Arrow Native Storage with SkyhookDM

Proposal for IRIS-HEP Fellowship Program (January 10 - July 10, 2021)

Jayjeet Chakraborty
4th Yr, B.Tech, Undergraduate
National Institute Of Technology, Durgapur, India

Overview

Apache Arrow is a columnar in-memory format for seamless data transfer between different big data systems. It mitigates the need for serializing and deserializing data. It has native abstractions for use in Big Data storage systems. We aim to convert SkyhookDM into an Arrow-Native storage system by utilizing the Object class SDK provided by Ceph to add a layer in its storage side using the Arrow C++ SDK to allow querying and processing of tabular datasets stored as objects in Apache Arrow format both in the storage and client side. We aim to upstream the Rados specific implementations of the Arrow C++ SDK also. Native support for Arrow will allow applications such as Coffea Processors, and ServiceX transformers to seamlessly interact with SkyhookDM, as well as other storage systems.

Expected Deliverables

1. Object class libraries that use RADOS object classes SDK to run queries and computations on tabular data partitions stored in Apache arrow format in the storage side.
2. Create example applications with Dask, Ray, or Spark to portray how to use Arrow with Ceph as its storage backend and Integrate Coffea processors with SkyhookDM.
3. Documentation and tutorials of how to create applications using SkyhookDM.

4. Automated scripts for continuously deploying and testing these new extensions, leveraging our work for summer 2020 as part of the IRIS-HEP fellowship.
5. Report on performance benchmarks on the implementation of client and storage side processing.

Project Timeline

My course work usually takes **12 - 15 hours** a week, so the project timeline is prepared taking into consideration the time spent in coursework and exams. Starting from January 10, 2021 till July 10, 2021, I have described the timeline as follows,

- **Week 1 - 2**
  - Study the Apache Arrow C++ SDK and find out the relevant abstractions that can be utilized directly and those which need to be implemented.
  - Discuss design decisions with my mentors and prepare an implementation plan and requirements document.

- **Week 3 - 4**
  - Setup an unit testing environment using `gTest` and look into how to mock `librados` using libraries like `gMock`. Write a mock for `librados`.
  - Start with implementing the Client side API implementations using a `librados` mock as the backend. Write tests simultaneously to follow a test-driven development approach.

- **Week 6 - 8**
  - Continue working on the Client side and stabilize the C++ API. Start implementing the CLS side for reading/writing Arrow Tables to/from Rados Objects.
  - Also, Work on bringing the CLS code base into the Arrow source tree to reuse several utility functions and to merge the Arrow and CLS build systems.

- **Week 9 - 10**
  - Continue building the Arrow CLS by implementing Scan operations like SELECT, PROJECT, AGGREGATE using the Arrow framework and Rados Object classes SDK on the Storage side. Finalize the initial implementation and get reviewed by mentors.

- **Week 11 - 12**
  - Develop a plan to set up a multi-node testing environment on River SSL and gather initial benchmarks of the implementation. Find bugs, performance issues due to zero-copy violations, edge cases and fix them.

    - Start writing Python bindings for the C++ side of the client.

- **Week 14 - 16**
    - Work towards having the Python bindings tested and finalized to be able to prepare sample notebooks and open a PR with the Rados implementation to the Arrow upstream repository.
    - Also, implement API's for Schema discovery, Partition discovery, metadata and statistics storage for query optimizations.

- **Week 17 - 18**
    - Write a few example applications using Dask, Ray, or Spark with our SkyhookDM Python client to show how to leverage Ceph as the storage backend while using Arrow.
    - Complete work on the example applications and publish them as tutorials or blogs.

- **Week 20 - 21**
    - Start exploring the possibility and design of using SkyhookDM as a cache for ServiceX.
    - Also, start work on the Coffea processor integration with SkyhookDM to perform Dask based parallel computations of ROOT files stored as Arrow IPC objects in the OSDs.

- **Week 22 - 24**
    - Complete the initial end-to-end Coffea processor integration and ServiceX integration with SkyhookDM, write blogs, prepare example notebooks, guides, and documentation.
    - Gather benchmarks for the Coffea + SkyhookDM+ ServiceX pipeline.

Exam Timeline

- **Week 5:** Continuous Assessment I
- **Week 13**: Continuous Assessment II
- **Week 19:** Final Semester Exams

These milestones have been structured so that they align with the NSF milestones for Fall 2020 of the SkyhookDM project that is part of the IRIS-HEP project.

Bibliographical Information and Previous Work

I am a senior year B.Tech student studying Computer Science from the National Institute Of Technology, Durgapur, India. I have been an IRIS-HEP 2020 Summer Fellow (6/15/2020 -

10/15/2020), working on the project "Reproducible Large Scale Experiments on SkyhookDM-Ceph" (see report here). I have also been working with CROSS, UC Santa Cruz on the Popper 2.0 project since Google Summer Of Code 2019 and have added various new features including additional runtimes like Singularity and resource managers like Kubernetes and Slurm. I also worked on popperizing MLPerf workflows, workflows for infrastructure automation, and workflows for running ML/AI workload in HPC.

Contact Info

Name: Jayjeet Chakraborty

Email: jchakra1@ucsc.edu, jayjeetchakraborty25@gmail.com

GitHub: https://github.com/JayjeetAtGithub

Twitter: https://twitter.com/heyjc25