Mutation Testing for stream processing engine

Project, CS 731: Software Testing
Hardik Khandelwal (IMT2020509)
Tejas Sharma (IMT2020548)

Goal

The project aims to understand and implement mutation testing through open source tools and write effective test cases for a python based data stream processor.

About the Code

Github : https://github.com/Tejsharma15/Stream-Mutator

Our real time stream processing engine consists of two parallel threads-

1. Data generator : Generates data every second with the specified attributes at the given input
2. Consumer : Reads the generated data on the fly and contains several inbuilt algebraic aggregate functions such as sum, min/max, average, standard deviation, variance, palindrome finding etc.
3. Driver : Runs the generator and consumer parallelly in threads. It collects results from both producer and consumer and plots the throughput graph.

Mutation Testing

1. Unit level testing : We have used the python module - Mutpy for unit level mutations. Mutpy supports the standard unittest module for loading test cases and applies mutation on AST level.
2. Integration level testing : We have written our own tool for integration mutation testing in python. It does basic parsing of the given modules and extracts methods and parameters with their types. Then it performs following mutations:
    a. Parameter value replacement
    b. Function call replacement
    c. Unary operator insertion in arguments

For each mutation applied, it runs the test cases on both original and mutated program and generates the mutation score at the end.

Steps to Run

```
git clone https://github.com/Tejsharma15/Stream-Mutator
python3 -m venv env
source env/bin/activate
pip install requirements.txt
cd mutationTesting
```

To run unit tests -

```
mut.py -target ../source/consumer.py -unit-test matpyTests.py -m
```

To run integration tests -

```
python3 integration.py ../source/consumer.py
```

Results

Unit level mutation testing results -

```
    - [# 164] ROR consumer: [0.03913 s] killed by test_generateData (matpyTest.CalculatorTest)
    - [# 165] ROR consumer: [0.04340 s] survived
    - [# 166] ROR consumer: [0.03968 s] killed by test_generateData (matpyTest.CalculatorTest)
    - [# 167] ROR consumer: [5.00669 s] timeout
    - [# 168] ROR consumer: [0.04236 s] killed by test_generateData (matpyTest.CalculatorTest)
 [*] Mutation score [151.13527 s]: 75.0%
    - all: 168
    - killed: 97 (57.7%)
    - survived: 42 (25.0%)
    - incompetent: 0 (0.0%)
    - timeout: 29 (17.3%)
```

Integration level mutation testing -

```
----------------------------------------------------------------------------------------------------
Mutation :  ['../source/consumer.py', 'findStdStream', 335, '        result = -484\n', '        result = calculate_std_dev(df)\n']
F
======================================================================
FAIL: test_findStdStream (test_cases.IntegrationTests)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/tejsharma/personalProjs/ST/IMT2020548_assignment2/IMT2020548_assignment2/streamMutator/mutationTesting/test_cases.py", li
ne 79, in test_findStdStream
    self.assertEqual(findStdStream("thread_1", 1, 1)[0], 3.2186953878862163)
AssertionError: -484 != 3.2186953878862163

----------------------------------------------
Ran 1 test in 0.008s

OK
----------------------------------------------
Number of mutants created :  30
Number of mutants killed :  22
MUTATION SCORE :  0.7333333333333333
```

Contributions

Tejas - Stream Consumer program, test cases, unit level mutation
Hardik - Data generator + Integration level mutation tool