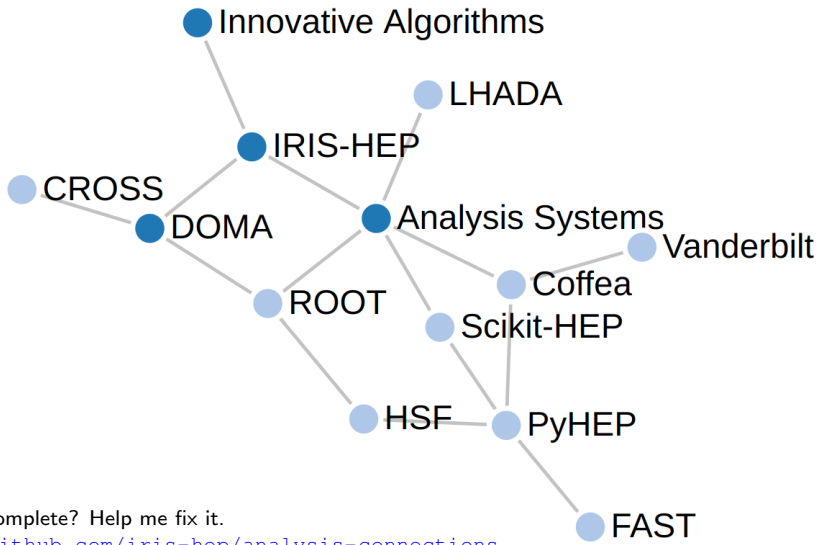
Skyhook for query systems

Jim Pivarski

Princeton University – IRIS-HEP

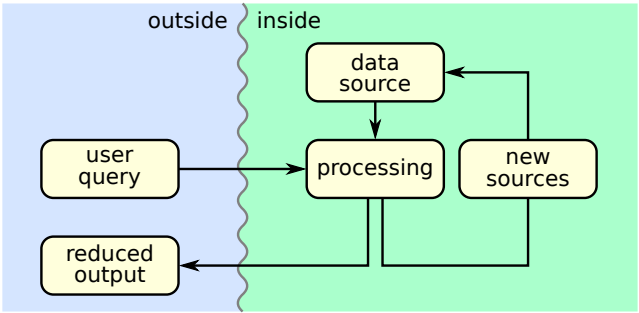
April 24, 2019

My view of the world



Biased? Incomplete? Help me fix it.
https://github.com/iris-hep/analysis-connections

Query System: big data on server, users request reduced output



- Mark Neubauer, Ben Galewsky: query scheduling, data delivery, iDDS
- Gordon Watts, Emma Torro, Mason Proffitt: query expression language
- Jim Pivarski, Henry Schreiner: event data processing, histogram aggregation

Coffea

Columnar **O**bject **F**ramework **F**or **E**ffective **A**nalysis

Matteo Cremonesi, Lindsey Gray, Oliver Gutsche, Allison Hall,
Bo Jayatilaka, Igor Mandrichenko, Kevin Pedro, Nick Smith [FNAL]

Performing two complete CMS analyses

- Dark Higgs search
- Boosted SM $H \to b\bar{b}$

using my awkward-array event processing, Ben's distributed processing with Spark, and Andrew Melo's Spark cluster at Vanderbilt.

Data granularity

We've found that it's useful to work with analysis data as individual ROOT branches converted into jagged arrays, rather than event records.

- efficient processing in Python
- analysis code readability
- columnar granularity

Data granularity

We've found that it's useful to work with analysis data as individual ROOT branches converted into jagged arrays, rather than event records.

- efficient processing in Python
- analysis code readability
- columnar granularity

One-plot queries (not yet realistic) would require $< 1\%$ of the columns but nearly all of the rows (events).

Whole-analysis workflows (demonstrated with Coffea) require $\sim 10\%$ of the columns with loose trigger cuts.

Data granularity

We've found that it's useful to work with analysis data as individual ROOT branches converted into jagged arrays, rather than event records.

- efficient processing in Python
- analysis code readability
- columnar granularity

One-plot queries (not yet realistic) would require $< 1\%$ of the columns but nearly all of the rows (events).

Whole-analysis workflows (demonstrated with Coffea) require $\sim 10\%$ of the columns with loose trigger cuts.

For applications like this, the unit of data is columns, not files or events.

Data delivery to query system

One of IRIS-HEP DOMA's projects is to create an intelligent Data Delivery System (iDDS), which would respond to requests for data in high-level terms ("muon kinematics in the first million events of 2018 data"), rather than low-level terms ("bytes 1024–2048 of some/file.root").

Data delivery to query system

One of IRIS-HEP DOMA's projects is to create an intelligent Data Delivery System (iDDS), which would respond to requests for data in high-level terms ("muon kinematics in the first million events of 2018 data"), rather than low-level terms ("bytes 1024–2048 of some/file.root").

For query applications, we would be requesting column segments, maybe with very simple cuts (e.g. trigger lines). Complex filtering happens in query system.

Data delivery to query system

One of IRIS-HEP DOMA's projects is to create an intelligent Data Delivery System (iDDS), which would respond to requests for data in high-level terms ("muon kinematics in the first million events of 2018 data"), rather than low-level terms ("bytes 1024–2048 of some/file.root").

For query applications, we would be requesting column segments, maybe with very simple cuts (e.g. trigger lines). Complex filtering happens in query system.

This is a good match to what Skyhook can deliver:

- format-aware data delivery;
- reformatting: decompression, recompression, concatenation, slicing;
- intelligence close to the source.

The ROOT file format is complicated

Just finding the bytes for "muon p_T in events 1–1000" is a non-trivial task, requiring infrastructure like ROOT, uproot, groot, root4j, etc.—which places limits on how "close to storage" the intelligence can go.

The ROOT file format is complicated

Just finding the bytes for "muon p_T in events 1–1000" is a non-trivial task, requiring infrastructure like ROOT, uproot, groot, root4j, etc.—which places limits on how "close to storage" the intelligence can go.

However, each logical entity *has* a fixed byte range in the files. A full framework can identify these ranges and a save them to an index to simplify lookups.

The ROOT file format is complicated

Just finding the bytes for "muon p_T in events 1–1000" is a non-trivial task, requiring infrastructure like ROOT, uproot, groot, root4j, etc.—which places limits on how "close to storage" the intelligence can go.

However, each logical entity *has* a fixed byte range in the files. A full framework can identify these ranges and a save them to an index to simplify lookups.

`https://github.com/diana-hep/uproot-skyhook` uses uproot to map logical row/entry numbers and column names to TBasket locations in the ROOT files. The index is formatted as Flatbuffers (Jeff's preferred format)—any process that can read Flatbuffers can deliver jagged arrays from the indexed ROOT files.

Sample: the Flatbuffers schema

```
include "interpretation.fbs";

enum Compression: int {
  none = 0,
  zlib = 1,
  lzma = 2,
  old = 3,
  lz4 = 4
}

table Branch {
  local_offsets: [ulong] (required);
  page_seeks: [ulong] (required);
  compression: Compression;
  iscompressed: [bool];
  compressedbytes: [uint];
  uncompressedbytes: [uint] (required);
  basket_page_offsets: [uint] (required);
  basket_keylens: [uint];
  basket_data_borders: [uint];
}
```

```
table Column {
  interp: uproot_skyhook
      .interpretation_generated
      .Interpretation (required);
  title: string;
}

table File {
  location: string (required);
  uuid: string (required);
  branches: [Branch] (required);
}

table Dataset {
  name: string (required);
  treepath: string (required);
  colnames: [string] (required);
  columns: [Column] (required);
  files: [File] (required);
  global_offsets: [ulong] (required);
  location_prefix: string;
}
```

Current status—before seeing Jeff's talk today!

- What Skyhook provides is a good match to what iDDS is trying to achieve.
- The unit of data for a query system is a column segment.
- I provided Jeff with a way to generate mappings from high-level entities—row/entry numbers and column names—to low-level byte positions.
- Once a prototype works, it could be tested with Coffea.