Migration of Conditions Clients to AthenaMT

First experience

Vakho Tsulaia (LBNL)

ATLAS Software TIM, Glasgow
June 6, 2016

Purpose of this exercise

- Migrate few conditions clients from **`CaloHiveEx/CaloHiveExOpts.py`** to the new Conditions Data Access infrastructure
- Get first experience with migration of the client code
- Identify issues with the new system and provide feedback to the core developers (Charles)
- Finally, come up with a list of instructions and recommendations for the algorithmic code developers
  - Yet to be done…

Trivial example

- **`CaloUtils/CaloLCClassificationTool`**
  - Client of one conditions folder
  - Registers callback for updating private data member:

```
const DataHandle<CaloLocalHadCoeff> m_data;
```

  - In the callback, just retrieves the object from the Detector Store and does nothing else

```
sc = detStore()->retrieve(m_data,m_key);
```

- Actions required for this client:
  1) Replace the old `m_data` data member with `ReadCondHandleKey`

```
SG::ReadCondHandleKey<CaloLocalHadCoeff> m_rchk;
```

  2) Drop the callback function
  3) Add the conditions DB folder to the `CondInputLoader` list of folders (job options):

```
topSequence.CondInputLoader.Load +=
[ ('CaloLocalHadCoeff','/CALO/HadCalibration2/CaloEMFrac') ]
```

Not so trivial example

- **`CaloUtils/CaloLCWeightTool`**
  - Client of one conditions folder
  - Registers callback for updating private data member:

```
const DataHandle<CaloLocalHadCoeff> m_data;
```

  - In the callback, retrieves the object from the Detector Store …

```
sc = detStore()->retrieve(m_data,m_key);
```

  - … and populates local cache (private data member)

```
std::vector<int> m_isampmap;
```

- The existence of such local cache means
  - We need to introduce a Conditions Algorithm
  - This Conditions Algorithm will create a Conditions Object (corresponding to the Tool's cache) and write it to the Conditions Store using a Write Conditions Handle

Not so trivial example (contd.)

- **Actions required for this client**. Introduce **two new classes:**
  1) New Conditions Data Object

```
class CaloLCWeightObj {
   …
 private:
   std::vector<int> m_indices;
};
```

  2) New Conditions Algorithm
     - Transform the callback function of `CaloLCWeightTool` into `execute()` of this algorithm
     - The `execute()` needs to create new instance of `CaloLCWeightObj` and store it into Conditions Container (using **Write Conditions Handle**)

Not so trivial example (contd.)

- **Actions required for this client**. Changes in **`CaloLCWeightTool`**
  1) Replace the old `m_data` data member with `ReadCondHandleKey`

```
SG::ReadCondHandleKey<CaloLocalHadCoeff> m_rchkCaloLocalHadCoeff;
```

  2) Replace the old `m_isampmap` cache with `RedCondHandleKey`

```
SG::ReadCondHandleKey<CaloLCWeightObj> m_rchkCaloLCWeightObj;
```

  3) Drop the callback funstion

- Finally, add the conditions DB folder to the `CondInputLoader` list of folders (job options):

```
topSequence.CondInputLoader.Load +=
[ ('CaloLocalHadCoeff',
'/CALO/HadCalibration2/H1ClusterCellWeights') ]
```

Complex example

- **`CaloTools/CaloNoiseToolDB`**
  - Client of **4 conditions folders**
  - Registers 2 callback functions
    - First callback for 1 conditions folder
    - Second callback for 3 conditions folders
  - In the callbacks
    - Updates several private data members
    - Sets a **flag** for updating local cache
  - The **flag** is checked at the beginning of several **public methods of the tool.** If the flag has been set then the cache update is triggered

Complex example (contd.)

- Like in the previous ("not so trivial") example, here we also need to introduce new Conditions Algorithm and Conditions Data Object

- But, due to the complexity of the code, it was not so obvious how to design the Conditions Data Object and how to implement `execute()` of the Conditions Algorithm

- For now implemented several not-so-clean shortcuts just to get the example going
    - For the sake of proof of principle …

- Clean implementation requires expertise of the developer of this particular Tool, or at least somebody who has a good knowledge of what this code is doing

Status, next steps

- After migrating aforementioned clients the **`CaloHiveExOpts.py`** example happily runs in AthenaMT in 20.8.X-VAL nightlies with `--threads=1`
  - No validation of the results has been done so far, but at least there are no crashes
- Next steps
  - Run tests with more that one thread
  - Test the new conditions infrastructure in **serial Athena**
  - Test the new conditions infrastructure with jobs in which **conditions change during event loop** (this is not the case for `CaloHiveExOpts.py`)
- Write code migration instructions for the developers of conditions clients
- At some point we need to **put everything into dev/devval**
  - And also back-port to stable releases, in order to avoid an avalanche of branch tags in client packages…