Progress Report
Week of August 31, 2020

Edris Qarghah

Enrico Fermi Institute
University of Chicago

Weekly Goals

- Real Network
  - Find stable perfSONAR pairs
    - Extract pairs from `trace_derived_v2` scan
  - Determine stable paths between PS pairs
    - Create scan of `ps_trace`
    - Determine how to iterate through scan efficiently
    - Determine centrality of nodes/edges
- Adapting elements of Toy Network
  - Edge omission detection
  - Network visualization

Daily Log

Monday, August 31

- Network Analytics Discussion
  - OSG All Hands talk slides for Shawn
  - Register for OSG meeting
  - A team has been working on creating issues in the network and trying to use that to map out nodes and grid points.
- Spoke with Ilija about goals for the week.
  - Get a clean sample from ElasticSearch, using a few hours of data.
  - Take only stable pairs (i.e., routes do not change).
  - Track these pairs over a few days to see if we can determine changes.
- Updated login function, updating `['atlas-kibana.mwt2.org']` to:
  `[{'host': 'atlas-kibana.mwt2.org', 'port': 9200, 'scheme': 'https'}]`
- Debugged issues with previous network_analytics Main (possibly caused by being partway through changes when I went out sick).

Tuesday, September 1

- Improved documentation in network_analytics (previous comments proved insufficient when debugging).
- Factor out `query_analysis`
- Added `is_ps_pair`, `stable_routes`, `get_stable_routes`, etc.
- Modified `add_bad_pair` to handle updating global variables `routes`, `ps_pairs`, `ps_adj`, and `bad_pairs` whenever a bad pair is added.
- Added new reasons for a pair being bad, that is determined when identifying routes: 'unstable' (meaning there is more than one sha for the route in the given time period) and 'looping'.

Wednesday, September 2

- Update slide for Shawn's OSG meeting presentation.
- Added `reset_vars`
- Spoke with Ilija
  - Important paths (used by many)
  - Stable paths
  - Plot on x, y
  - Find which ones have changed
  - Reflect paths that got effected
  - Don't like approach: Arbitrary choice of three hours.
  - Seven days worth of data. Collect all source/dest pairs. Sha and timestamp. Next time find the same document but with different SHA. Calculate difference in time. Half-life of a path.
  - Plot: x average lifetime of path. y is number of paths that have that kind of stability.
  - Don't consider paths at all, only edges. All edges. What are the edges that are most frequently used (core of the network)? 50 most used edges, connect in small tiny network. See how the network increases when allowing more edges. Should see start connecting. For each edge, how important is it.
- Added `route_changes` (recording time stamps when sha changed for a route) and `get_route_changes`.
- Learned the meaning of RTFM.

Thursday, September 3

- Informal SAND meeting
  - Discussed issue with `filter_path` in python.
  - Found that the solution is to include scroll id and shards
- Iterating through an hour's worth of data takes 10 seconds. Doing something (i.e., recording details) while iterating through an hour's worth of data takes over a minute. This is too slow to feasibly be able to do work on days worth of data.
- Appending individual lines to a dataframe is SLOW, writing to a csv and then converting to a df would probably be more efficient.
- Writing to csv worked, but Ilija suggested using pyarrow instead
- Develop plot on 50 minutes worth of data.
- Read everything once, save it and then play with plots afterward.
- Save a day at a time into parquet. Then develop on 1, expand at the end with all of them.
- Used a 16 gb ram ML instance to run life of path on a full 7 days of data and write it to parquet
- Iterated through 7 days of PS trace data, saving individual days of data to parquet.

Friday, September 4

- Loaded and read a day of data at a time to create frequency chart of nodes on paths
- Graphed stability of paths
  - Determined when routes changed
  - Determined length of time PS pairs were connected by any given path
  - Plotted results in log and linear scale and learned out to reverse the orientation of an axis in MatPlotLib
- Created weekly reports

Achievements

I learned about...

- Pandas
  - Means of creating DataFrames and, specifically, that its best to create them wholesale rather than create and then append them (as that requires copying the whole DataFrame again).
  - Drop columns, add columns, renaming, column calculations, etc.
  - Aggregations and groupings (used this to get counts)
  - Flattening lists (requires Python 3.6+)
  - Removing multi-indexing using reset index.
- Writing large amounts of data
  - CSV and PyArrow take roughly the same amount of time to write to, but PyArrow takes up a tenth the space.
- Chaining generators (”yield from” in python 3.5+)
- Improving ES Scans
  - Removing fields using `_source = [list of fields]`
  - Removing data metadata using `filter_path` (requires keeping shards and index, otherwise query returns nothing).

I created...

- Improved ES scans for pulling data from `ps_trace` and `trace_derived_v2`.
- “Stable routes” as determined by whether the route changed at all over a given time period (we ultimately ditched this approach)
- Means of saving scan results (list -¿ dataframe -¿ pyarrow) and loading
- Edges from lists of hops
- Frequency chart of nodes on paths
- Means of determining route life
  - Determined when routes changed
  - Determined length of time ps pairs were connected by any given path
  - Plotted the results in MatPlotLib

Roadblocks

Questions

- How much does direction matter when trying to determine edges that have performance issues?

Problems

- Updated pandas relies on python 3.6+ (0.24.2 is the highest available in 3.5)

Challenges

- Iterating through scans was taking too long.
  - It took 10 seconds per hour to iterate through and do nothing.
  - It took over a minute to do some basic calculation on an hours worth of data (this meant 3.5+ hours to do the same on 7 days worth of data).
- My iPython kernel died multiple times, but I was able to work around these restrictions by increasing the amount of RAM on my ML instance to 16 GB and by batching.

Plans for Next Week

- Get edge statistics
  - Convert list of nodes to lists of edges
  - Determine frequency of edges
- Draw core network
  - Plot the n most frequent edges
- Network Tomography
  - Determine criteria to trigger investigation (i.e., spike in OWD, increased packetloss)
  - Determine edges along troubled routes that are now omitted