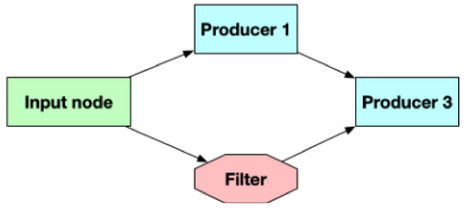
Multi-Threaded Graph Processing Framework For Dune

Clifton Terwilliger, University of Alaska Anchorage – SIST Intern

Poster# [FERMILAB-POSTER-22-119-STUDENT]

Project Introduction

The dune project will require a lot of data collection, and data processing. Thus, logically will require a lot of multi-threaded code. However, good multi-threaded code is very hard to write, and tends to be a very time-consuming process. This can lead to poorly written multi-threaded code, which can be very hard to debug and make thread safe. My project for my internship this summer was to help explore and create a framework that should help solve these problems. The framework was created in C++ using an existing library adapted to our use case and is based on graph processing.



Example Graph

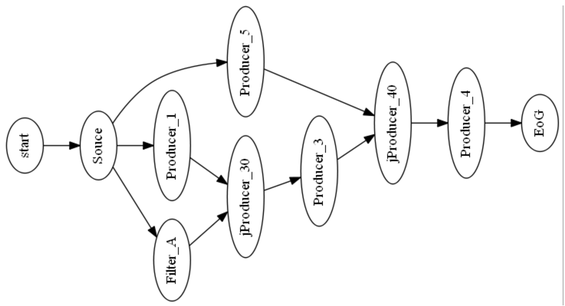
Framework Emphasis

This project had an emphasis on using modern C++ features to streamline the process of using the framework. These modern features allowed for automation that was not previously possible. These new features include dynamic type deduction, and template guides and many more. Furthermore, there was a large emphasis on making the framework hard to make thread unsafe.

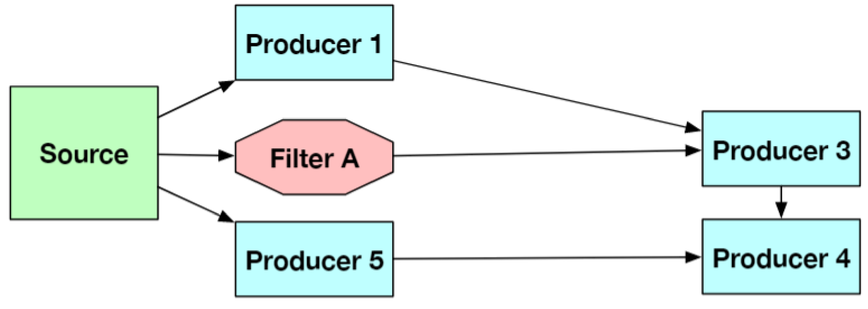
Details

Ideally, all that the user of the framework would only have to input for a name, and identifier for data(key) and function for each node. Then list the connections in the graph. The framework should handle the rest.

Under the hood, the framework automatically does many things. First, it's automatically deducing what kind of node the user is adding to the graph, then deduces what data types and stores and inputs data to the function based on the associated keys. Then it constructs the graph based on the edges given by the user. Some additional nodes are added to properly control data flow.



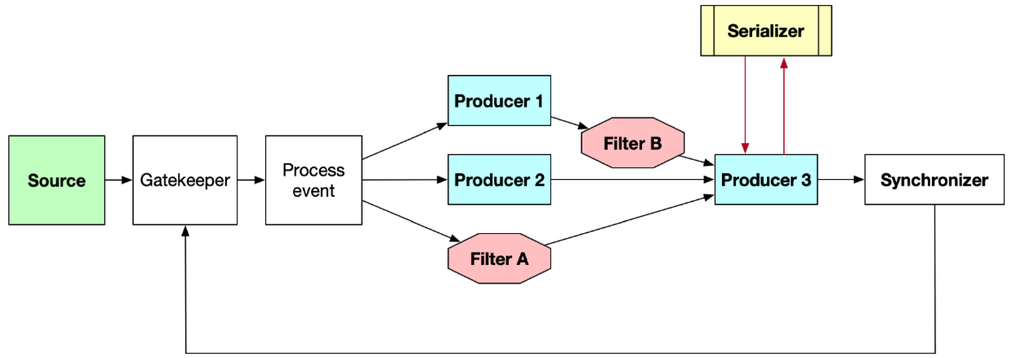
Example of What The Framework Produces



Example graph

Desired Framework Attributes

The main goal was to make a framework that is easy to use. However, the framework should also be lightweight, as to save resources and time. Additionally, the framework should be able to take full advantage of allocated resources. Overall, this framework could potentially reduce code development time, decrease required computing resources, and most importantly hopefully reduce errors due to poorly written code.



Example graph of Final Framework

Acknowledgements

Thanks to my supervisor Kyle Knoepfel, for guiding me, and giving me aid through out this project. Especially his help with modern C++ features. Additionally, I would like to thank Arden Warner, and Jonathan Eisch, my mentors during this internship. Special thanks to the SIST committee for making this project and internship possible.

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.