Numerical Debugging through Reverse Execution: Trace Construction and Reverse Replay in HPC Programs

---

Thesis advisors:

- David DEFOUR, Full Professor, Univ de Perpignan Via Domitia

Location: University of Perpignan Via Domitia, France

Duration: 4-6 months

Keywords: IEEE754, floating-point computation, LLM

Compensation: ~630€/months

---

1. Introduction and Context

High-performance computing (HPC) programs are often subject to numerical errors due to the nature of floating-point operations and numerical approximations. These errors can accumulate or propagate unpredictably in complex computations. To analyze these errors, many tools are available, mainly focusing on **forward error analysis**, which introduces small perturbations to the input and measures their effects on the output.

However, there are no standard tools for performing **backward analysis**, which would involve analyzing a final result and tracing it back to the input values that caused an error. This process, known as **reverse execution**, is challenging due to the non-reversibility of floating-point operators and to the complexity of reconstructing the history of computations, especially in an HPC environment where programs handle large datasets with optimized parallel algorithms.

This thesis aims to propose an innovative solution by constructing an **execution trace** of a program, containing all the information necessary to trace back numerical errors by reversing the operations performed. Specifically, we will use the PIN tool to capture this trace and develop scripts that enable the reverse execution of instructions.

2. Problem Statement

Numerical errors in HPC programs are difficult to diagnose due to their complexity and the nature of floating-point computations. Current analyses focus primarily on the impact of errors on the output based on small perturbations in the input, limiting developers' ability to understand how and why these errors occur.

The main challenge in backward debugging is the **construction of an informative execution trace**, including all intermediate values and types of operations performed. Once this trace is obtained, the difficulty lies in the ability to replay this execution backward, much like watching a movie in reverse, to understand how errors evolved from the output back to the input.

3. Objectives

The primary objective of this thesis is to develop a **reverse execution** debugging tool for HPC programs. The specific sub-objectives are:

1. **Execution Trace Construction**: Use the PIN tool to instrument a program and collect all necessary execution information, including numerical values and types of operations (addition, multiplication, etc.).
2. **Reverse Instruction Replay**: Develop a script or a suite of tools that can replay a program's execution in reverse order to trace back from the output and identify the causes of numerical errors.
3. **Performance Evaluation**: Measure the impact in terms of performance of generating and using the execution trace in complex HPC programs. The tool will be tested on representative case studies, such as numerical simulations or physics-based numerical methods.
4. **Case Studies and Validation**: Apply the developed method to one or more real-world HPC programs and demonstrate its utility for identifying and understanding numerical errors.

4. Methodology

1. **Literature Review**: Start with a review of existing techniques in numerical error analysis, tracing tools such as PIN, and reverse execution approaches in related contexts (e.g., rollback in parallel simulations, automatic differentiation, reverse debugging).
2. **Instrumentation with PIN**: Use the PIN tool (a dynamic instrumentation framework developed by Intel) to insert trace points into the HPC program. PIN allows recording register values, operation types, and other details of program execution.
3. **Execution Trace Construction**: Build an execution trace containing all the necessary information to replay the program in both directions. The trace will include arithmetic operations and intermediate computation values.
4. **Reverse Instruction Replay**: Develop a script or tool capable of replaying the execution in reverse. The tool will interpret the execution trace and reconstruct, step by step, the program’s state by tracing back from the final result to the input.
5. **Experimentation and Validation**: Test the solution on a set of HPC programs with different numerical characteristics (i.e. gauss solver with pivoting). Measure the tool's ability to identify the sources of numerical errors.