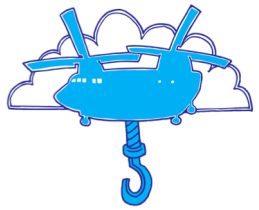
SkyhookDM projection-only pushdown and Arrow dataset integration into Skyhook objects

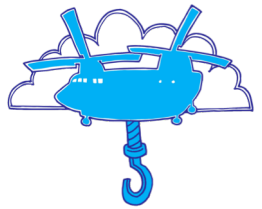
Xiongfeng Song

(Mentored by Jeff LeFevre, Ivo Jimenez, Noah Watkinns)



CROSS | CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

SkyhookDM – “Programmable storage”

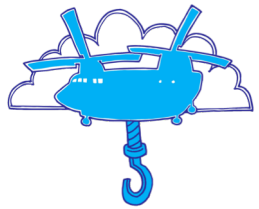


- Utilizes and extends Ceph distributed object storage with customized C++ "object classes”

- Supports database operations such as SELECT, PROJECT, AGGREGATE to be offloaded (i.e., pushed down) directly into the object storage layer

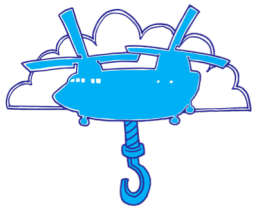
- Supports row-based Flatbuffers and column-based Arrow

Goal Overview



- Phase 1: Skyhook projection-only pushdown for Arrow tables
- Phase 2: In storage Arrow ListArray datatype operations
- Phase 3: Setup Arrow-native SkyhookDM Architecture

Phase 1: Why we want to do this?



For example:

`SELECT a, b WHERE a < 2 && c > 1` => ["a", "b", "c"]

`SELECT a, b WHERE a > 1` => ["a", "b"]

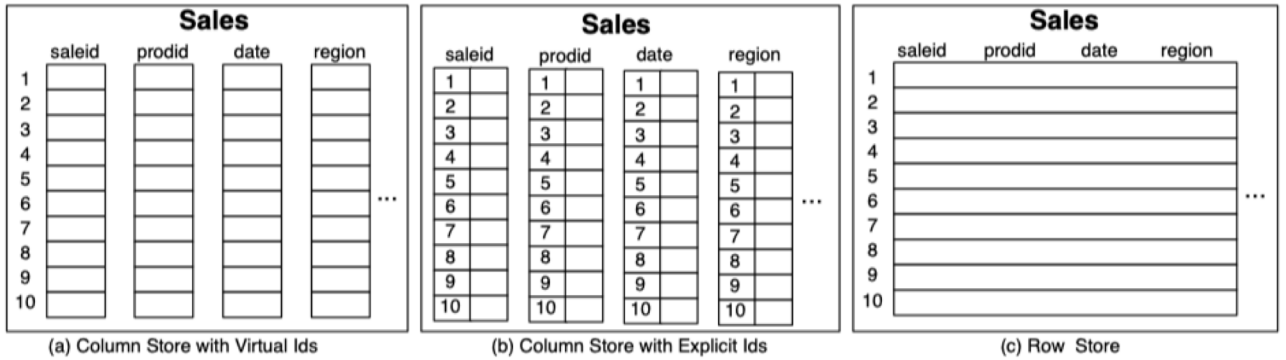
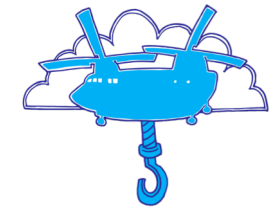


Figure 1.1: Physical layout of column-oriented vs row-oriented databases.



What we want to implement.

Input Query
projection: muon
predicate: A=true, B=true

run-query.cc
Set query schema and query predicate for cls

query schema (for projection): muon

query predicate: A=true, B=true

Modify the query schema as {muon, A, B} and query predicate as empty, pass it to cls.

launch aio_exec

exec_query_op
process query using cls

return raw_result

use_cls returns all columns which are related to projection and predicate

query.cc

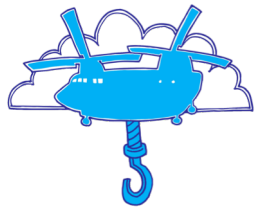
Real predicate will be done in query.cc's more_processing

if more processing

Currently, only if !use_cls, it is possible to do more processing

query.cc
use sky_qry_schema and sky_qry_preds to process the previous result

Use more processing, sky_qry_schema is muon, sky_qry_preds is A=true, B=true

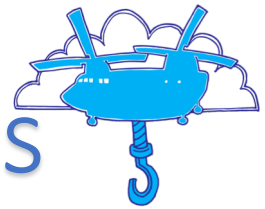


Major commits:

- Add a new flag pushdown-cols-only (default false) to the run-query.cc. When the flag is set, the query execution will only push down columns projection to storage level and leave predicates processing to the more_processing step in query.cc.
- Reconstruct schema and preds in more processing, fixed print data
- Append columns directly when there is no predicate and row_nums is empty
- Extract indexing lookup from exec_query_op

See https://github.com/uccross/skyhookdm-ceph-cls/pull/55

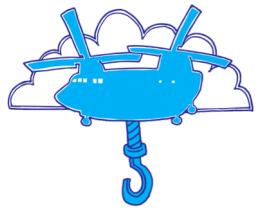
Phase 2: List array and reducers operations



- A common data format in HEP data is Jagged/Awkward Array.
- Currently these jagged arrays are stored in skyhook as Arrow::List types.
- Each entry in the column (row) is a list of arbitrary and non-uniform length.

- We would like to implement reducer ops like: min, max, count…
- However, we found arrow compute APIs

Apache Arrow compute APIs



api_scalar: Add(), Subtract(), And, Or, XOR, IsIn()…

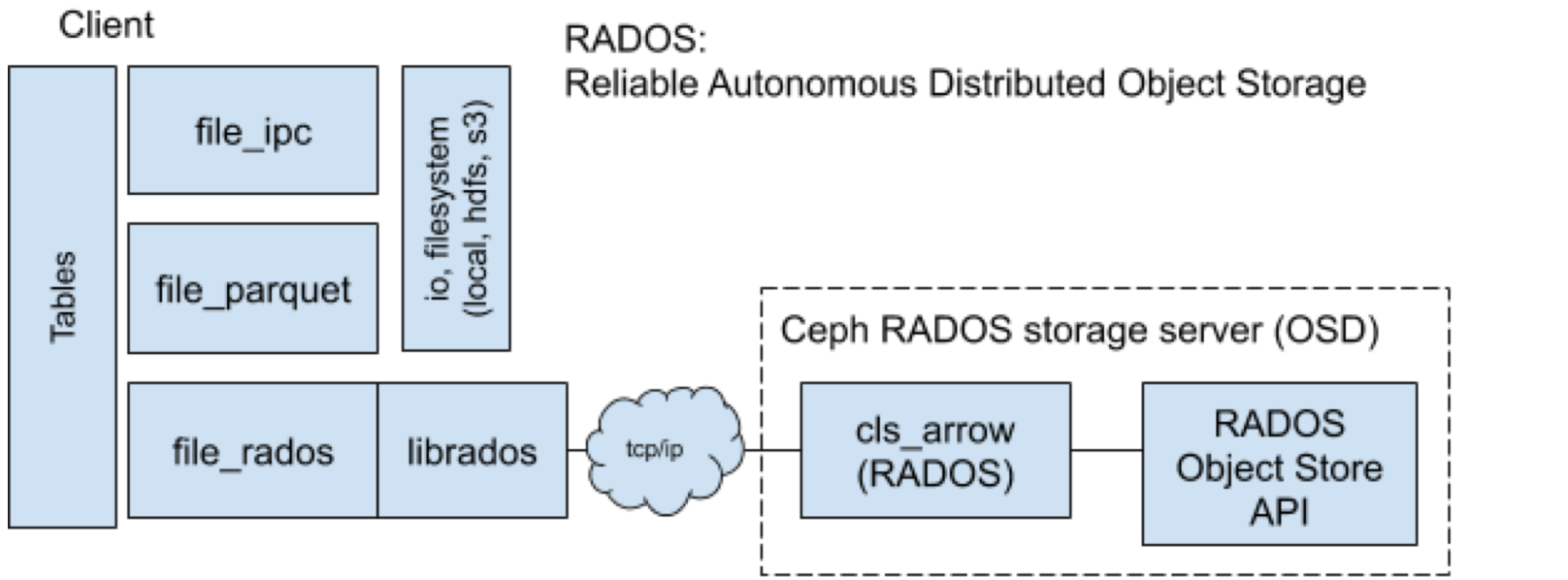
api_aggregate

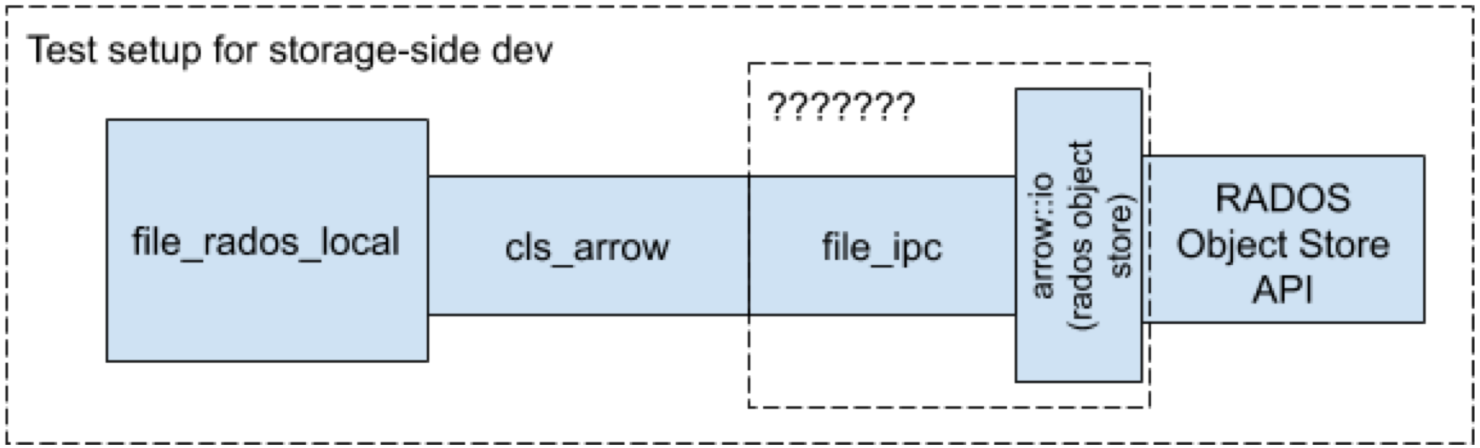
api_vector: Filter(), Take()…

Aggregations

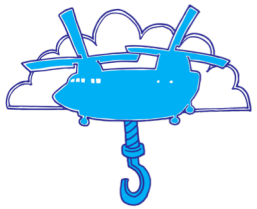
Function name	Arity	Input types	Output type	Options class
count	Unary	Any	Scalar Int64	`CountOptions`
mean	Unary	Numeric	Scalar Float64	
min_max	Unary	Numeric	Scalar Struct (1)	`MinMaxOptions`
sum	Unary	Numeric	Scalar Numeric (2)	

Phase 3: Move to Arrow!





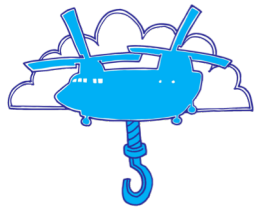
Works done for phase 3

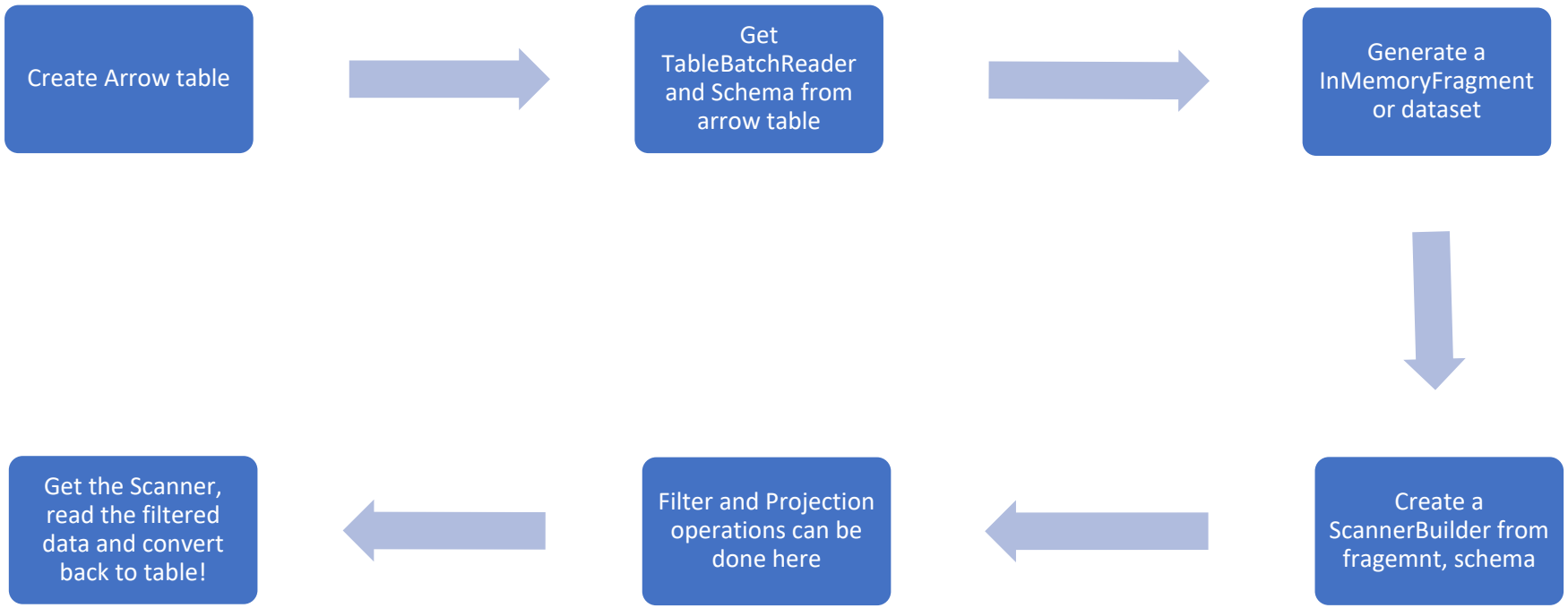


- Issue 1: arrow-cls setup for running unit tests

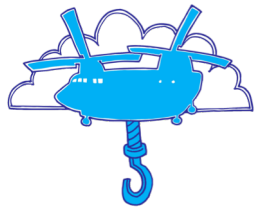
- Issue 2: InMemoryFragment interface to object store’s API

In Memory Fragment





In Memory Fragment



```
/// \brief Scanner is a materialized scan operation with context and options
/// bound. A scanner is the class that glues ScanTask, Fragment,
/// and Dataset. In python pseudo code, it performs the following:
///
///  def Scan():
///    for fragment in self.dataset.GetFragments(this.options.filter):
///      for scan_task in fragment.Scan(this.options):
///        yield scan_task
class ARROW_DS_EXPORT Scanner {
```

```
/// \brief Set the filter expression to return only rows matching the filter.
///
/// The predicate will be passed down to Sources and corresponding
/// Fragments to exploit predicate pushdown if possible using
/// partition information or Fragment internal metadata, e.g. Parquet statistics.
///
/// \param[in] filter expression to filter rows with.
///
/// \return Failure if any referenced columns does not exist in the dataset's
///         Schema.
Status Filter(std::shared_ptr<Expression> filter);
```

Thank you !