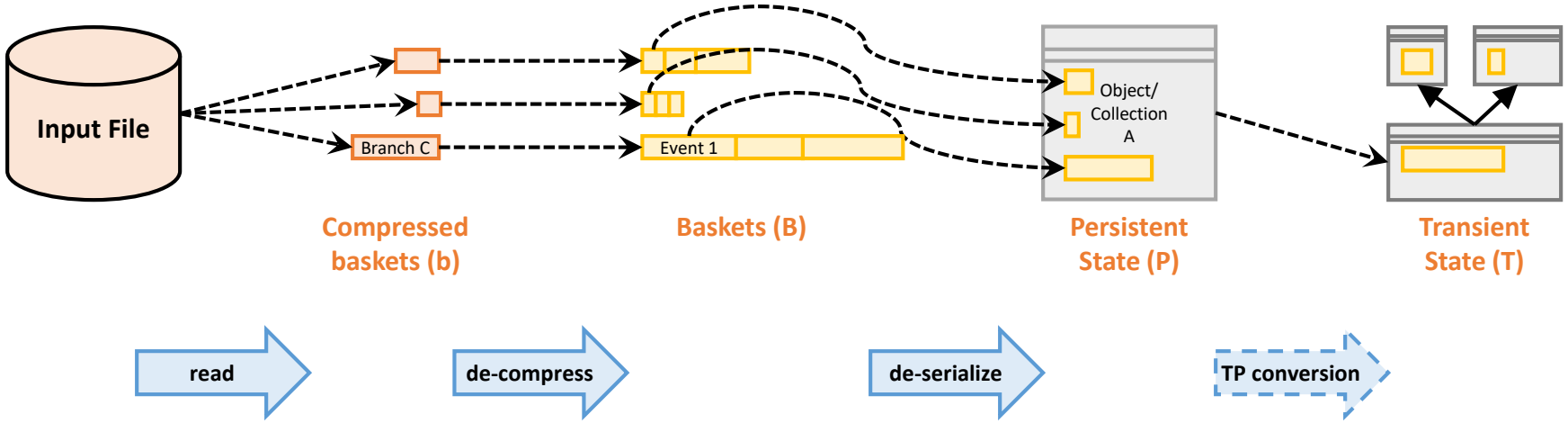
Discussion starter: HEP Experiment and ROOT I/O

Peter van Gemmeren (ANL)

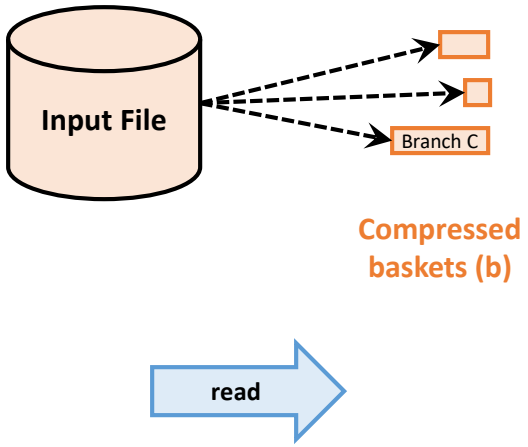
Outline

- This is not a ‘talk’.
- But I assembled a few diagrams to walk through what ROOT I/O means to HEP experiments.
  - Okay, mainly ATLAS, but at high level it all looks the same
  - No news for anyone in HEP, but maybe a good starting point to talk about instrumenting I/O in ROOT

ROOT ‘I/O’ in three (3 ½) steps: Reading

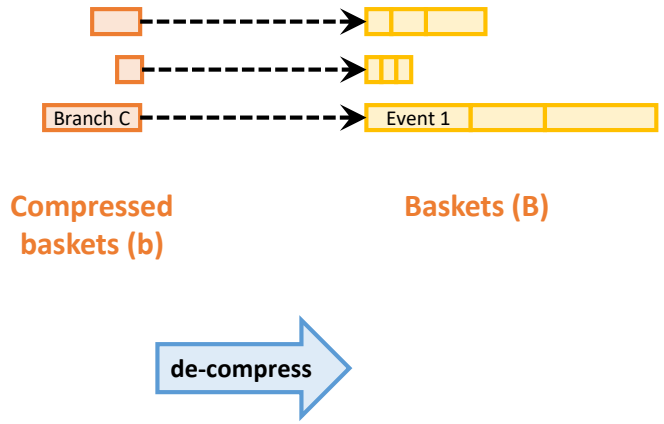


1st step: ROOT I/O



- ROOT reads (compressed) basket of data:
  - Each basket corresponds to a **TBranch** in the **TTree** and can contain many entries/events of data.
    - Configured on write via auto-flush and split-level.
  - ROOT can use a **TTreeCache** to read/cache branches and minimize read operations.
    - Optimize for sequential access, adapted to out-of-sequence reads (e.g. multithreading)

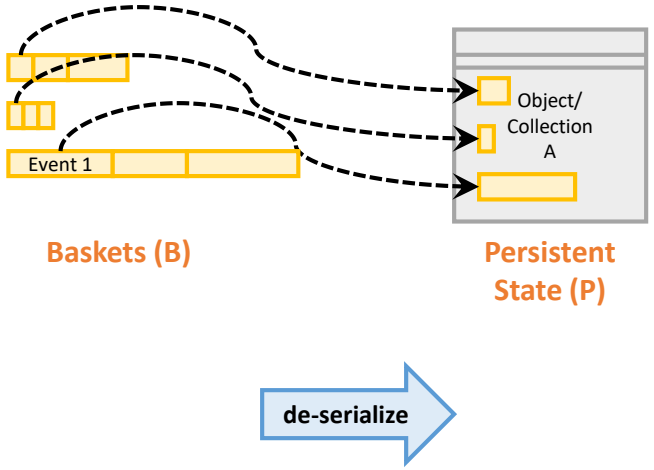
2nd step: Compression



- ROOT decompresses baskets
  - ROOT implements: ZLIB, LZMA, LZ4 and (new) ZSTD algorithms.
  - Compresses whole baskets, all entries/events
    - Often that is what is needed
    - Selective event reading (e.g. in distributed processing) may suffer performance penalty.
    - Decompression is on demand (when branches are accessed).

3rd step: Serialization

- ROOT de-serializes baskets
- The decomposition is the job of a Streamer.
  - Every class to be stored in a file has a Streamer
  - When ROOT writes an object to file, it also writes the description of that class with it.
    - This description is implemented in the StreamerInfo class.
  - When reading an object from a file, the system uses the StreamerInfo list to decode an object.



- Splitting into TBranches
- If a data member is an object, the data members of this object are also split into branches (up to the limit set by the value of split).
  - Pointed to, std::string or array data member will not be split.

Optional step: T/P Conversion

- Not part of ROOT
- Some Experiments (for some data) may use simpler persistent state objects to encapsulate the state of some (very complex) transient classes:
  - This can also allow for schema evolution
  - Or custom (domain specific) compression