Updating HPS Readout

Introduction

- The current HPS readout framework is not ideal.

Introduction

- The current HPS readout framework is not ideal.
  - We use an event loop system, but employ drivers that depend on data across many events.

Introduction

- The current HPS readout framework is not ideal.
  - We use an event loop system, but employ drivers that depend on data across many events.
  - The current framework was designed for the old 2014 readout system, and is not an intuitive, natural structure for the current readout system.

Introduction

- The current HPS readout framework is not ideal.
  - We use an event loop system, but employ drivers that depend on data across many events.
  - The current framework was designed for the old 2014 readout system, and is not an intuitive, natural structure for the current readout system.
  - The current system is very awkward for developing any driver that depends on more than one system.

Proposal Framework

- Consider a new framework consisting of “readout drivers” which all connect to a central “data management module.”

Proposal Framework

- Consider a new framework consisting of “readout drivers” which all connect to a central “data management module.”
- A readout driver performs several functions:

Proposal Framework

- Consider a new framework consisting of "readout drivers" which all connect to a central "data management module."
- A readout driver performs several functions:
  - Declares a "local time offset" – the amount of time between the time an object is produced versus its real time.
    - For instance, clusters are offset by an amount of time equal to half the clustering time window. Thus, a cluster with a seed hit with time $t = x$ ns will actually be produced at time $t = (x + t_{\text{window}})$ ns.

Proposal Framework

- Consider a new framework consisting of "readout drivers" which all connect to a central "data management module."
- A readout driver performs several functions:
  - Declares a "local time offset" – the amount of time between the time an object is produced versus its real time.
  - Declares which collections upon which it depends.
    - This allows the management driver to know the total time offset of the driver's output.
    - For instance, clustering depends on calorimeter hits. If calorimeter hits have an offset of t_{hits}, then the total offset for clusters is $t_{\text{hits}} + t_{\text{window}}$.

Proposal Framework

- Consider a new framework consisting of “readout drivers” which all connect to a central “data management module.”
- A readout driver performs several functions:
  - Declares a “local time offset” – the amount of time between the time an object is produced versus its real time.
  - Declares which collections upon which it depends.
  - Declares which collections (and their object types) that it produces.
    - This allows the management driver to know which collections to track and what the time offsets on them will be.

Proposal Framework

- Consider a new framework consisting of “readout drivers” which all connect to a central “data management module.”
- A readout driver performs several functions:
  - Declares a “local time offset” – the amount of time between the time an object is produced versus its real time.
  - Declares which collections upon which it depends.
  - Declares which collections (and their object types) that it produces.
  - Creates readout objects and passes them to the data management driver.
    - Readout drivers will request the data they need in a certain time range from the management driver.
    - Drivers may further output their data as appropriate to the management driver for use by other readout drivers.

Proposal Framework

- This solves a number issues:

Proposal Framework

- This solves a number issues:
  - Removes the need for drivers to each separately buffer data.
    - All data is buffered in the management module.
    - Drivers get a list of a needed collection in a defined time range.
    - Drivers do not directly depend on the somewhat unnatural (for this usage case) event structure directly.

Proposal Framework

- This solves a number issues:
  - Removes the need for drivers to each separately buffer data.
  - Removes awkward code structure.
    - The current code structure makes sense for the 2014 trigger system, but is not natural and difficult to follow in the present system.
    - New version is much more straightforward, and provides a simpler framework for development.
    - Improves code readability and maintainability going forward.

Proposal Framework

- This solves a number issues:
  - Removes the need for drivers to each separately buffer data.
  - Removes awkward code structure.
  - Removes need for drivers to know about other drivers.
    - The management module knows the time offsets of all collections, so individual drivers only need to poll it to see if a collection is available in a given time range.
    - This allows drivers to be truly independent of each other, and to be modified without affecting other drivers. A driver need only be responsible for declaring its own behavior.

Proposal Framework

- This solves a number issues:
  - Removes the need for drivers to each separately buffer data.
  - Removes awkward code structure.
  - Removes need for drivers to know about other drivers.
  - Modifying collection output is simplified.
    - The current readout driver is quite convoluted.
    - The management module can be designed with more straightforward output behavior to simply the addition of new collections to readout, like hodoscope hits, GTP clusters, or truth information.

Conclusion

- The current readout system is poorly documented and not well suited to the modern readout structure.
  - This induces issues with modifying readout output to include new systems, like the hodoscope, or new components, like truth information.
  - It also induces problems with linking multiple systems with different time offsets together.
- A modified, streamlined, and centralized system can alleviate the structural problems and remove the time offset issue.
  - A central management module can learn and account for time offsets for each driver without each driver needing to know these itself.
  - Much of the convolution comes from legacy 2014 behavior support, which can be safely removed for a more structurally appropriate system now.