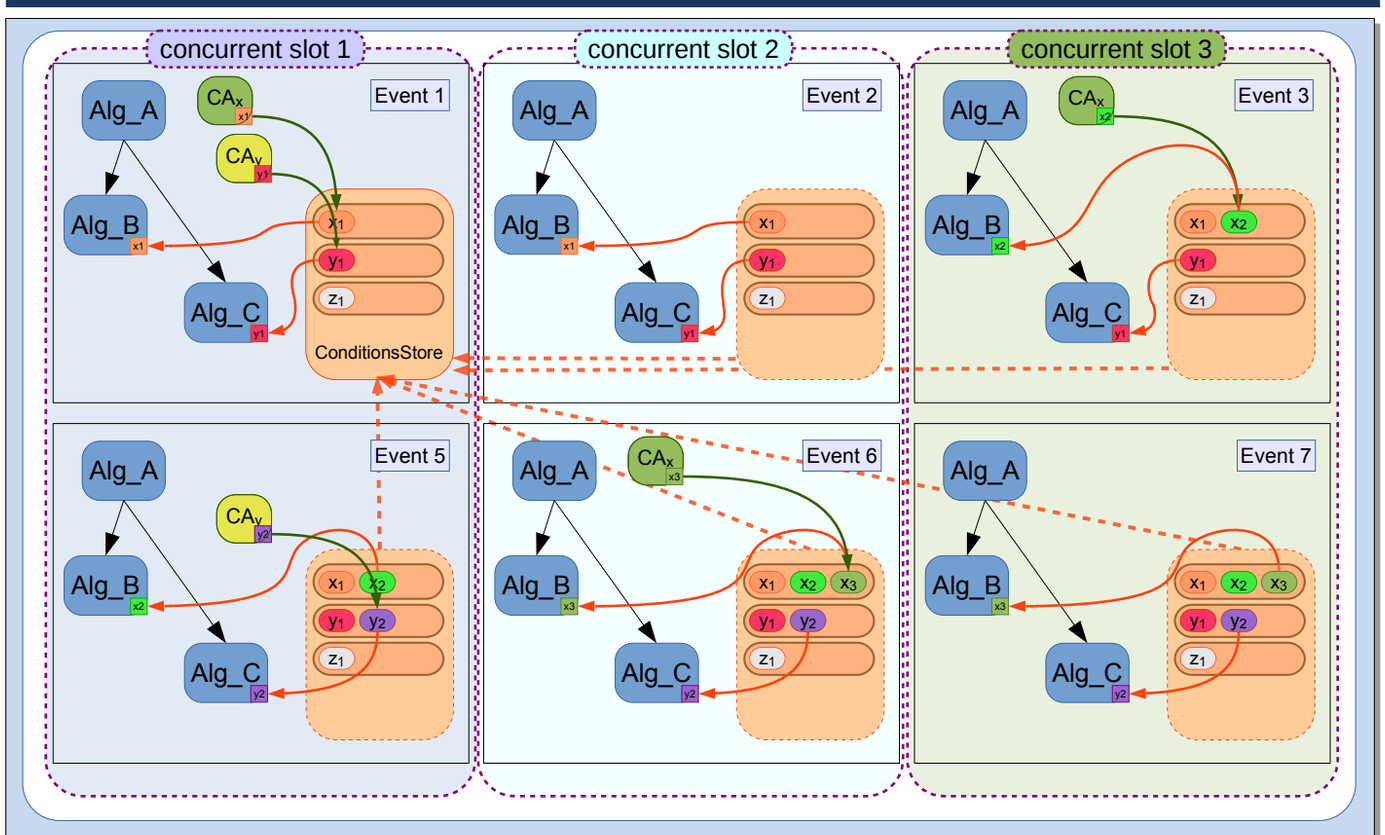
CHEP 2018 Poster Session - July 9 2018

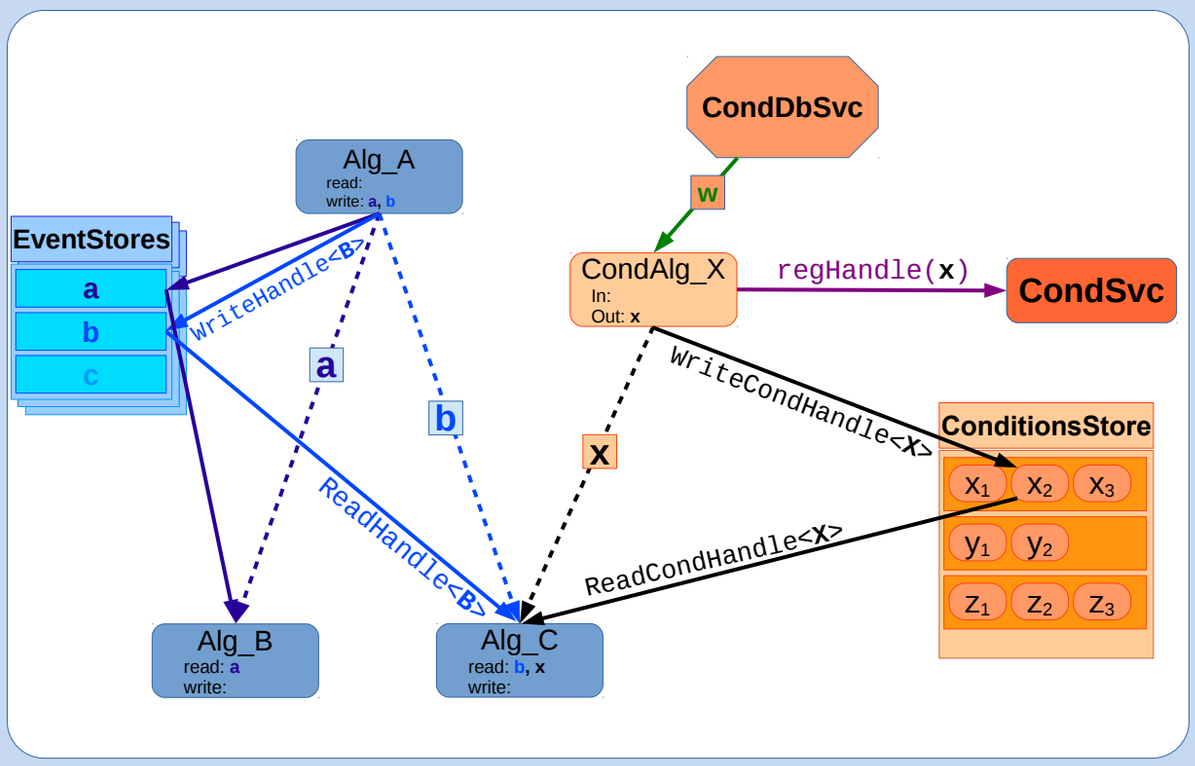
Conditions Data Handling In The Multithreaded ATLAS Framework

Multi-cache Conditions Store



- Conditions data is stored in containers indexed by name and type in the ConditionStore, and ordered as one entry per IOV.
  - Conditions data of type **T** stored in **`CondCont<T>`**
  - Implemented as a ConcurrentRangeMap, allowing efficient lookup with no locking, and locked writing with concurrent reading

Condition Handles



- Algorithms use smart references (DataHandles) to interact with all event data.
- ConditionHandles are special instances that use the current event information to point to the appropriate entry in the ConditionsStore.
  - Conditions data of type **T** referenced via **`CondHandle<T>`**
  - Can be accessed via derived or base types

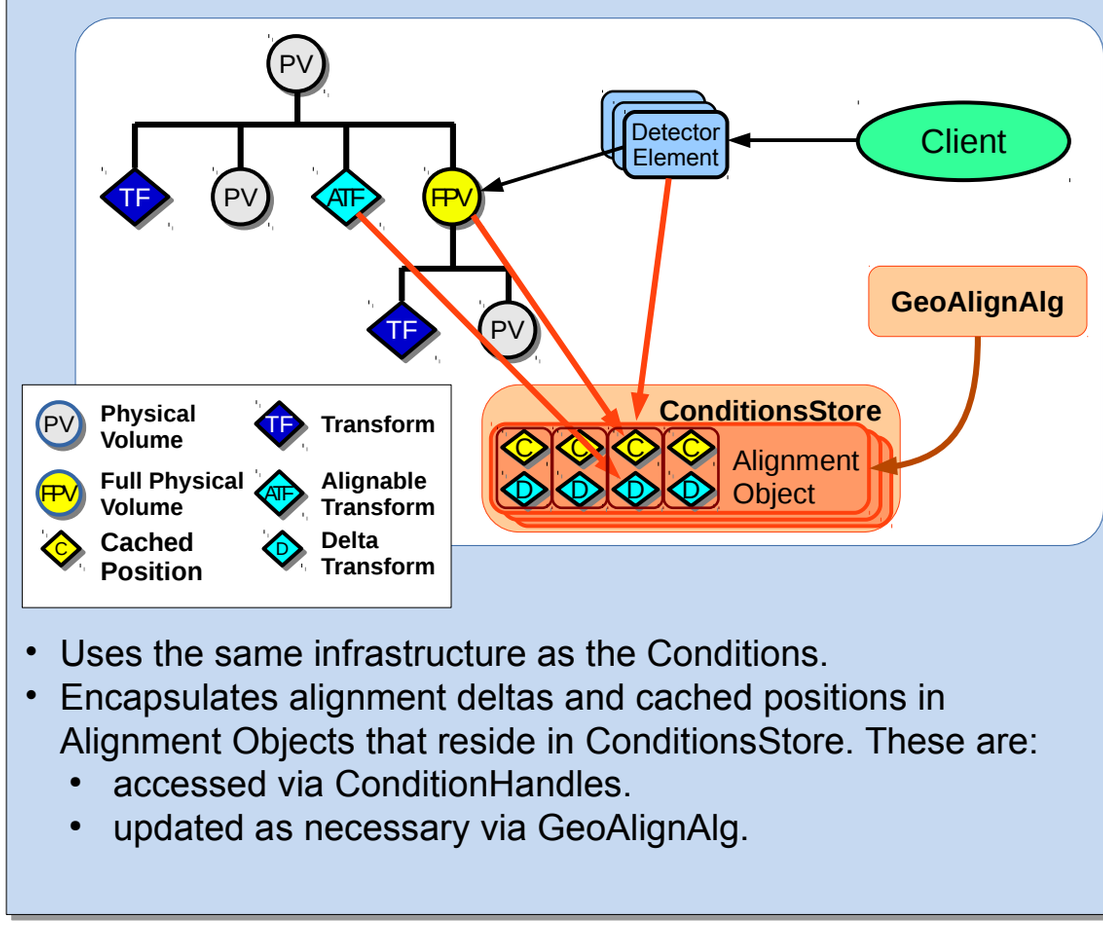
Migration Status

- Client migration to Condition Algorithms is in progress.
  - Significant amount of work due to thread safety considerations.
- Support for execution in serial Athena is maintained by means of special sequences that execute Condition Algorithms before regular ones.
- Support for HLT use cases (all conditions loaded before execution of first event) is maintained.

Multithreaded Conditions Access

- In order to minimize memory usage, the ConditionsStore is shared between all concurrent events.
- The structure of the ConditionsStore is a multi-cache, where data is held in containers. Containers are indexed in the store by name and type, internally ordered by Interval of Validity (IOV)
- Conditions are recorded in the ConditionsStore by special Condition Algorithms that interact with the Conditions database service and perform all necessary calculations to transform raw conditions into derived ones before the recording is performed.
- Smart references called ConditionHandles are used by Algorithms to write and read conditions data. These are guaranteed to always point to the correct data for the current event.
- At the start of the job, the scheduler builds a directed acyclic graph of all Algorithms based on their data dependencies to determine the execution order. At this time, Conditions Algorithms are removed from the data flow realm, and managed separately. Unlike regular Algorithms which are always executed in each event by the scheduler, Condition Algorithms are only executed when the data they manage enters an IOV that is not already present in the ConditionsStore.

Geometry and Detector Description



- Uses the same infrastructure as the Conditions.
- Encapsulates alignment deltas and cached positions in Alignment Objects that reside in ConditionsStore. These are:
  - accessed via ConditionHandles.
  - updated as necessary via GeoAlignAlg.

Garbage Collection

In order to minimize memory usage, containers in the ConditionsStore are periodically purged of entries that are no longer needed.

- Maintain a circular buffers of *timestamp* and *run+LumiBlockNumber* for last *N* events.
- When an object is added to a Condition Container, add note to clean it *M* events later.
- At that time, examine objects in the container, oldest first. If none of the keys from the last *N* events match it, then remove this object.
- Stop when we find an object that matches a key, or when there is only one object left.

- Cleaning can be performed synchronously from the event loop, or asynchronously via a Threading Building Block task.

C. Leggett, I. Shapoval, S. Snyder, V. Tsulaia for the ATLAS Collaboration