CMSSW to Avro

JIM PIVARSKI (PRINCETON)
NHAN TRAN (FNAL)

DIANA HEP
NOVEMBER 7, 2016

CMS workflow with Big Data tools

Idea: a working example porting CMS analysis (monotop) to Big Data tools (Scala/Spark)

https://indico.cern.ch/event/505613/contributions/2228345/

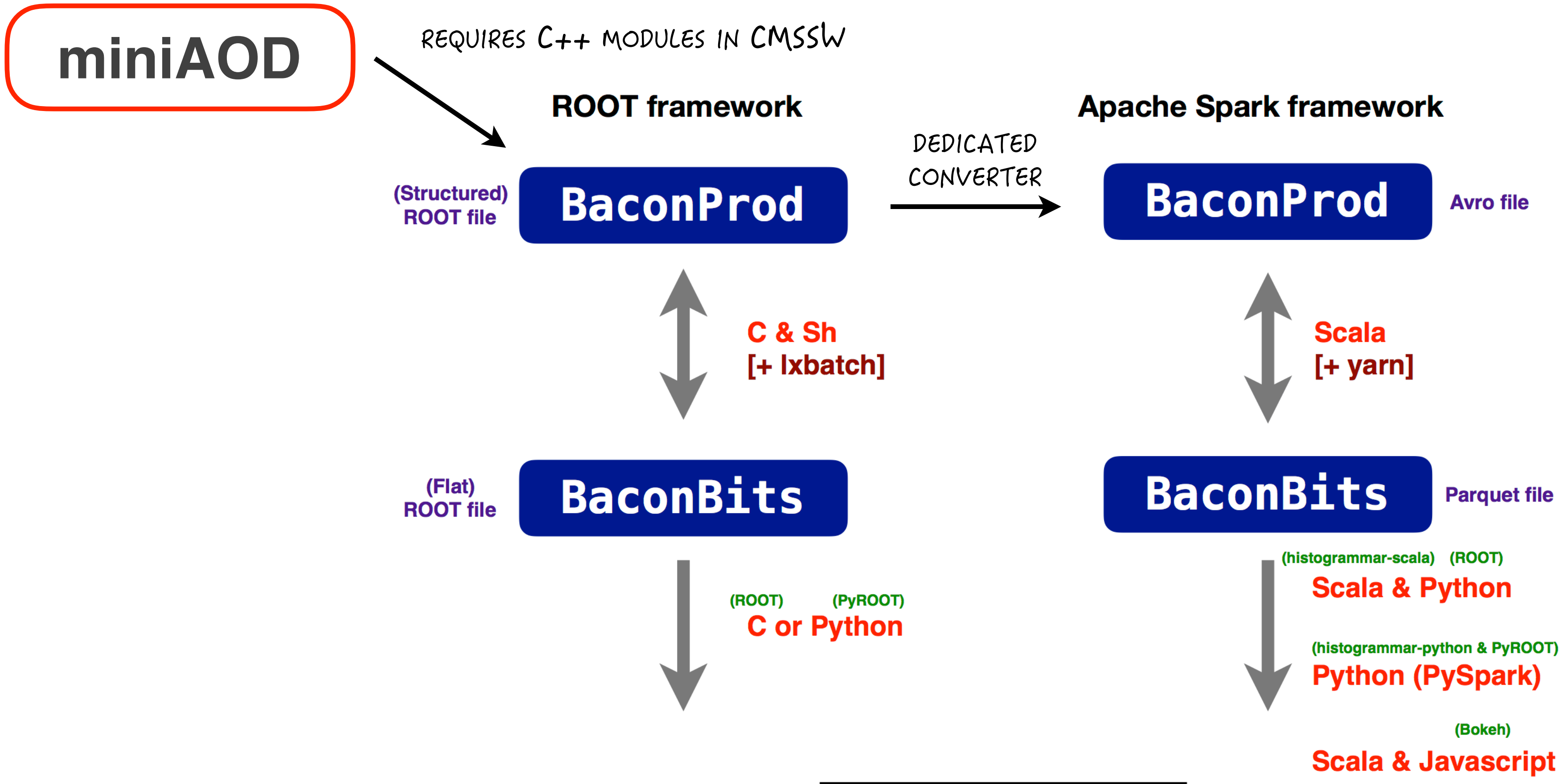
CHEP talk by O. Gutsche et al.

Typical CMS analysis workflow:

miniAOD (EDM ROOT format)

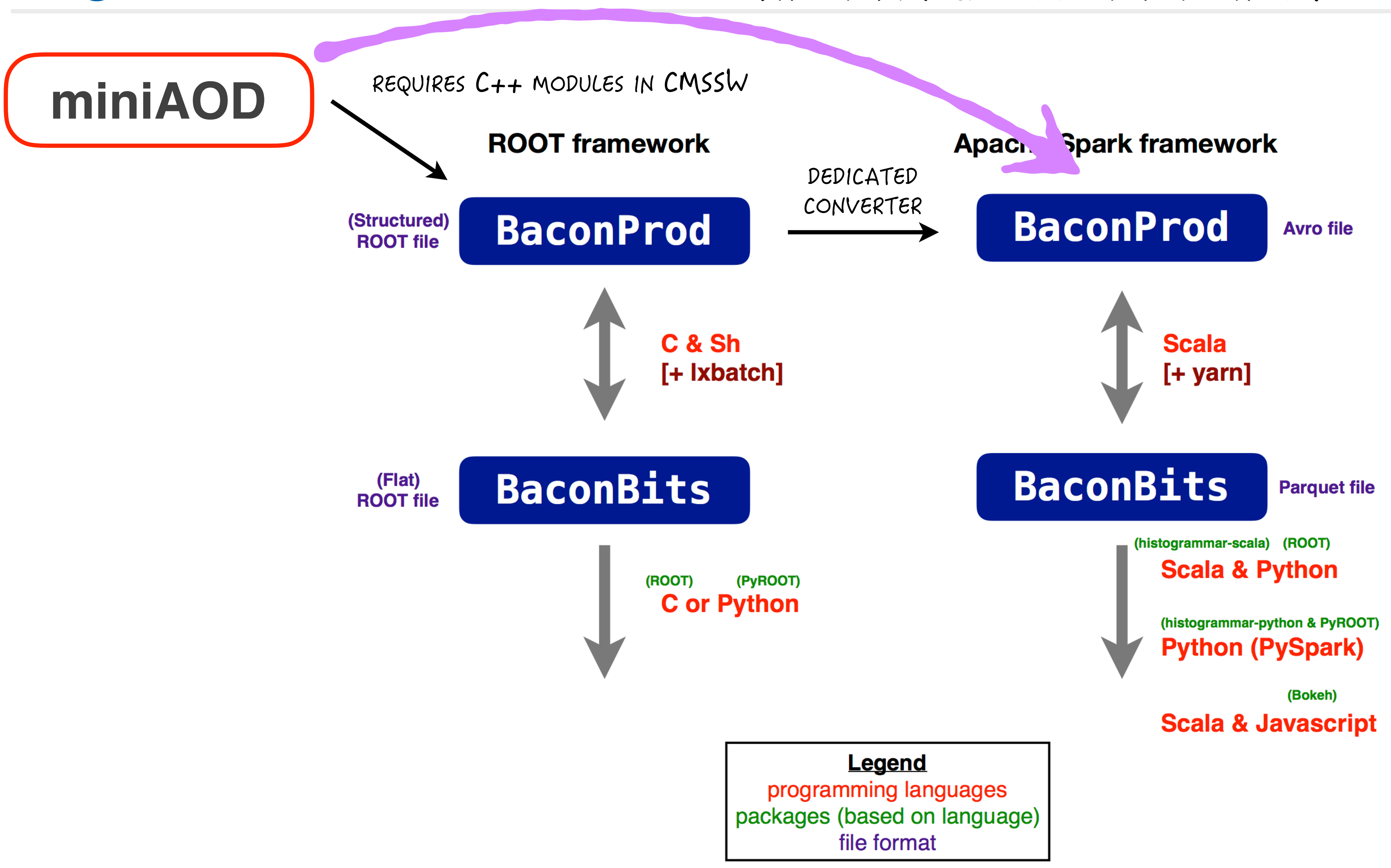
- ☛ user-defined, large-scale ntuples [BaconProd]
  - ☛ analysis ntuples (small scale) [BaconBits]
    - ☛ plots and fits

"Big data in HEP"



Legend
programming languages
packages (based on language)
file format

"Big data in HEP"



CMSSWToAvro

Proof-of-principle:

Build example of reading in miniAOD and writing out Avro

- Avro, preferred data format at the time

Code documented here:

https://github.com/nhanvtran/CMSSWToAvro

Steps:

- Build/install Avro-C, link to CMSSW
- Read in miniAOD files, write out Avro files

Documentation:

https://avro.apache.org

https://avro.apache.org/docs/current/api/c/

CMSSWToAvro

Some issues to be solved in the installation of Avro-C

https://issues.apache.org/jira/browse/AVRO-1844

Fairly trivial to link to CMSSW

```
`scram setup avro`
```

Very simply use case:

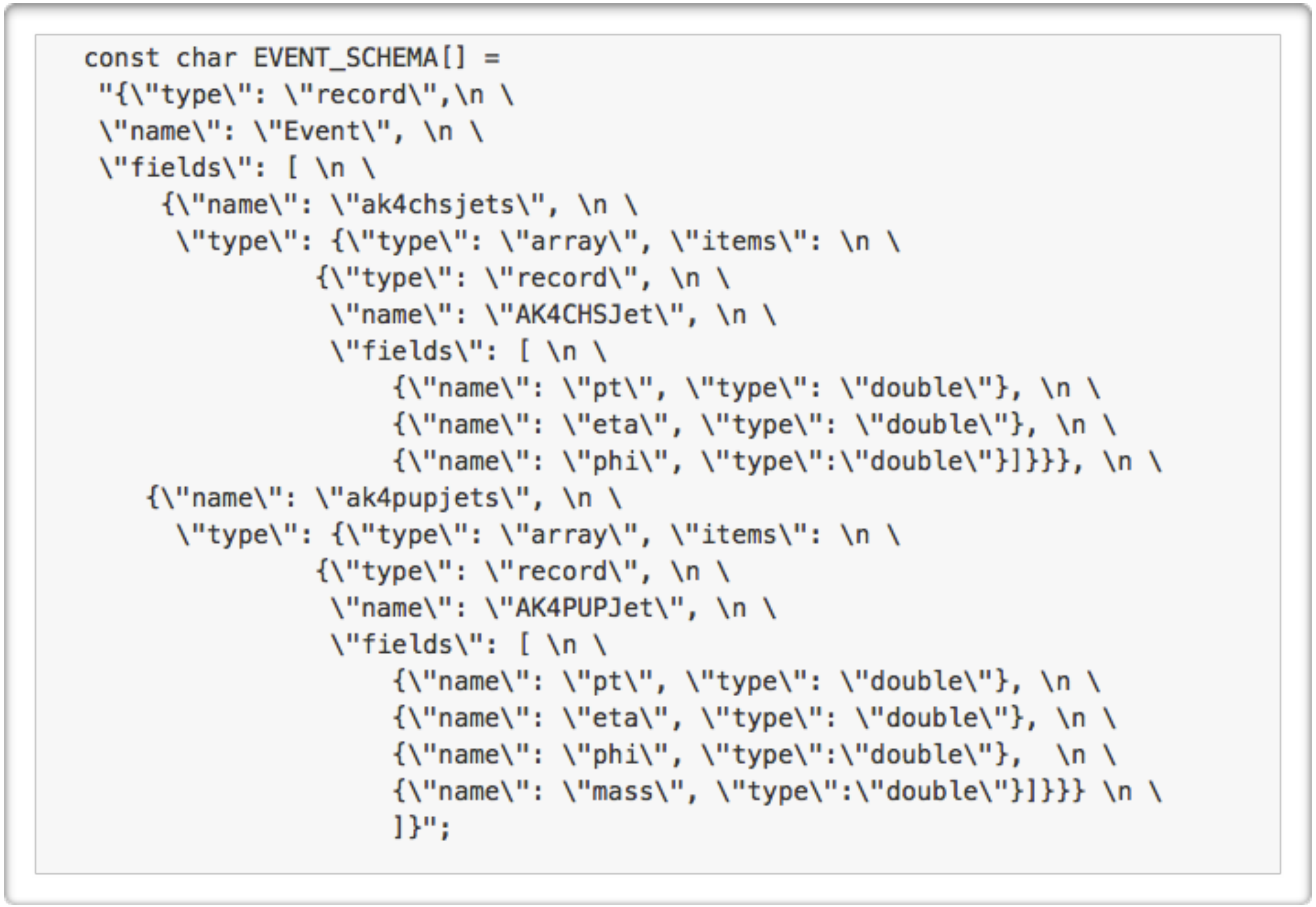
- Read in a collection of jets with non-trivial structure
- Write out to Avro
  - *this can often involve additional c++ processing of the jet collection*

Simple steps for user:

- define schema using JSON format
- fill schema
- write out file

schema

Example is intentionally meant to have structure often used in physics

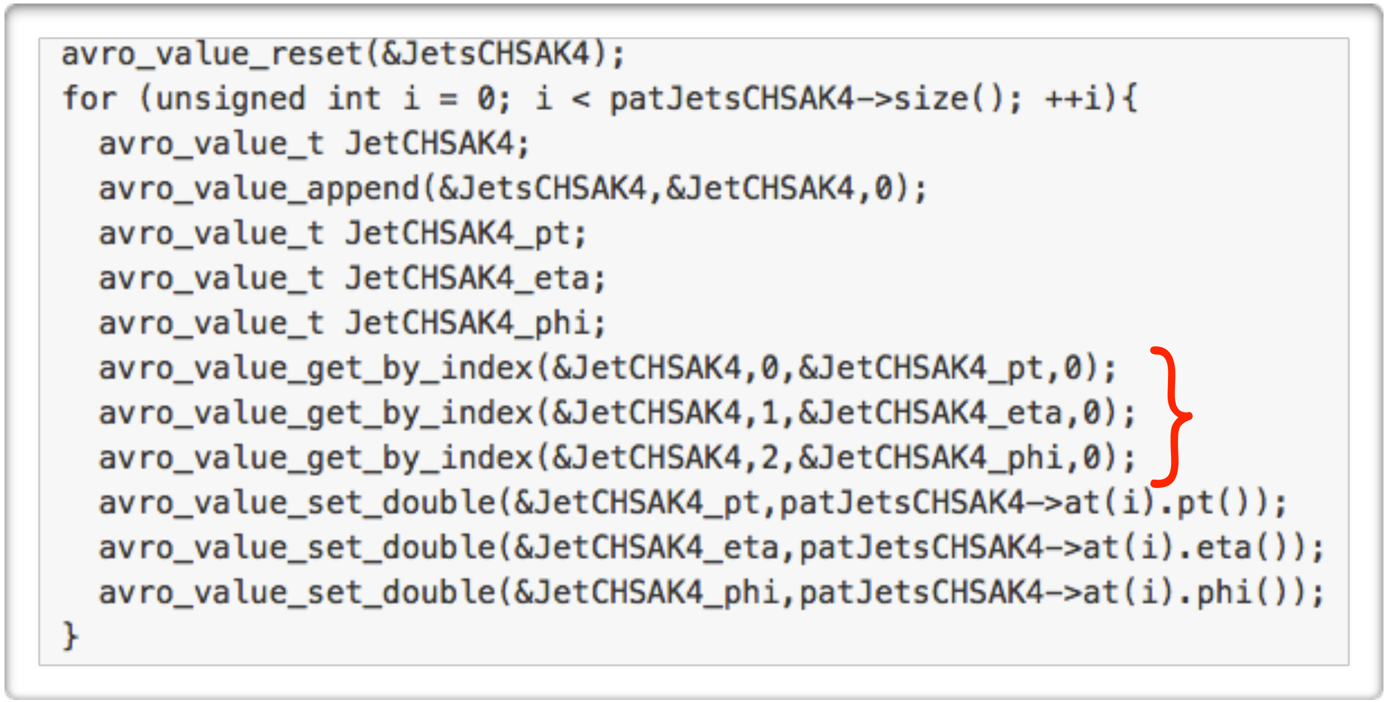


```
const char EVENT_SCHEMA[] =
 "{\"type\": \"record\",\n \
 \"name\": \"Event\", \n \
 \"fields\": [ \n \
     {\"name\": \"ak4chsjets\", \n \
      \"type\": {\"type\": \"array\", \"items\": \n \
               {\"type\": \"record\", \n \
                \"name\": \"AK4CHSJet\", \n \
                \"fields\": [ \n \
                    {\"name\": \"pt\", \"type\": \"double\"}, \n \
                    {\"name\": \"eta\", \"type\": \"double\"}, \n \
                    {\"name\": \"phi\", \"type\":\"double\"}]}}}, \n \
    {\"name\": \"ak4pupjets\", \n \
      \"type\": {\"type\": \"array\", \"items\": \n \
               {\"type\": \"record\", \n \
                \"name\": \"AK4PUPJet\", \n \
                \"fields\": [ \n \
                    {\"name\": \"pt\", \"type\": \"double\"}, \n \
                    {\"name\": \"eta\", \"type\": \"double\"}, \n \
                    {\"name\": \"phi\", \"type\":\"double\"},  \n \
                    {\"name\": \"mass\", \"type\":\"double\"}]}}} \n \
                    ]}";
```

Schema, in a real scenario, could be defined externally in JSON or YAML formats

filling schema

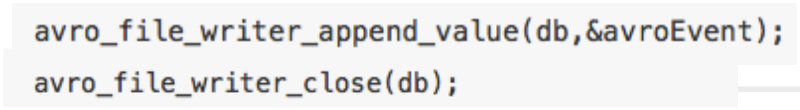
FILL "BY INDEX", ORDER MATTERS OR CAN BREAK THE FILLING



```
avro_value_reset(&JetsCHSAK4);
for (unsigned int i = 0; i < patJetsCHSAK4->size(); ++i){
  avro_value_t JetCHSAK4;
  avro_value_append(&JetsCHSAK4,&JetCHSAK4,0);
  avro_value_t JetCHSAK4_pt;
  avro_value_t JetCHSAK4_eta;
  avro_value_t JetCHSAK4_phi;
  avro_value_get_by_index(&JetCHSAK4,0,&JetCHSAK4_pt,0);
  avro_value_get_by_index(&JetCHSAK4,1,&JetCHSAK4_eta,0);
  avro_value_get_by_index(&JetCHSAK4,2,&JetCHSAK4_phi,0);
  avro_value_set_double(&JetCHSAK4_pt,patJetsCHSAK4->at(i).pt());
  avro_value_set_double(&JetCHSAK4_eta,patJetsCHSAK4->at(i).eta());
  avro_value_set_double(&JetCHSAK4_phi,patJetsCHSAK4->at(i).phi());
}
```

THIS IS A SOMEWHAT FRAGILE PROCEDURE,
RESTRICTED BY THE FACT THAT SCHEMA IS DEFINED AT NOT COMPILED (LIKE ROOT)

FINALLY, WRITE OUT FILE...



```
avro_file_writer_append_value(db,&avroEvent);
avro_file_writer_close(db);
```

outlook

An example of how to incorporate big data formats into CMSSW

- miniAOD $\Longrightarrow$ AVRO
- fairly straightforward to link libraries and write to appropriate format

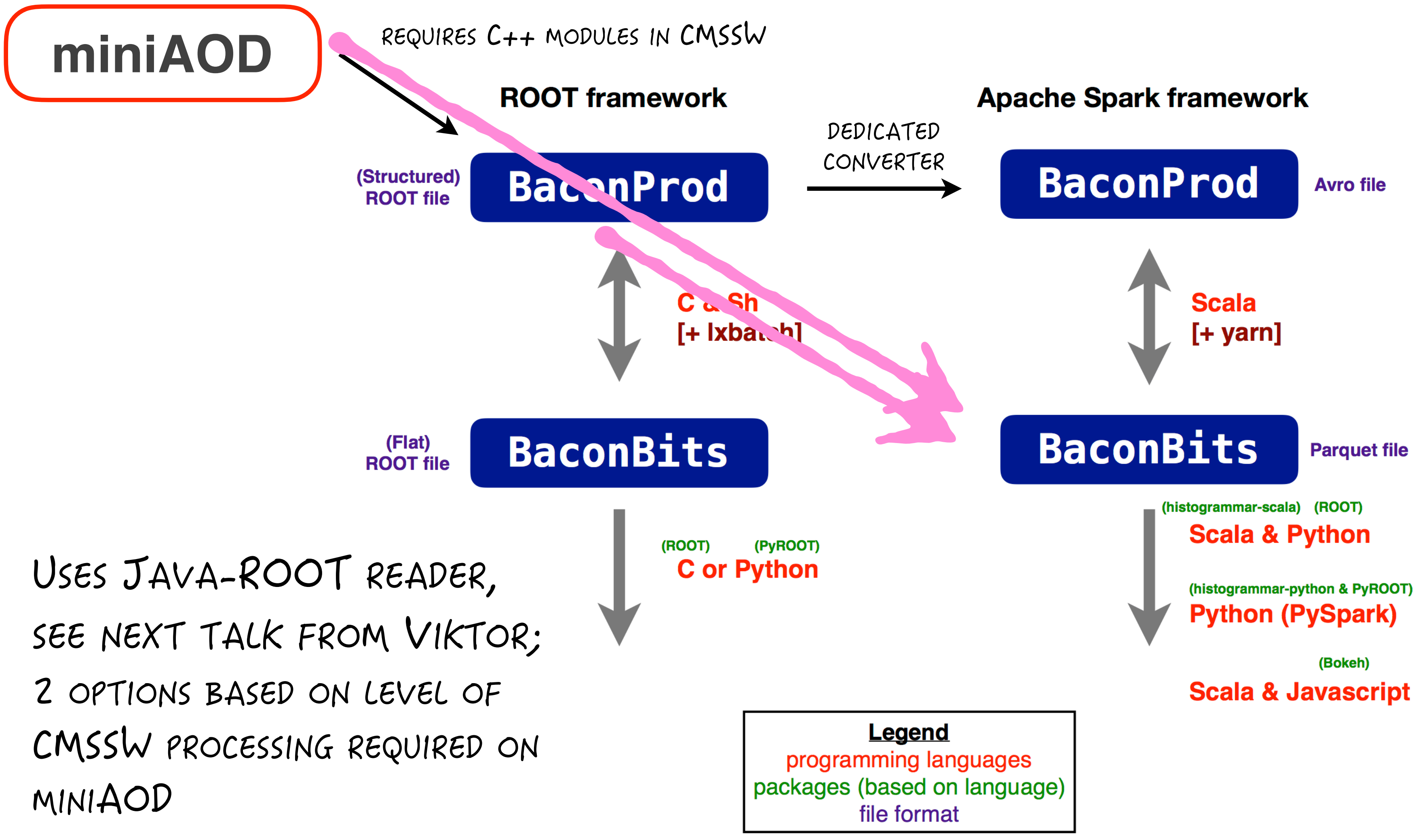
This will feed into the rest of the “big data” analysis from CHEP

- Conceptually, a choice to define schema at run time
- Next possible step: write a wrapper for users so they are not exposed to the fragile parts of Avro-C

Next:

- Other more HEP friendly Big Data tools?
- Other formats?

Other options



USES JAVA-ROOT READER, SEE NEXT TALK FROM VIKTOR; 2 OPTIONS BASED ON LEVEL OF CMSSW PROCESSING REQUIRED ON MINIAOD