Interactive Visual Analysis Tool for Anomaly Provenance Data

Alicia Guite[1], Tanzima Z. Islam[1], Christopher Kelly[2], Wei Xu[2]
[1]*Texas State University,* [2]*Brookhaven National Laboratory*

***Abstract*—Chimbuko is a framework for detecting real-time performance anomalies incurred by large-scale applications. Understanding the source of anomalous behaviors is difficult due to the high volume of information stored by Chimbuko in a provenance database. This undergraduate research project aims to intuitively display this high volume of information without overwhelming users. We then integrate our analysis and visualization techniques into a publicly available framework called DASHING. This project facilitates interactive user investigation of anomaly provenance in large-scale applications.**

***Index Terms*—Real-time anomaly detection, Anomaly provenance analysis, Visualization.**

I. INTRODUCTION

Chimbuko [1] is a real-time anomaly detection framework developed by Brookhaven National Laboratory as part of the CODAR co-design center. Large-scale applications such as NWChem [2], XGC [3; 4] running on the world's largest supercomputers such as Summit [5] and Crusher [6] (the pre-exascale testbed) use Chimbuko to inform users about anomalies. Here, an anomaly refers to aberration in function execution time. For instance, if a specific call path of a function takes 10 seconds on average to execute, then the same call path taking 100 seconds is deemed anomalous.

Chimbuko's current visualization tools only display statistical data, but not where anomalies appear or how anomalies affect ranks in a multi-process execution. Moreover, each provenance database can have millions of anomalies reported even for short simulations. For instance, the largest anomaly provenance database analyzed has more than $16M$ anomalies reported for an NWChem run with 42 workers spread across three nodes. Displaying a large amount of provenance data in an intuitive and relevant manner is challenging, requiring the development of new analyses and visualizations. Our work addresses this challenge where we provide an automated tool for interactively analyzing and visualizing the provenance of performance anomalies occurring for applications using Chimbuko.

The contributions of this work are: (a) Design and develop four different visualizations for presenting anomaly information in a top-down manner where users can interact with the visualizations to gain more insights as needed. (b) Analyze several large anomaly provenance databases having the number of entries between 200K to greater-than 16M. Each analysis only takes a few seconds. (c) Integrate our methods in a publicly available analysis and visualization framework called DASHING [7], to make them available to the research community.

II. BACKGROUND AND RELATED WORK

Chimbuko's anomaly provenance schema is described in detail in their user guide [8]. Among many information, Chimbuko provides information about the call stack, the rank ID, the elapsed time, and the severity of the anomalies. The severity score is proportionate to the elapsed time of the function. An anomalous function can have different call paths. For instance, the call path `foo->bar->anom` to an anomalous function `anom` is different from another call path `baz->kaz->waz->anom`. Figure 1a shows an example of how Chimbuko stores information about a call path.

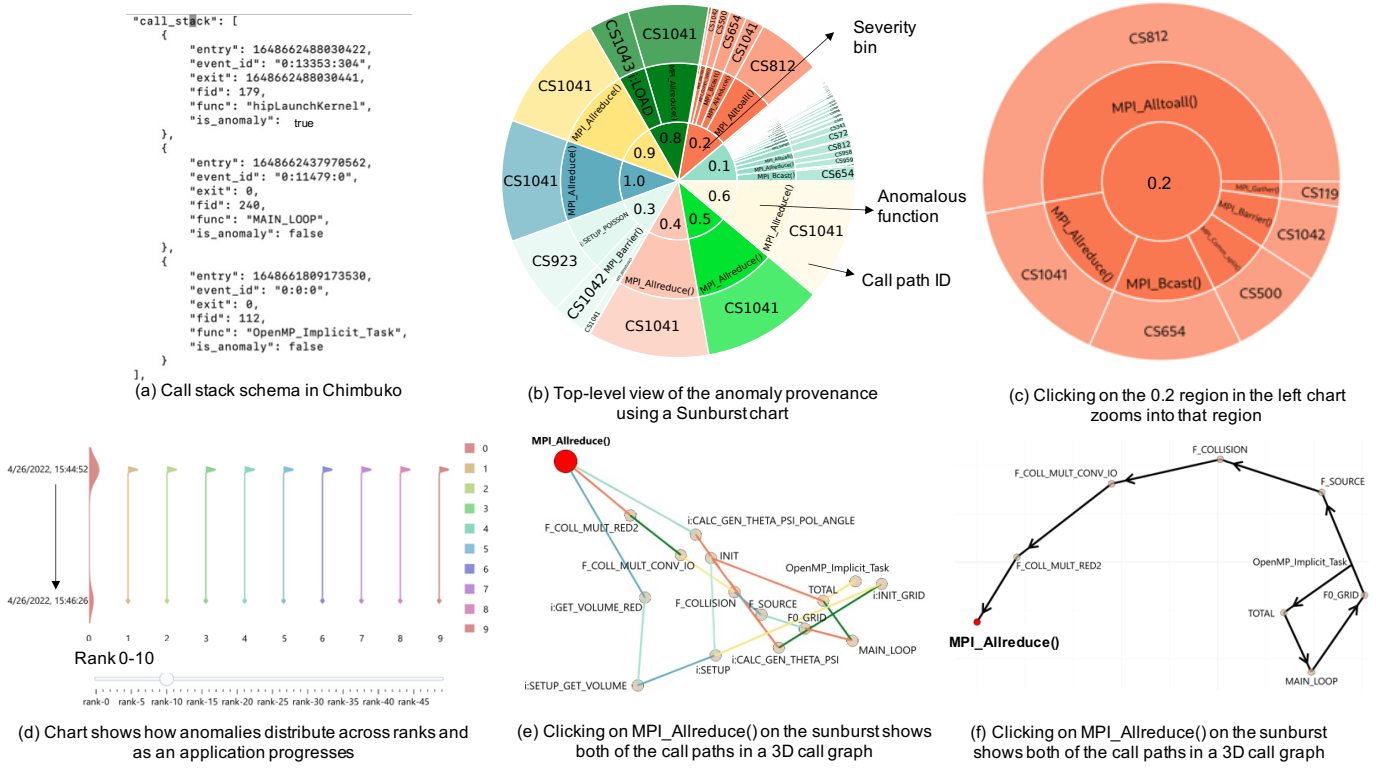
DASHING [7] is a publicly available programmable framework that enables users to deploy their custom analysis and visualizations easily. Currently, DASHING provides two analysis—resource importance based on hardware performance counters, and comparative analysis between performance profiles of a pair of applications. None of the existing techniques apply to the anomaly provenance use case. Hence, this research extends the current capabilities of DASHING to support anomaly provenance analysis for Chimbuko.

III. DESIGN AND IMPLEMENTATION

With the goal of easy interpretability, we have created several visualizations corresponding to several research questions.

Top-down view of anomalies (Figure 1b) presents a large amount of hierarchical information in a visually intuitive manner. A single sunburst chart represents the characteristics of an entire provenance database. Hovering over a slice also displays more details. The innermost circle in each pie slice shows the severity bin, the middle layer shows the anomalous function names, and the outermost layer shows the call path IDs for each of the anomalous functions. The pie slices are sorted based on the aggregated total amount of time spent by the anomalous functions of that group. Each section of each circle represents its relative contribution to its parent. The anomaly slice with the highest aggregated importance is presented along the three o'clock axis, and the rest are sorted in the counter-clockwise direction. This top-down visualization enables users to quickly discover the problematic functions in large-scale applications. This discovery will guide users to optimize those specific functions first.

We use Plotly's Sunburst chart in the graph objects library to implement it. This visualization answers the following research questions: (a) Which functions incur frequent anomalies and how severely do they affect the overall application? (b) Which call paths are the root causes of anomalies? Is it always the same one or are there multiple call paths that can lead to



(a) Call stack schema in Chimbuko

(b) Top-level view of the anomaly provenance using a Sunburst chart

(c) Clicking on the 0.2 region in the left chart zooms into that region

(d) Chart shows how anomalies distribute across ranks and as an application progresses

(e) Clicking on MPI_Allreduce() on the sunburst shows both of the call paths in a 3D call graph

(f) Clicking on MPI_Allreduce() on the sunburst shows both of the call paths in a 3D call graph

Fig. 1. Overall view of our research.

the same function being anomalous? Each section is clickable, expanding that section to fill the whole chart (Figure 1c).

Anomaly distribution across ranks (Figure 1d) shows anomaly distribution across ranks as a simulation progresses. Along X-axis, we show ten ranks at a time, and along Y-axis we show the execution times (starting from the top). The slider at the bottom allows users to scroll between groups of ranks. Since an application can run with hundreds of MPI ranks, the slider enables the visualization to scale for large applications. We implement this visualization using Plotly's Violin chart. The legends on the right show the ranks being displayed, and a user can click on the legends to toggle the visibility of the corresponding distribution chart. This visualization has been designed to address the research question: how do anomalies distribute across ranks and time steps?

Call graph and call path (Figure 1e, f) showcases the entire call graph of an anomalous function. We color the anomalous function in red to highlight, and show function calls along different anomalous call paths using separate colors. Though difficult to display on a paper, the graph is 3D and can be rotated, allowing users to analyze anomaly origins with relative ease. We also present a directed graph for individual call paths with an ID of the form "CSXYZ" (the outermost circle in the sunburst chart). Call paths are shown in 2D, and the arrows show the progression of the function calls. We implement the call paths and graphs using the networkx [9] library.

IV. Preliminary Results

In this poster, we present the analysis of a dataset from the XGC application (200K entries), which is a fusion simulation code to model gyrokinetic plasma physics. It only takes DASHING a few seconds to analyze and visualize the dataset. The poster presents an example configuration file to use in DASHING to reproduce the results. **Figure 1b** shows that for the XGC application with 512 ranks on the Crusher system, functions with small execution times belonging to the 0.1 anomaly bin accumulate the largest amount of wasted computation. The next anomaly group belongs to 0.2 severity. We also observe that the `MPI_Allreduce` and `MPI_Barrier` functions are most often anomalous. **Figure 1d** shows that the behavior of all ranks are marked as anomalous at the beginning since there is no other baseline to compare with. The straight line shows a lack of anomalies for the rest of the program execution. The application developers validated this observation as rank 0 conducts a lot more tasks (e.g., I/O, data structure setup) at the beginning and end of the simulation than other ranks. **Figure 1e** shows the call graph of the `MPI_Allreduce()` function. By clicking on `MPI_Allreduce()` on Figure 1b, a user is able to visualize the entire call graph of the function that includes multiple call paths. By clicking on a call stack ID, a user is able to visualize a specific call path in Figure 1f.

V. Conclusions

In this undergraduate-led research, we design and develop analysis techniques and visualizations for understanding the provenance of function execution-time-based anomalies. We also integrate our methods in the DASHING framework, which will be made publicly available if the poster is accepted. We will also include analysis from a large-scale run of the NWChem application with 16M anomalies in the final poster.

REFERENCES

[1] C. Kelly, S. Ha, K. Huck, H. Van Dam, L. Pouchard, G. Matyasfalvi, L. Tang, N. D'Imperio, W. Xu, S. Yoo *et al.*, "Chimbuko: A workflow-level scalable performance trace analysis tool," in *ISAV'20 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, 2020, pp. 15–19.

[2] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. J. Van Dam, D. Wang, J. Nieplocha, E. Aprà, T. L. Windus *et al.*, "Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations," *Computer Physics Communications*, vol. 181, no. 9, pp. 1477–1489, 2010.

[3] S. Ku, C.-S. Chang, and P. Diamond, "Full-f gyrokinetic particle simulation of centrally heated global itg turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry," *Nuclear Fusion*, vol. 49, no. 11, p. 115021, 2009.

[4] C.-S. Chang and S. Ku, "Spontaneous rotation sources in a quiescent tokamak edge plasma," *Physics of Plasmas*, vol. 15, no. 6, p. 062510, 2008.

[5] Oak Ridge National Laboratory, "Summit Supercomputer," https://www.olcf.ornl.gov/summit/.

[6] ——, "Crusher Supercomputer," https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html.

[7] T. Islam, A. Ayala, Q. Jensen, and K. Ibrahim, "Toward a programmable analysis and visualization framework for interactive performance analytics," in *2019 IEEE/ACM International Workshop on Programming and Performance Visualization Tools (ProTools)*. IEEE, 2019, pp. 70–77.

[8] B. N. Laboratory, "Chimbuko User Guide," https://chimbuko-performance-analysis.readthedocs.io/en/latest/io$_s chema/schema.html provenance-database-schema$.

[9] A. Hagberg and D. Conway, "Networkx: Network analysis with python," *URL: https://networkx. github. io*, 2020.