Tracing and Visualization of GPU Offloading Using LTTng and Eclipse Trace Compass

Hrutvi Barad, and Yonghong Yan, College of Computing and Informatics, UNC Charlotte

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

Introduction

- GPU offloading, in which computations shift from the CPU to the GPU, is a transformative force in improving application performance through parallel processing. Understanding the subtleties of GPU execution is critical for realizing the GPU's full potential in application optimization.
- The current study looks into advanced visualization techniques designed for GPU offloading. The goal is develop visualization tools to developers, for better explaining parallelism, identifying performance bottlenecks, and speeding code optimization. These visualization tools will empower developers to maximize GPU utilization, hence improving overall program performance by efficiently tackling intrinsic problems.

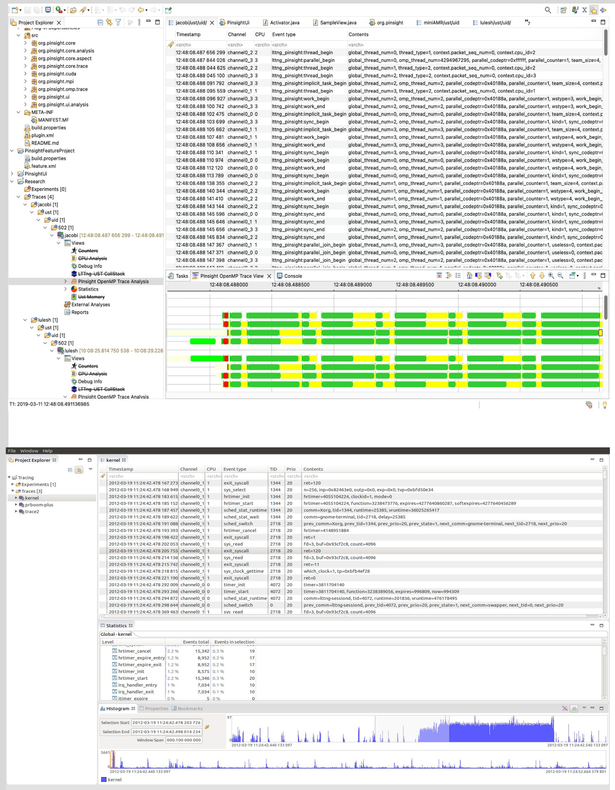
Objectives

- Visualization of data movement between CPU and GPUs: Develop visual representations enlightening the dynamics of Data movement between CPU and GPU memory, aiming to illustrate transfer efficiency and patterns.
- Visualization of concurrency between CPUs and GPUs: Create visualizations of concurrent activity between CPUs and GPUs to help us comprehend parallel processes and synchronization.
- Visualization of performance gaps between applications and CPU/GPU devices, ad resources utilization: Use visual analytics to find performance gaps between applications and CPU/GPU devices, focusing on resource utilization inconsistencies for a more comprehensive examination.
- Identification of performance hotspot for optimization: Using visual analysis, identify and highlight performance hotspots in order to locate areas within applications or system architecture for targeted optimization techniques.

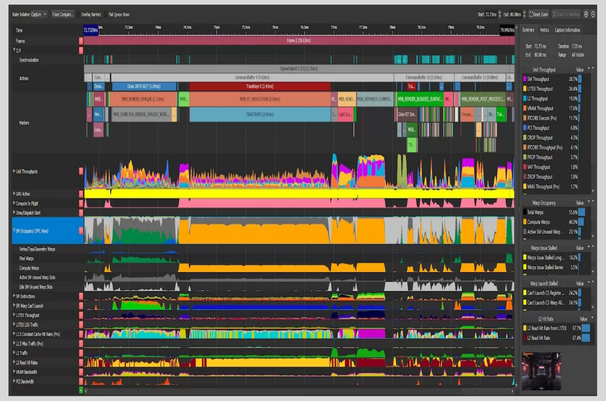
Method

- **Data collecting:** Utilize advanced tracing tools like LTTng and Trace Compass to capture comprehensive data encompassing CPU-GPU interactions, system concurrency, data movement events, and performance gaps within applications and CPU/GPU devices.
- **Visual Analysis Tools:** Use visualization tools like as graphs, charts, and heat maps to visually depict the collected data, indicating bottleneck locations.
- **Bottleneck Identification:** Analyze the visual representations to discover performance bottlenecks, focusing on regions of inefficiency or resource contention within the CPU-GPU system.
- **Validation of recommendations:** Refine recommendations based on continual data gathering and performance assessments by iterating the analysis and optimization cycle.

Results



- Visual Representation: Displays traced system events or metrics across a specified time period.
- Temporal Analysis: Depicts the chronology or sequence of recorded events, allowing for a chronological evaluation of system activity.
- Data Patterns: Draws attention to patterns, irregularities, or trends in the traced data, assisting in the identification of system behavior or performance fluctuations.
- Efficiency Metrics: Depending on the traced data, various system efficiency metrics or resource use patterns can be displayed.
- Trace Compass: Frequently provides customization tools, allowing users to focus on specific events or metrics based on their analysis needs.



FInal Goal for our research

Conclusions

- Conclusively, this study introduces novel visualization methods for understanding GPU offloading, tackling important issues in program optimization. This study explores the graphical analysis and comprehensive data gathering of GPU events using powerful tools such as LTTng, Tracecompass, and Pinsight.
- The results emphasize how important it is to visualize GPU operations in order to comprehend parallelism, spot bottlenecks, and optimize code for better performance. This project attempts to close the gap in the available tools by iteratively improving visualization techniques, enabling developers to efficiently maximize GPU utilization.
- The contributions of this work include providing clear visual depictions of GPU offloading events, assisting with well-informed judgments regarding code optimization, and enabling the maximum utilization of GPU capabilities to improve program performance.

References

- Kannan, Ajaykumar . "Offloading to the GPU: An Objective Approach." *Research Gate*,
- "Q&A: Visualizing Code Traces with Eclipse Trace Compass | Eclipse News, Eclipse in the News, Eclipse Announcement." *Newsroom.eclipse.org*, 21 Dec. 2021, newsroom.eclipse.org/eclipse-newsletter/2021/december/qa-visualizing-code-traces-eclipse-trace-compass. Accessed 29 Nov. 2023.