Visualizing the Behavior of Large Programs

Hoa Nguyen
Scientific Computing and Imaging Institute, University of Utah
Lawrence Livermore National Laboratory
Email: hoanguyen@sci.utah.edu

Greg Bronevetsky
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Email: bronevetsky@llnl.gov

Abstract—As High Performance Computing simulations grow more complex, debugging and optimizing them becomes vastly more difficult. This makes it necessary to design visualization tools that help developers understand the behavior of their applications. We propose novel visualizations and effective mechanisms to interact with them to support developer efforts to easily and efficiently analyze the behavior of large applications. These visualizations address some of the major issues still plaguing program behavior analysis, including reducing human error, improving speed and quality of decision making, and mitigating program size and complexity issues in analyzing behaviors. A case study on HPC programs demonstrates that our methods can help developers efficiently explore the behavior of their applications.

I. INTRODUCTION

Effective debugging and optimization of large scientific applications take up half of the overall development time [1] and require a detailed analysis of their behaviors. Debug logging and performance tracing are tools for such analysis tasks and are used in a wide range of application and systems. It is natively supported by tools such as HPC Toolkit [2], Parodyn tool’s Performance Consultant [3], and SCALASCA’s cube visualization [4]. However, as these logs grow large, they overwhelm developers ability to find the key bits of information among the MBs of GBs of aggregate text and numeric data. This makes it important to develop visualization tools that help developers understand these complex logs and their relationship to application semantics and performance.

This poster presents our work on novel visualizations of information from text and performance logs. Section III presents a hierarchical view that organizes log information and enables developers to focus on the portions of the log most relevant to their task. Section IV focuses on novel visualizations of multi-dimensional numerical and categorical data. These visualizations help developers to focus their attention on the key regions of their logs and identify important relationships among different dimensions of collected data. Our visualizations were implemented on top of the Sight application analysis framework.

II. PRIOR WORK

A. Program Behavior Visualization

Although there exist many systems for visualizing application performance information, work on visualizing behavior over time is limited. Vampir [5] provides a framework for analyzing parallel software applications. It supports high-level information about parallel applications but still leaves it difficult to correlate this information with the detailed log.

B. Multidimensional Visualization

Developers are interested in a wide range of application metrics, including performance data such as elapsed time and cache miss rates, as well as high-level metrics such as matrix sparsity or kinetic energy of the simulated system. To understand the relationship between these data dimensions they require effective multidimensional visualization techniques. In this work we build upon prior visualization approaches: A Scatter Plot (SCP) [6] is a simple plot of points used to investigate the relationships between 2, 3 and 4 dimensions using the location and colors of points. Parallel Coordinates Plot (PCP) display axes for data dimensions as parallel lines.

III. VISUALIZATION FOR PROGRAM BEHAVIORS

A. The Complexity of Program Behaviors

The large size of application logs makes it necessary to organize and present them to developers in a way that enable them to focus in detail on just the regions that are relevant to a given task. To this end we developed a hierarchical graph that supports zooming, expand and collapse interactions.

B. Interactive Flow Graph

1) Design: The proposed graph organizes segments of the application logs that share some properties (e.g., collected during a given function call) into separate graph nodes. Numerical and text labels on the nodes denote the corresponding application region and arrows denote the flow of control and data among application regions. Repetitive structural patterns of program behavior (e.g., nested loops, recursion or repeated communication) are shown by nesting sub-graphs inside each other.

2) Flow Graph Algorithm: We implemented the above visualization and connected it to the logs exported by the Sight tool. Sight logs explicitly encode hierarchical containment relationships of log regions, which makes it easy to connect log regions to elements in our hierarchical visualization. By connecting the visual elements to the log regions they encode our visualization makes it possible to interactively search through the overall structure of the log, zoom into the log regions of most interest, and alternate between the high-level hierarchical view and detailed view of individual log entries.
We conducted a case study to apply our visualization method to different programs, including a simple Fibonacci sequence program and the AMG2103 benchmark [7]. Figures 1 and 2 show the results, which confirm that developers can easily use our method to explore large program behaviors.

IV. MULTIDIMENSIONAL VISUALIZATION

A. Correlation Coordinate Plots

To help application developers understand relationships in multi-dimensional performance and behavior data we implemented SCP and PCP visualizations and developed a novel visualization that addresses their limitations. SCP (Figure 3b) can help identify relationships in 2, 3 and 4 dimensions but it cannot show all combination of dimensions PCP (Figure 3c) presents data trend but it requires heavy user interaction for complete exploration. These problems led us to develop a new visual encoding called Coorelation Coordinate Plot (CCP). CCP is laid out on a single major axis, which represents the best linear fit through the data, and a minor axis that indicates the distance from this line. The major coordinate axis also serves as an indicator as to the direction of correlation. Data elements are transformed into the correlation coordinate space and displayed accordingly as in Figure 3a.

B. Snowflake Visualization

We also propose a new multidimensional visualization, called Snowflake Visualization (Figure 3d), which is an interactive focus+context style circular design. The focus region enumerates the correlations between a given dimension and all other dimensions. The outer region enumerates all other pairs of dimensions to provide further context. User interactions such as swapping, zooming, selection enables shifting dimensions in and out of the focus region to enable exploration of the entire dataset with \( O(n) \) interactions.

C. Evaluation

We conducted an user study with 25 people for 3 experiments in 1 hour per test. The results indicate that CCPs/Snowflake improves the identification of direction and strength of correlation, with higher accuracy and faster speed than SCP, PCP. This design thus balances space efficiency, comprehensive visualization and economy of interactions.

V. CONCLUSION

The Interactive Flow Graph visualizes program behaviors efficiently via a hierarchical visual encoding, graph algorithm and interactions. This visualization can be applied to logs and traces from large scale programs to help developers analyze complex program behaviors. Multidimensional Visualizations help developers identify correlation and other patterns in application data, reduces human error, improve speed and quality of decision making on exploring program behaviors. In the future work, we will apply our methods for larger scale program behaviors.

We implemented these visualizations on top of the SIGHT tool, using its data collection and aggregation mechanisms. Our source code available at https://github.com/bronevet/sight/tree/hoaViz.

ACKNOWLEDGMENT

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, and was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-POST-658233).

REFERENCES


