Monitoring DUNE Data Streaming Efficiency

Lisa Paton - University of Alaska Anchorage, SIST Intern
13 August 2021

Introduction

The Deep Underground Neutrino Experiment (DUNE) is an international collaboration to study neutrinos at the Long-Baseline Neutrino Facility (LBNF), which is currently under construction at the Sanford Underground Research Facility in South Dakota.

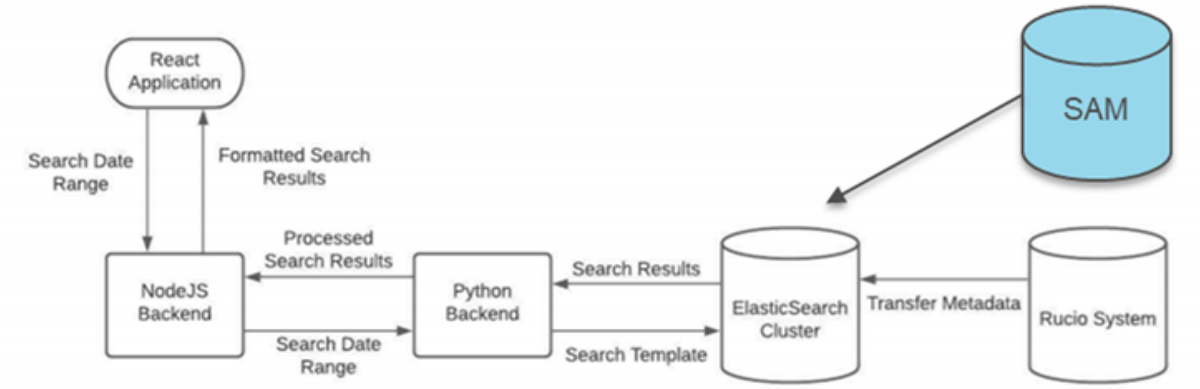
There are 36 computing sites available for DUNE to run jobs, 15 of which have available storage. Since there isn't storage at every site, it is necessary to stream data to run simulation and analysis tasks. Streaming is used to transfer files since the entire file is not always needed to run a job. This reduces network traffic and speeds up the time to run a job because it takes more time to transfer a whole file than to stream the needed portions.

File streaming is done using the Xroot protocol, which is run by Xrootd.

Methods

React App Overview

React App, an open-source JavaScript library, is used to build user interfaces for web applications. This year, a group of interns at Oregon State University set up a React application to monitor transfer data from Rucio, a file transfer system. The students (Luke Penner, Zachary Lee, Lydia Brynmoor, and Sean Gilligan) worked under Dr. Heidi Schellman to build this intuitive data transfer display.



Above is an Architecture Diagram of React App, adapted from [1].
I've added in a blue cylinder SAM (sequential access metadata). This is to show the changes made from a copy of the app I worked on this summer.

File transfer systems and ElasticSearch

Data from a file transfer system (either SAM or Rucio depending on the version being used) is fed into ElasticSearch.

SAM (Sequential Access via Metadata), is a data-handling system to store and retrieve files and associated metadata, including a complete record of the processing that has used the files. SAM, which has been in use for data handling since before 2001 [5], is slowly being phased out. Rucio, named after the donkey from Don Quixote, is a replica manager. The program knows where things are, and can move files from one location to another [6]. With time, rucio will replace the file transfer portion of SAM.

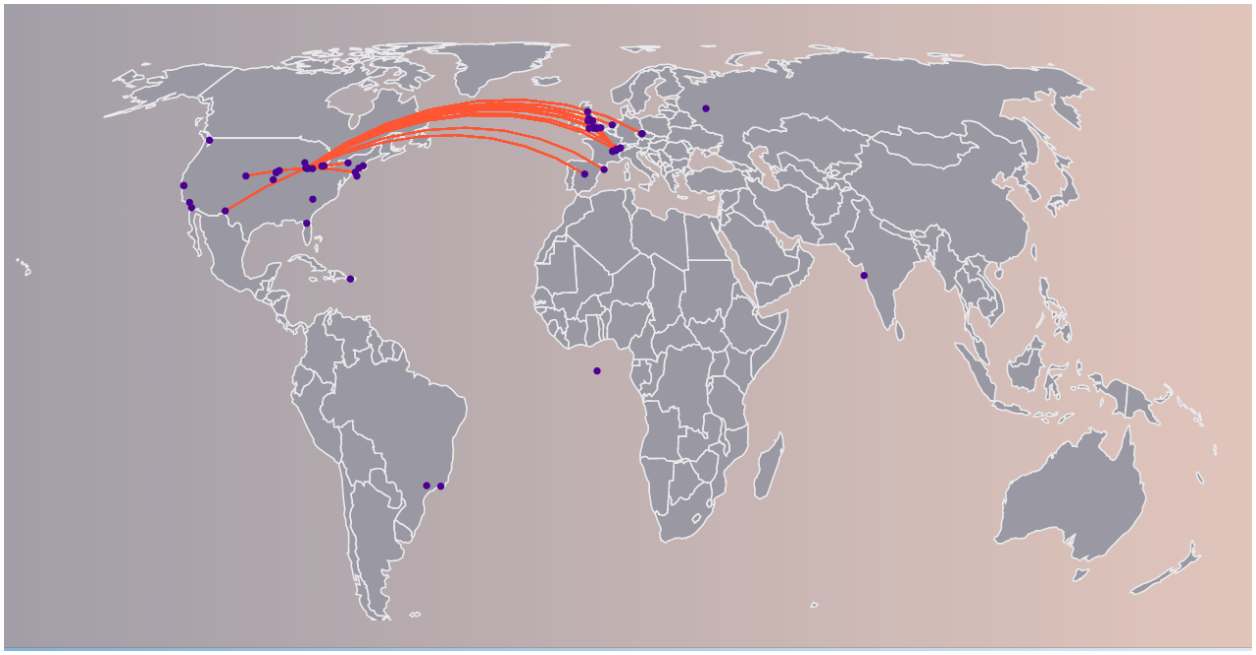
ElasticSearch is a scalable application that stores data with record-like structures to make data accessible. ElasticSearch deletes data after 6 months, and will not return more than 10,000 events in a query.

Python Backend

ElasticSearch is queried by the Python backend. This is where most of the changes are between one copy to the other. By changing the search template, we were able to get Xroot streaming data instead of Rucio transfer data. Once the data has been collected from ElasticSearch, it is reformatted to be compatible with the NodeJS backend.

NodeJS and ReactApp

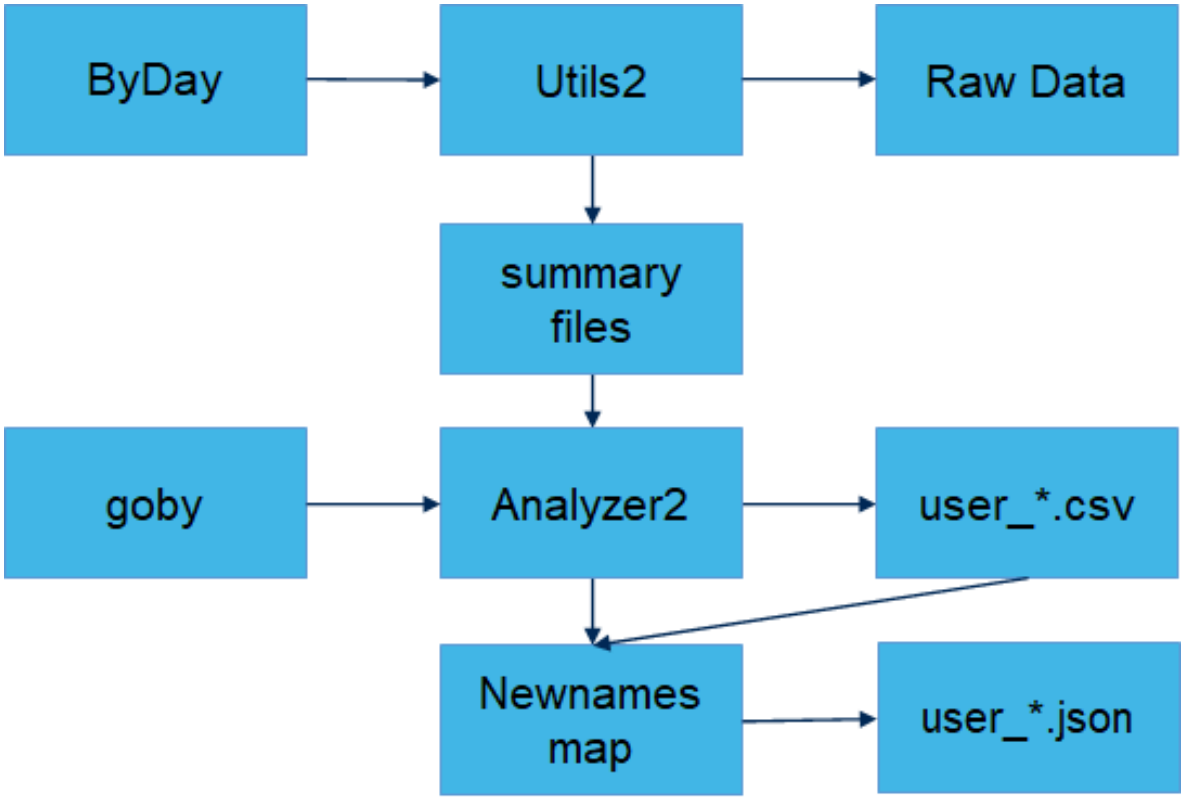
The NodeJS backend displays the data onto the React web app, where a user can select a date range to display data for.



Here is an example of a transfer display map, a screenshot taken from http://fermicloud514.fnal.gov:3000/. Each blue dot represents a DUNE computing site and connecting red lines show data transfer paths.

Python Backend Overview

The python backend exists to query ElasticSearch data and to make that data available to the NodeJS backend. This is done by running the data fetching sequence shown in the flowchart on the next page.



First, running Utils2.py queries ElasticSearch, which holds the SAM Xrootd data, and creates raw data files. This data is then sorted and separated by day and project ID to make a day-by-day summary of transfers. ByDay.py is a short script that runs Utils2.py for each day in the specified date range taken from the user at the React App.

Next, user_*.csv files are made from the summary files by running Analyzer2.py

disk,user,date,process_id,timestamp,duration1.fnal.gov,dunepro,2021-07-02,16085913,2021-07-02T06:47:20.895Z,134162.20600008965,8228122221,dunepro,reco,skipped,us_fnal.gov,0.06132965807818114,dunepro-fife_wrap_20210630_142757_3076055,np04_raw_run005786_0031_dl7.root,raw,dunepro-21652711-0-fnpc7696.fnal.gov
,file_size,username,application,final_state,site,rate,project_name,file_name,data_tier,node
fndca1.fnal.gov,dunepro,2021-07-02,16085932,2021-07-02T06:47:09.453Z,63823.24799990654,8243280609,dunepro,reco,consumed,us_fnal.gov,0.1291579615160305,dunepro-fife_wrap_20210630_142757_3076055,np04_raw_run005786_0008_dl9.root,raw,dunepro-21652791-0-fnpc7512.fnal.gov
fndcao.fnal.gov,dunepro,2021-07-02,16085913,2021-07-02T06:47:20.895Z,134162.20600008965,8228122221,dunepro,reco,skipped,us_fnal.gov,0.06132965807818114,dunepro-fife_wrap_20210630_142757_3076055,np04_raw_run005786_0031_dl7.root,raw,dunepro-21652711-0-fnpc7696.fnal.gov

This example of a user_*.csv file shows DUNE Xroot transfers from 30 June 2021 to 01 July 2021. At this step, accompaning graphs are produced using the ROOT package. (To save time on processing, the graphing commands have been commented out in the Analyzer2test.py script).

From there, CSV files are put through newnamesmap.py. This is the step where we make our data compatible with the NodeJS backend. At this step, we translate source names between naming conventions (thus a script is named) and reformat the user_*.csv files to user_*.json files. Here is what the 30 June 2021 to 01 July 2021 CSV file looks like as a JSON file.


```
{
   "data": [
      {
         "Name": "np04_raw_run005786_0008_dl9.root",
         "Source": "fndca1.fnal.gov",
         "Destination": "us_fnal.gov",
         "File_size": "8243280609",
         "Start_time": "2021-07-02T06:47:09.453Z",
         "File_transfer_time": "63823.24799990654",
         "transfer_speed(MB/s)": "0.1291579615160305",
         "transfer_speed(B/s)": 129157
      },
      {
         "Name": "np04_raw_run005786_0031_dl7.root",
         "Source": "fndca1.fnal.gov",
         "Destination": "us_fnal.gov",
         "File_size": "8228122221",
         "Start_time": "2021-07-02T06:47:20.895Z",
         "File_transfer_time": "134162.20600008965",
         "transfer_speed(MB/s)": "0.06132965807818114",
         "transfer_speed(B/s)": 61329
      }
   ]
}
```


When translating a CSV file to a JSON file, some information is dropped because the JSON files only need to show what the NodeJS backend is looking for. In this example, data tier and node information was dropped in the second record.

Utils2.py, ByDay.py, and Analyzer2.py remained largely unchanged between the Rucio and the SAM copies. The noninvasive changes that did occur (like adding command-line arguments), happened in alternate scripts named Analyzer2test.py and ByDaytest.py.

Caching and Concatenation System

While querying ElasticSearch through the python backend, some issues came about. Either the user at the React web app would wait for many minutes for their data to display, some queries would not show all the data from the selected range, or the machine hosting the python backend would run out of working memory. Each of these issues could be addressed by using a caching system.

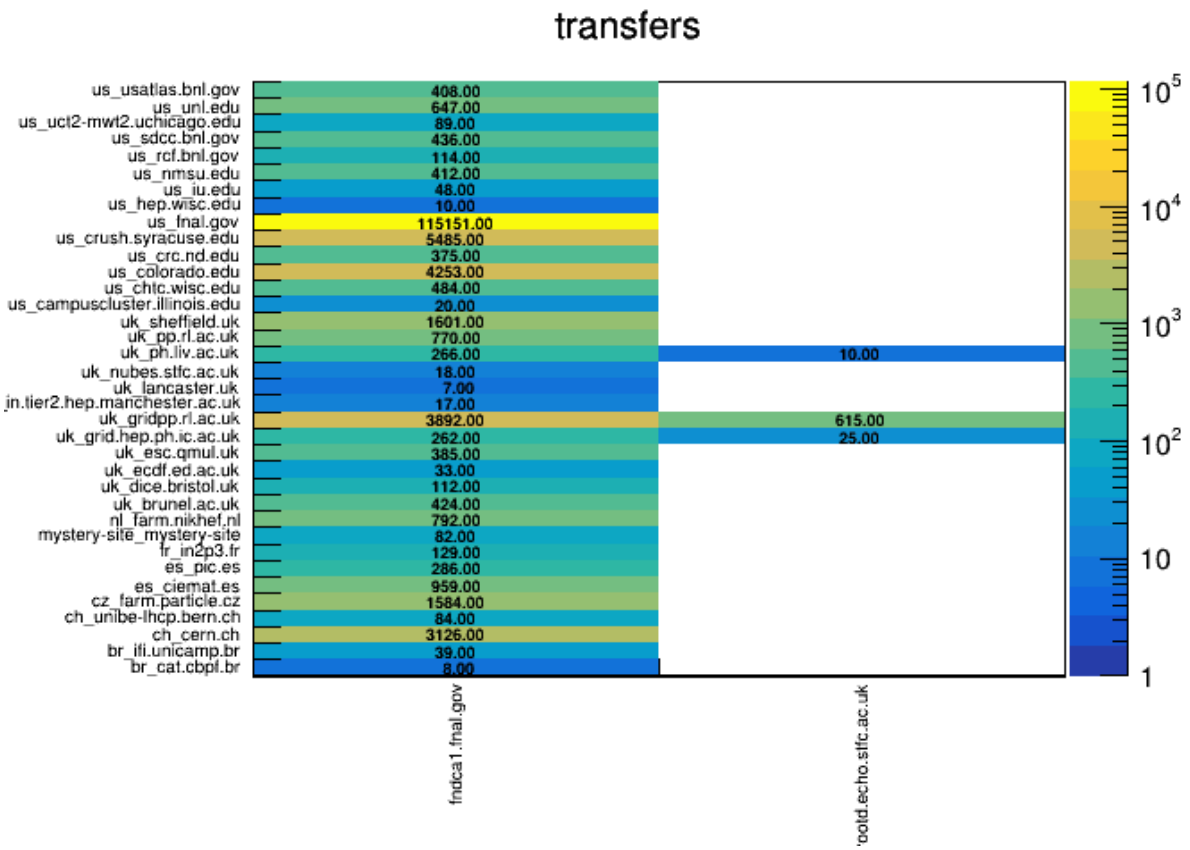
By preparing user_*.csv files for each day before a user asks for that data, we could concatenate the JSON output chunks for each day until the file spans the requested dates. We make per-day CSV files by running goby.py. Similar to ByDay.py for Utils2.py, goby.py runs Analyzer2test.py on day-long intervals over a specified date range. Then at the newnamesmap stage (previously called csvtojsonmulti), the JSON outputs are made, concatenated, and then exported to the NodeJS backend.

By caching and concatenating, we remove Analyzer2.py (or Analyzer2test.py) from the chain of programs a React App user would need to wait for before seeing their data displayed, making load times significantly quicker. Caching transfer data per day also prevents queries from exceeding 10,000 entries. Similarly, the virtual machine could not run out of RAM while making a CSV file for a single day. (We found that we didn't run out of RAM until running Analyzer2.py over a month-long range.)

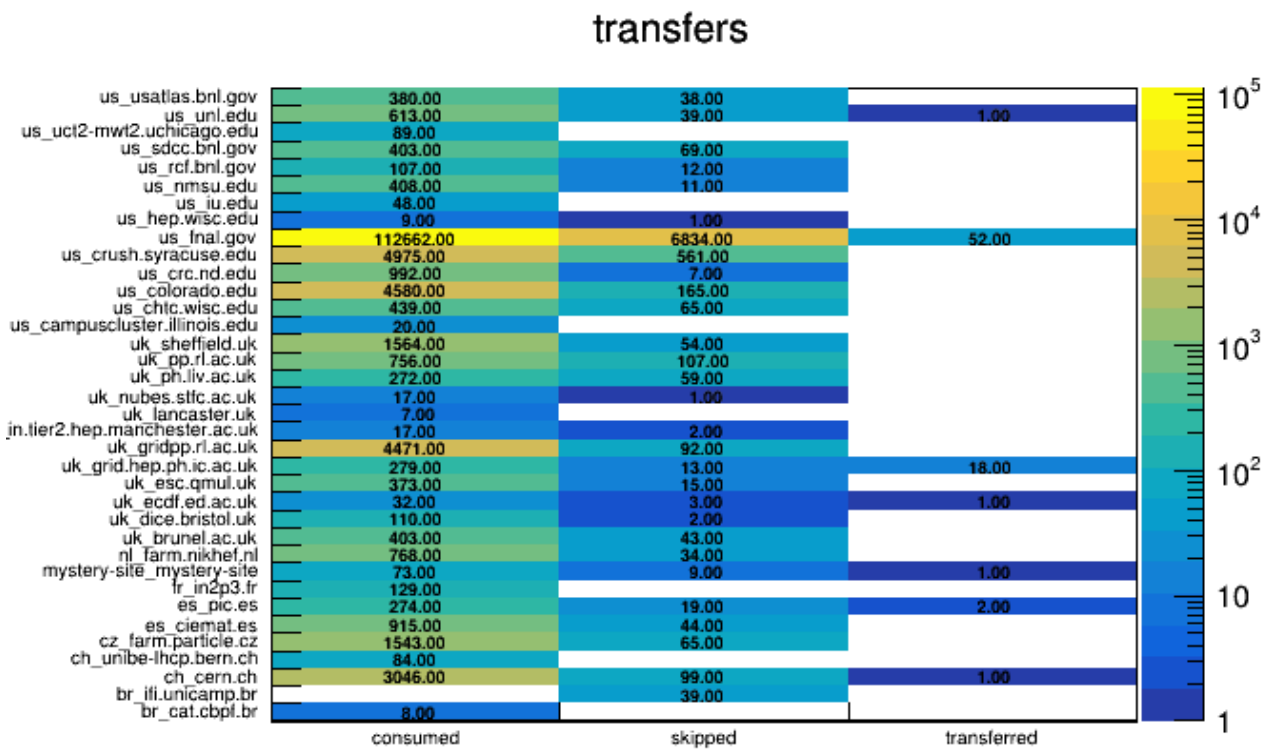
Hiccups in Data Fetching

While querying data and making daily CSV files for this year, we found holes in our data. The entire months of May and June didn't have any records in ElasticSearch for DUNE xroot transfer data. After comparing with Dr. Schellman and Kibana, we came to the conclusion that there were SAM transfer records, but these had been filtered out in some queries. We suspect that after filtering out SAM transfer events for DUNE that are not related to simulation or analysis, that there was nothing to show. These events are likely mechanical or diagnostic processes that don't need to be included for analysis. However, this gap in data should be explored more.

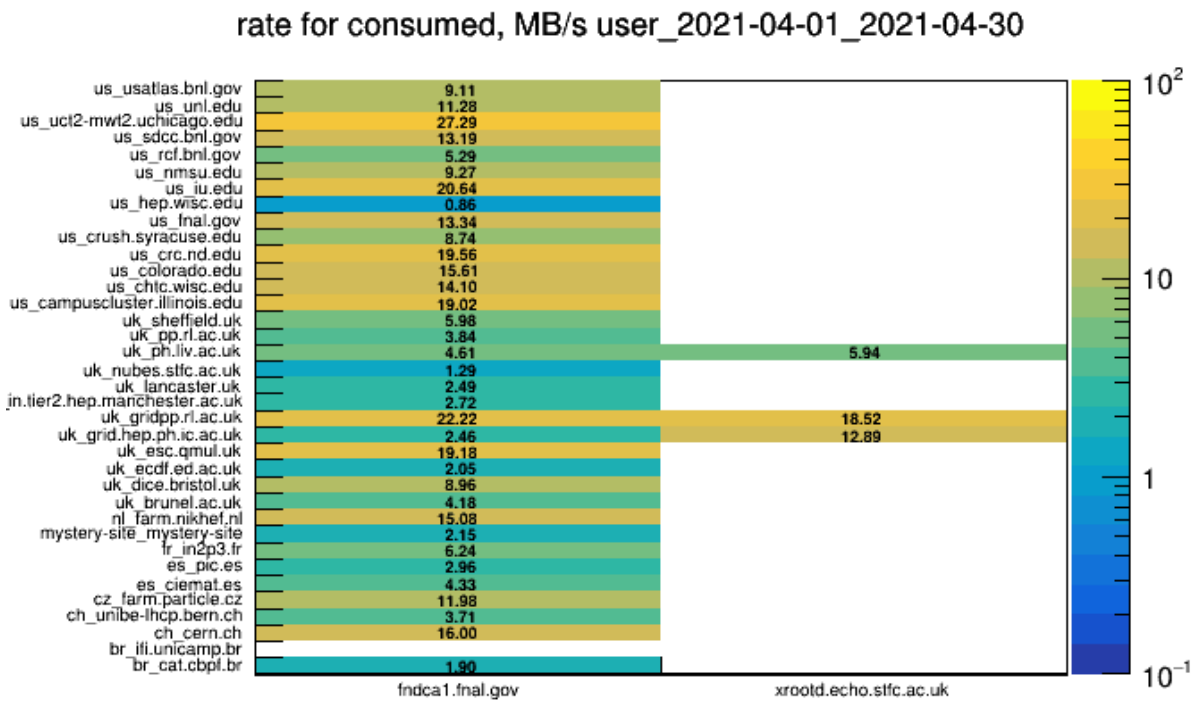
Analysis



This is an example of a graph made while running Analyzer2.py. It shows a heat map of DUNE Transfers over the month of April 2021. Sources are shown along the bottom, while destinations are along the left side. It's clear to see that most transfers in April were from Fermilab (FNAL) to Fermilab. In the future, this graph will look more evenly distributed as more data is collected overseas, and as more computing tasks are processed at other locations.



Like the first graph, this is a heat map of DUNE xroot transfers from April 2021 but separated instead by file state. Consumed files have been transferred, marked as successful, and closed. Skipped files have been closed, but were not marked as successful. Transferred files have been copied to the local disk, but have not been closed yet.



Once again, we have a heat map from April 2021 of DUNE SAM transfer data, this time showing the transfer rate of consumed files. We can see that this map is much more uniform than the previous two, where the yellow color denotes 100, rather than 10,000. In April, we had a range of rates between 0.86 and 27.29. Differences in streaming rates could be attributed to variations in bandwidth between the sites, other network traffic, slow local networks, slow local disks, etc.

Distance is an important factor when considering the time it takes to transfer data, since it will take more time to travel over longer paths. This should be taken into account when running Analysis tasks to save time and speed up job completion.

Conclusions

We were successfully able to display xroot transfer data from SAM through the React App. This will be useful when continuing xroot streaming data.

SAM project records, along with ElasticSearch records, give us plenty of information to determine the characteristics of DUNE applications. For analysis tasks, streaming should be done from the nearest copy to save time. This is not as important for tasks that don't use as much CPU, like reconstruction jobs. [5]

Future Work

More data streaming analysis needs to be done in preparation for DUNE. It will be important to predict network constraints (like traffic) and requirements (like bandwidth). To optimize performance, it will be useful to find the best job distributions for given bandwidths.

The data fetching sequence needs to be automated for the react app to continue working due to the limited life of ElasticSearch records (otherwise we could lose data). This process has been started for ByDay.py, but not yet for goby.py.

Since there have been some beneficial changes made in this branch, but have not yet been implemented in the other branches (and vice versa), the XrootParser branches should be merged.

Acknowledgments

I cannot give enough thanks to my supervisor Steven Timm, Heidi Schellman, the OSU team (Luke Penner, Zachary Lee, Lydia Brynmoor, and Sean Gilligan), my mentors (Aisha Ibrahim, Andres Quintero Parra, and Brian Vaughn), to the SIST Committee for making this possible, and to Beth Spangler for recommending me to apply.

References

[1] L. Penner, Z. Lee, L. Brynmoor, *Monitoring International Data and Connections*. Oregon State University, 2021.

[2] A. Dorigo, P. Elmer, F. Furano, A. Hanushevsky, *XROOTD - A highly scalable architecture for data access*. DE-AC02-76-SFO0515, Stanford University, 2004

[3] Terekhov, Igor & White, Victoria. Distributed data access in the sequential access model at the D0 experiment at Fermilab, article, July 5, 2000; Batavia, Illinois. University of North Texas Libraries, UNT Digital Library

[4] L. Janyst, New XRootD client plug-ins. Cern IT Department, 2014

[5] H. Schellman. Study of DUNE xrootd transactions. 2021.

[6] M. Barisits, T. Beermann, F. Berghaus, et al. Rucio: Scientific Data Management. Comput Softw Big Sci 3, 11 (2019).