Parallel Code Coverage

Ecse 420 - Final Project

Project 21
Alexa Normandin
Alexis Franche
Maxime Cardinal
Oliver Murphy

Intro & Problem

- Problem
  - Testing is a big part of software development. It is time consuming and expensive
  - Our goal is to save developper time in finding the best input to test code

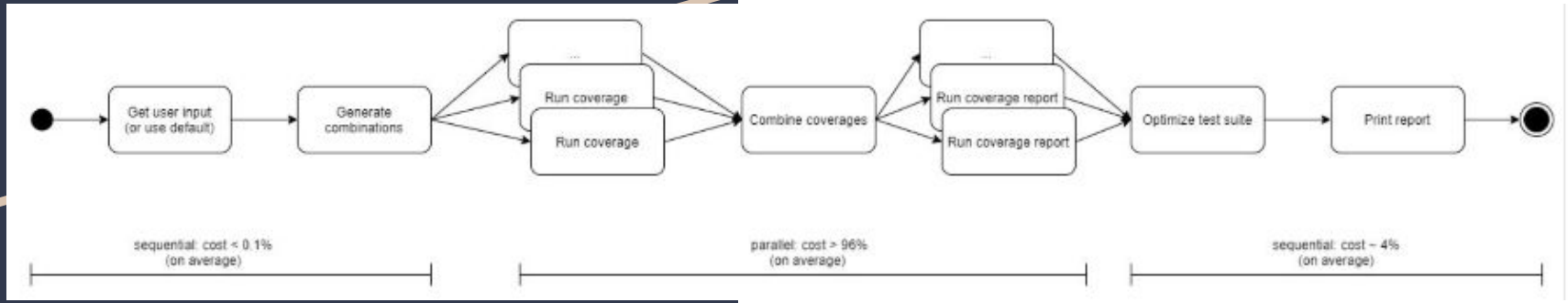
- Our tool
  - Aims to feed testers a set of input to their code that will result in a maximum statement coverage.
  - Minimize the size of the input and maximize the coverage percentage on the code.

Design Approach

- Coverage library
- Failed Attempt with PyCuda and CUDA
- Multiprocessing library
- Threading Vs. Multiprocessing
- Python Global Interpreter Lock
- Multiprocessing constraints
- Instantiating more Processes than CPU cores available using multiprocessing

Parallelization Scheme

- Before parallelization:
  - Generate all possible combinations of inputs
- In parallel:
  - Run each combination of inputs using the coverage library



Live Demo

- CMD Command:

```
python test_coverage_optimization.py path_to_program arg1 arg2 arg3 ...
```

- Greedy Approach
  - Missed Lines vs. Hit Lines
- Report

```
Test Suite Coverage Optimization
--------------------------------

Code coverage: 0.9285714285714286
With inputs: [['1.1', '4.7'], ['1.3', '3.8']]

Total lines covered: 13
Lines covered: [1, 2, 4, 5, 7, 8, 10, 12, 14, 16, 19, 20, 22]

Total lines missed: 1
Lines missing: [17]
```

Q & A

Thank you for your attention!

Do you have any questions?

```
1  Prints (int a, int b) {
2  int result = a+ b;
3  If (result> 0)
4      Print ("Positive", result)
5  Else
6      Print ("Negative", result)
7  }
```