Memory and data groupings

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

Kyle J. Knoepfel

DUNE/LDRD monthly meeting

13 July 2022

The primary thrusts of my LDRD

- **Deliverable 1 is to find a framework design that supports the processing of flexible (i.e. user-defined) data groupings.**

  Requires a data model that is less rigid that *art*'s (under exploration)

  Should support an interface that can easily support DUNE concepts (e.g. APAs)

The primary thrusts of my LDRD

- **Deliverable 1 is to find a framework design that supports the processing of flexible (i.e. user-defined) data groupings.**

  Requires a data model that is less rigid that *art*'s (under exploration)

  Should support an interface that can easily support DUNE concepts (e.g. APAs)

  But also can lead to better memory usage…

The primary thrusts of my LDRD

- **Deliverable 1 is to find a framework design that supports the processing of flexible (i.e. user-defined) data groupings.**

  Requires a data model that is less rigid that *art*'s (under exploration)

  Should support an interface that can easily support DUNE concepts (e.g. APAs)

  But also can lead to better memory usage…

The *art* way

```
void process(art::Event& e)
{
  for (auto const& apa : all_apas()) {
    process(apa);
  }
  write_products(e);
}
```

User code

The primary thrusts of my LDRD

- **Deliverable 1 is to find a framework design that supports the processing of flexible (i.e. user-defined) data groupings.**

  Requires a data model that is less rigid that *art*'s (under exploration)

  Should support an interface that can easily support DUNE concepts (e.g. APAs)

  But also can lead to better memory usage…

The *art* way

```
void process(art::Event& e)
{
  for (auto const& apa : all_apas()) {
    process(apa);
  }                                  User code
  write_products(e);
}
```

The better way

```
void process(art::Event& e)
{
  for (auto const& apa : all_apas()) {
    process(apa);                    User code
    write_products(apa);
  }
}
```

The big picture

- **For dealing with memory usage…**

  The framework can provide an API for mitigating memory usage (e.g. removing transient data products from memory)

  I would rather provide a framework solution that can accommodate a construct that is meaningful to the experiment (e.g.):

  1. A configuration notifies the framework that a trigger record should be decomposed into data from APAs.
  2. The framework then understands that data created from APAs can be written to files/flushed from memory while processing APAs; it need not wait to write/flush until the full trigger record is processed

The big picture

- **For dealing with memory usage…**

  The framework can provide an API for mitigating memory usage (e.g. removing transient data products from memory)

  I would rather provide a framework solution that can accommodate a construct that is meaningful to the experiment (e.g.):

  1. A configuration notifies the framework that a trigger record should be decomposed into data from APAs.
  2. The framework then understands that data created from APAs can be written to files/flushed from memory while processing APAs; it need not wait to write/flush until the full trigger record is processed

- **This makes sense to me, in principle. But…**

My concerns

- To identify code that could be simplified with a new framework approach, I looked for explicit loops over (e.g.) geometrical constructs like APAs.

My concerns

- **To identify code that could be simplified with a new framework approach, I looked for explicit loops over (e.g.) geometrical constructs like APAs.**

- **I found very few.**

  This could be evidence that:

  1. I'm not looking in enough places
  2. The very few loops matter very much
  3. The problem isn't as severe as I thought it was

  I think it's more a combination of 1 and 2 than 3.

My concerns

- **To identify code that could be simplified with a new framework approach, I looked for explicit loops over (e.g.) geometrical constructs like APAs.**

- **I found very few.**

  This could be evidence that:

  1. I'm not looking in enough places
  2. The very few loops matter very much
  3. The problem isn't as severe as I thought it was

  I think it's more a combination of 1 and 2 than 3.

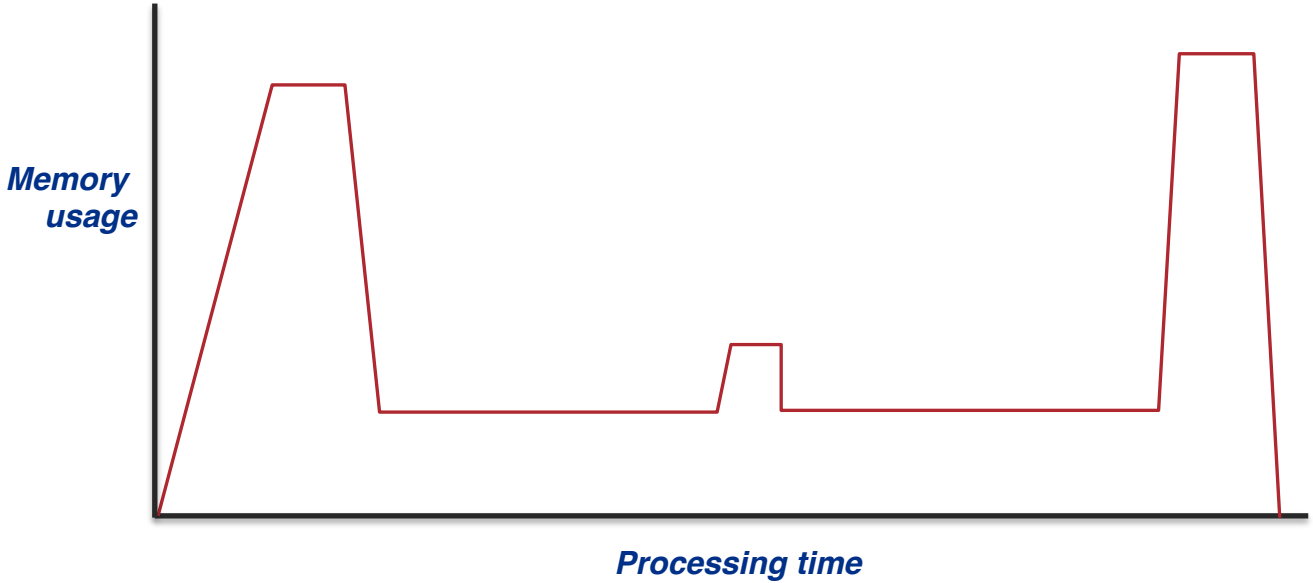
- **It is not worthwhile to pursue a solution that will have little impact…**

My concerns

- **Example:** you can’t optimize in a vacuum

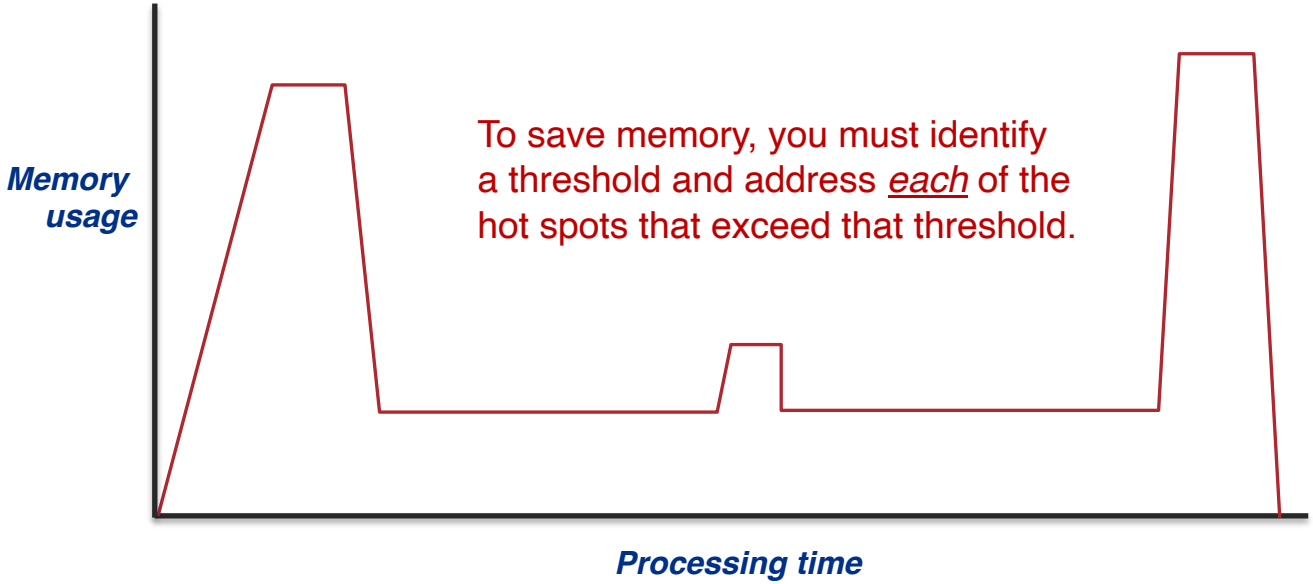
My concerns

- **Example:** you can't optimize in a vacuum



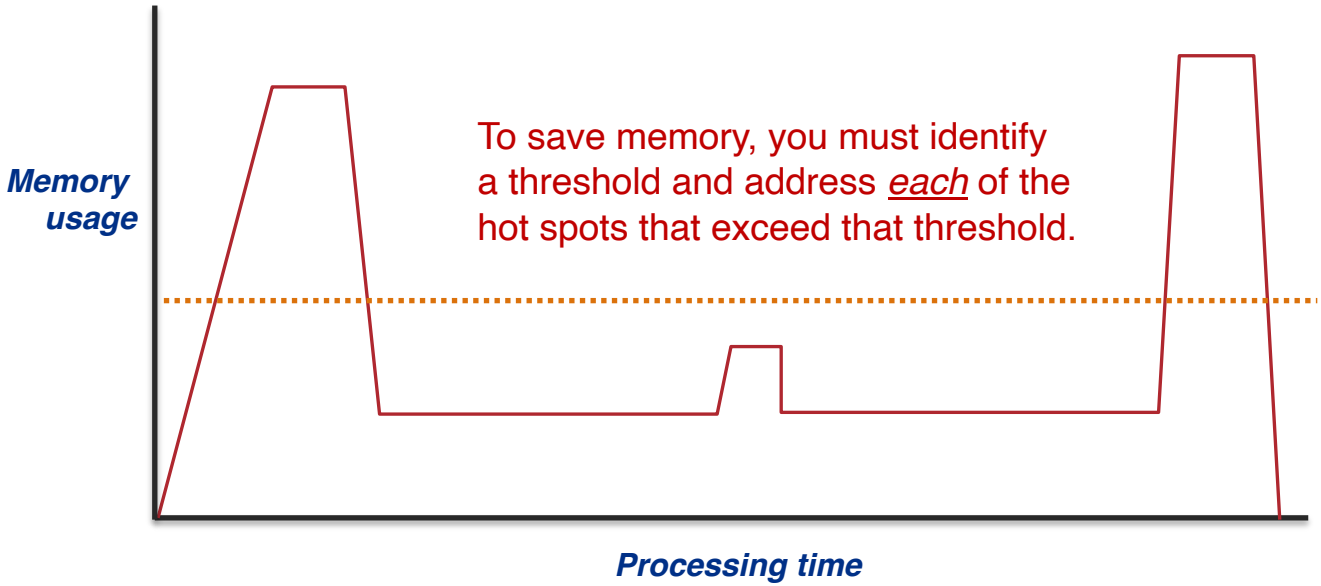
My concerns

- **Example:** you can't optimize in a vacuum



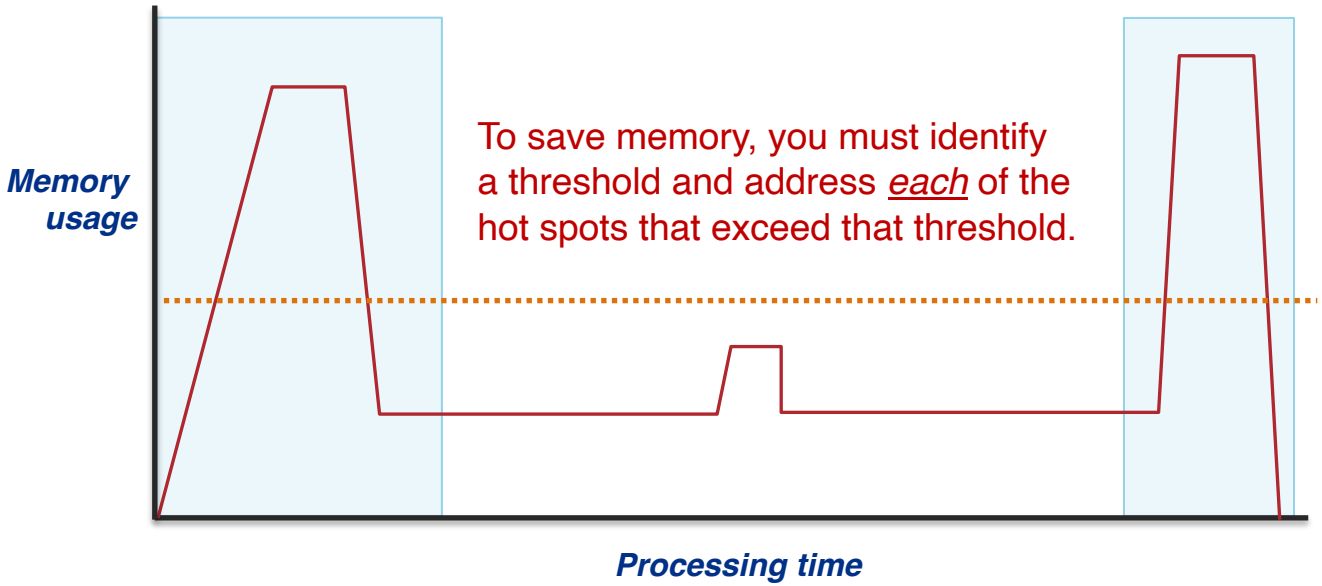
My concerns

- **Example:** you can't optimize in a vacuum



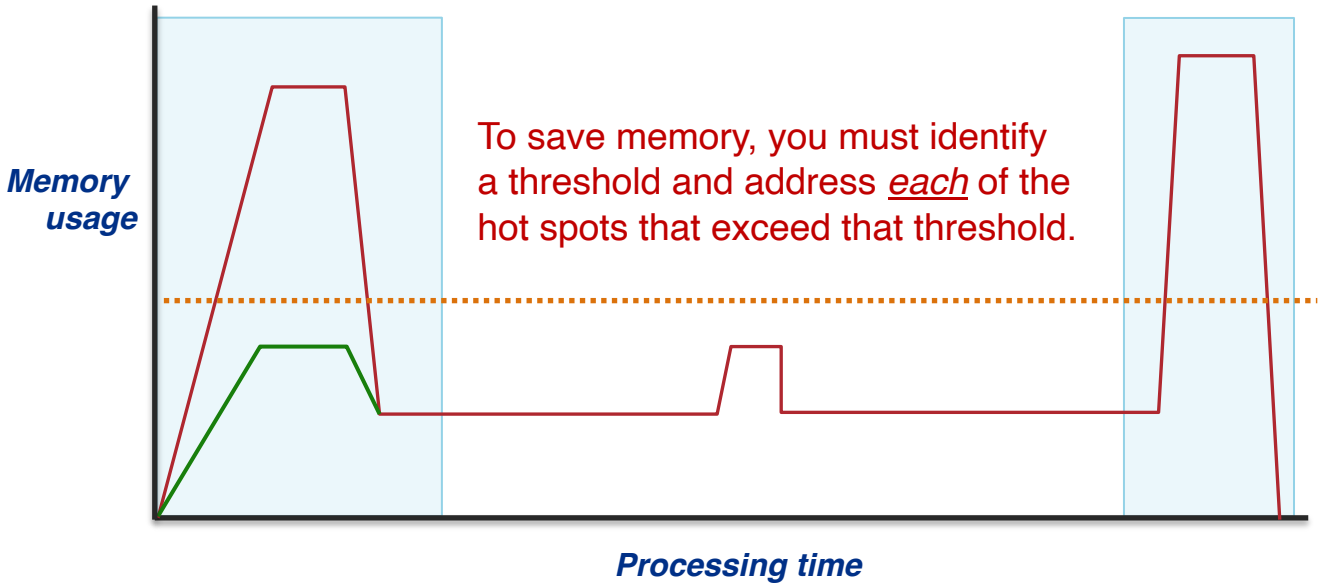
My concerns

- **Example:** you can't optimize in a vacuum



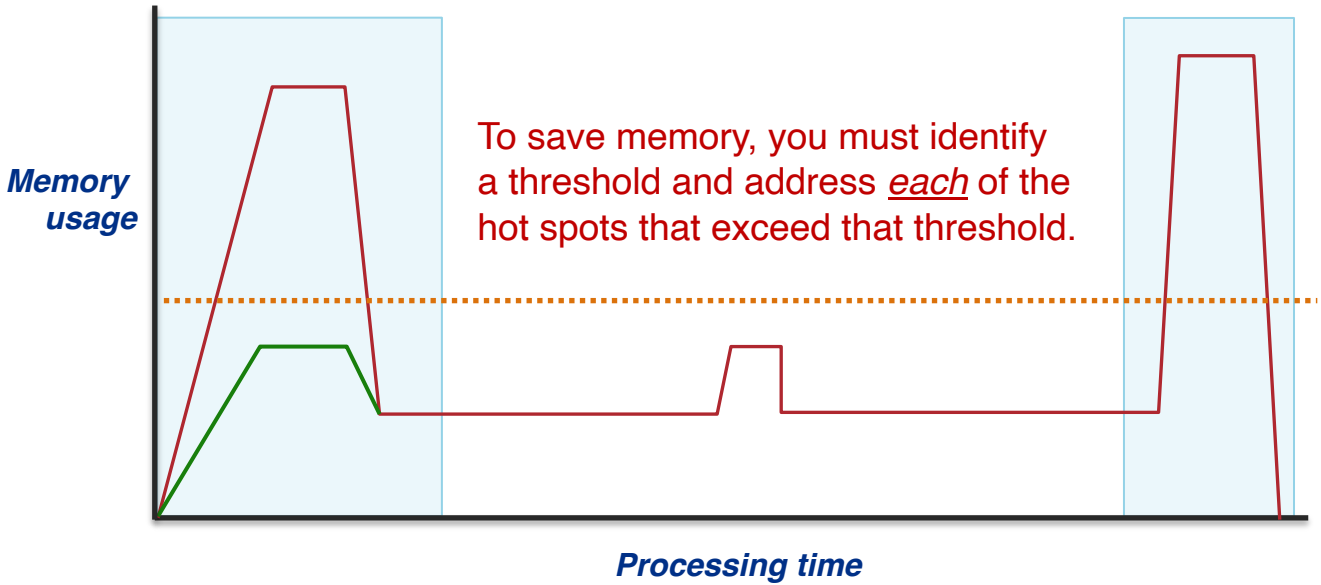
My concerns

- **Example:** you can't optimize in a vacuum



My concerns

- **Example:** you can't optimize in a vacuum



- I do not want to find a solution that can address only part of the problem.

To avoid a premature solution…

- We need a reasonably complete understanding of memory usage for each of the workflows that will be used with the framework.

  There are existing workflows using *art* that can help inform this, and I will look at these soon:

  - `prodgenie_nu_dune10kt_1x2x6.fcl`
  - `standard_g4_dune10kt_1x2x6.fcl`
  - `standard_detsim_dune10kt_1x2x6.fcl`
  - `standard_reco_dune10kt_1x2x6.fcl`

- Do you have other ideas?