CROSS | CENTER FOR RESEARCH IN OPEN SOURCE SOFTWARE

iris hep Institute for Research & Innovation in Software for High Energy Physics

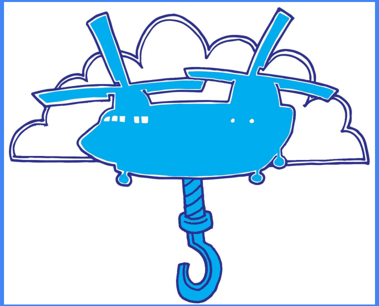
# SkyhookDM: Ability to Push Back Query Execution to Client in Case of Overloaded OSDs

By Eshan Bhargava
Mentor: Jianshen Liu

## Background

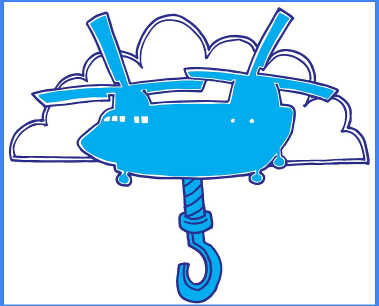
- SkyhookDM
  - Client Server Architecture
  - An extension of Ceph for scalable storage of tables
- Work is offloaded to Ceph to process, rather than the client doing work
  - The Arrow Dataset API allows the definition of SQL-style expressions to filter data when reading data from files.
  - Normally, expressions are applied after the client receives data from a file system.
  - But with Skyhook, these expressions are applied at the server-side.
    - Reduces data sent to client.

## Problem



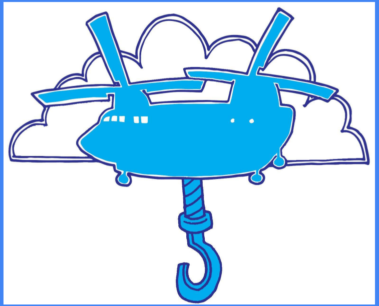
- Problem: Storage servers in Ceph may be busy
  - If clients are always pushing down, the server may become overloaded.
  - Then may want to reject the request to apply the expressions to the data.
  - For improved performance, it may be optimal to ‘pushback’ the filtering to the client.
- Solution: Apply expressions at Client
  - The storage receives the read request, and decides that it cannot apply the filter to the data.
  - Filters and data are pushed back to the client.
  - When the client receives this result, it can apply the expression to the data.

## Progress



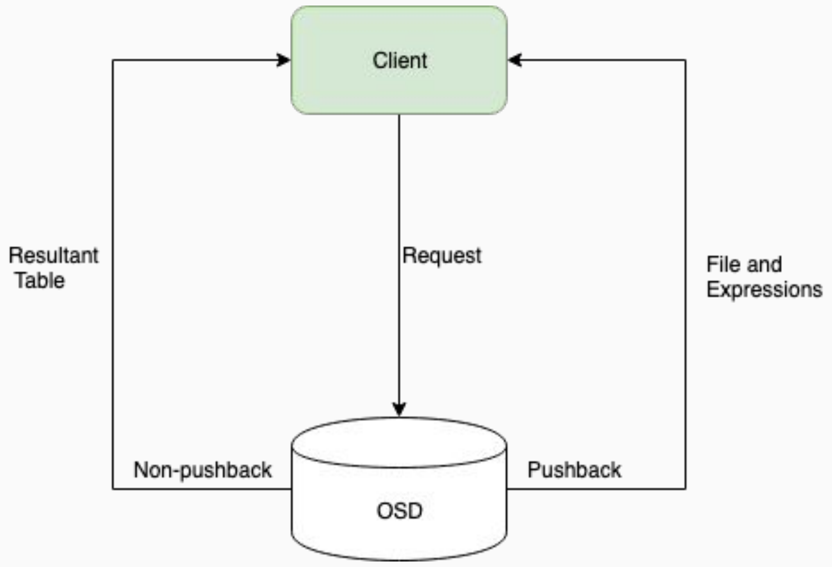
- We have finished the main function
    - For the parquet file format, we have better resource estimation.
    - We can also support the feather file format, but we are working on getting better resource estimation.
- Working on benchmarking

## Implementation

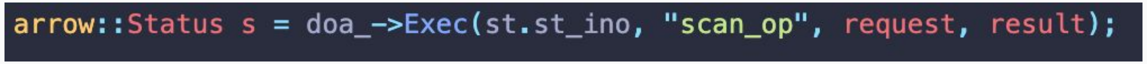


- Iterate over a parquet file's rows and columns to find the total_uncompressed_size of the file in bytes.
  - The iteration is over the metadata of parquet files, so the overhead of making the pushing back decision on the server side is negligible.
- Then compare that value to a threshold representing the available RAM of the system.
- We compare the CPU load from the sysinfo API to another threshold, representing system load.
- Return a predefined status code, representing pushback.
- Use status detail class to return this code to the client.
- The expressions and unprocessed data are returned from the server in a bufferlist.
- At client, if the scan_op returns this code, apply filtering.

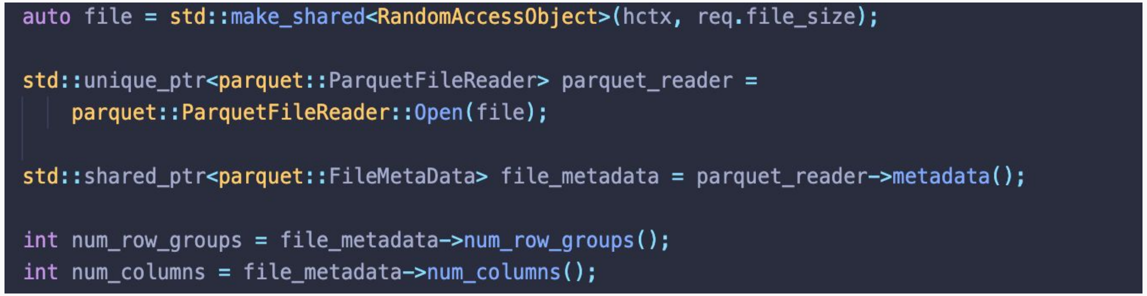
## High Level Figure



## Implementation

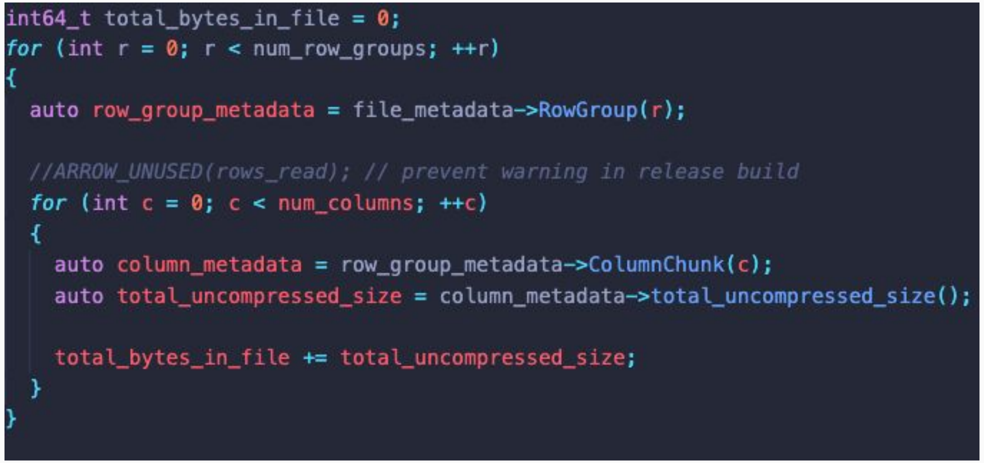


```cpp
arrow::Status s = doa_->Exec(st.st_ino, "scan_op", request, result);
```

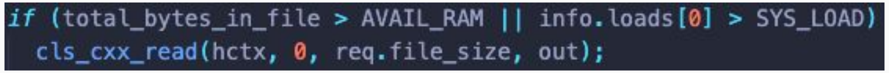


```cpp
auto file = std::make_shared<RandomAccessObject>(hctx, req.file_size);

std::unique_ptr<parquet::ParquetFileReader> parquet_reader =
    parquet::ParquetFileReader::Open(file);

std::shared_ptr<parquet::FileMetaData> file_metadata = parquet_reader->metadata();

int num_row_groups = file_metadata->num_row_groups();
int num_columns = file_metadata->num_columns();
```

# Implementation

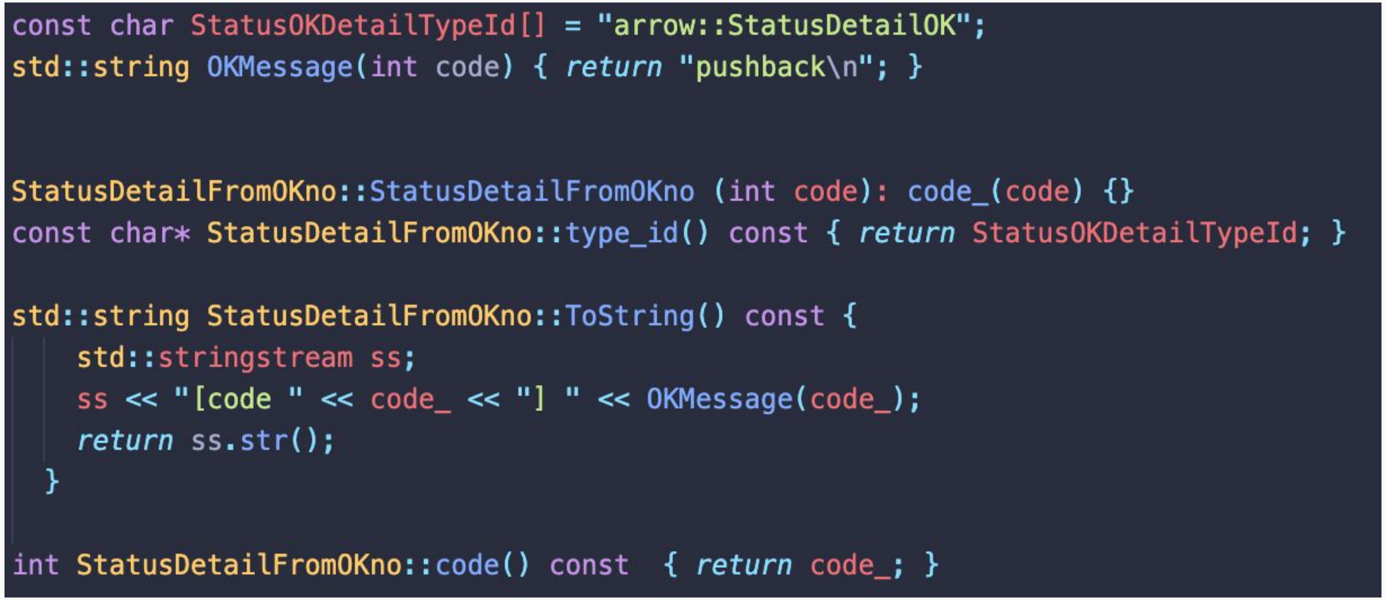


```
int64_t total_bytes_in_file = 0;
for (int r = 0; r < num_row_groups; ++r)
{
  auto row_group_metadata = file_metadata->RowGroup(r);

  //ARROW_UNUSED(rows_read); // prevent warning in release build
  for (int c = 0; c < num_columns; ++c)
  {
    auto column_metadata = row_group_metadata->ColumnChunk(c);
    auto total_uncompressed_size = column_metadata->total_uncompressed_size();

    total_bytes_in_file += total_uncompressed_size;
  }
}
```



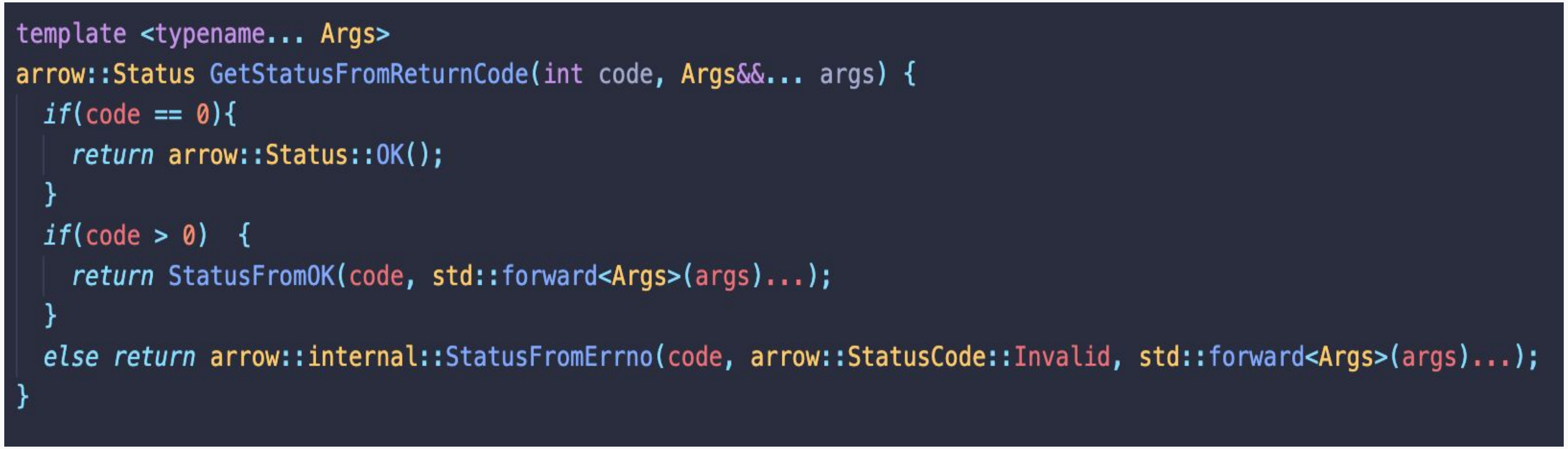
```
if (total_bytes_in_file > AVAIL_RAM || info.loads[0] > SYS_LOAD)
  cls_cxx_read(hctx, 0, req.file_size, out);
```

# Implementation

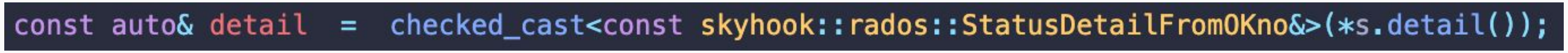


```
const char StatusOKDetailTypeId[] = "arrow::StatusDetailOK";
std::string OKMessage(int code) { return "pushback\n"; }


StatusDetailFromOKno::StatusDetailFromOKno (int code): code_(code) {}
const char* StatusDetailFromOKno::type_id() const { return StatusOKDetailTypeId; }

std::string StatusDetailFromOKno::ToString() const {
    std::stringstream ss;
    ss << "[code " << code_ << "] " << OKMessage(code_);
    return ss.str();
  }

int StatusDetailFromOKno::code() const  { return code_; }
```

## Implementation



```
template <typename... Args>
arrow::Status GetStatusFromReturnCode(int code, Args&&... args) {
  if(code == 0){
    return arrow::Status::OK();
  }
  if(code > 0)  {
    return StatusFromOK(code, std::forward<Args>(args)...);
  }
  else return arrow::internal::StatusFromErrno(code, arrow::StatusCode::Invalid, std::forward<Args>(args)...);
}
```



```
const auto& detail  =  checked_cast<const skyhook::rados::StatusDetailFromOKno&>(*s.detail());
```

# Thank You!

- I would like to thank CROSS and IRIS-HEP for giving me the opportunity to work with Skyhook.
- I would also like to thank my mentor, Jianshen Liu, for his help.
- Questions?