Visual Analysis of Cloud Computing Performance Using Behavioral Lines

Abichal Ghosh aghosh55@asu.edu
Bharath Balaji bkbalaji@asu.edu
Pavan Kumar Raja praja3@asu.edu
Saicharan Papani spapani@asu.edu
Sakshi Sinha ssinha@asu.edu
Takshshila Rawat trawat2@asu.edu

I. Introduction

The goal of this project is to build a visualization for datasets with a high volume and variety of data. With the visualization that we built, users will be able to gain insights on datasets that contain data points over a wide range of metrics that have a varying correlation with each other. To demonstrate our technique, we have taken the dataset of multiple clusters of servers operating in AWS. Servers that belong to a cluster run the same application. These servers are monitored and the performance metrics of these servers are recorded. Along with this, we also use the logs for each application in our analysis. Generally, a highly scalable cloud computing environment consists of many nodes and each node may or may not have its own database. Modern large-scale applications contain thousands of nodes and it gets really complicated to monitor the performance of these nodes and to debug issues on the fly. We have developed a system that provides both of these features. In case of complete outage or system failures, there are ways provided by major cloud providers to have a disaster recovery option and spin up new servers within seconds. But analyzing what went wrong is a challenge. We provide ways to analyze the performance shocks and issues of all the servers on live systems. Our dataset consists of metrics like CPU, Memory, Disk, and Network utilization for all the nodes in the cluster. We also have the application logs for these nodes.

If we had to visualize one day's worth of data, for a single node it is going to be 1440 data points. Since we have 4 metrics, that is going to be more than 5000 data points. There could be anywhere between 10 to 1000 nodes in a cluster. In large-scale, highly-available applications, we may have to process 50 Million data points. Visualizing so many data points with traditional techniques will not be effective. It will be very difficult to gain insights from such visualizations. On top of the approaches mentioned in the paper [1] , we have implemented new features and techniques that help users visualize complex data like the one discussed here. We offer up to 15 different plots at the same time as part of the analysis tool that we built.

Some of the visualizations presented as part of the project are based on similarity scores. In a fleet of thousands of nodes, many nodes are going to be running the same applications. This includes auto-scaling groups, stand-by replicas, etc. Analyzing each node individually gives us more information, but when we have thousands of nodes, most nodes are gonna have similar performance metrics. So, we provide a similarity-based visualization. We calculate the similarity scores between each node and group them based on this score. Users can see the general trend of the metrics based on the similarity scores. That is, nodes that are using high resources or nodes that have failed, etc. If a group of nodes behaves as expected they may not need more analysis. If an anomaly is encountered, there will be spikes in the similarity plots and users can dive deeper by selecting a group of nodes or a single node to gain more insights.We concentrate on the correlations between the nodes, how they act in connection to one another, and grouping the nodes that act similarly using the force method given in the study. The behavior analysis aids users in identifying faulty, ineffective, or suspicious nodes.

II. Visualization design

The complete visualization design can be broken down into 4 main parts(charts), the behavioral stacked area chart(at the bottom of the visualization), the Metric chart(middle left), the behavioral chart(upper left), and detailed plots(upper right). An overview of the complete visualization can be seen in Fig 1. We built the system in such a way that it is highly interactive and responsive to user actions. We made improvements in how the dataset is consumed and visualized. We implemented a technique to visualize logs for the given dataset. Logs provide useful information regarding the events that happened in the system. If the visualization shows any anomalies like sudden spikes or dips, it means the system could be behaving in an unexpected way. Usually, developers check the logs to debug such issues. We have provided a mechanism to directly pull the logs close to the timestamp where the user clicks on the chart. We have added extensions to all parts of the visualization. More details about the original implementation (from the paper) and extensions will be described in each of the sections below.

A. Stacked Area chart

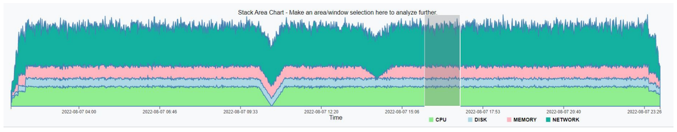


Fig. 2. Stacked Area Chart

The stacked area chart [2] (Fig 2) can be considered to be the starting point of the visualization design. The stacked

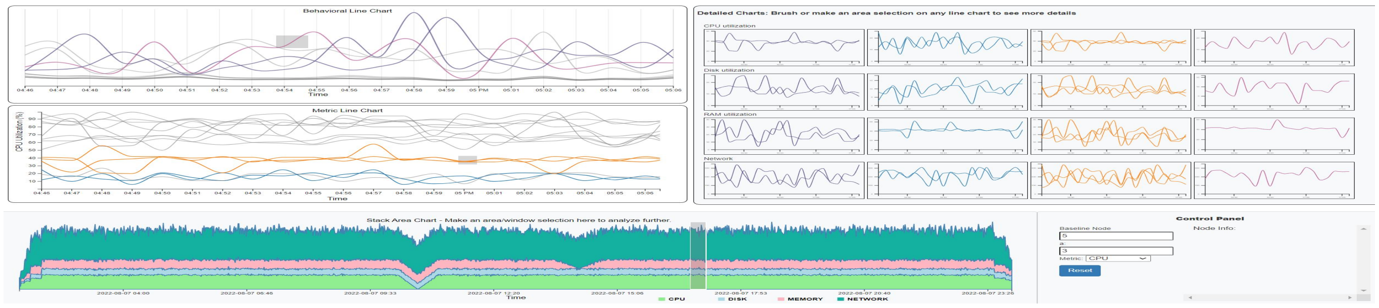


Fig. 1. Project Visualization

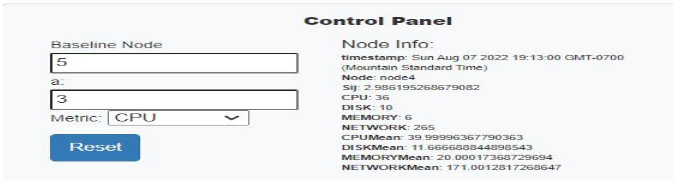
area chart provides an overview of all the attributes of the nodes over the whole time period. These attributes include CPU Utilization, Disk Utilization, Memory Utilization, and Network Utilization. Each of these attributes is visualized using different colors in the form of a stacked area chart. It contains a stack of normalized metric values plotted over time. At each timestamp, the height of each stack is calculated by taking the average of the normalized attribute value of a specific metric for all the nodes. The calculated value is used to plot the stack-area chart. The user can visualize how each metric changes over time after aggregating across all the nodes.

Extension: We allowed the user to flexibly choose the window width. The window can also be moved by keeping the width the same.

- **Marks**:
  - Lines: Lines that separate stacked metrics over a duration of time. The area between lines is colored according to the metric represented by the top line.
- **Channels**:
  - X position: Corresponding to the time.
  - Color: Each metric is visualized with a different color
  - Thickness/Area: Each stack area represents an aggregated view of a specific attribute or metric across all the nodes over a duration of time.
- **Interactions**:
  - Brush [3]: Brush selection has been added on the stacked area chart over the time axis (x-axis). Upon brushing/selecting over a particular time frame(Fig 2), the start and the end time data is provided to the node line chart and the behavioral line chart for plotting the data of the individual nodes within the given time stamp. The user can brush across the x-axis only to select the time window they want to visualize further.

B. Control Panel (Extension)

It contains multiple user-controllable settings and below is the list as show in Fig 3. This control panel is not present in the paper instead we added it explicitly for having a good understanding of the nodes and their behavior.



Control Panel

Baseline Node
5
a:
3
Metric: CPU
Reset

Node Info:
timestamp: Sun Aug 07 2022 19:13:00 GMT-0700 (Mountain Standard Time)
Node: node4
Sij: 2.986195268679082
CPU: 36
DISK: 10
MEMORY: 6
NETWORK: 265
CPUMean: 39.99996367790363
DISKMean: 11.666688844898543
MEMORYMean: 20.00017368729694
NETWORKMean: 171.0012817268647

Fig. 3. Control Panel

- For computing the behavioral line, we use a baseline node and calculate how similar the other node is. In the original implementation, the baseline node is considered a static node but we have given the user capability to choose the baseline node.
- For computing mean and standard deviation at a time stamp t, we use the values from t+a to t-a. So we have given users the opportunity to select the desired value of a.
- Metric selection dropdown. Metric line charts can be drawn based on different attributes or metrics. By selecting the necessary metric, the user can view that metric line chart which is plotted by using that metric.
- When hovered on a node in the behavioral line chart, we can see all the (Node Info) intermittent values that the algorithm generated at that timestamp for finding the similarity.

C. Metric Line Chart (Extension)

This chart is present in the middle left part of the visualization design. The line chart provides an overview of one particular attribute utilization of all the nodes present in the data in some given time frame. The attribute to be visualized in the line chart is selected from the metric drop-down in the control panel as shown in figure Fig 4. The time frame of the line chart is linked to the time frame which can be selected in the Stacked Area chart as mentioned above. This graph is not present in the original implementation and has been added here as an extension. The original paper only plots the behavioral similarity graph but our implementation allows the user to compare the metric values and the similarity values simultaneously.

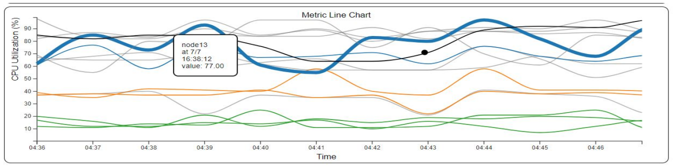
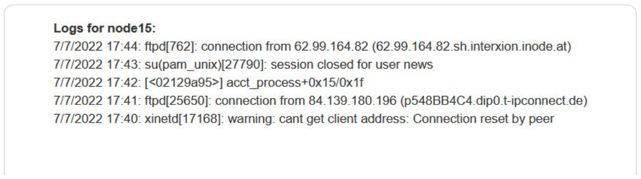


Fig. 4. Node Line Chart

- **Marks**: Line: Lines have been used in this chart for visualizing the direct values of the metric of a given node in a time stamp.
- **Channels**:
  - X position: corresponding to the timestamp of the nodes.
  - Y position: corresponding to the value of the selected attribute for a node.
  - Opacity: low if not selected, high if clicked.
  - Color: grey if not selected, black if clicked. Hues of Red, blue, green, and orange if selected via bushing.
  - Thickness/Size: increases on hovering.
- **Interaction**:
  - Hover: On hovering over the lines, the line gets highlighted(by increasing the thickness of the line) and the details of the node corresponding to the line show up as a tooltip can be seen in Fig 4. The details include the name(number) of the node, the date, the time, and the value on which the mouse is currently on.
  - Brush [3]: On brushing over a particular part of the line chart, the nodes corresponding to the lines in the brushed area get selected and get colored in hues of red, blue, green, and orange. The information of the nodes which have been selected and the particular color then gets updated on the Detailed Chart (on the right). The detailed chart displays all the metrics of the selected nodes in one of its columns.
  - Click: On clicking a particular line, the logs of the node corresponding to the line and the point clicked shows up in the form of a popup, as shown in Fig 5. The line clicked also gets highlighted(by increasing the opacity) and a circle comes up corresponding to the point where the line was clicked.

Extension: The implementation of this entire feature was an extension. This plot and its interactions were not present in the original paper.



```
Logs for node15:
7/7/2022 17:44: ftpd[762]: connection from 62.99.164.82 (62.99.164.82.sh.interxion.inode.at)
7/7/2022 17:43: su(pam_unix)[27790]: session closed for user news
7/7/2022 17:42: [<02129a95>] acct_process+0x15/0x1f
7/7/2022 17:41: ftpd[25650]: connection from 84.139.180.196 (p548BB4C4.dip0.t-ipconnect.de)
7/7/2022 17:40: xinetd[17168]: warning: cant get client address: Connection reset by peer
```

Fig. 5. Logs

D. Behavioral Line Chart

This entire implementation is about showing how similar or dissimilar the nodes are with each other in a distributed and non-distributed cloud system that has auto-scaling features (scales based on user demands as shown in Fig 6. Also multiple nodes work together to serve the customers and nodes might be running the same or different kinds of application).

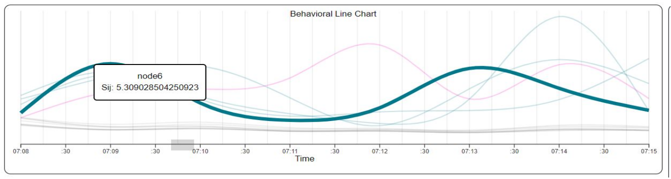


Fig. 6. Behavioral

- Why do we need this kind of information and visualization?
  - It is hard to go and check every metric for an individual node in the cloud system. And manually going through such trace levels is not possible all the time. So it is better to have a combined view or visualization of information that was created by taking all metrics for the nodes. This helps in tracking down which node is behaving abnormally and which property might be causing this issue, etc. in a few glances and a small analysis from our visualization.
- How is it going to help us in tracking node issues, etc?
  - As answered in the above question, we implemented some algorithm that brings a condensed view of all the metrics for all the nodes. This is done by condensing the 4 metrics i.e CPU utilization, Network, Memory utilization, and Disk into some kind of 1 metric and plotting (Similar to dimensionality reduction). The output of this is a behavioral line chart that shows how each node is behaving with respect to the baseline ideal node. This gives clear visualization of which node is abnormal.

Force Algorithm (We can also term it as Similarity Algorithm):

- In order to avoid sharp jumps as data points enter or exit the window, we compute a gaussian kernel, as is generally considered appropriate for temporal sampling.
- For each node i, dimension d, and time t we consider a window from $t + a$ to $t - a$ and calculate the mean and standard deviation:

$$\mu_{i,d}(t) = \frac{\sum_{k=a}^{-a} D_{i,d,t+k} * w(t,k)}{\sum_{k=a}^{-a} w(t,k)}$$

$$\sigma_{i,d}(t) = \sqrt{\frac{\sum_{k=a}^{-a} (D_{i,d,t+k}^2 - \mu_{i,d}^2(t)) * w(t,k)}{\sum_{k=a}^{-a} w(t,k)}}$$

where

$$w(t,k) = e^{\frac{-(t-k)^2}{k^2}}$$

- After calculating the mean and standard deviation for every time, for every node, and for every metric, we find the similarity between two nodes using the below formulae:

$$S_{ij} =$$

$$\left(\sum_{m=1}^{n} \sqrt{ \left(\frac{\mu_{m,i} - \mu_{m,j}}{\mu_{m,max} - \mu_{m,min}} \right)^2 + \left(\frac{\sigma_{m,i} - \sigma_{m,j}}{\sigma_{m,max} - \sigma_{m,min}} \right)^2 } \right)^{-1}$$

  - Where i, j are the nodes and n is the number of metrics (currently we have 4).
  - And S_{ij} is computed for every pair of nodes at each time stamp t.
- If the baseline ideal node is considered as i, then we compute the similarity of S_{ij} where $(1 <= j <= numberofnodes)$ at every t. And calculate the force value using the below formulae and plot the behavioral chart. Here F_a is the attractive force (Hooke's law), and F_r is the repulsive force (Coulomb's law). Coefficients k_1 and k_2 are chosen to guarantee that the two nodes have very similar Fa and Fr when they reach the minimum distance d_{min}.
- **Marks**: Line: Lines have been used in this chart for visualizing the direct values of the metric of a given node in a time stamp.
- **Channels**:
  - X position: corresponding to the timestamp of the nodes.
  - Y position: corresponding to the value of the selected attribute for a node.
  - Opacity: low if not selected, high if clicked.
  - Color: grey if not selected, black if clicked. Hues of Red, blue, green, and orange if selected via bushing.
  - Thickness/Size: increases on hovering.
- **Interaction**:The following interactions have been implemented for this line chart-
  - Brush [3]: On brushing over a particular part of the line chart, the nodes corresponding to the lines in the brushed area get selected and get colored in hues of red, blue, green, and orange. The information of the nodes which have been selected and the particular color then gets used by the Detailed Chart.
  - Hover: On hovering over the lines, the line gets highlighted(by increasing the thickness of the line) and the details of the node corresponding to the line show up as a tooltip can be seen in Fig 6. The details include the name(number) of the node, the date, the time, the value on which the mouse is currently, and the similarity score. More details about this similarity line which includes the average of metrics can be seen in the control panel can be seen in Fig 3.

Extension: We added a baseline ideal node that can be selected by the user and similarities with respect to this node get changed. Hence the obvious change in the behavioral line chart. Apart from that we added hovering, and tooltip behavior and can even explore intermittent values in the force algorithm.

E. Detailed Plots

These plots are on the right side of the visualization window as shown in Fig 7. The detailed plots consist of a matrix of multiple line charts corresponding to the nodes selected via brushing in the node/behavioral line charts. Each row of this chart indicates an attribute or metric that the nodes are plotted against - CPU utilization, RAM utilization, Disk memory utilization, and network load.

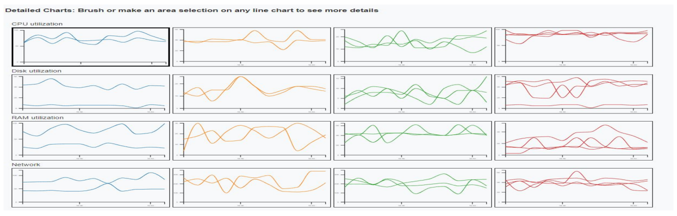

Detailed Charts: Brush or make an area selection on any line chart to see more details
CPU utilization
Disk utilization
RAM utilization
Network

Fig. 7. Detailed

Each column in this visualization corresponds to a brush selection on the metric line graph or the behavioral graph. There can be a maximum of 4 selections and upon a newer selection, after the 4th the oldest selection is replaced with a new one. The 4 columns are given the colors based on where and when the selection happened as mentioned above in the node and behavioral line charts sections.

- **Marks**: Line, Lines have been used in this chart for visualizing the direct values of the attributes of the selected nodes in the time stamp
- **Channels**:
  - X position: corresponding to the timestamp of the nodes.
  - Y position: corresponding to the value of the selected attribute for a node.
  - Color: Red, blue, green and orange based on the selection via bushing from the behavioral and/or node line charts.
- **Interaction**:The following interactions have been implemented for this line chart-
  - Selection: Each smaller plot can be clicked to pop them out and examine in detail as shown in Fig 8. Each line is tagged with its corresponding node id and is useful in seeing granular details. This is an added interaction
  - Marks:
    - ∗ Lines: Each line indicates a unique node in the selection.
    - ∗ Circle: Indicates the end point of a line.
    - ∗ Text: Used to label each node with their corresponding id.
  - Channels:
    - ∗ X position: corresponds to passage of time.
    - ∗ Y position: corresponds to magnitude of the selected attribute.
    - ∗ Color: Categorical - Used to differentiate between nodes.

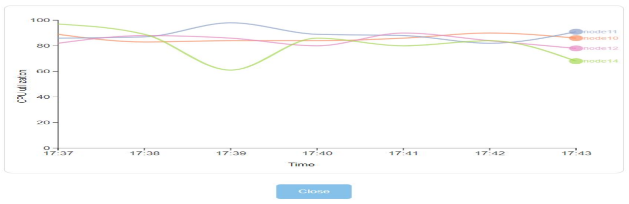


Fig. 8. Detailed Popup Chart

Extension: Clicking on any subplot in this area will open a new chart as a popup. This popup will contain all the nodes that were plotted in the subplot. Additionally, it shows the behavior of each of the selected nodes in the given timeframe.

III. Dataset description

The original paper did not release its dataset. We also could not find a relevant dataset that suited the problem we are trying to solve. So we spawned instances in AWS with AMIs of different nodes running different applications. We had 15 instances and made them run for 3 days. While the instances were running we also captured the application logs. Our dataset consists of 5 CSV files. We have data for the CPU, Memory, Disk, Network, and Logs. This data was downloaded from running instances in AWS. We initially had 3 days' worth of data with data points every minute. This data point represents the average value over the 1-minute interval. We had 15 running instances. With 5 metrics from 15 instances at a 1-minute interval for 3 days, our program was running a bit slow. The calculations for plotting the behavioral lines are computationally heavy. So we decided to use only one day's worth of data. Below is the data abstraction for our dataset.

Dataset Type: Tabular

- Dataset Type: Tabular
- CPU Utilization ($cpu\_p.csv$): Contains the CPU utilization at a given timestamp as a percentage of maximum CPU capacity averaged over a 1-minute interval.
  - Type - Ordered (Quantitative)
  - Cardinality: 95
- Disk Utilization($disk\_p.csv$): Contains the Disk utilization at a given timestamp as a percentage of maximum Disk read/write capacity averaged over a 1-minute interval.
  - Type - Ordered (Quantitative)
  - Cardinality: 62
- **Network Utilization:** Contains the total network usage which includes the network upload and network download in bytes per second at a given timestamp averaged over a 1-minute interval.
  - Type - Ordered (Quantitative)
  - Cardinality: floating point [24.0 - 341.0]
- **Memory Utilization:** Contains the RAM utilization averaged over 1-minute intervals as a percentage of maximum RAM available.
  - Type - Ordered(Quantitative)
  - Cardinality: 94
- **Timestamp:**
  - Type - Ordinal (temporal)
  - Cardinality: Time [07/08/2022 01:40 - 07/08/2022 22:25]
- **Logs** ($logs.csv$)**:** Contains the log information for all the 50 nodes for the entire day.

IV. Case studies

A. *Case study 1: This visualization technique can be used to examine the macroeconomic situations of different regions.*

- **Dataset:** Economic situation of different regions over a period of many years as measured by metrics such as Unemployment, Consumption, CPI indices, and Trade with other regions. This can be combined with a record of major events or news that happened in this period.
  Our visualization technique is good at identifying correlations among different parts of the dataset. So we can analyze complex financial data that are highly correlated. In a dataset with the above-mentioned metrics, we will be able to analyze how each metric affects other and it impacts productivity as a whole.
- **Behavioral Line Charts:** Users can use this chart to identify how similar or different a region's overall economic situation with another region. For example, if there is a war in the East, the countries part of that region will have poor economic output. So, they will be grouped together in the similarity plot. This can be applied to states within a country as well. Using this we can cluster different regions based on their economic situation at each time step.
- **Region-wise line charts:** The user can select the metric they want to visualize. For example, Trade. Trade is something that involves two regions. If the user can see a spike in trade for one region, it must be accompanied by an increase in trade in one or many of the other regions. This can be spotted in this chart. Also, if for some reason, there is a sudden spike or dip in one metric, the reason can be analyzed by clicking on the line. This will pop up a window with the news that happened before that time. Both the above charts (Fig 4 and Fig 6) allow the user to select a line or a group of lines to visualize further. With this technique, we can narrow down regions that are of interest to the user. Making a selection would update the Detailed Charts
- **Detailed Chart:** This is highly useful to see region-level and metric-level data for different regions side-by-side. This is the highest level of comparison. The user can click on any part of the chart to bring a popup to see the plot even closer. Users can see how each metric changed over time for different regions that are plotted alongside.

B. *Case study 2: Analyzing Climate Change.*

- **Dataset:** Dataset of parameters such as Humidity, Temperature, Pollution, Water level, Air Quality, Number

of Industries, Economy growth, and CO2 emission for different zones (tropical, temperate) over many years.
Additionally having a log of events or natural disasters such as Tsunamis, Cyclones, and Storms would help in detailed analysis.

- **Stacked area charts [2]:** The stacked area chart provides construction across the above-mentioned attribute across regions. This can help visualize how much each region contributes to climate change across the globe. There can be added layers of comparison between the worsening climate and economic growth of corresponding regions. Each channel (color) in the chart can represent a unique region.
- **Behavioral Line Charts:** Behavioral lines provide a detailed view of the changes in the attribute associated with climate change over the period selected via a stacked area chart.
This helps in analyzing changes in climate change across each country. It'll be used to understand the impact of each country and give an overview of countries showing similar behavior through force lines.
- **Detailed line charts:** Plotting all the attributes for a selection of interest provides a detailed view that is useful in comparing or contrasting the effects and causation between the countries in the selection. These charts are plotted using the actual magnitude associated with their data (unlike behavioral lines), therefore a user sees the actual magnitude of the impact associated with each country in the selection.
- **Individual Multi Line Chart:** The detailed line chart provides a comparison of selected countries. However, it is not granular to the level of an individual country. Users can click on any of the detailed line charts to get an individual line chart associated with selection and attribute. This plot gives granularity where the user can closely follow all the countries in the selection in detail.

V. Discussion

Through this project, we learned a lot about complex visualizations such as stacked area charts, area selection using brushes, and most importantly using force algorithms to change behavioral lines dynamically. We didn't find the original data as the paper doesn't release the dataset, so we spawned instances in AWS and downloaded the metrics data. We used that data to visualize similar interpretations. We now know how to visualize any dataset and show the story behind that data in an interactive way. The data that was downloaded from AWS was pretty huge. As mentioned earlier we had data for 3 days with intervals of 1 minute. This was the case for all the metrics multiplied by 15 nodes.

The computation behind calculating the behavioral lines and force value has the complexity of O(n^2). Since we had a huge amount of data to process and limited computing resources (our laptop), the visualizations took time to get updated. We tried to improve on the algorithm and reduce its time complexity. Our early improvement efforts were focused on reducing the number of repeated computations. We used dynamic programming to store the computed values and use these results to compute future values. The problem with this was that we were now using extra memory because we are storing the intermediate results. Storing intermediate results also slowed down the process because since we are using JavaScript, it runs on the browser. So we are limited by the amount of memory available to the browser to store the results. So this approach didn't work. We then tried to precompute the force values. As we already have the dataset and the algorithm implemented in code, we could easily pre-compute the force values and store them in the file. We could later just use this file when plotting the data. This approach worked. But then our system will not be dynamic.

Our goal was to design a system that can handle streaming data. We wanted to make it real-time or near real-time so that it can actually be used in an enterprise setting. So we discarded the idea of precomputing. We decided that the way to achieve this is by consuming the data in chunks. We would consume the data one-day at a time. Even for streaming data, we could batch the data every couple of minutes and process it that way to make it near real-time.

Future improvements could be in the area of the algorithm time complexity. We are greatly limited by the O(n^2) complexity of the algorithm. We can make the system more scalable if we had a better algorithm that calculates the similarity score. One approach could be to have a trade-off between accuracy and run time. We can group nodes based on their metrics and calculate partial similarity and stop computing if the partial similarity is below a threshold. This would still work but the output would not be as accurate.

VI. Results

We were able to successfully implement a visualization technique that can be used to analyze highly complex and correlated data with multiple metrics. Our system is highly interactive and flexible compared to the original paper implementation. We provided the users options to select the baseline node, the "a" value, options to compare behavioral data with actual data, and a lot of interactions in all these plots. Our visualization is information-rich. Since the original dataset contains a lot of data, we made efforts to show as much data as possible (when required) while simultaneously keeping the cognitive overload down. The capabilities provided by our extension is user friendly and easy to understand. We faced many challenges while working on this project, technical and non-technical, but we were able to tackle them. We made the system as responsive as possible to window adjustments etc. We were able to build different techniques to visualize all parts of the dataset and provide insights based on which users can take timely decisions.

References

[1] Chris Muelder, Biao Zhu, Wei Chen, Hongxin Zhang, and Kwan-Liu Ma. Visual analysis of cloud computing performance using behavioral lines. *IEEE transactions on visualization and computer graphics*, 22(6):1694–1704, 2016.

[2] How to create a stacked area chart with d3. https://medium.com/@louisemoxy/how-to-create-a-stacked-area-chart-with-d3-28a2fee0b8ca. Accessed: 2018-12-15.

[3] Multi-line chart focus + context w/ mouseover tooltip. https://observablehq.com/@connor-roche/multi-line-chart-focus-context-w-mouseover-tooltip. Accessed: 2022-04-14.