Interactive Interface for StarVZ Plot Visualization

Christian Einhardt Sousa Filho[1], **Vinícius Garcia Pinto**[1]

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Rio Grande - RS - Brasil

christianeinhardt2002@gmail.com, vinicius.pinto@furg.br

Abstract. *This work introduces an interactive interface developed in R using the Shiny framework for visualizing plots generated by StarVZ. Visualization of execution traces is a key tool for performance analysis of parallel applications, helping to identify inefficiencies and optimize resource utilization. The interface allows users to load data, configure different visualization parameters, and switch between ggplot2 and plotly-generated graphs. The goal is to provide a more intuitive and flexible experience for execution trace analysis, making it easier to extract insights from processed data.*

1. Introduction

Execution trace analysis is a fundamental step in understanding the performance of parallel applications. However, this task can be challenging due to the complexity of the generated data and the need for tools that enable a clear and interactive visualization. Several tools have been developed for execution trace visualization [Pinto et al. 2021, ViTE Team 2024]. One such tool is ViTE [ViTE Team 2024], an open-source trace explorer that supports the visualization of execution traces in Pajé or OTF format. A recent work [Ordronneau 2024] discusses ViTE improvements, ensuring better performance and usability in trace analysis.

In this context, this work proposes an interface to facilitate the visual exploration of data generated by StarVZ [Lucas Mello Schnorr et al. 2022], a tool used to visualize and analyze execution traces from StarPU. The primary objective is to make this analysis more accessible, allowing users to interact intuitively with the data without manually handling files and complex configurations.

2. Technologies and Tools Used

To enable efficient visualization and analysis of StarPU execution traces, a combination of tools was utilized. This section describes the key technologies used in the development of the Shiny-based interface for StarVZ.

StarVZ is a visualization tool designed to analyze execution traces from StarPU, a runtime system for multicore architectures. It allows users to examine various performance aspects, such as task execution timelines, resource usage, and scheduling efficiency. Effectively visualizing this data is crucial for optimizing parallel applications and understanding their computational behavior.

To make this analysis more accessible and interactive, an interface was developed in R using the Shiny framework [Chang et al. 2024]. It allows users to interact dynamically with the data, adjusting parameters in real-time.

The integration of StarVZ with Shiny enhances the accessibility and usability of execution trace analysis, providing a more intuitive experience and enabling users to adjust visualization parameters and interact with the data without manually handling files or complex configurations.

The interface was developed using tools from the R ecosystem to ensure an interactive and flexible experience. Shiny serves as the foundation of the application, enabling the creation of a dynamic interface, while the shinyjs [Attali 2021] package facilitates controlling the visibility and behavior of interface elements. The shinyFiles [Pedersen et al. 2022] package simplifies file and directory selection.

For data visualization, the interface combines ggplot2 [Wickham 2016] for static graphs and plotly [Sievert 2020] for interactive graphs, allowing detailed exploration of the data. Data manipulation and processing are optimized using the tibble [Müller and Wickham 2023] package, ensuring efficiency when handling large datasets. Additionally, the StarVZ [Lucas Mello Schnorr et al. 2022] package is used for loading and structuring execution traces.

With these tools, the interface facilitates the visual exploration of execution traces, providing a more intuitive and accessible analysis experience for users.

3. Developed Interface

The developed interface (Figure 1) consists of two main components: a side control panel and a main visualization area. The side panel contains options for data loading, visualization parameter configuration, and activation of different functionalities. The main area displays the generated graphs based on the processed data, enabling users to interact and explore the information intuitively.

Figure 1 illustrates the visualization of an execution trace from a real parallel application. Specifically, it represents the execution trace of a loop-based application using the StarPU `dmdas` scheduler, running on a machine with 10 CPU threads and 4 GPUs. The top panel shows the iteration progress, while the second panel provides an enriched space-time view with task dependencies and resource idleness. The dataset used for this visualization was obtained from [Pinto et al. 2021], where further details on the execution environment and methodology can be found.

The key functionalities of the interface include:

- Selection of directory and configuration file for data loading;
- Activation and deactivation of different visualization layers, such as legends, makespan, and tasks;
- Generation of static graphs using ggplot2 (Figure 2);
- Interactive visualization with plotly, enabling zooming and data selection;
- Provides options to select labels, set the number of task levels, and input a custom tasks list.
- Captures clicks on the interactive plot to automatically update the tasks list (Job IDs), facilitating easier tracking and analysis.
- Automatic interface adjustments based on selected parameters;
- Buttons for switching between different visualization modes.

The interface is designed to be responsive and dynamically adapt to the amount of displayed information. Additionally, the integration with shinyjs enables dynamic interface manipulations, such as conditional element display and automatic panel layout adjustments.

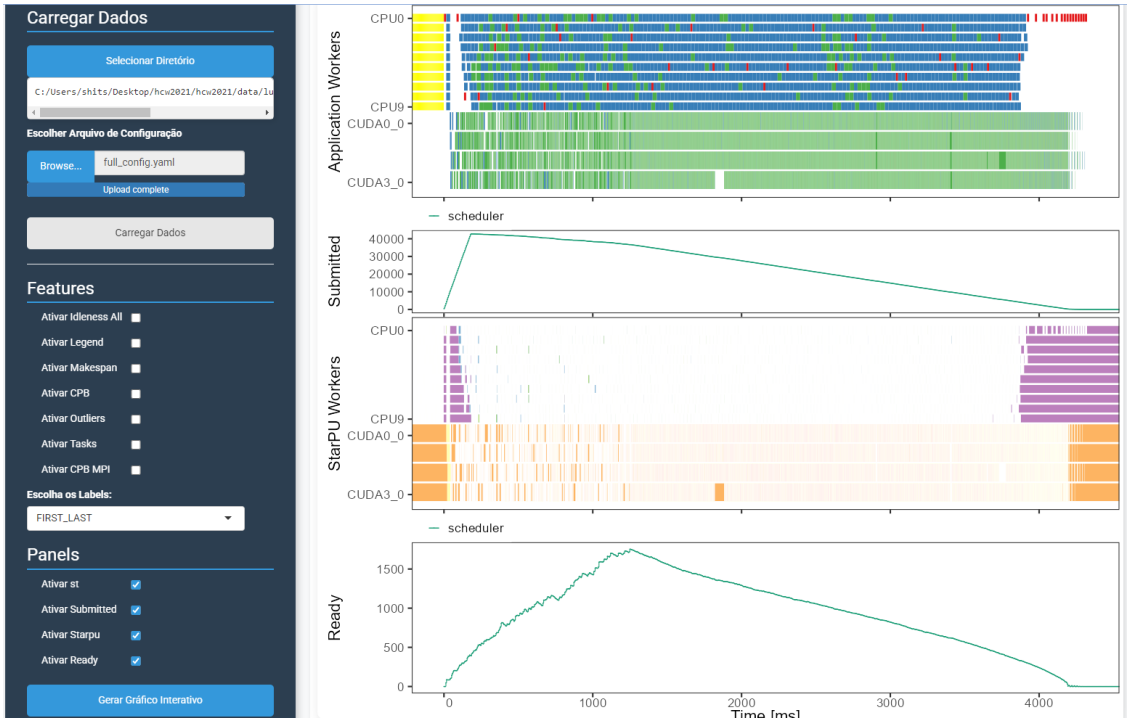

Figure 1. Interface developed in R for StarVZ plot visualization.

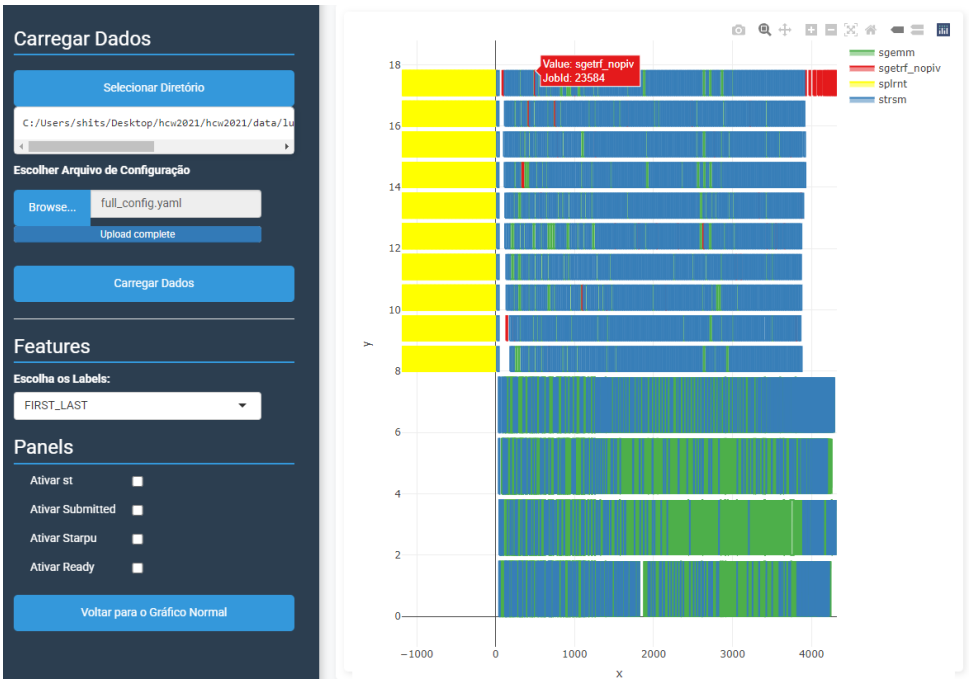

Figure 2. Plotly visualization.

4. Conclusion and Future Improvements

The Interactive R interface developed with Shiny for StarVZ significantly facilitates the visualization of StarPU execution traces, making the analysis more accessible and interactive. The use of modern technologies such as ggplot2 and plotly enhances user experience by enabling interactivity and customization of data visualization.

However, performance remains a limiting factor. Optimizing the performance, especially for large datasets, is an essential point for further improving the tool's scalability and responsiveness. In addition to performance enhancements, future work should focus on extending the system's functionalities. This includes enabling side-by-side comparisons of different execution traces, allowing seamless integration with cloud-based platforms, and implementing new visualization options that enrich the overall experience.

A publicly available repository at `https://github.com/chrisesf/StarVZ-Interface.git` includes the source code for the new StarVZ interface. This study was partially financed by FAPERGS (23/2551-0000861-0) and PDE/FURG (EPEC 2024/2025).

References

Attali, D. (2021). *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*. R package version 2.1.0.

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. (2024). *shiny: Web Application Framework for R*. R package version 1.9.1.

Lucas Mello Schnorr, Vinicius Garcia Pinto, Lucas Leandro Nesi, and Marcelo Cogo Miletto (2022). *starvz: R-Based Visualization Techniques for Task-Based Applications*. R package version 0.7.1.

Müller, K. and Wickham, H. (2023). *tibble: Simple Data Frames*. R package version 3.2.1.

Ordronneau, C. (2024). Développement et maintenance du logiciel vite.

Pedersen, T. L., Nijs, V., Schaffner, T., and Nantz, E. (2022). *shinyFiles: A Server-Side File System Viewer for Shiny*. R package version 0.9.3.

Pinto, V. G., Leandro Nesi, L., Miletto, M. C., and Mello Schnorr, L. (2021). Providing in-depth performance analysis for heterogeneous task-based applications with starvz. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 16–25.

Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC.

ViTE Team (2024). *ViTE: Trace Explorer*. ViTE is a trace explorer. It is a tool to visualize execution traces in Pajé or OTF format for debugging and profiling parallel or distributed applications. It is an open source software licenced under CeCILL-A.

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.