Missing the Trees for the Branches: Graphical-Scripting Interaction with Large-Scale Calling Context Trees

CONNOR SCULLY-ALLISON, University of Arizona, USA
KATHERINE E. ISAACS (ADVISOR), University of Arizona, USA
OLGA PEARCE (ADVISOR), Lawrence Livermore National Laboratory, USA

Additional Key Words and Phrases: visualization, calling context tree, user-interface, literate programming

ACM Reference Format:
Connor Scully-Allison, Katherine E. Isaacs (Advisor), and Olga Pearce (Advisor). 2021. Missing the Trees for the Branches: Graphical-Scripting Interaction with Large-Scale Calling Context Trees. 1, 1 (October 2021), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRO

In large scale performance analysis of HPC code, there is a tension between the visualizations provided by automated tools and the flexibility afforded by custom scripting. Visualizations give users a unique perspective on their data and enable insights which are not possible with tabular representations. For calling context trees (CCT), in particular, node-and-link visualizations give users a view of performance data that corresponds to their mental model. Unfortunately, while current scripting workflows support the writing of code, they are not well supported by existing visualizations. GUI based tools often restrict users to rigid modes of interaction. Furthermore, many general purpose visualization libraries are not well configured to support the complex multivariate data collected by HPC profilers, do not scale well to fit real HPC datasets, and often require exporting and viewing visualizations in a context divorced from where the code is developed. To ease this tension and support modern analysis workflows, we use human-centric design methodologies to marry scripting and graphical interaction paradigms into a scalable, interactive CCT visualization embedded in Jupyter notebooks.

2 RELATED WORK

Many related works on visualizing calling context trees use a traditional node-link based layout [4, 9, 10]. The node link design is intuitive for users and easy to understand. Unfortunately, these solutions suffer from poor scalability, remove users from the context of their code, and show only one metric at a time.

Other works eschew the traditional node-link layouts to address scalability concerns [7, 8]. The first uses "ring-charts" to encode call hierarchies in layers radiating out from a central node. The second aggregates call sites by module into a Sankey diagram. Both representations are effective at scaling large profiles but suffer from over-abstracting the

Authors' addresses: Connor Scully-Allison, cscullyallison@email.arizona.edu, University of Arizona, Tucson, Arizona, USA, 85721; Katherine E. Isaacs (Advisor), kisaacs@cs.arizona.edu, University of Arizona, Tucson, Arizona, USA, 85721; Olga Pearce (Advisor), pearce8@llnl.gov, Lawrence Livermore National Laboratory, Livermore, California, USA, 94550.

structure of a CCT too far from an analyst's mental model of their program. Furthermore, it is difficult to highlight particular functions using these designs.

Finally, CCTs are often rendered using indented tree structures[1, 2, 5]. While indented tree structures can be an effective way to show trees in the same context as scripts (in a console, notebook or vertical window), they lack meaningful interactivity and do not scale well, making inefficient use of vertical and horizontal space.

3 INTRODUCING THE GRAPHICAL SCRIPTING INTERACTION PARADIGM

Our interactive tree visualization was built to to interact with Hatchet[3], a python library for programmatic performance analysis that uses a hybrid dataframe-callgraph as its core data structure called a GraphFrame. This visualization supports scripting and graphical interaction paradigms by making visualization part of the scripting workflow. Traditionally, users would have to operate on their data in scripts or visualizations exclusively with no direct communication between the two mediums; changes made to a programmatic call graph need to be re-implemented visually when the data is uploaded to a visualization tool and vice versa.

By contrast, our paradigm, enabled with Juypter magics, allows users to filter their GraphFrame programatically and then visualize that modified data on the fly. It also allows users to use mouse manipulation to prune their CCT and get back that reduced data for further analysis. This gives analysts the benefits of both graphical and script-based interaction with their performance data; all without leaving their current working environment.

Semi-structured interviews with three individuals experienced in performance analysis has informed development of our paradigm by elucidating *how* analysts use scripts and graphical tools and what they expect from their tools. We use these insights to mitigate pain points and enhance the overall utility of our solution.

4 MAKING LARGE MULTIVARIATE TREES UNDERSTANDABLE

To address common scalability issues presented by real HPC profiles we automatically collapse un-interesting subtrees with a process we call *pruning*. We first identify potential regions of interest using box-plot outlier detection [6] on a user selected metric. After locating outliers, and preserving the parents of outliers to maintain context, we collapse sub-trees containing non-outliers into surrogates. These surrogates inform the user about the collapsed trees in the following two ways:

(1) If the total metric values of all call sites in a removed sub-tree sums to zero our surrogate node is displayed as a triad of greyscale nodes.
(2) If the the total metric values in the removed sub-tree is nonzero, we aggregate the sub-tree into a rectangle. It is colored using the same scale as regular nodes using the aggregate metric as its encoded data.

In addition to pruning, this visualization uses multivariate encoding to show correlation between two different metrics. Users expressed a desire to view speedup and time on one graph as their intersection on a call-site provides richer information than one metric alone. For example, by choosing the *speedup* metric as node color and the *exclusive time* metric as node size, the large nodes colored with low speedup values are highlighted to the user and indicate potential optimization targets at a glance.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-ABS-825366

REFERENCES

[1] Laksono Adhianto, Sinchan Banerjee, Mike Fagan, Mark Krentel, Gabriel Marin, John Mellor-Crummey, and Nathan R Tallent. 2010. HPCToolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience* 22, 6 (2010), 685–701.

[2] Robert Bell, Allen D Malony, and Sameer Shende. 2003. Paraprof: A portable, extensible, and scalable tool for parallel performance profile analysis. In *European Conference on Parallel Processing*. Springer, 17–26.

[3] Abhinav Bhatele, Stephanie Brink, and Todd Gamblin. 2019. Hatchet: Pruning the overgrowth in parallel profiles. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–21.

[4] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. 2003. Graphviz and dynagraph – static and dynamic graph drawing tools. In *GRAPH DRAWING SOFTWARE*. Springer-Verlag, 127–148.

[5] Markus Geimer, Pavel Saviankou, Alexandre Strube, Zoltán Szebenyi, Felix Wolf, and Brian JN Wylie. 2010. Further improving the scalability of the Scalasca toolset. In *International Workshop on Applied Parallel Computing*. Springer, 463–473.

[6] Jorma Laurikkala, Martti Juhola, Erna Kentala, N Lavrac, S Miksch, and B Kavsek. 2000. Informal identification of outliers in medical data. In *Fifth international workshop on intelligent data analysis in medicine and pharmacology*, Vol. 1. Citeseer, 20–24.

[7] Philippe Moret, Walter Binder, Alex Villazón, and Danilo Ansaloni. 2010. Exploring large profiles with calling context ring charts. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*. 63–68.

[8] Huu Tan Nguyen, Abhinav Bhatele, Nikhil Jain, Suraj P Kesavan, Harsh Bhatia, Todd Gamblin, Kwan-Liu Ma, and Peer-Timo Bremer. 2019. Visualizing hierarchical performance profiles of parallel codes using callflow. *IEEE transactions on visualization and computer graphics* 27, 4 (2019), 2455–2468.

[9] Huu Tan Nguyen, Lai Wei, Abhinav Bhatele, Todd Gamblin, David Boehme, Martin Schulz, Kwan-Liu Ma, and Peer-Timo Bremer. 2016. VIPACT: a visualization interface for analyzing calling context trees. In *2016 Third Workshop on Visual Performance Analysis (VPA)*. IEEE, 25–28.

[10] Josef Weidendorfer, Markus Kowarschik, and Carsten Trinitis. 2004. A tool suite for simulation based analysis of memory access behavior. In *International Conference on Computational Science*. Springer, 440–447.