Project Advisors: Alexander Sim, John Wu

Studying Data Access Patterns Using dCache Logs

Student: Jacob Aldrich

Problem and Significance

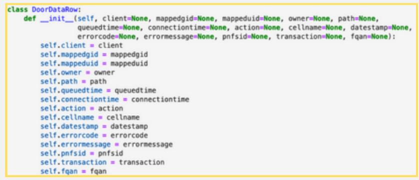
dCache is a storage management system which is used to speed up operations of high-energy physics data from the US ATLAS community. In this research project, we seek a method for predicting the 'popularity' of certain datasets in advance, in order to speed up operations within the dCache system. If we are able to determine how popular a dataset may be in the future, we can use this information in order to create an effective cache-replacement policy, speeding up the performance of our storage management system by an order of magnitude. This is not only beneficial in the context of HEP data, but has the potential to be applied across many different domains, as data storage systems are essential systems in use around the globe.

Background

The storage management system dCache is the disk cache for a large collection of high-energy physics (HEP) data, primarily from the A Toroidal LHC ApparatuS (ATLAS) experiment. Storage space on dCache is much smaller than all the data required for ATLAS, which reside on extremely slow tapes across various data centers. If we are able to place the most commonly accessed files in our cache, we would often not need to use lagging tapes to retrieve data, speeding up retrieval times by a substantial amount. It is like the difference between traveling by plane and a car. To determine which datasets are the most popular, we look closely at our dCache metadata and perform EDA in order to generate insights into what makes datasets 'popular'. We also create a data pipeline which transforms our metadata into a format which we can easily perform data analysis on.

Data Set

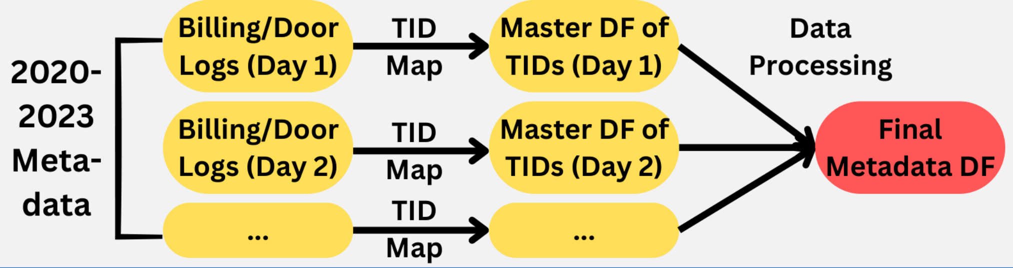
The data used in this research project was acquired from dCache server logs, beginning in late 2020 and ending mid 2023. These server logs can be represented as Pandas DataFrames, of which two are used to generate our insights: Billing and Door DFs.



These logs contain essential information about transactions in the cache, for example the type of transaction, total time required, bytes transferred, or (most importantly) the path of the file that the transaction is directed toward.

Data Pipeline

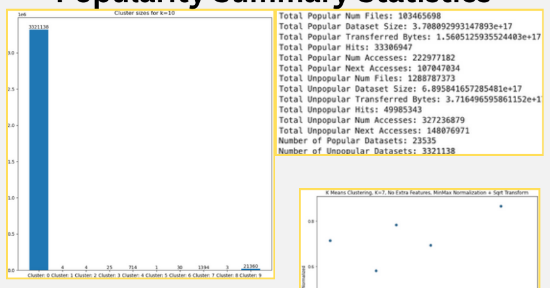
To convert our raw data logs into a usable format for EDA, a data pipeline was created to create a Master DataFrame of datasets with all the necessary (and unnecessary) metadata for data analysis. Each dataset is uniquely identified by its TID (ex. A single experiment or simulation). By analyzing filepaths, it is possible to map files to their TIDs and perform EDA on the dataset level, rather than the file level.



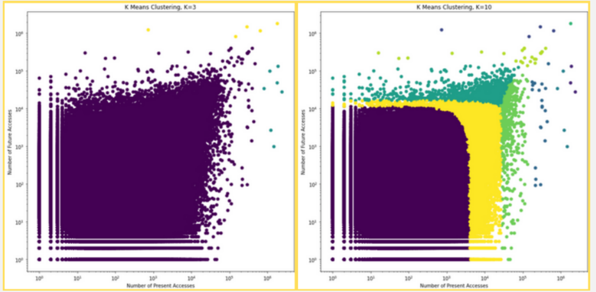
Data Visualizations/Analysis

Various data analyses were performed on our Final Metadata DF in order to generate insights into how a ML-Based Cache Replacement Policy can be implemented.

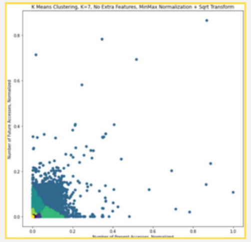
Popularity Summary Statistics

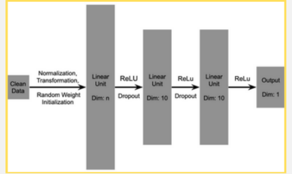


Visualizing # of Daily Accesses vs # of Tomorrow's Accesses



Normalization Transformation Testing





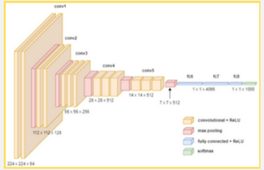
Proposed NN Architecture

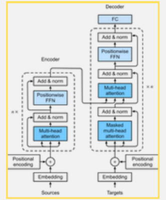
Main Findings

The main conclusions reached during this research project are that:

1. There is a clear cutoff between popular and unpopular' datasets using access counts as the deciding variable.
2. Our data visualizations suggest that we are able to differentiate popular and unpopular datasets.
3. While we can identify datasets which may become popular in the future, we cannot confidently identify datasets which will be unpopular.
4. There are much, much more unpopular datasets as opposed to popular ones. This goes for files as well.
5. A combination of frequency/recency is required for our model to be as accurate as possible.

Towards the Future





This paper is just the first part of a year-long research project. During the second half of this project, we plan to continue seeking out ways to effective build a prediction algorithm for the purpose of identifying popular datasets from the ATLAS experiment. One goal for this prediction algorithm is to make it an online algorithm, so our model can learn while being used in a cache-replacement policy. Another goal for this project is to test out various different ML methods which can be applied to this problem, to make our predictions as accurate as possible. How should we train our model to achieve the best performance? What model should we use to get the best performance? What techniques can we apply during training and prediction to achieve better results? While the first half of this project focuses on EDA and understanding the ATLAS/dCache system, the future of this project lies in building the best model possible.