V Tsulaia, LBNL

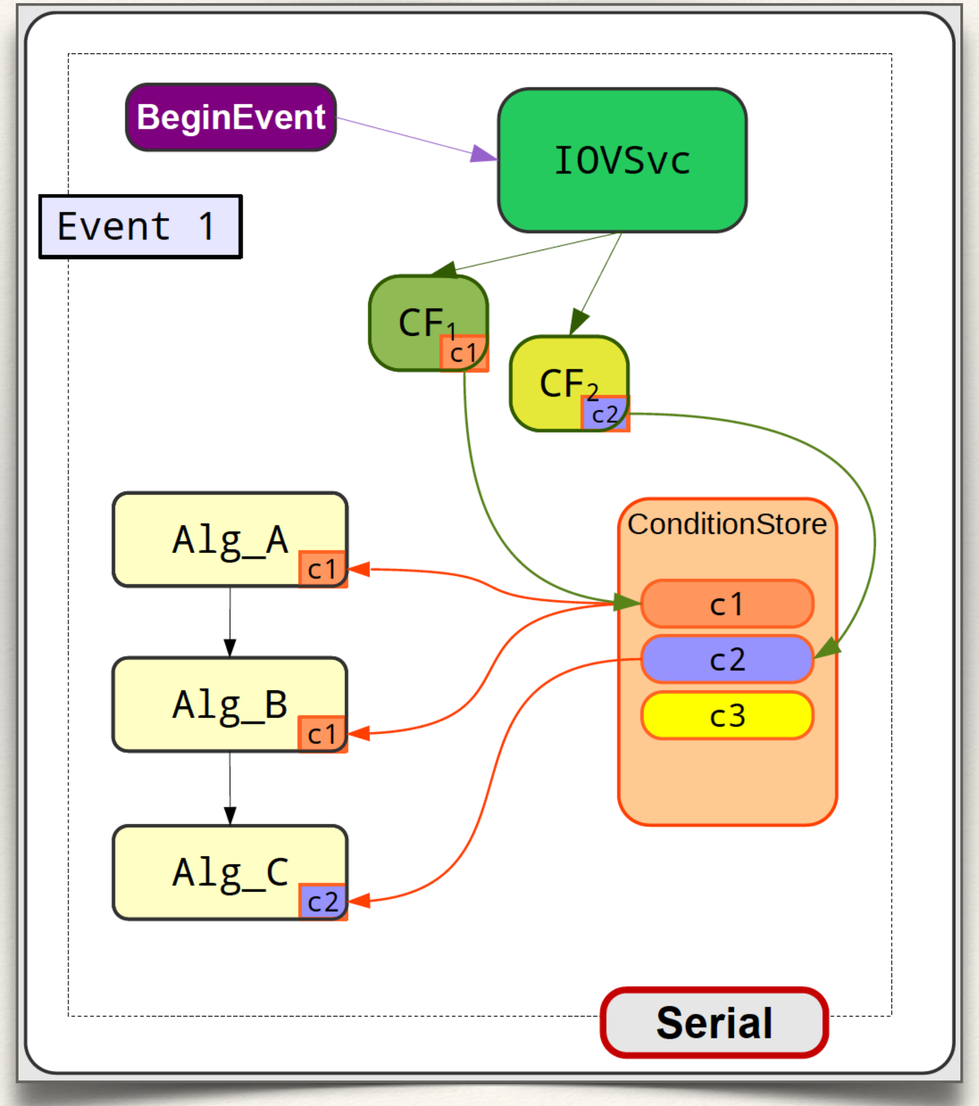
Condition Data Access in AthenaMT

AthenaMT Developer Workshop
CERN, Sep 18-22, 2017

Contents

Serial Processing with Conditions

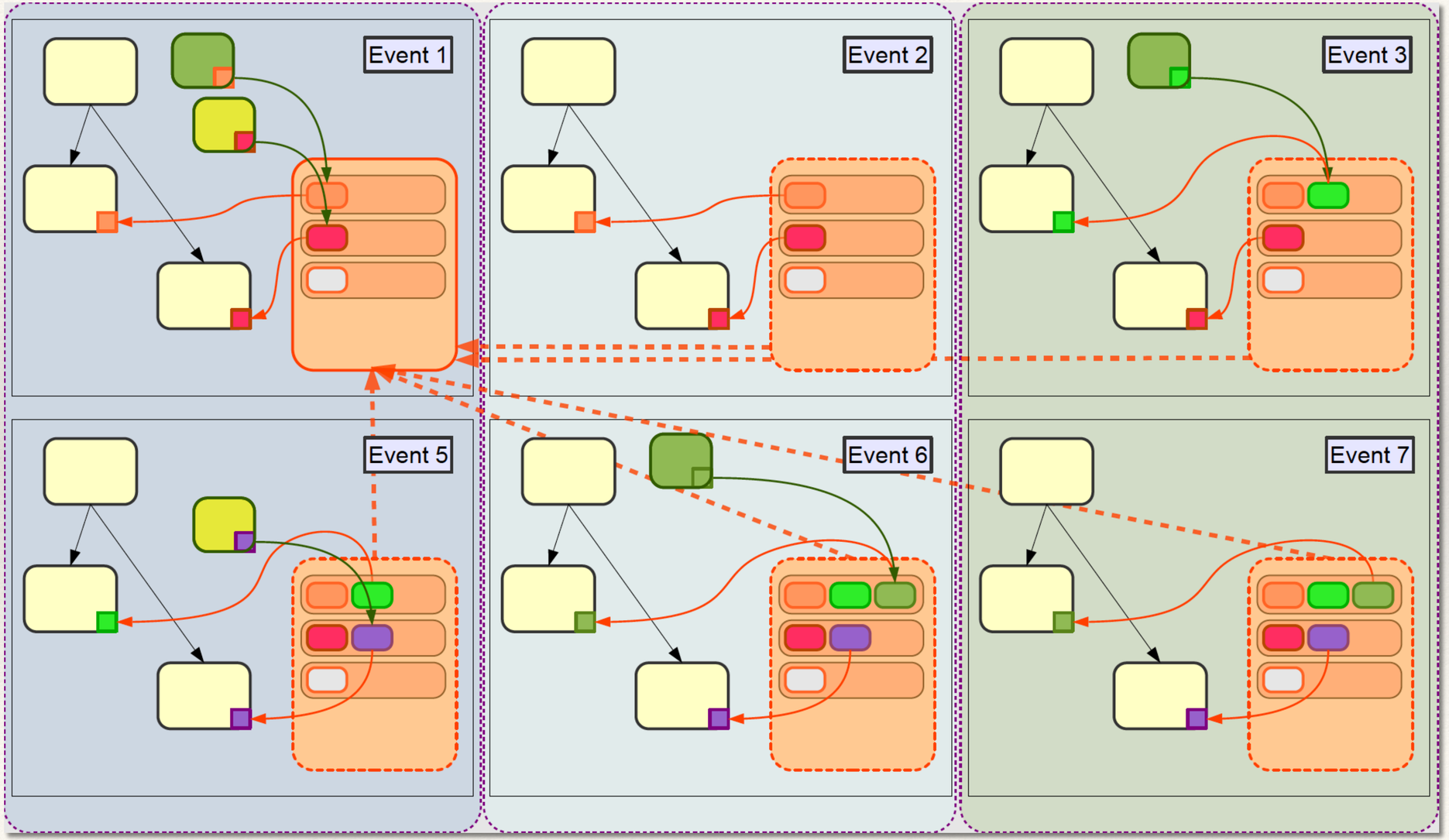
- Within given event, all framework elements process data from the same IOV
- Condition clients are blind to the IOV, retrieve data from the **Detector Store**
- Data retrieval from the Condition DB is a responsibility of **IOVDbSvc**
- At the start of every event (**BeginEvent**):
  - **IOVSvc** checks the validity of all condition objects
  - If some object is no longer valid, it is **overwritten** in the store with a new version fetched from the database
  - For all updated objects IOVSvc triggers callback functions



Multi-Cache Condition Store for AthenaMT

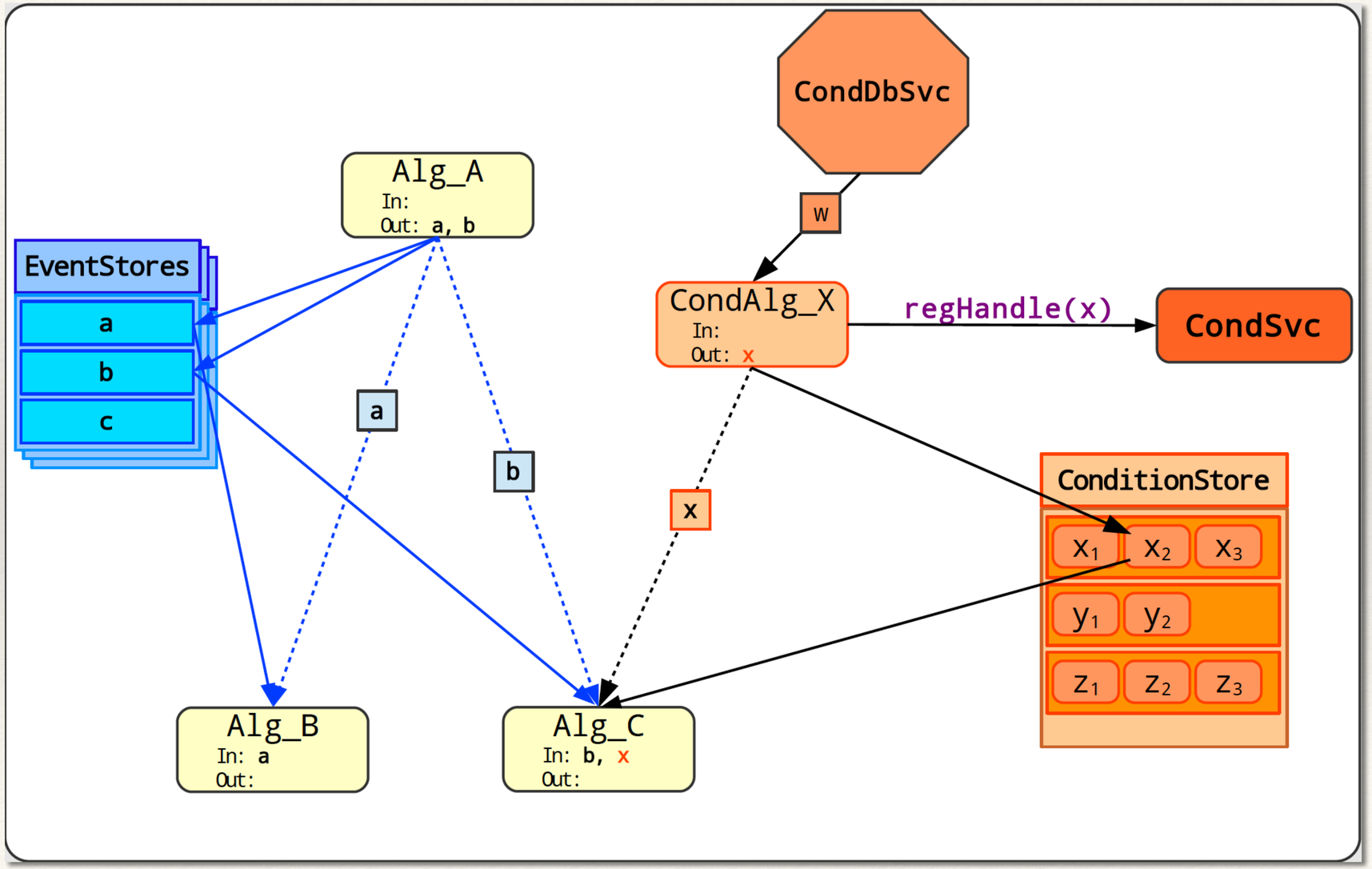
- **Single multi-cache Store** for Condition Data: **Condition Store**
- Each **store element is a Container** that holds multiple instances of Condition Data Objects of single type, one per IOV: **Condition Container**
- Client **accesses the data via smart handles**, which point to the appropriate entry in the Condition Container for a given event: **Condition Handle**
  - From the client's perspective, these objects look like any other object in the Event Store (keyed with an unique identifier)
  - **Client Algorithms declare a data dependency on the condition data object**
- Updating functions are scheduled by the framework
  - **IOVSvc callback functions are migrated into specialized algorithms: Condition Algorithm**
  - These Algorithms are scheduled only when they enter new IOV

Multi-Cache Condition Store for AthenaMT



Condition Handles

- Read Condition Handle
  - Used for read-only access to condition objects in the Condition Store
- Write Condition Handle
  - Used by Condition Algorithms for creating new entries of Condition Objects inside Condition Containers



Functionality

- During initialize(), Condition Algorithms register their Write Condition Handles with the **Condition Service**
- At the start of each event the Scheduler will:
  - Query Condition Service to determine which CondOBjID-s are valid/invalid
  - Query the Execution Flow Graph to find producer algorithms for these objects
  - If any object produced by Condition Algorithm is not valid, schedule the Algorithm to execute, otherwise mark it as already executed
- Only those Condition Algorithms that need to produce new data for given event will execute

CondInputLoader Algorithm

- **CondInputLoader** plays a special role in AthenaMT
  - It triggers the retrieval of Raw Condition Objects from either IOVDbSvc cache or from CondDB
  - It stores Raw Condition Objects into Condition Store using Write Condition Handles …
  - … and by this way downstream clients can access them via Read Condition Handles
- All Condition Objects (COOL folders) need to be declared to CondInputLoader at the configuration step
- For doing this use an updated interface to the **conddb** object in python. Example:

```
conddb.addFolder("CALO","/CALO/HadCalibration2/CaloEMFrac", className="CaloLocalHadCoeff")
```

Migration of Condition Clients

For more details - offline reading, code examples - see
TWiki Documentation

Core Components

- Core components of the Condition Access infrastructure were enabled in RecExCommon back in March:
  - **CondSvc**
  - **ConditionStore**
  - **Condition Sequencer** `AthCondSeq`. For more details about sequencers see the presentation by Charles on Monday.
    - CondInputLoader is added to AthCondSeq inside RecExCommon
    - Users are expected to add their Condition Algorithms to AthCondSeq by themselves

Trivial Examples

- It is relatively simple to migrate those Condition Clients which **don't produce Derived Condition objects**:
  - The client registers a data handle on the condition object, **or …**
  - … the client registers callback on the condition object, but inside the callback it only retrieves the object from the Detector Store and does nothing else
- Migration strategy for such clients
  - Access Condition Objects via **Read Condition Handles**
  - Configure CondInputLoader to read required data from Condition DB
    - Add COOL folder(s) to CondInputLoader's list

Complex Examples

- Some Condition Clients **do produce Derived Condition Data**
- Derived Condition Data can be represented as
  - **A well defined object**, which is updated in the Detector Store at the end of the callback function
  - **A well defined object**, which is kept by the Client as private data member and which is updated inside the callback function
  - **A number (one or many) of private data members** (either basic C++ types or user-defined types) of the client, which are updated inside the callback function
- **Better yet:** some clients (tools) may have local cache which gets updated outside of the callback
  - A flag to trigger the cache update is set within the callback

Complex Examples (contd.)

- Strategy for the migration
  - (If necessary) Define new class for the Derived Condition Object
  - Migrate the existing callback function into a new Condition Algorithm
    - Will use **Read Condition Handle** for Raw Condition Objects and **Write Condition Handle** for writing Derived Condition Object into Condition Store
  - Access Derived Condition Objects via **Read Condition Handles** in downstream clients
  - Configure CondInputLoader to read required data from Condition DB

Clients migrated so far

- List of classes which have references to ReadCondHandle and WriteCondHandle:

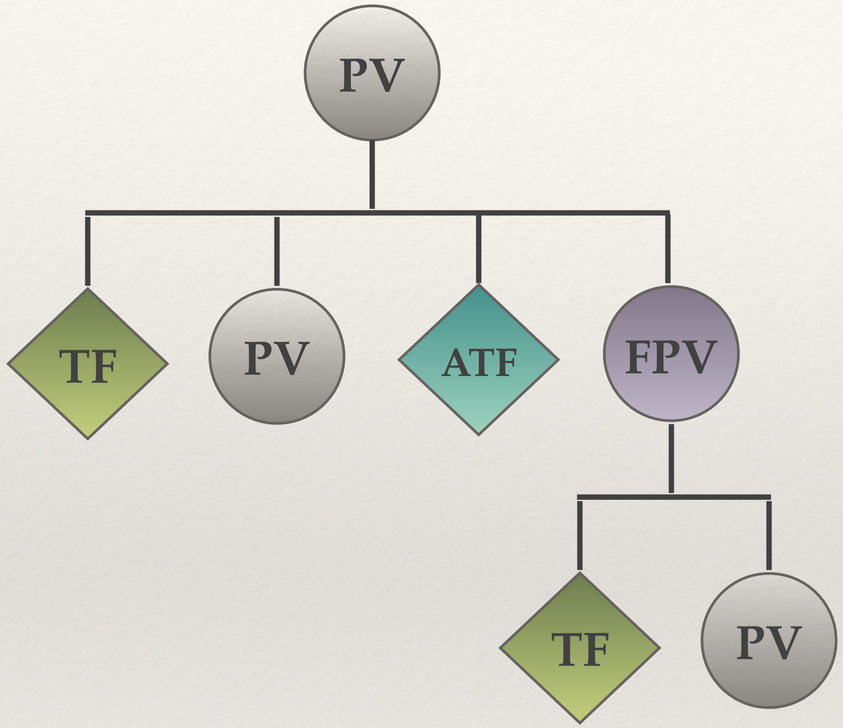
```
Calorimeter/CaloUtils/CaloLCClassificationTool
Calorimeter/CaloUtils/CaloLCOutOfClusterTool
InnerDetector/InDetCalibAlgs/PixelCalibAlgs/PixelCalibCondAlg
InnerDetector/InDetConditions/PixelConditionsTools/PixelRecoDbTool
InnerDetector/InDetConditions/SCT_ConditionsServices/SCT_MonitorConditionsCondAlg
InnerDetector/InDetConditions/SCT_ConditionsServices/SCT_MonitorConditionsSvc
InnerDetector/InDetConditions/SCT_ConditionsServices/SCT_TdaqEnabledCondAlg
InnerDetector/InDetConditions/SCT_ConditionsServices/SCT_TdaqEnabledSvc
LArCalorimeter/LArBadChannelTool/LArBadChannel2Ascii
LArCalorimeter/LArBadChannelTool/LArBadChannelCondAlg
LArCalorimeter/LArBadChannelTool/LArBadFeb2Ascii
LArCalorimeter/LArBadChannelTool/LArBadFebCondAlg
LArCalorimeter/LArRecUtils/LArADC2MeVCondAlg
LArCalorimeter/LArRecUtils/LArFlatConditionsAlg
LArCalorimeter/LArRecUtils/LArHVScaleRetriever
LArCalorimeter/LArRecUtils/LArOnOffMappingAlg
```

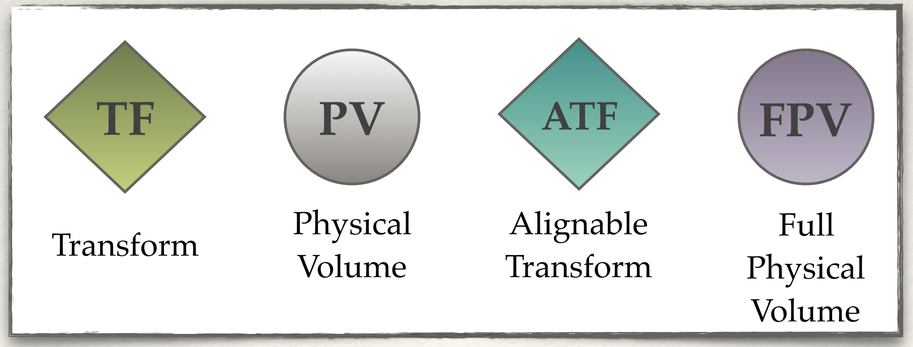
- Tests and examples:

```
Control/AthenaBaseComps/test/propertyHandling_test.cxx
Control/AthenaExamples/AthExHive
Control/DataModelTest/DataModelTestDataCommon
```

Alignments in AthenaMT

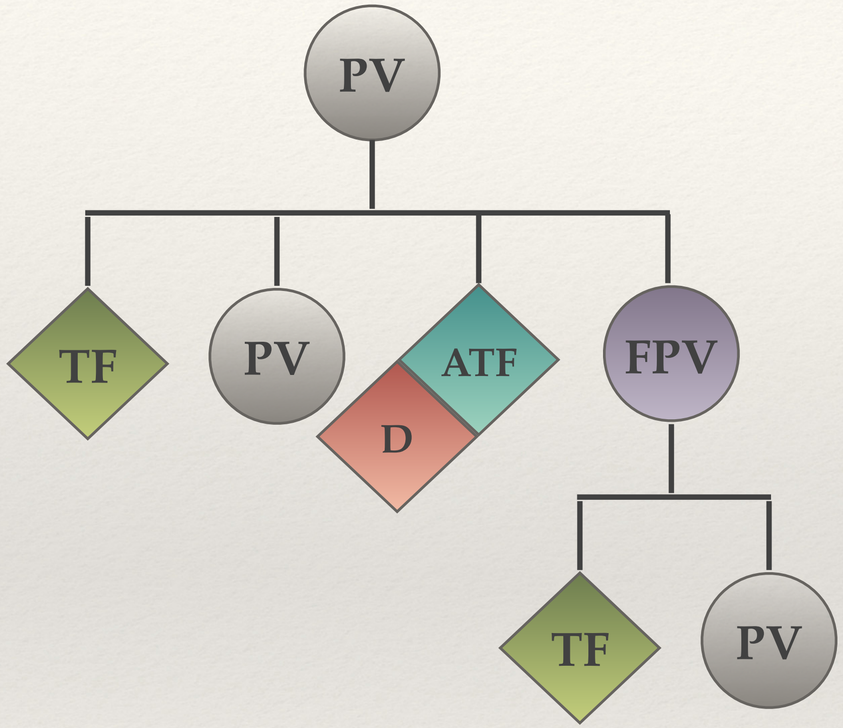
Static GeoModel Tree

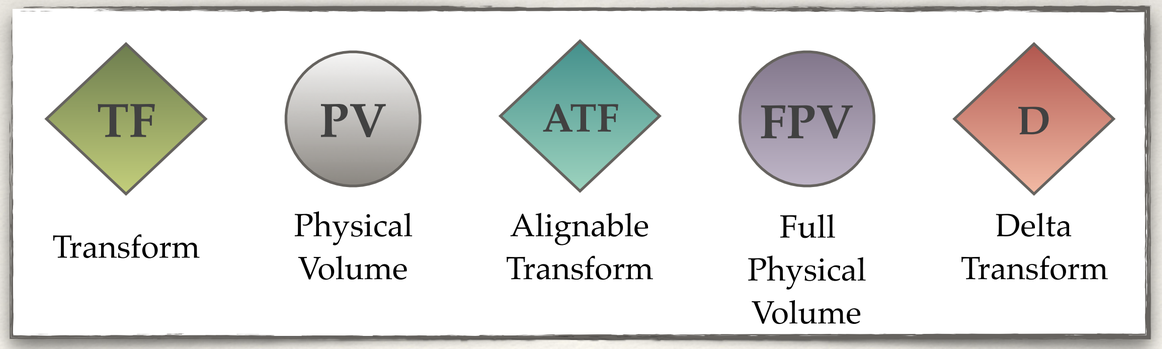




- Physical Volume
  - Basic building block of the tree
- Full Physical Volume
  - The one which will be queried by the clients for its **global position**
  - Computes and **caches** its global position on first query
- Transform
  - Cannot be altered after construction
- Alignable Transform
  - Can be altered multiple times. Set Delta/Clear Delta
  - **Caches the Delta**

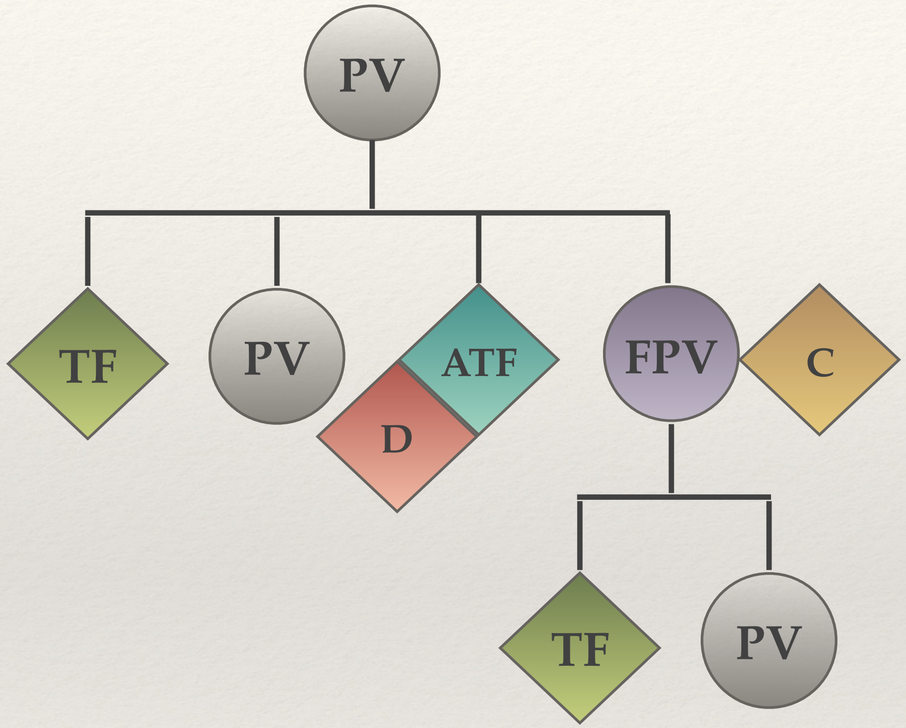
Applying Alignment Corrections



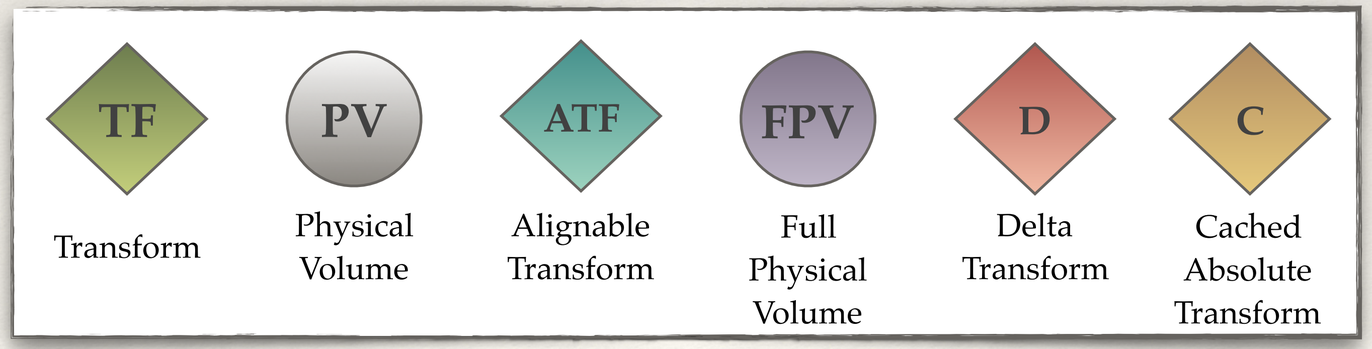


- Alignment Corrections are stored in the **Conditions Database** in a form of **Delta Transforms**
  - Delta in local (wrt the parent volume) coordinate system
- Alignment data is read in **Callbacks** and is applied to the AlignableTransforms using setDelta() functions
- An AlignableTransform caches its Delta in a private data member

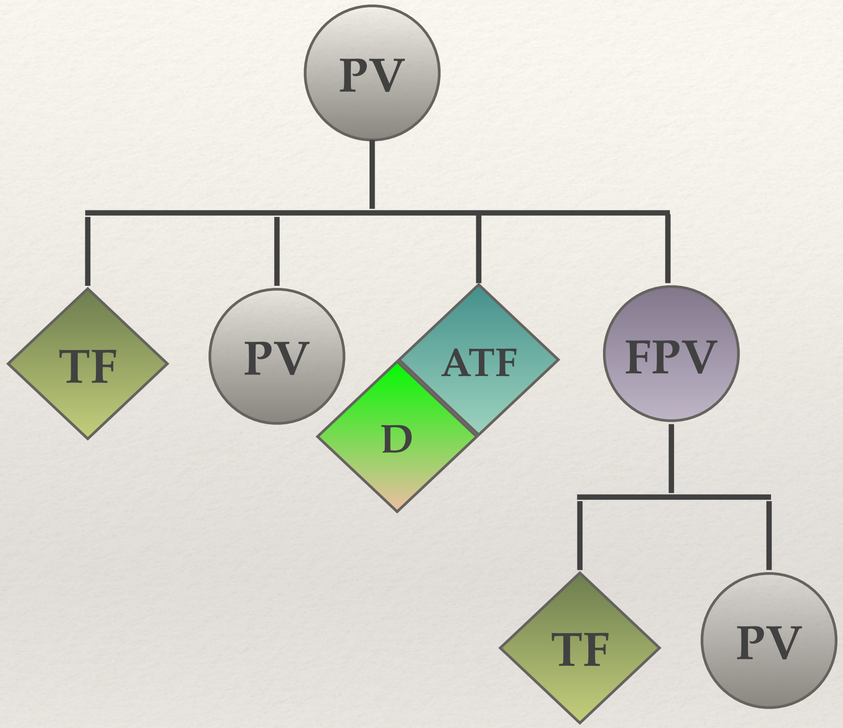
Caching Absolute Positions of Full Physical Volumes



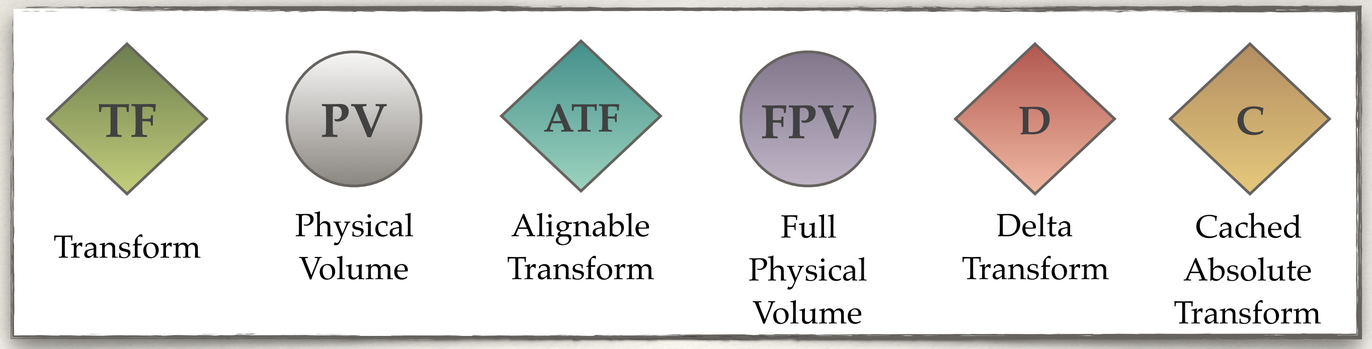
- When a Client queries a Full Physical Volume for its **Absolute Position** (i.e. position in the global coordinate frame) …
- … the FPV **computes** its Absolute Position and **caches it** in a private data member



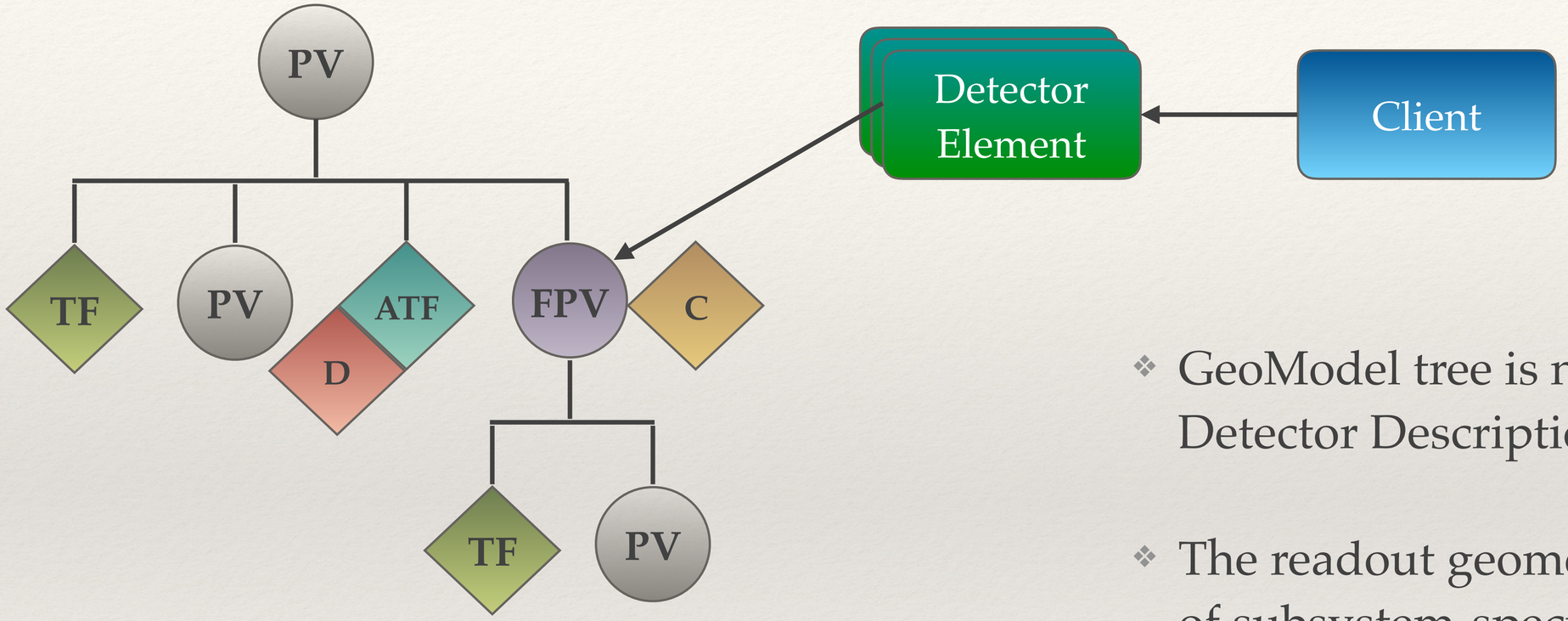
Updating Alignments

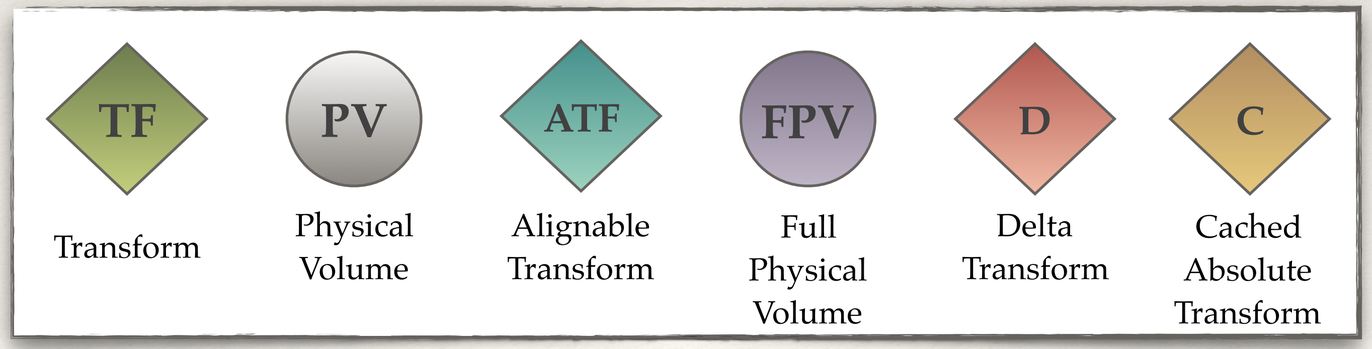


- When **alignments change** during the job …
- … the callback **overwrites the existing Deltas** with new values read from the DB
- At the same time the cached Absolute Positions of FPV-s are cleared



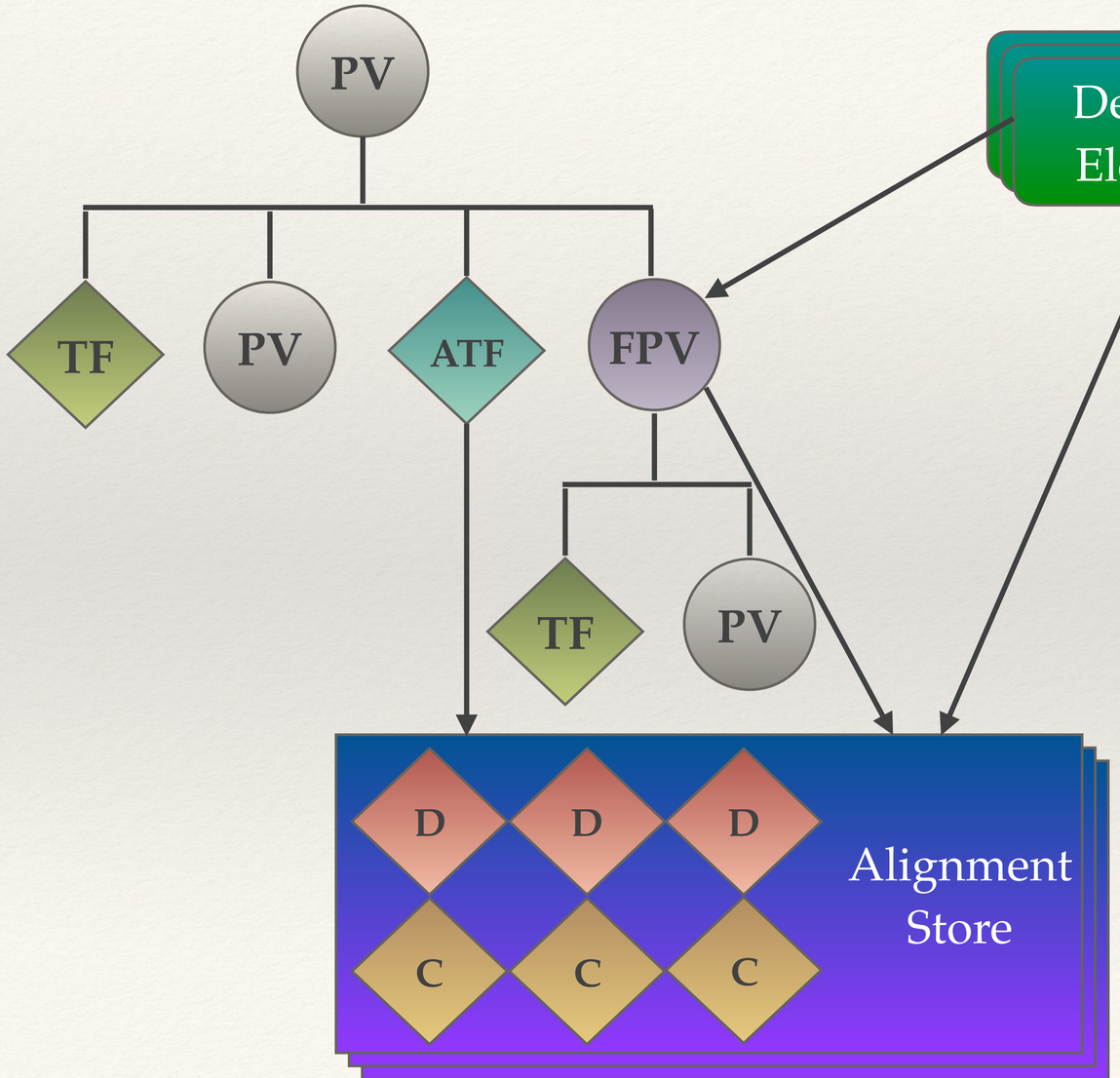
Readout Geometry

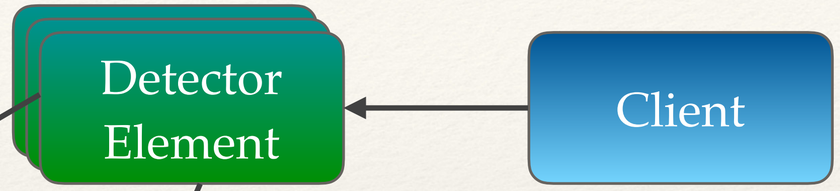




- GeoModel tree is not exposed to Detector Description clients
- The readout geometry layer consists of subsystem-specific **Detector Elements**
- Each Detector Element has a pointer to Full Physical Volume

MT-friendly implementation





- **Alignment Store** is a **Condition Object**
- By making Detector Elements aware of the Alignment Store we can **make the transition completely transparent to Detector Description Clients**

Status and Next Steps

- Prototype implemented ~1 year ago
  - **Core modification:** affected 10 classes out of 70+ in **GeoModelKernel**
  - **Client modification:** tested with **TRT_GeoModel** only in serial jobs
  - The prototype is not yet publicly available (not even in SVN)

- Next steps: give top priority to this task and start working on it ~next week
  - Implement core changes and merge them onto master
  - Start adiabatic migration of the clients, with a help of experts from sub-detectors