Tracing Optimization for Performance Modelling and Regression Detection

Kaveh Shahedi, Heng Li
Polytechnique Montreal

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

01
The Problem

What is Performance Modelling?

Observe
Software Behavior

- Num. of Inputs
- Complexity of Inputs
- Functions Executions

What is Performance Modelling?

Observe
Software Behavior

- Num. of Inputs
- Complexity of Inputs
- Functions Executions

↓

Predict
Performance Metrics

- CPU Utilization
- Execution Time
- Throughput

What is Performance Modelling?

Observe
Software Behavior

- Num. of Inputs
- Complexity of Inputs
- Functions Executions

↓

Predict
Performance Metrics

- CPU Utilization
- Execution Time
- Throughput

Performance Regression?

The performance model **MUST** be able to **DETECT** the performance **regressions**

In this study, the performance model is a regression model, functions tracing data are the inputs, and the program’s execution time is the output.

The Trade-Off

A: GOOD

Performance Model Precision

Using **functions tracing**, the **accuracy** of the **performance model** is **great**!

```
Performance Model

R2 Score: > 99%
Mean Error: < 5%
```

The Trade-Off

A: GOOD

Performance Model Precision

Using **functions tracing**, the **accuracy** of the **performance model** is **great**!

Performance Model

```
R2 Score: > 99%
Mean Error: < 5%
```

BUT

B: BAD

The Overhead of Tracing

The **added overhead** to the **system** regarding the **execution time** and **storage usage** is **massive**!

Resources

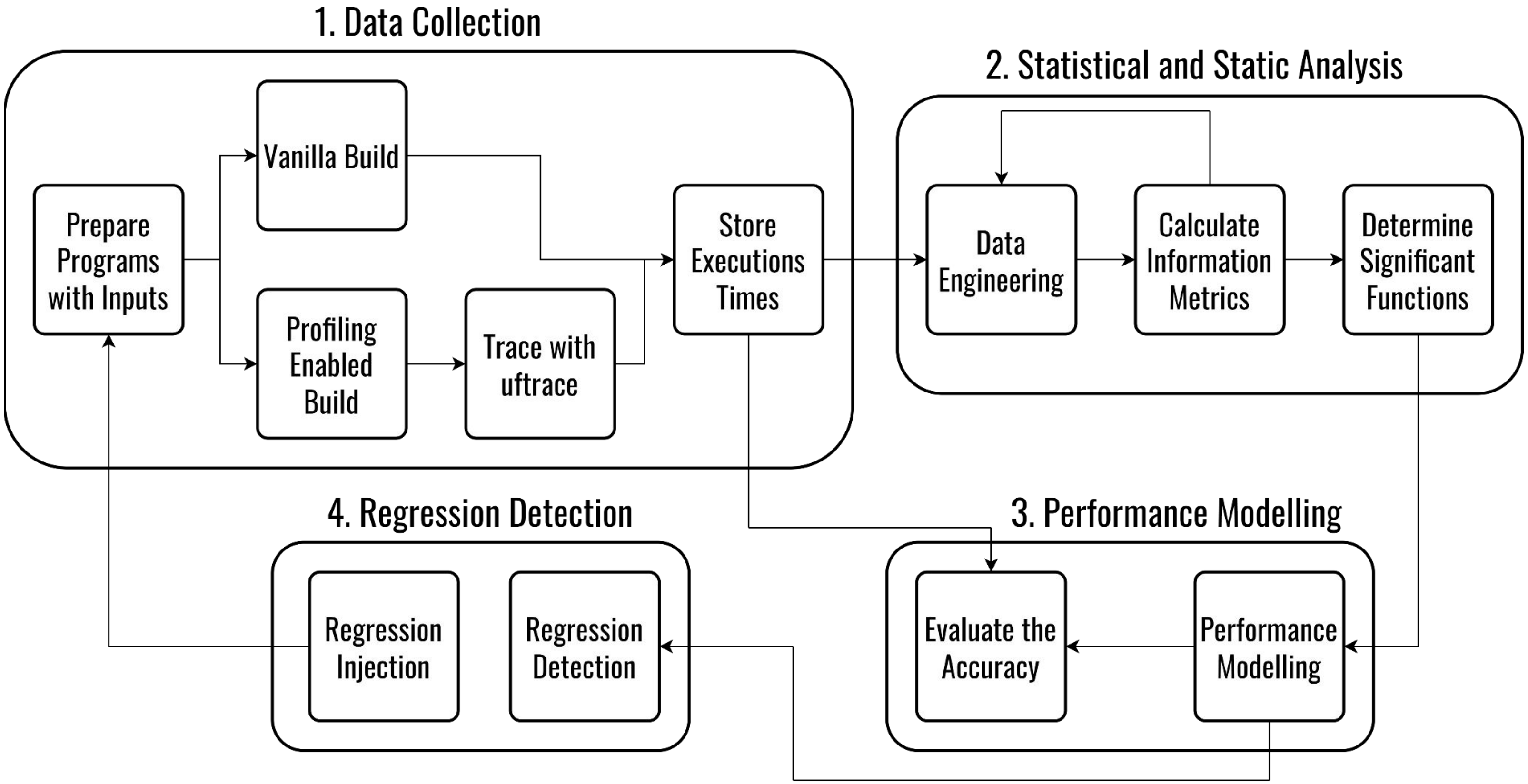
```
Mean Time Overhead: > 55%
Mean Storage Overhead: >> 1000%
```

But, **with respect to performance modelling, not all** of the **functions** have **significant impact** on the **model** and can be **removed** from **tracing**

02

The Methodology

Overall Steps



1. Data Collection

I. **Programs (Benchmarks)**
   A. SPEC CPU 2017: 631.deepsjeng_s (Int), 638.imagick_s (FP)
   B. SPEC MPI 2007: 104.milc
   C. SU2
   D. PARSEC: freqmine

1. Data Collection

I. **Programs (Benchmarks)**
    A. SPEC CPU 2017: 631.deepsjeng_s (Int), 638.imagick_s (FP)
    B. SPEC MPI 2007: 104.milc
    C. SU2
    D. PARSEC: freqmine

II. **Input Generation ->** More than 10k combinations of inputs

1. Data Collection

I. **Programs (Benchmarks)**
   A. SPEC CPU 2017: 631.deepsjeng_s (Int), 638.imagick_s (FP)
   B. SPEC MPI 2007: 104.milc
   C. SU2
   D. PARSEC: freqmine

II. **Input Generation ->** More than 10k combinations of inputs

III. **Run the Programs in Vanilla Mode and Full Tracing, and Store the Execution Times along with Storage Usage**

2. Statistical and Static Analysis

I. **Calculating Several Metrics of the Functions' Executions**
   A. Entropy, Coefficient of Variant, and Ridge Regression Coefficients
   B. Their union, intersection, or other combinations can be used

2. Statistical and Static Analysis

I. **Calculating Several Metrics of the Functions' Executions**
   A. Entropy, Coefficient of Variant, and Ridge Regression Coefficients
   B. Their union, intersection, or other combinations can be used

II. **Finding the Significant Functions**
   A. Sorting the functions based on their metrics
   B. Cluster them into two groups (significant and insignificant)

2. Statistical and Static Analysis

I. **Calculating Several Metrics of the Functions' Executions**
   A. Entropy, Coefficient of Variant, and Ridge Regression Coefficients
   B. Their union, intersection, or other combinations can be used

II. **Finding the Significant Functions**
   A. Sorting the functions based on their metrics
   B. Cluster them into two groups (significant and insignificant)

III. **Extracting Functions Characteristics**
   A. Complexity, LoC, Number of loops, Number of calls, etc.
   B. These characteristics will be used for the further steps of this study

3. Performance Modelling

I. **Vanilla Performance Model**
   A. Building a performance model with the fully instrumented data
   B. Very precise due to the number of data and their diversity

3. Performance Modelling

I. **Vanilla Performance Model**
   A. Building a performance model with the fully instrumented data
   B. Very precise due to the number of data and their diversity

System's performance has changed significantly

3. Performance Modelling

I. **Vanilla Performance Model**
   A. Building a performance model with the fully instrumented data
   B. Very precise due to the number of data and their diversity

System's performance has changed significantly

II. **Optimized Performance Model**
   A. The created performance model only with the significant functions
   B. The lost precision should be negligible

3. Performance Modelling

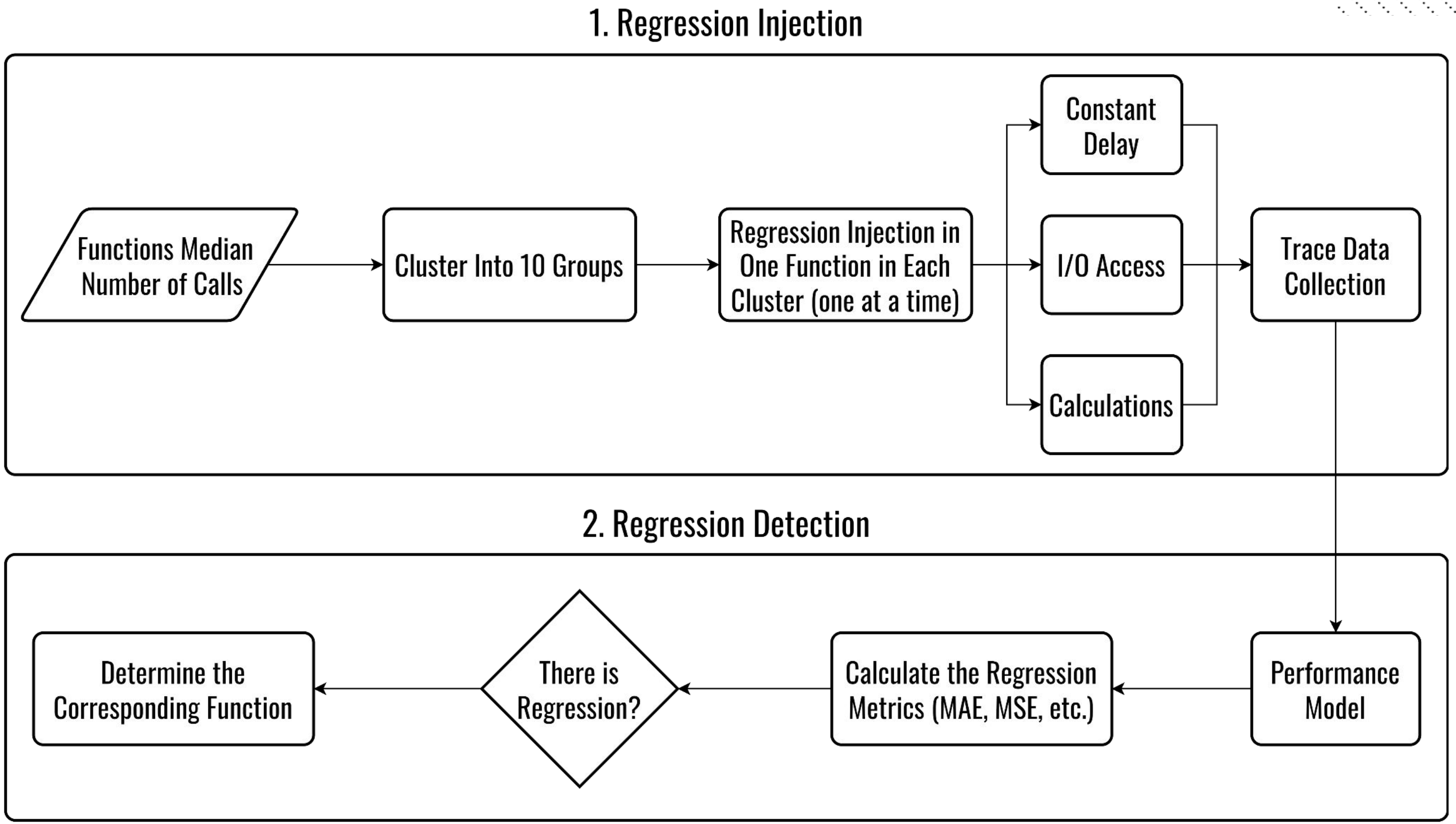
I. **Vanilla Performance Model**
   A. Building a performance model with the fully instrumented data
   B. Very precise due to the number of data and their diversity

System's performance has changed significantly

II. **Optimized Performance Model**
   A. The created performance model only with the significant functions
   B. The lost precision should be negligible

III. **Evaluating the Optimized Performance Model**
   A. The accuracy of the performance model itself
   B. Performance regression detection

4. Regression Detection



03
The Results

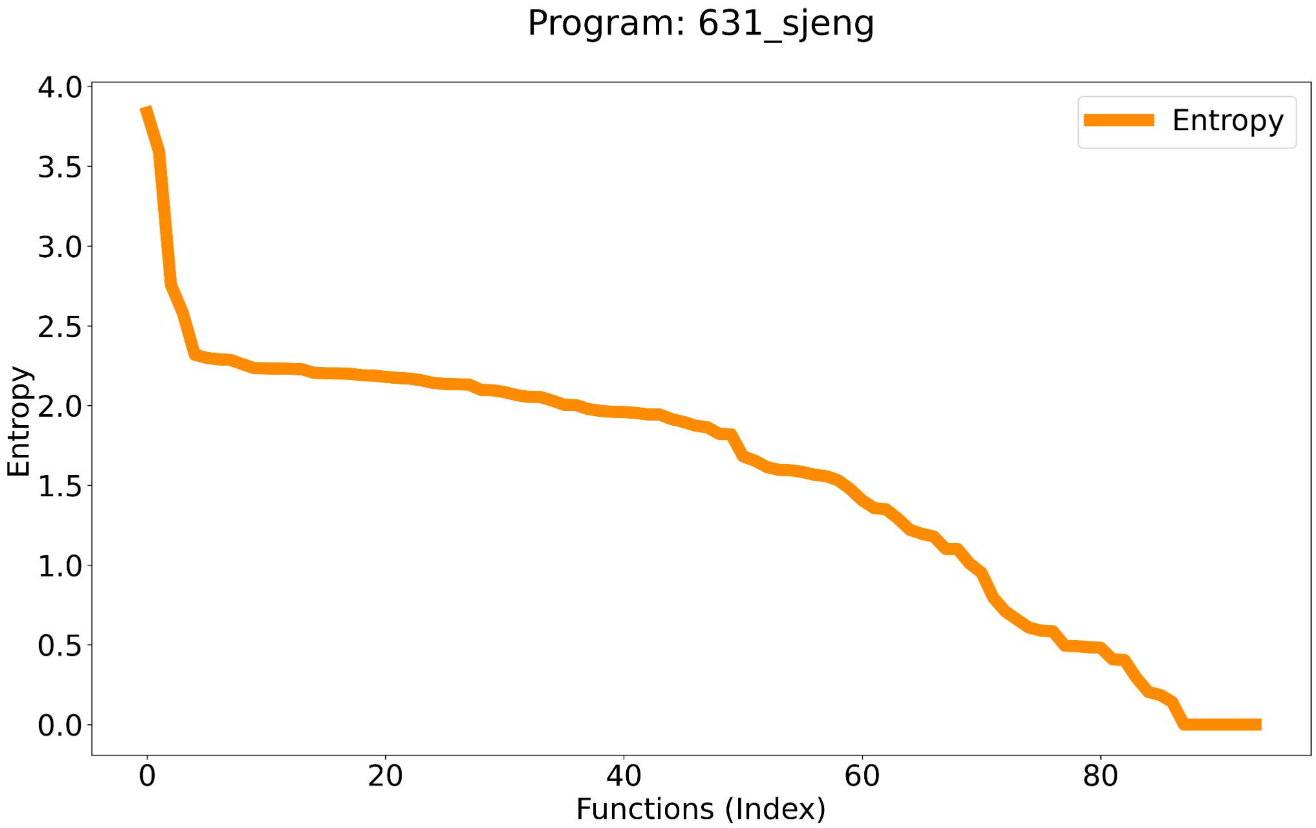
1. Data Collection

Programs Collected Data

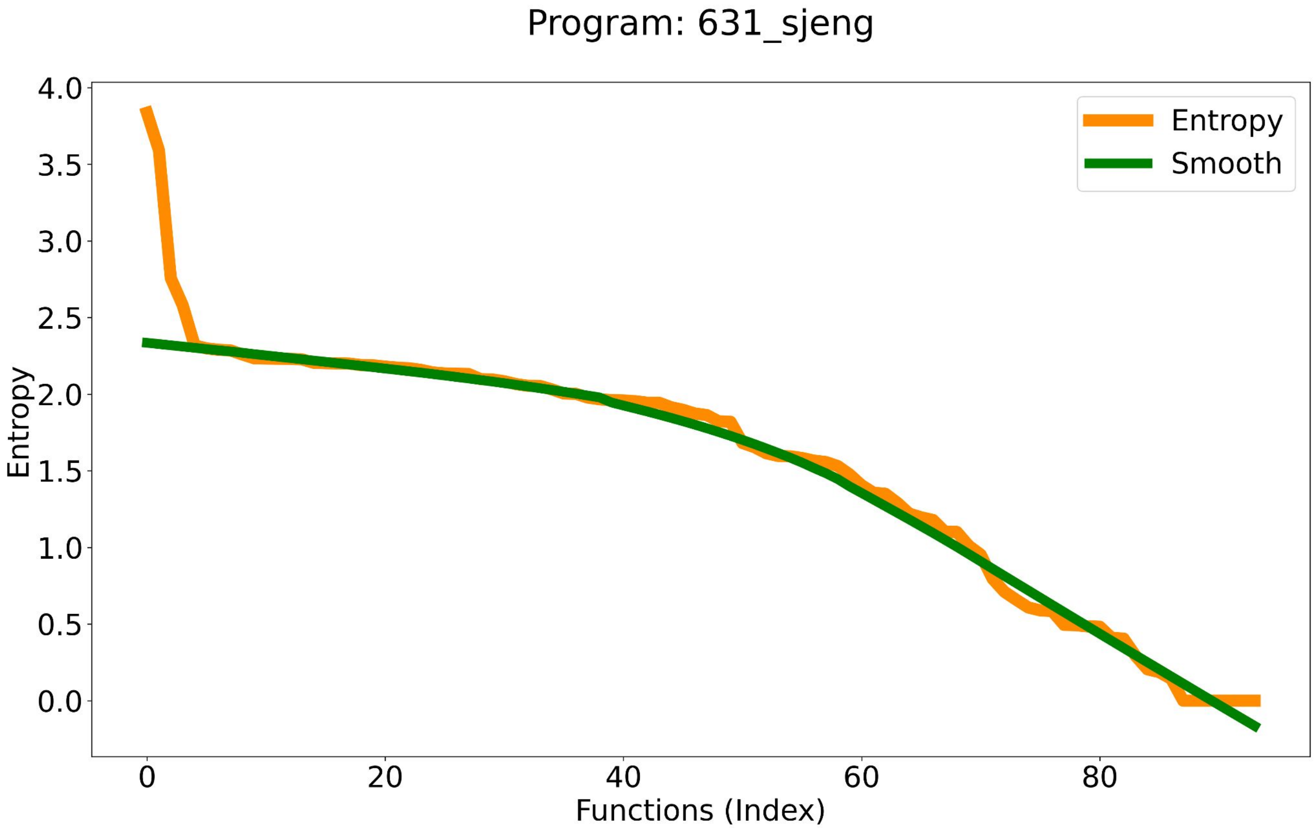
- 631.deepsjeng_s: 9,000 executions
- 638.imagick_s: 2,350 executions
- 104.milc: 4,650 executions
- SU2: 4,700 executions
- Freqmine: 4,200 executions

Times	Storage Usage	Parameters	Functions
Total execution times of the program (vanilla, fully traced)	The overhead of used storage by tracing	The input parameters for that specific execution	Self time, cumulative time, number of calls

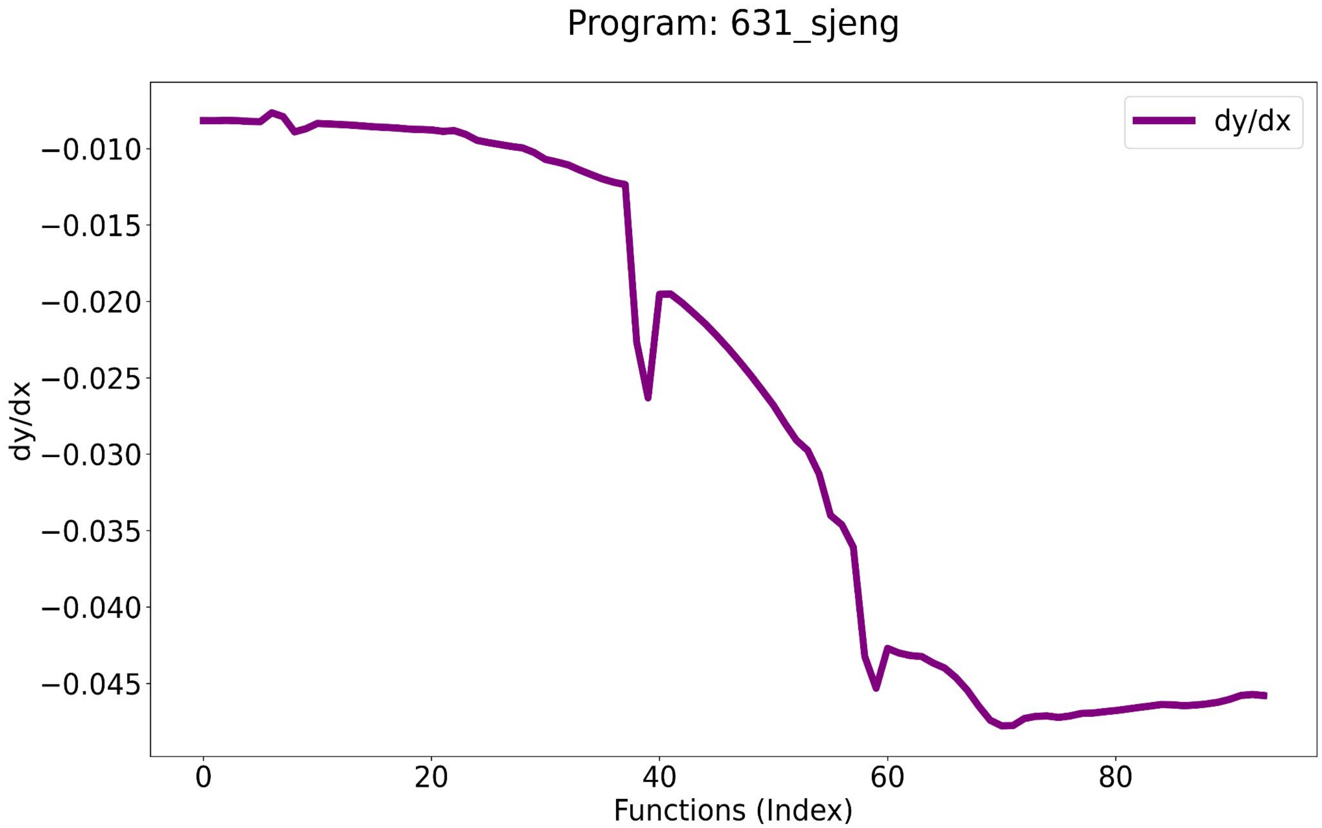
2. Statistical and Static Analysis



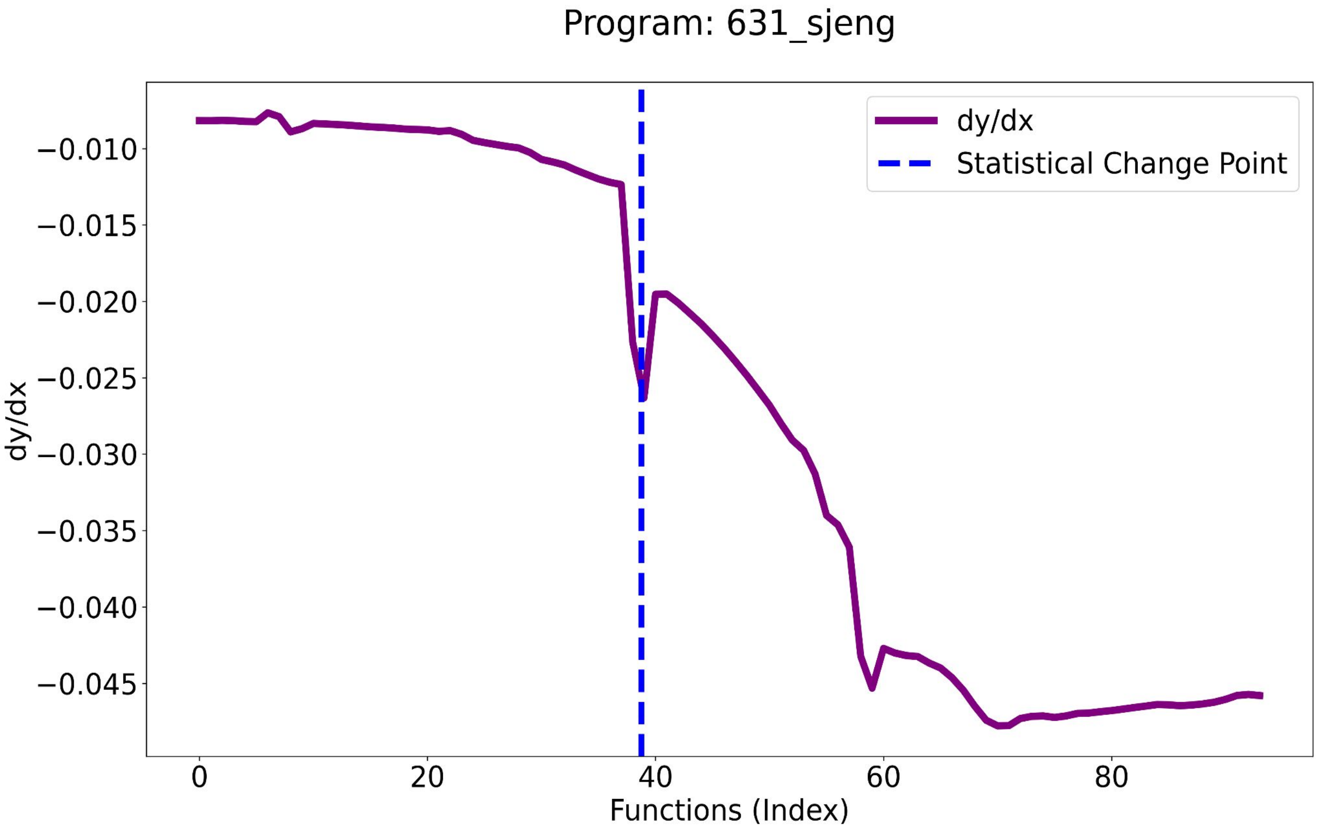
2. Statistical and Static Analysis



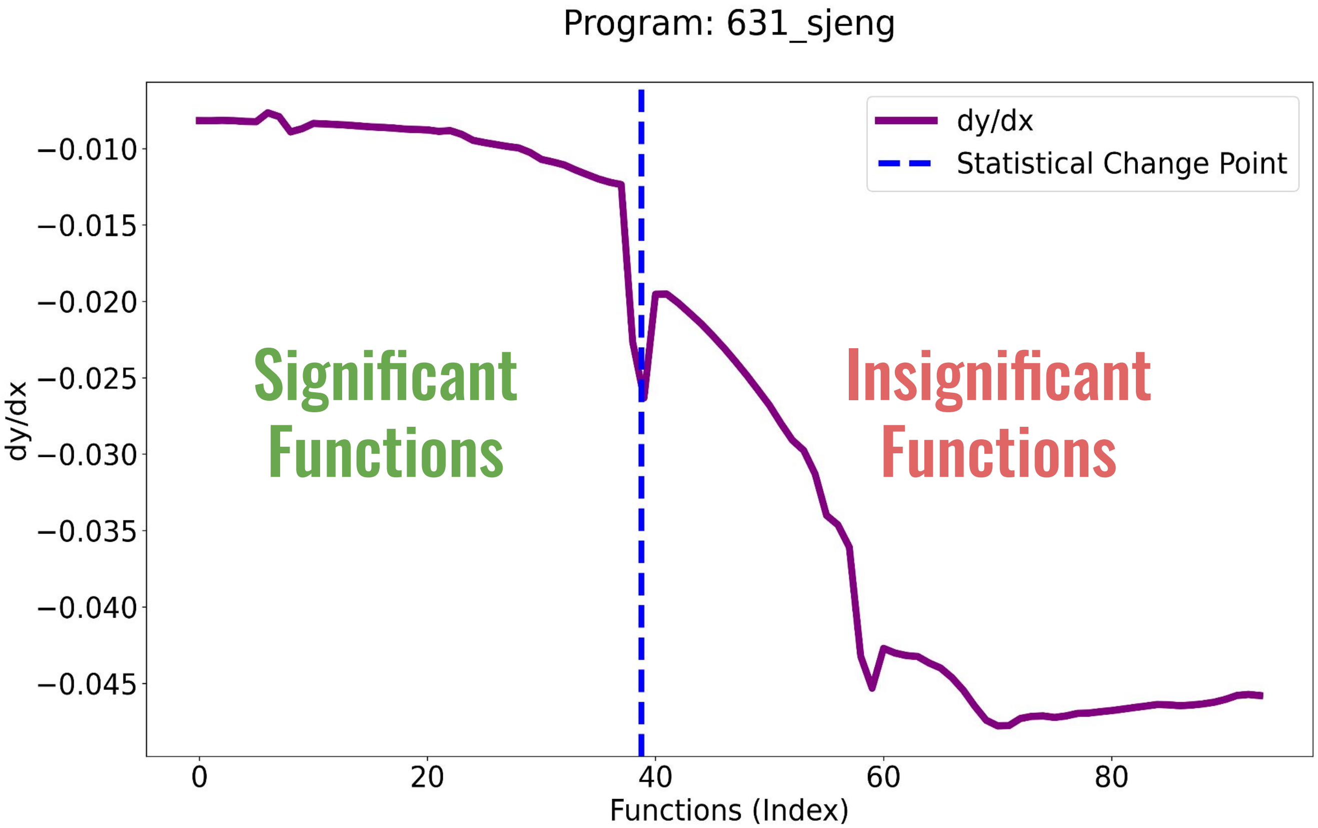
2. Statistical and Static Analysis



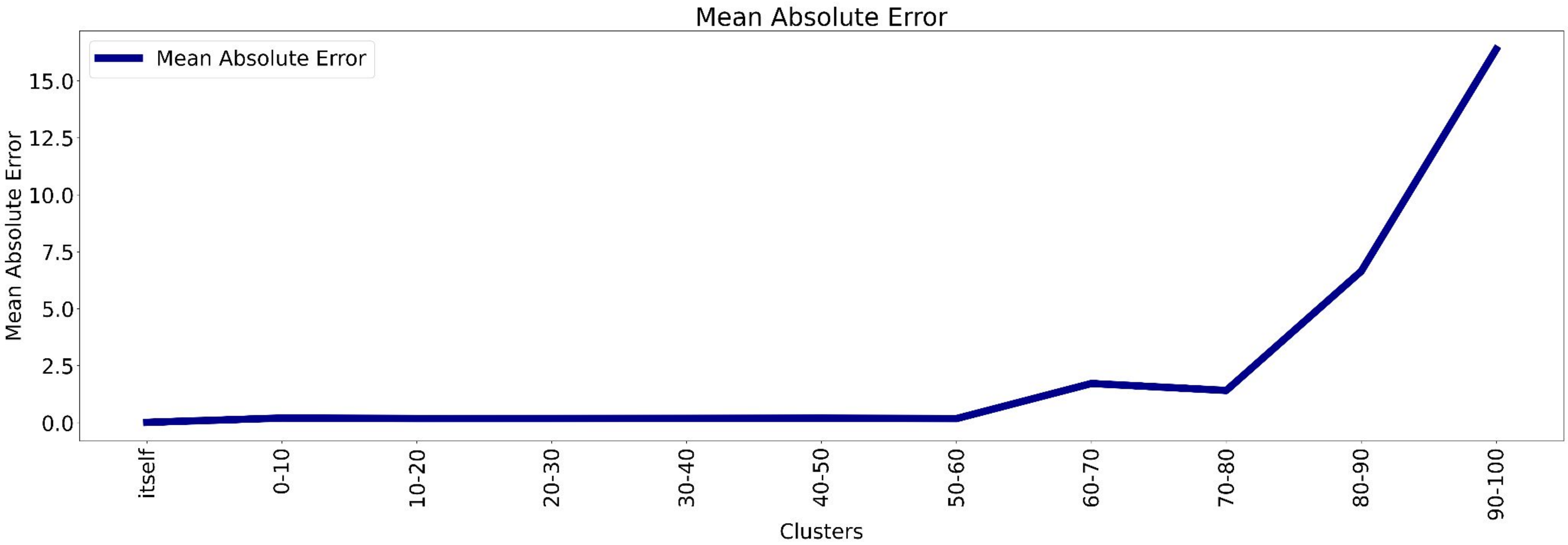
2. Statistical and Static Analysis



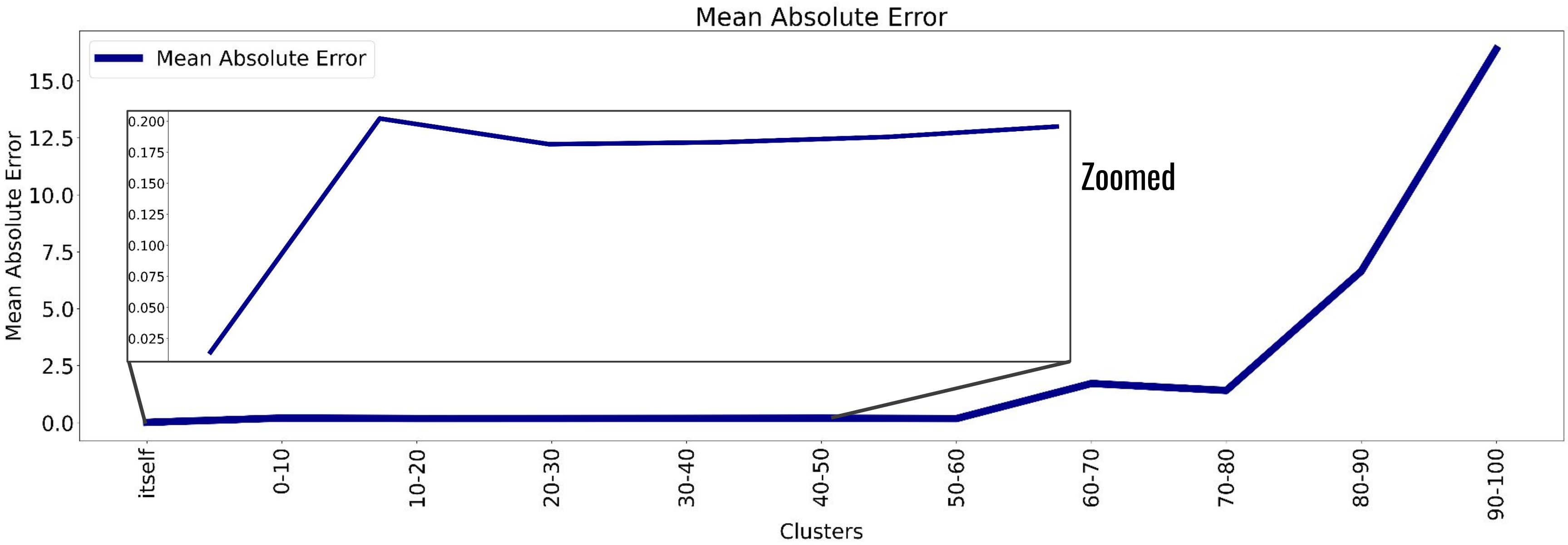
2. Statistical and Static Analysis



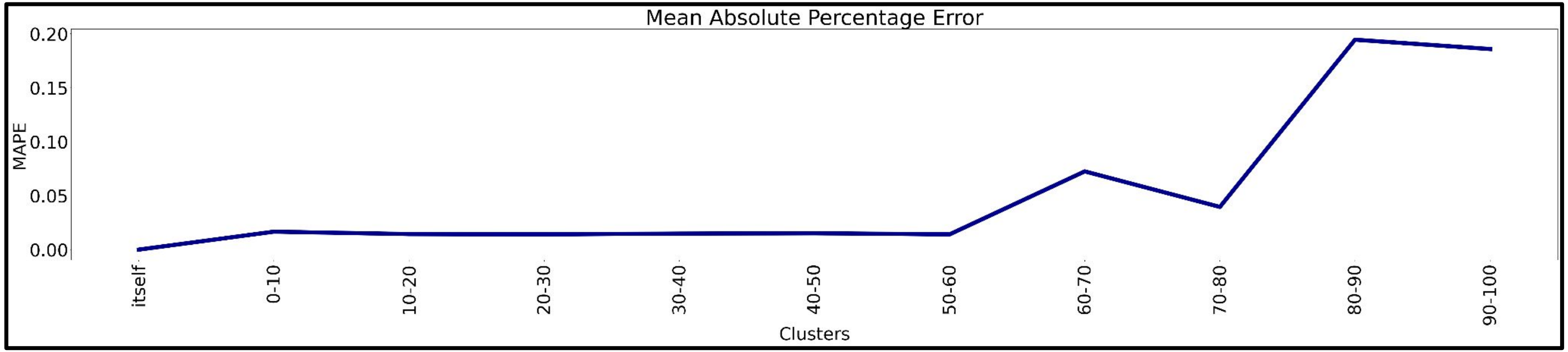
3. Regression Injection and Detection



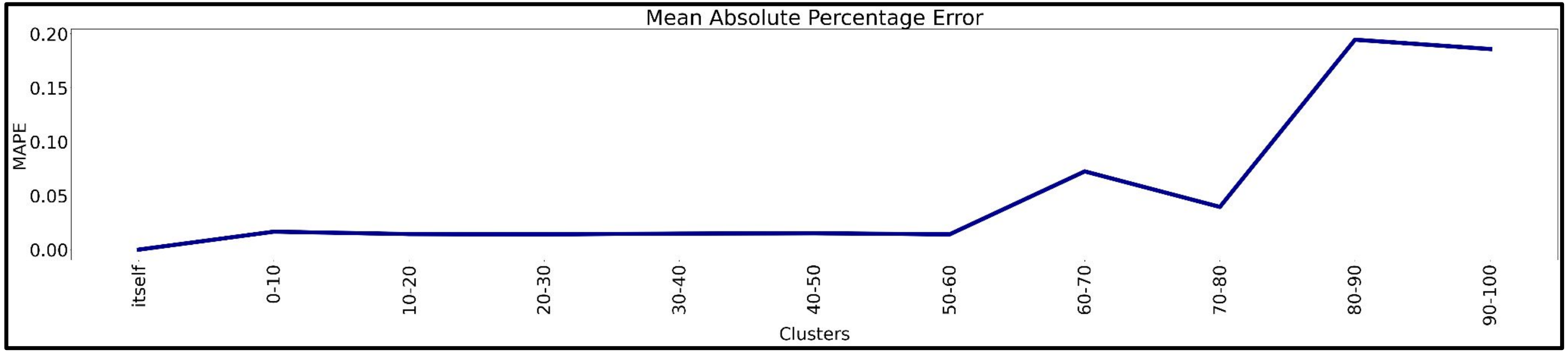
3. Regression Injection and Detection

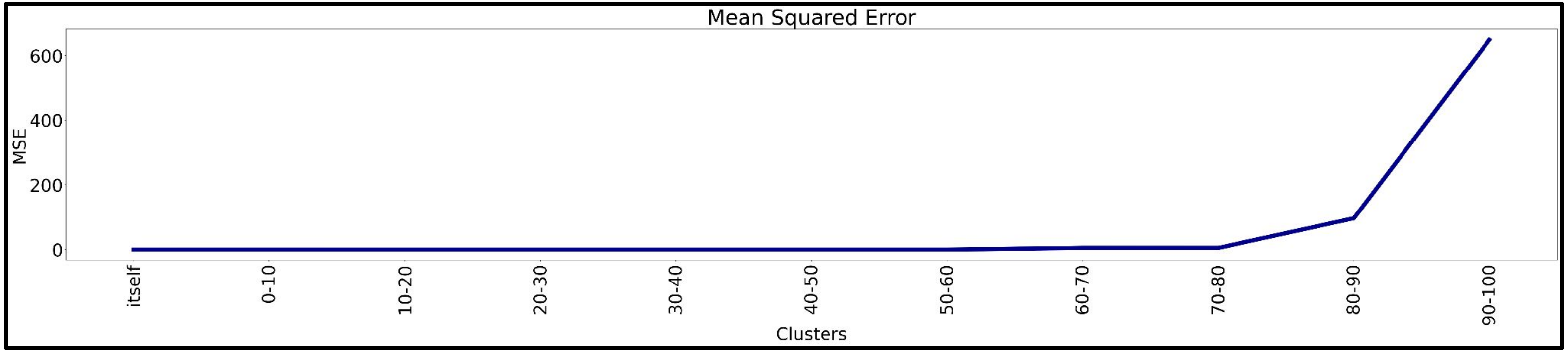


3. Regression Injection and Detection



3. Regression Injection and Detection





04 The Next Steps

Next Steps

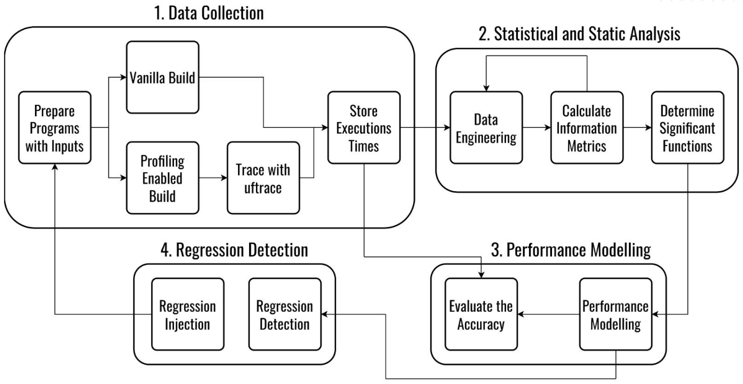
I. **Regression Detection Analysis**
   A. Investigate further the performance model's accuracy
   B. Change the injected regressions types

Next Steps

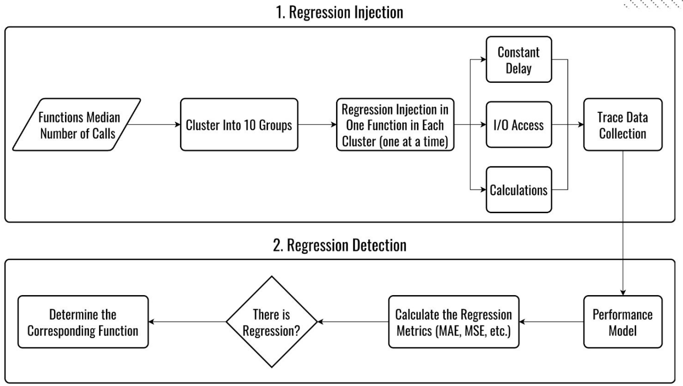
I. **Regression Detection Analysis**
   A. Investigate further the performance model's accuracy
   B. Change the injected regressions types

II. **Statistical and Static Analysis of the Characteristics of the Functions**
   A. Check whether it is possible (and accurate) to build an optimized performance model just through a statistical analysis of the program's source code
   B. Compare the impact of each function metric (LoC, Loops, etc.) on the performance model's accuracy

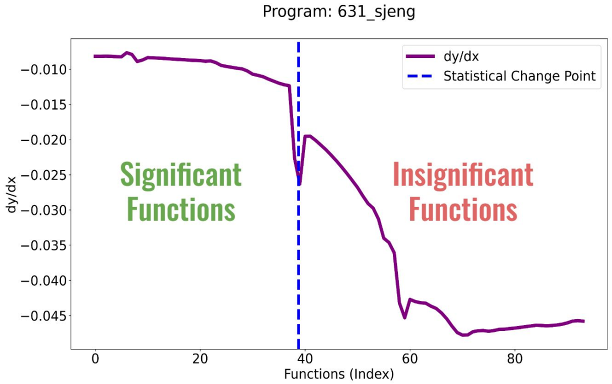
Overall Steps



4. Regression Detection



2. Statistical and Static Analysis



3. Regression Injection and Detection

linkedin.com/in/**kavehshahedi**

kaveh.shahedi@polymtl.ca