later; not FHiCL or Python)
Plugin handling	Boost DLL
Task scheduling	Intel TBB (yes, MPI is on the table)

- Prototype package https://github.com/knoepfel/sand/

  It will change ***a lot*** in the months ahead.

The basic idea

- The framework should support experiment-defined processing levels.

  Inheritance-based modules don't necessarily work well.

- `sand` uses template metaprogramming to process data of the "right type".

  This may change.

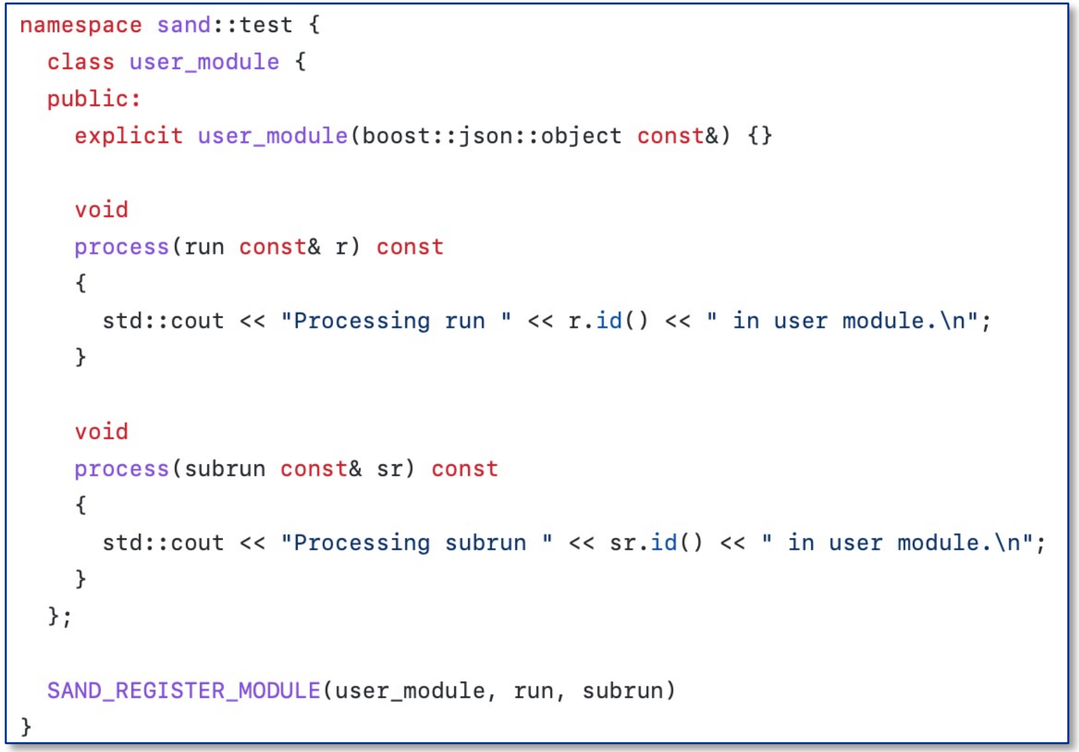
The basic idea

- The framework should support experiment-defined processing levels.

  Inheritance-based modules don’t necessarily work well.

- sand uses template metaprogramming to process data of the “right type”.

  This may change.



```
namespace sand::test {
  class user_module {
  public:
    explicit user_module(boost::json::object const&) {}

    void
    process(run const& r) const
    {
      std::cout << "Processing run " << r.id() << " in user module.\n";
    }

    void
    process(subrun const& sr) const
    {
      std::cout << "Processing subrun " << sr.id() << " in user module.\n";
    }
  };

  SAND_REGISTER_MODULE(user_module, run, subrun)
}
```

Thus far (< 1 month)

- Some coding
- Discussions with individuals (Tom, Andrew, Wes Ketchum, Barnali Chowdhury)

  Barnali and I are meeting somewhat regularly to discuss HDF5
- Looking through DUNE's workflow (thanks, Tom, for the FHiCL files)

Thus far (< 1 month)

- Some coding
- Discussions with individuals (Tom, Andrew, Wes Ketchum, Barnali Chowdhury)

  Barnali and I are meeting somewhat regularly to discuss HDF5
- Looking through DUNE’s workflow (thanks, Tom, for the FHiCL files)
- **Finding: It’s not clear yet what the data flow should look like.**

  Sometimes we want to decompose a trigger record into to APAs and then reassemble them.
  - What is the relevant “path” for this use case? Is it at the trigger record or APA level? Maybe both?
  - Would this use case be more easily supported with an on-demand system instead of a scheduled one?
  - What does it look like in code to “reassemble” a trigger record without blowing the memory budget?
- I will be working through these types of issues.