ATLAS in-file metadata and multi-threaded processing

Frank Berghaus[1], Attila Krasznahorkay[2] Tim Martin[3], Tadej Novak[4], Marcin Nowak[5], A.C. Schaffer[6], Vakho Tsulaia[7] and Peter van Gemmeren[1]
on behalf of the ATLAS Collaboration

[1]Argonne [2]CERN [3]University of Warwick [4]DESY
[5]BNL [6]Université Paris-Saclay [7]LBNL

May 19, 2021

Metadata

Merriam-Webster:

"data that provides information about other data"

- In HEP we mean, for example
  - Detector conditions
  - Run parameters
  - Simulation parameters
- For this talk:

  Metadata is information in event data files about the events and file

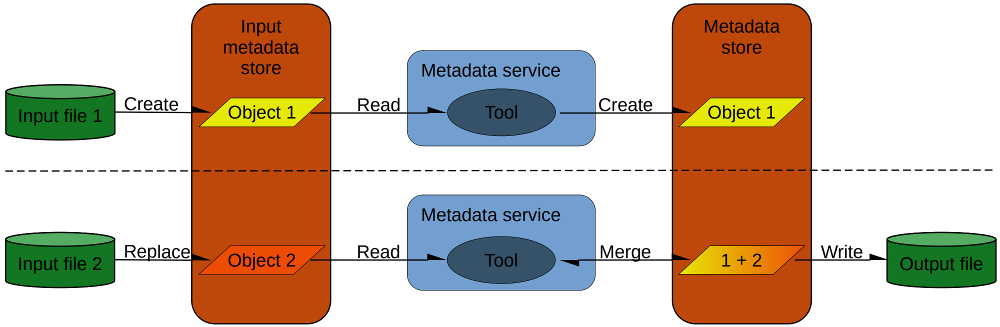
Traditional metadata

ATLAS in-file metadata

- In-file metadata can be split into logical categories according to
  - The data model containing the metadata, and
  - The dedicated software components controlling the information

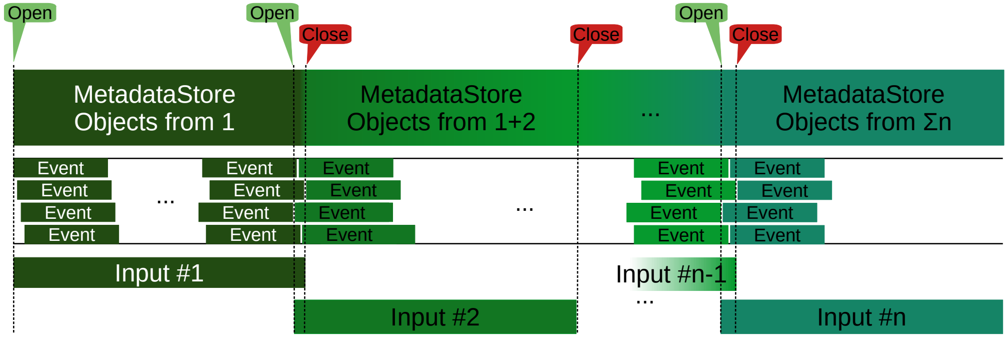
	Metadata Category	Information
Software	`DataHeader`	Event data location and content
Software	`Bytestream`	Run parameters
Software	`EventStreamInfo`	Event content summary (production)
Software	`EventFormat`	Event content summary (analysis)
Software	`FileMetaData`	Event and provenance summary (analysis)
	`IntervalOfValidity`	Lifetime other than file or event
Physics	`BookKeeper`	Event selection
Physics	`LumiBlockRange`	Luminosity blocks in-file
Physics	`TriggerMenu`	Trigger configuration
Physics	`Truth`	Event generation weights and names

In-file metadata infrastructure



- Stores making in-file metadata available to clients:
  - *InputMetaDataStore* — Cleared when new input file is opened. Filled with new file content
  - *MetaDataStore* — Tools move data from input store and add (merge) it with existing objects
- Designed for serial and multi-process environment — not thread safe

Concurrent event processing and in-file metadata


Open
Open
Close
Close
Open
Close
MetadataStore Objects from 1
MetadataStore Objects from 1+2
...
MetadataStore Objects from Σn
Event
...
Input #1
Input #2
Input #n-1
Input #n

- Input files are opened and closed in sequence
  - Overlap required to handle data access on-demand
- Events need metadata from multiple sources around file boundary

Transition to multi-threaded environment

- Must support:
  1. Event-less files
  2. Multiple instances of in-file metadata objects in the MetaDataStore
     - ★ On input file boundary jobs are expected to process events from different files

- Supporting event-less files
  - React to opening or closing files:
    - ★ Input: `BeginInputFile`, `EndInputFile`
    - ★ Output: `MetaDataStop`
  - Limit reactions to these incidents to framework components
    - ★ Use these reactions only when no reasonable alternative exists

- Support multiple versions
  - Ensure metadata is merged or accumulated from input
  - Provide interfaces for thread-safe access for objects where modifications may lead to conflicts
  - *Issue*: Single metadata store
    - ★ Last resort `MetaCont<>` — contains metadata objects keyed by input identifier

Metadata objects for software functionality

- `Bytestream`: A collection accumulates all run parameter sets
  - Only used for record keeping — minimal impact

Implemented uniform design and stream specific handling

- `EventStreamInfo`: Naturally supports addition
  - Lock during modification (*BeginInputFile*)
- `EventFormat`: Naturally supports addition
  - Lock during modification (*BeginInputFile*)
- `FileMetadata`: Keep single item
  - Ignore but warn about differences between first and subsequent input files

Interval of Validity [IOV]

- The **Interval Of Validity** is a time window in ns or range of runs and luminosity blocks
  - Metadata object stores values for all intervals relevant to the file
  - Stored values are valid for some or all events in a file
  - Designed to contain conditions — “laptop on an airplane” use-case
- Example information stored in interval of validity metadata objects:
  - *Simulation parameters* usually a single interval of validity covers many files
  - *Calorimeter voltages* usually have many intervals of validity in a single file
- Access in multi-threaded environment ensured with dedicated services and tools
  - Deprecate direct access to objects by clients

Metadata objects required for scientifically meaningful results

- Bookkeeping — tracking events removed by selection decisions
  - Clients access information using a utility software component
  - Dedicated software components ensures thread-safe access
- Luminosity block information
  - Access patterns during production were found to be thread safe
- Trigger — configuration and menu of trigger items
  - Stored a JSON serialized string — `boost::ptree` in memory
  - Access managed through dedicated software that ensure thread safety
- Truth information — strings identifying generator event weights
  - Addition of new information on `BeginInputFile`
  - Updated by thread-safe and multi-threaded software

Summary

- ATLAS stores a rich collection of information about data in data files
- The in-file metadata infrastructure supports multi-threaded simulation and reconstruction
- Duplication and obsolescence of features motivates a comprehensive multi-threaded redesign — for run 4

Thank you

Special case: event service

- Jobs process short event ranges sequentially
- Event range is written to file and staged out
  - AthenaMT $\implies$ events from multiple ranges in flight at range boundary
- Initially foreseen for simulation only

- Implement (part of) `StoreGateSvc` interface in `MetaDataSvc`
- Call `MetaDataSvc` instead of `StoreGateSvc`
  - Call inserts objects into metadata container by type, key, and range ID
- Output stream resolves metadata containers

Transient type	Key
`EventStreamInfo`	All
`IOVMetaDataContainer`	/Simulation/Parameters
`IOVMetaDataContainer`	/TagInfo

Metadata objects in simulation

Example in-file metadata content

Example metadata object	Example content
ByteStreamMetadata	Run number, luminosity block, beam energy
EventFormat	`AntiKt4LCTopoJets` is a `DataVector<xAOD::Jet_v1>`
TagInfo	AMITag, IOVDbGlobalTag, AtlasRelease
EventStreamInfo	event types (`IS_DATA`), processing tags (StreamESD), item list (a `xAOD::JetContainer` called `AntiKt4LCTopoJets`)
TriggerMenu	ItemName (`L1_EM3`), ItemVersion (1)

In-file metadata infrastructure components

- Service: `MetaDataSvc`
  - Maintain current architecture: management of tool through file incidents
- Tools: `IMetaDataTool`
  - Read new objects from `InputMetadataStore`
  - Create or append to containers in `MetadataStore`

- Stores: `StoreGateSvc`
  - Input InputMetadataStore
    - ★ Clean and repopulate on file open
  - Output MetadataStore
    - ★ Store metadata containers
    - ★ Append new content from input to containers on file open
- Handles:
  - ReadMetaHandle: exists
  - WriteMetaHandle: stashed
    - → probably will not resurrect

List of metadata tools

These tools implement `IMetaDataTool`:

- BookkeeperDumperTool
- BookkeeperTool
- ByteStreamMetadataTool
- CopyEventStreamInfo
- EventFormatMetaDataTool
- FileMetaDataCreatorTool
- FileMetaDataTool
- IOVDbMetaDataTool
- LumiBlockMetaDataTool
- MetaDataSvc
- ReadMeta
- TriggerMenuMetaDataTool
- TruthMetaDataTool
- xAODMetaDataCnvAthena