Time Series Updates for Traveler

List of my Pull Requests:

https://github.com/hdc-arizona/traveler-integrated/pull/49

https://github.com/hdc-arizona/traveler-integrated/pull/52

https://github.com/hdc-arizona/traveler-integrated/pull/57

Summary of first 6 weeks (May 18 – June 30):

In this google summer project, my goal is to make the Traveler interface more interactive and responsive by reducing visual lagging as well as providing more control (mouse interaction, file selector interface, etc.) over the frontend UI.

In the aim of providing faster data browsing experience, I followed a three phased approach to eliminate the visual latency. In the first phase, I created several API calls (*drawValues*, *newMetricData*, *ganttChartValues*) to present the data for the frontend UI. The APIs are created as a wrapper over a previously designed library utilizing summed area table data structure. Second, data fetching mechanisms are implemented through those APIs using asynchronous calls and stored in a unified cache object. Third, in the frontend interface, the older implementation with SVG is totally overhauled with a newer implementation using HTML5 Canvas.

The following charts have been created using HTML5 Canvas:

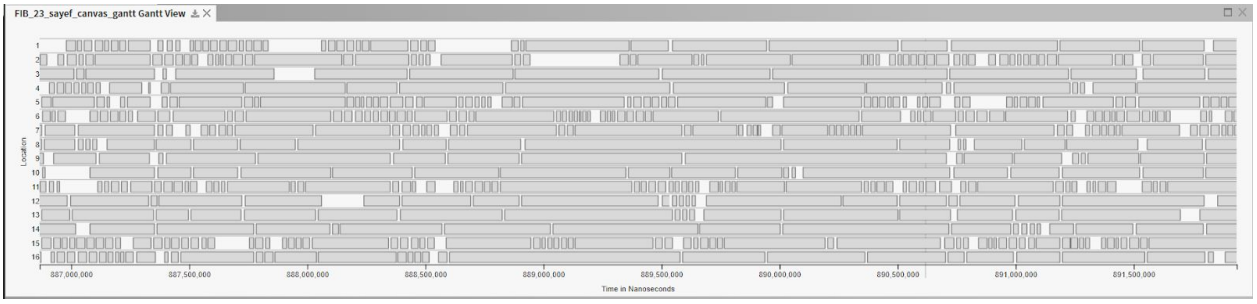

Figure: Gantt View

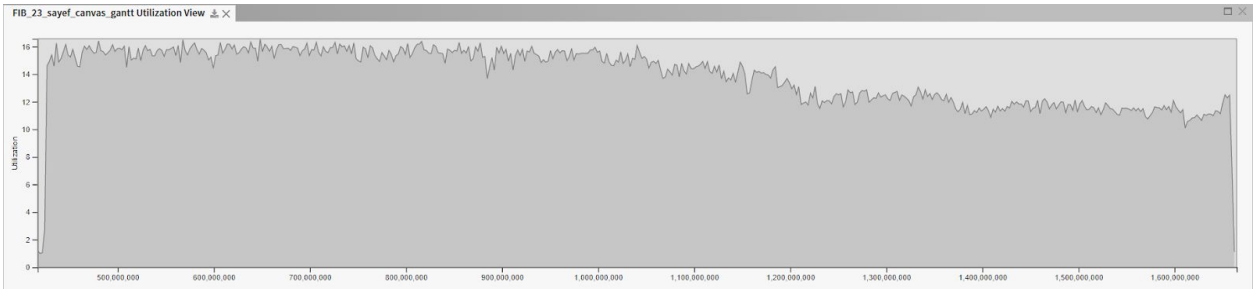

Figure: Utilization View

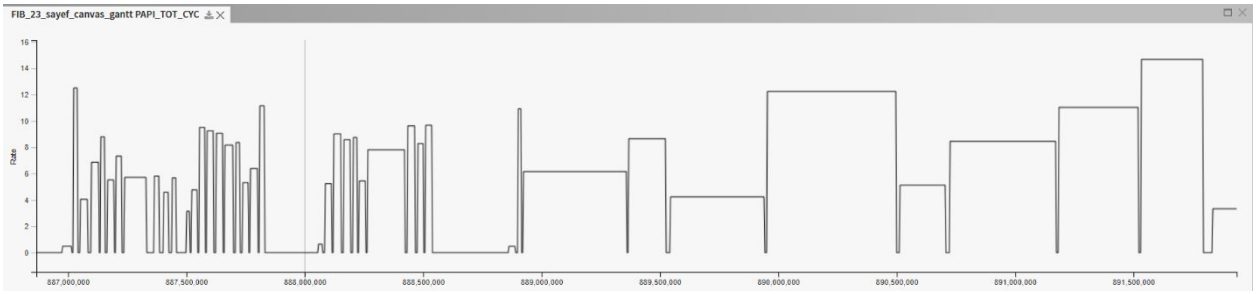

Figure: PAPI Metric (Total CPU Cycle)

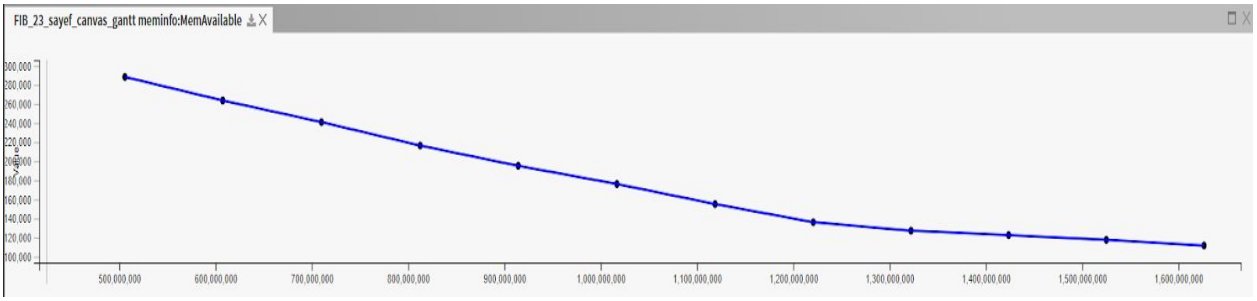

Figure: PROC Metric (Available Memory)

List of completed new features in the first evaluation:

- Summed area table technique implemented to fetch and organize data in the backend with the implementation of *sparseUtiliationList.py* class.
- Using HTML5 Canvas to draw the charts:
    - Gantt View
    - Utilization View
    - PAPI Metrics (PAPI_TOT_INS, PAPI_BR_MSP, etc.)
    - PROC Metrics (MemAvailable, Buffers, status:VmPeak, etc.)
- Newly included APIs:
    - *drawValues* (Utilization)
    - *newMetricData* (PAPI and PROC metrics)
    - *ganttChartValues* (intervals for the Gantt chart)
    - *getPrimitiveList* (intervals of a selected primitive or GUID)
    - *getIntervalInfo* (Fetch all metadata for an interval)
- Common mouse interaction implemented across all views:
    - Scrolling to zoom (in/out)
    - Panning to slide (left/right)
- Basic mouse interaction implemented in the Gantt View with a separate call to the backend for the metadata. (hover, single click, double click)
- An intermediate *cache* object utilized to store and maintain data.
- Sliding left pane window with the dataset list added.
- Locations ordered numerically on the Gantt View.