DynaQFocus: Focusing test prioritization on builds with test failures

Emad Fallahzadeh and Peter C Rigby
Concordia University
CSER 2021

emad.fallahzadeh@concordia.ca

https://users.encs.concordia.ca/~pcr/
peter.rigby@concordia.ca

Introduction

- Big companies like Google make lots of **changes** per minute
- They run thousands of **tests** to verify code changes
- They follow **Continuous Integration** (CI) process
  - Requires **rerunning** tests for each change
- It **delays** release in a rapid release environment
- **Test prioritization** can help

Related works

- Some studies focused on pre-submit test-case selection
- Others conducted test case prioritization after submitting the change
- Kim and Porter were pioneers in using historical test failures for test prioritization
- Elbaum et al. used a combination of pre-submit selection and post-submit prioritization

Our contribution

- Design a new test prioritization algorithm
  - Bases on the hypothesis that bugs might cluster
  - One failing test might be a clue for the other ones
- Analyze testing datasets features in big projects

Prioritization Algorithms

- BatchedFIFO (baseline)
- GoogleTCP
- DynaQFocus
- DynaQFocusFail

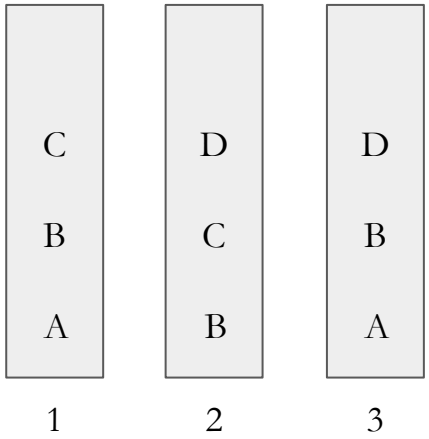
BatchedFifo Algorithm

Algorithm 1: BatchedFifo

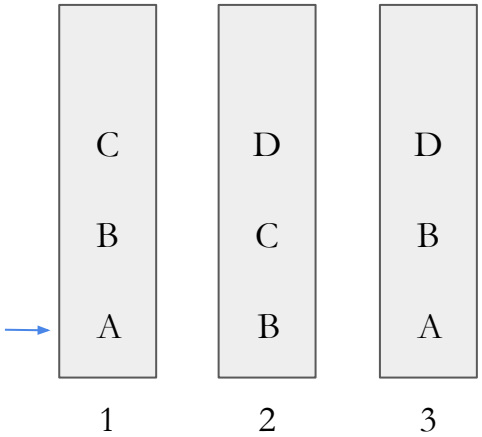
Result: BatchedFifo get builds one after another and runs one test from each.

```
1 while There are more builds do
2 |   fill dispatchQueue with b builds;
3 |   while dispatchQueue is not empty do
4 |   |   build = dispatchQueue.getNextBuild();
5 |   |   run(build.getNextTest());
6 |   end
7 end
```

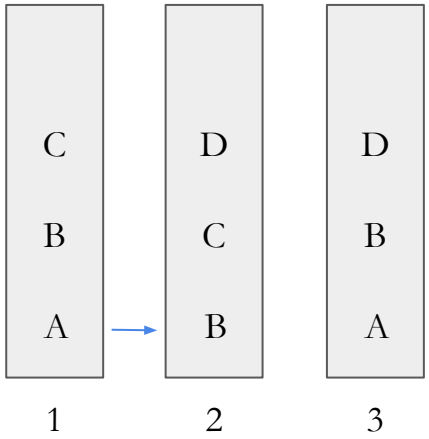
BatchedFifo Simulation



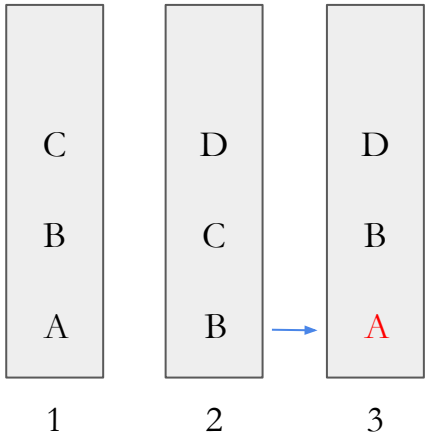
BatchedFifo Simulation



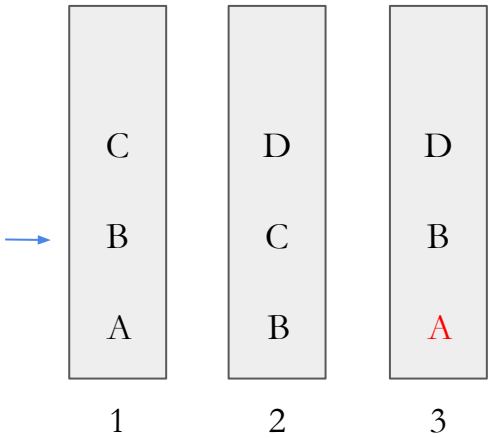
BatchedFifo Simulation



BatchedFifo Simulation



BatchedFifo Simulation



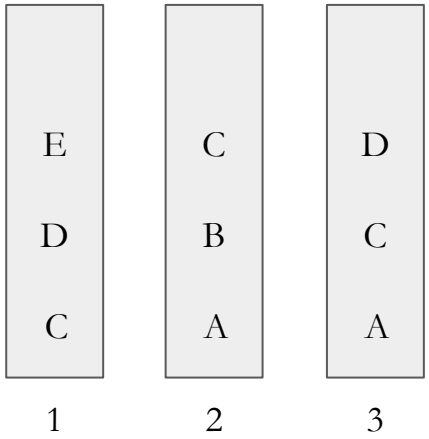
GoogleTCP Algorithm

Algorithm 2: GoogleTCP

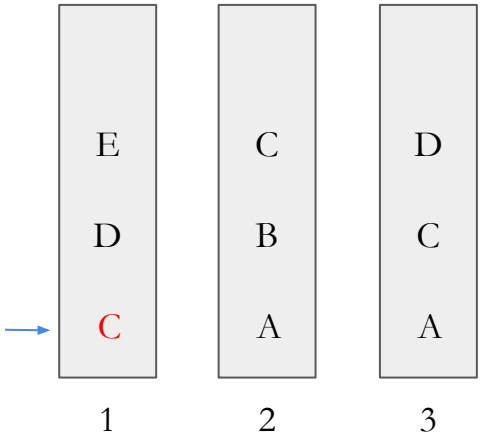
Result: GoogleTCP prioritizes the test cases for the next run after running the test cases inside the dispatch queue each time

```
1 while There are more builds do
2     fill dispatchQueue with b builds and order the test cases
        in each build based on their previous failures;
3     while dispatchQueue is not empty do
4         build = dispatchQueue.getNextBuild();
5         run(build.getNextTest());
6     end
7 end
```

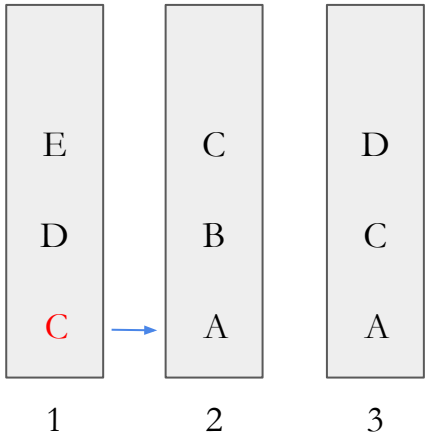
GoogleTCP Simulation



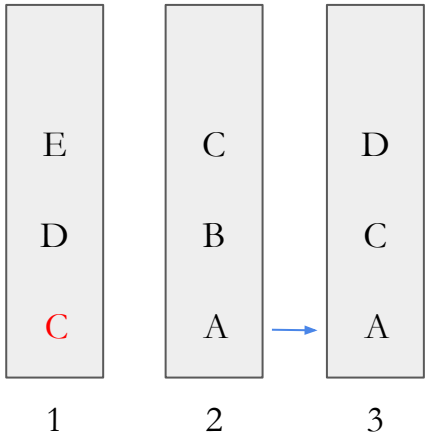
GoogleTCP Simulation



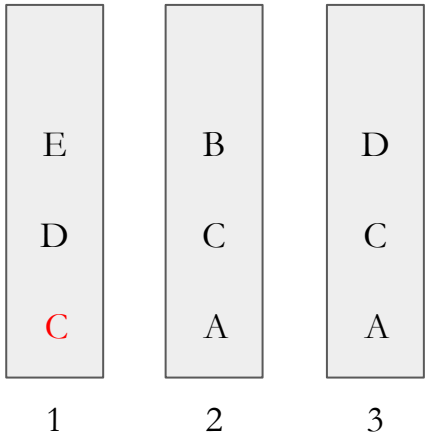
GoogleTCP Simulation



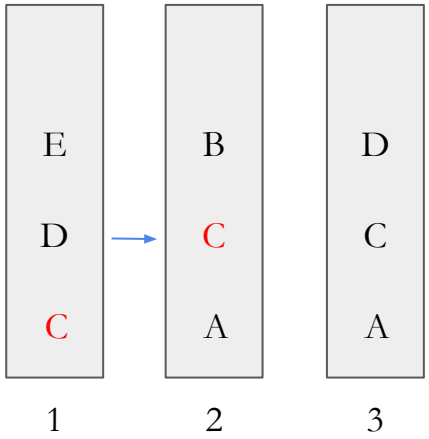
GoogleTCP Simulation



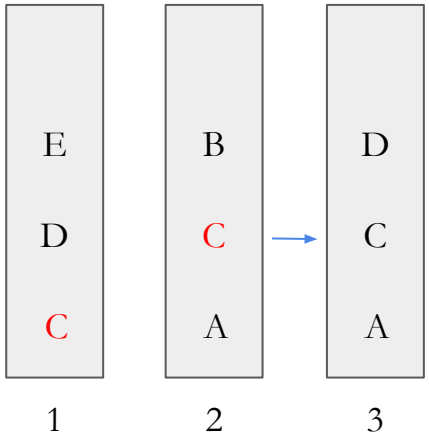
GoogleTCP Simulation



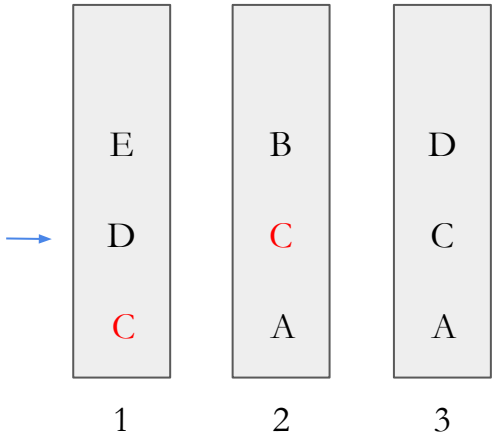
GoogleTCP Simulation



GoogleTCP Simulation



GoogleTCP Simulation



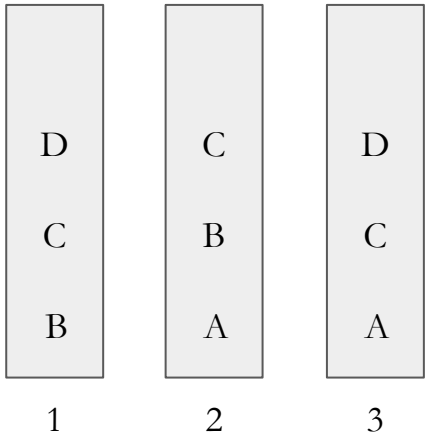
DynaQFocus

Algorithm 3: DynaQFocus

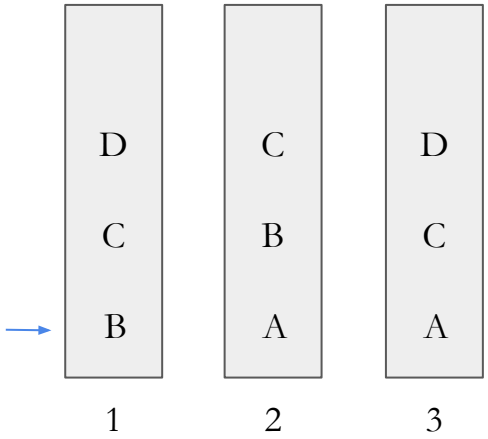
Result: DynaQFocus focuses on code changes that have a failing test, prioritizing the tests of this build above other builds.

```
1  /* we run tests from b builds to avoid
      starvation                                   */
2  while There are more builds do
3      fill dispatchQueue with b builds;
4      while dispatchQueue is not empty do
5          build = dispatchQueue.getNextBuild();
6          verdict = run(build.getNextTest());
7          /* on pass, go to the next build        */
8          if verdict == failure then
9              /* focus: run all tests for the current
                 build                              */
10             runAllTests(build);
11         end
12     end
13 end
```

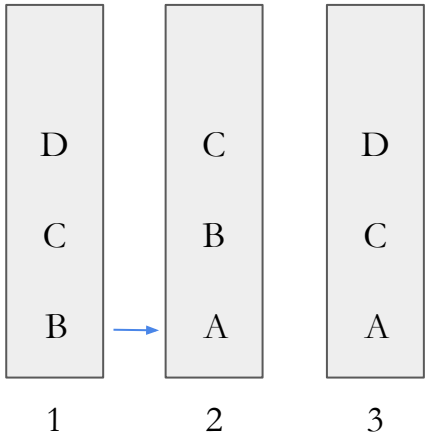
DynaQFocus Simulation



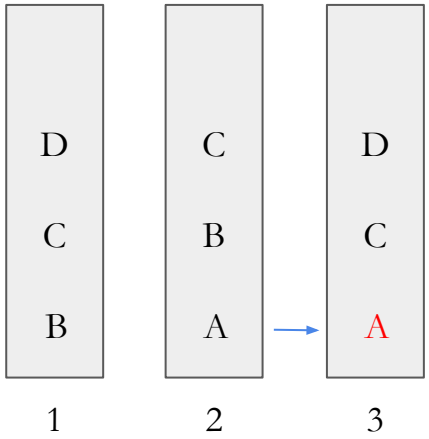
DynaQFocus Simulation



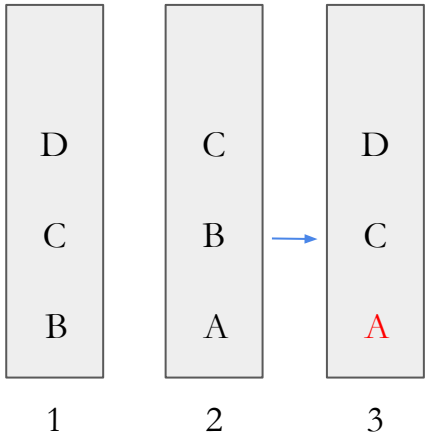
DynaQFocus Simulation



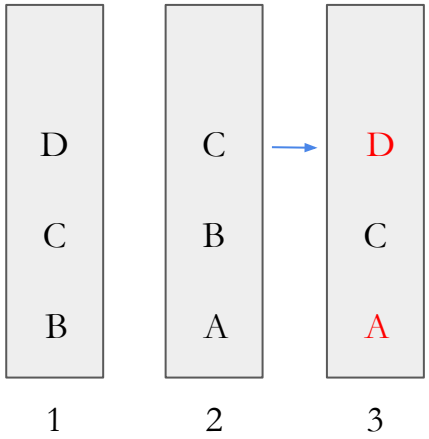
DynaQFocus Simulation



DynaQFocus Simulation



DynaQFocus Simulation



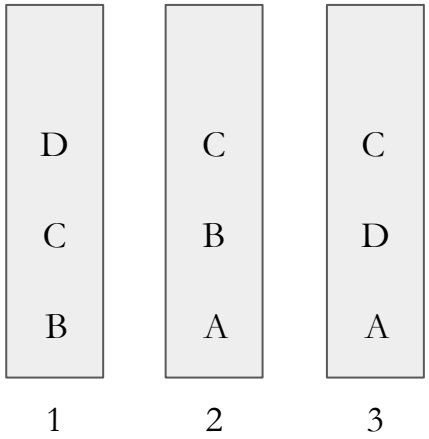
DynaQFocusFail

Algorithm 4: DynaQFocusFail

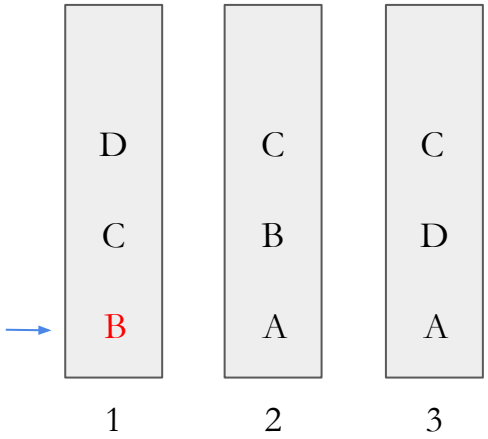
Result: DynaQFocusFail prioritizes the tests in builds based on their previous failures and also focuses on a buggy build.

```
1  while There are more builds do
2      fill dispatchQueue with b builds with tests prioritized in
         each build based on their previous failures. The more a
         test fails previously, the higher it will be in order;
3      while dispatchQueue is not empty do
4          build = dispatchQueue.getNextBuild();
5          verdict = run(build.getNextBuild());
6          /* on pass, go to the next build        */
7          if verdict == failure then
8              /* focus: run all tests for the current
                  build                             */
9              runAllTests(build);
10         end
11     end
12 end
```

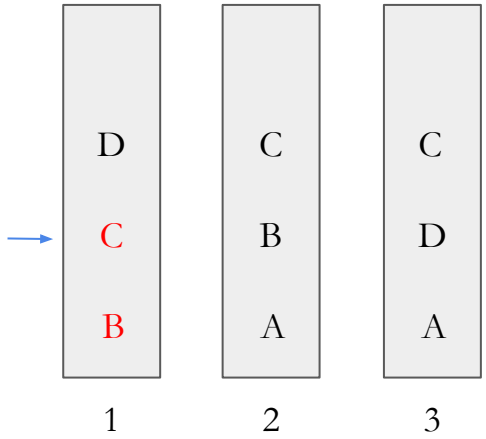
DynaQFocusFail Simulation



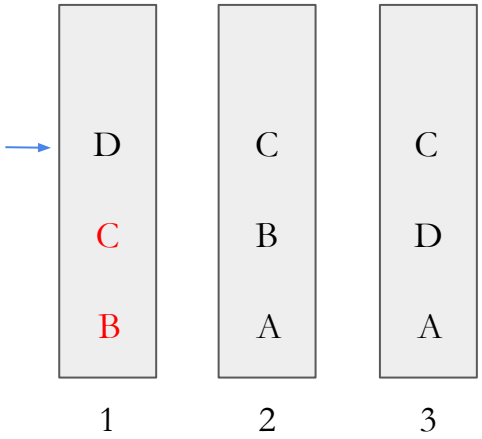
DynaQFocusFail Simulation



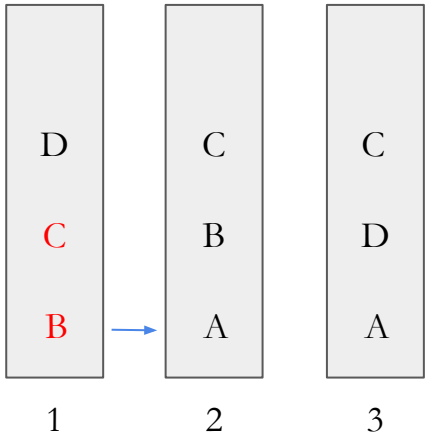
DynaQFocusFail Simulation



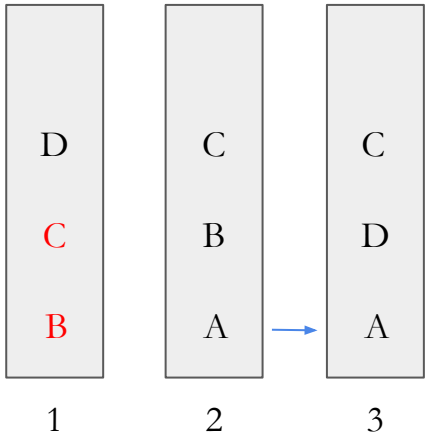
DynaQFocusFail Simulation



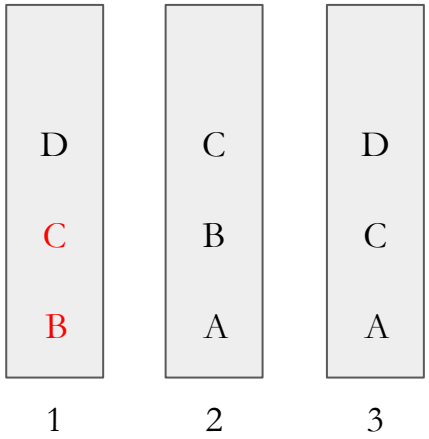
DynaQFocusFail Simulation



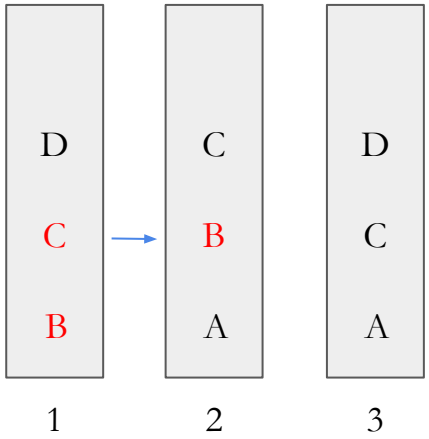
DynaQFocusFail Simulation



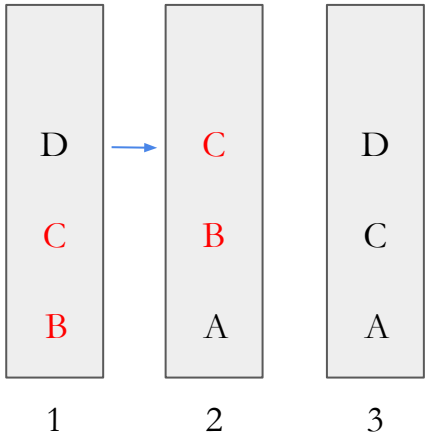
DynaQFocusFail Simulation



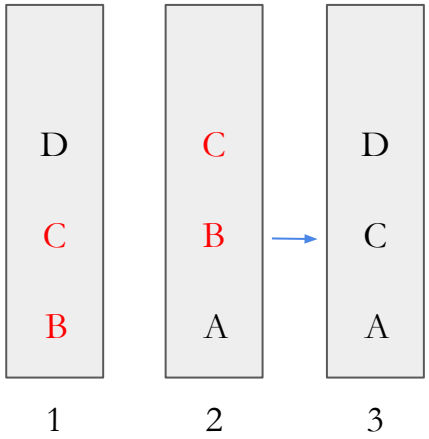
DynaQFocusFail Simulation



DynaQFocusFail Simulation



DynaQFocusFail Simulation



Datasets overviews

- Google
  - 3.5 million tests
  - 8,952 failing tests
    - Test failure ratio = 0.25%
  - Large builds
    - Consisting of up to 65,000 tests
- Chrome
  - 5.2 million tests
  - 810,514 failing tests
    - Test failure ratio = 15.4%
  - Small builds
    - Consisting of up to 139 tests

Evaluation Metric

- GainedRunOrder
- PercentageGain

GainedRunOrder

$$\textsc{GainedRunOrder}\ (\mathrm{A}) = \mathrm{RunOrderFail}(\mathit{FIFO}) - \mathrm{RunOrderFail}(\mathit{A})$$

PercentageGain

$$\text{PercentageGain}(A1, A2) = 1 - \textsc{GainedRunOrder}(A1) / \textsc{GainedRunOrder}(A2)$$

Experimental Results

Median GainedRunOrder Results

	Google	**Chrome**
GoogleTCP	8927	57
DynaQFocus	310	113
DynaQFocusFail	9407	221

PercentageGain Against GoogleTCP

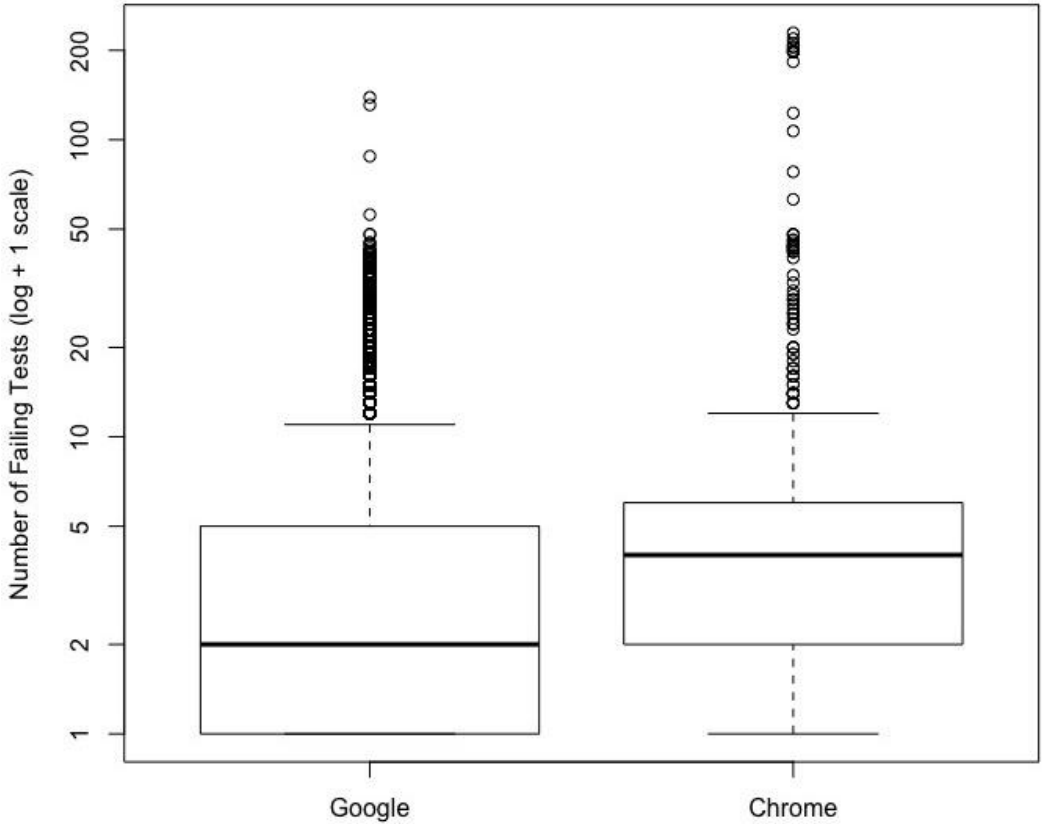
	Google	**Chrome**
DynaQFocus	-96.52%	98.24%
DynaQFocusFail	5.37%	287.71%

Discussion

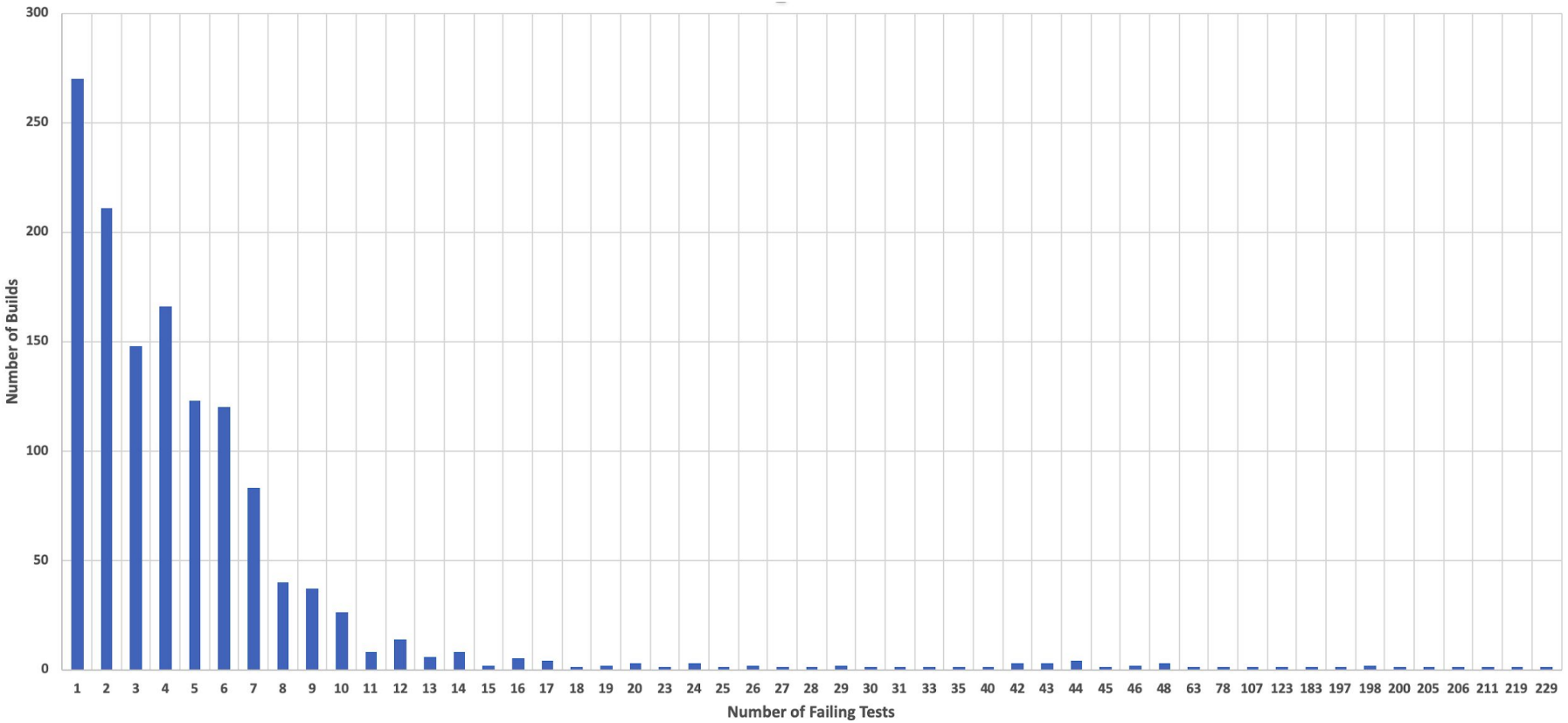
Build Level Failure Distribution

- Ratio of failures in each build
  - If the majority of tests are failures => prioritization does not help
  - If there are a few failures per build => focusing idea does not help
- How many builds we have for different number of test failures?

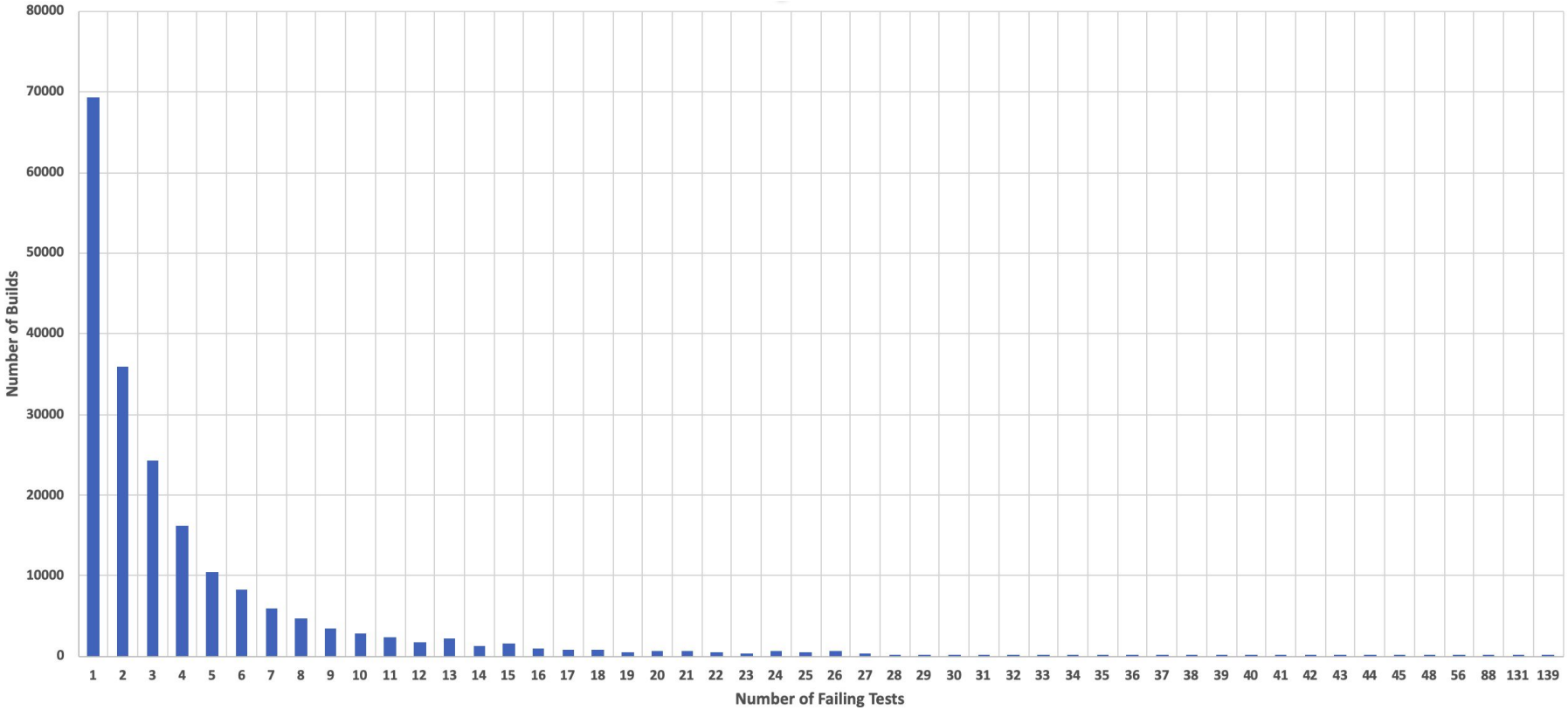
Build-level Failure Distribution



Build-level Failure Distribution of Google



Build-level Failure Distribution of Chrome



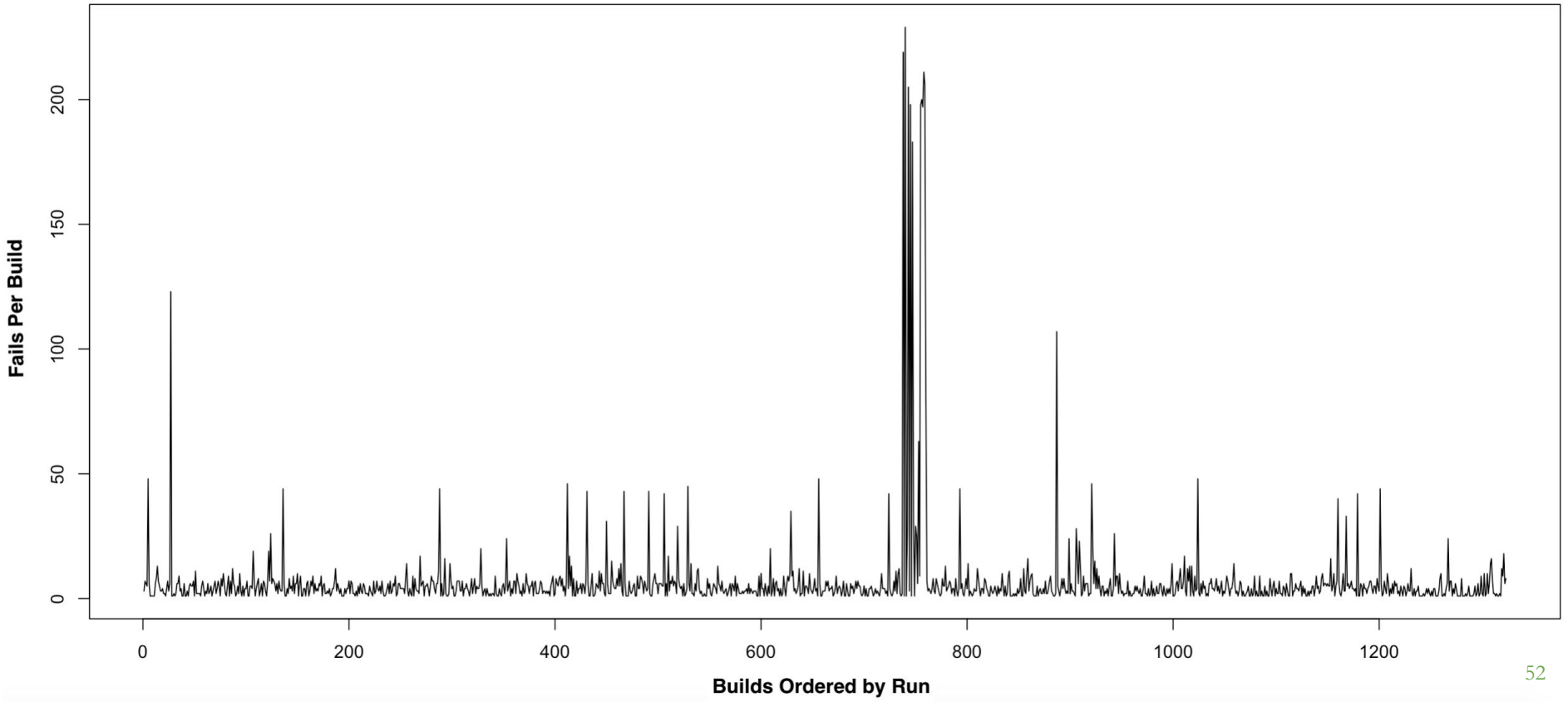
Conclusion

- We hypothesized that test failures cluster and the focusing idea might help
- DynaQFocusFail performs the best
- Results on Chrome are better than Google because of the failure ratio
- We have to consider failure distribution for the future designs

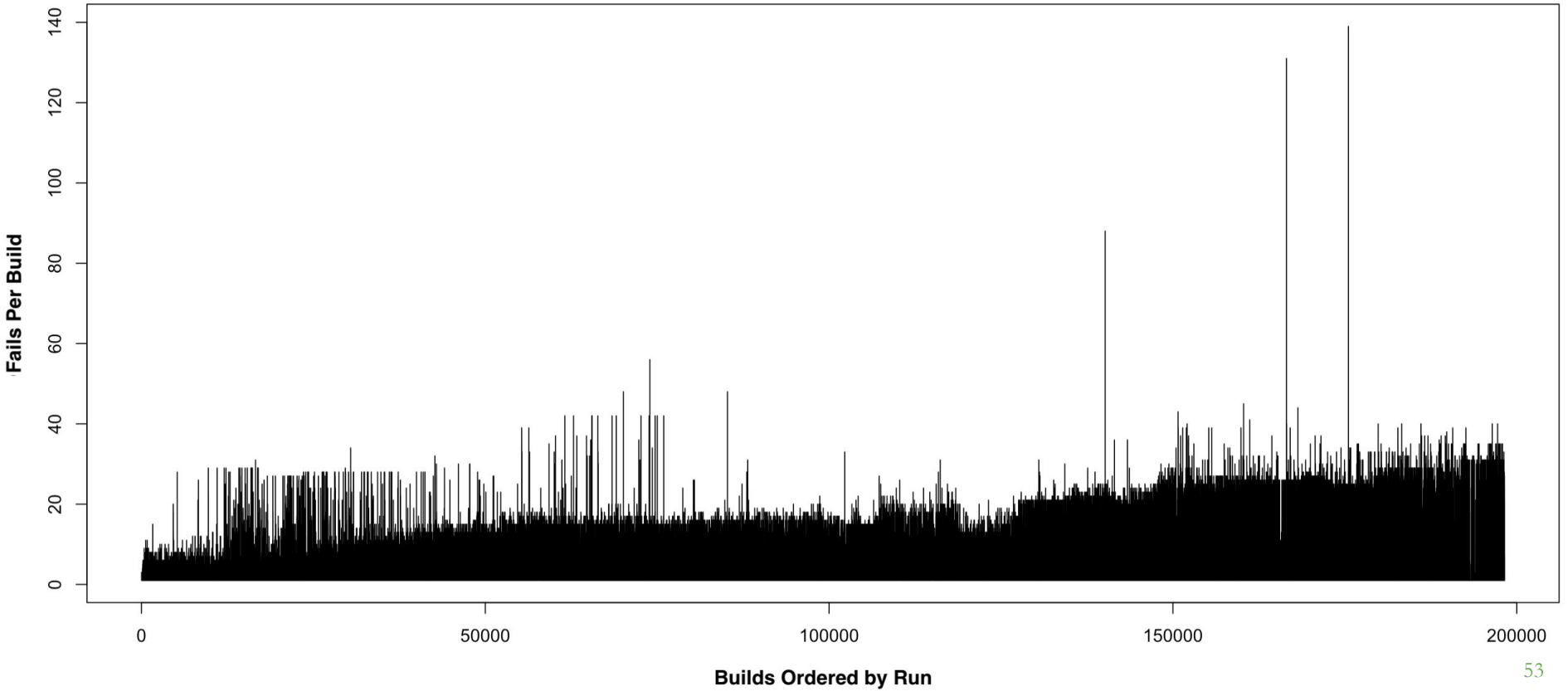
emad.fallahzadeh@concordia.ca

peter.rigby@concordia.ca

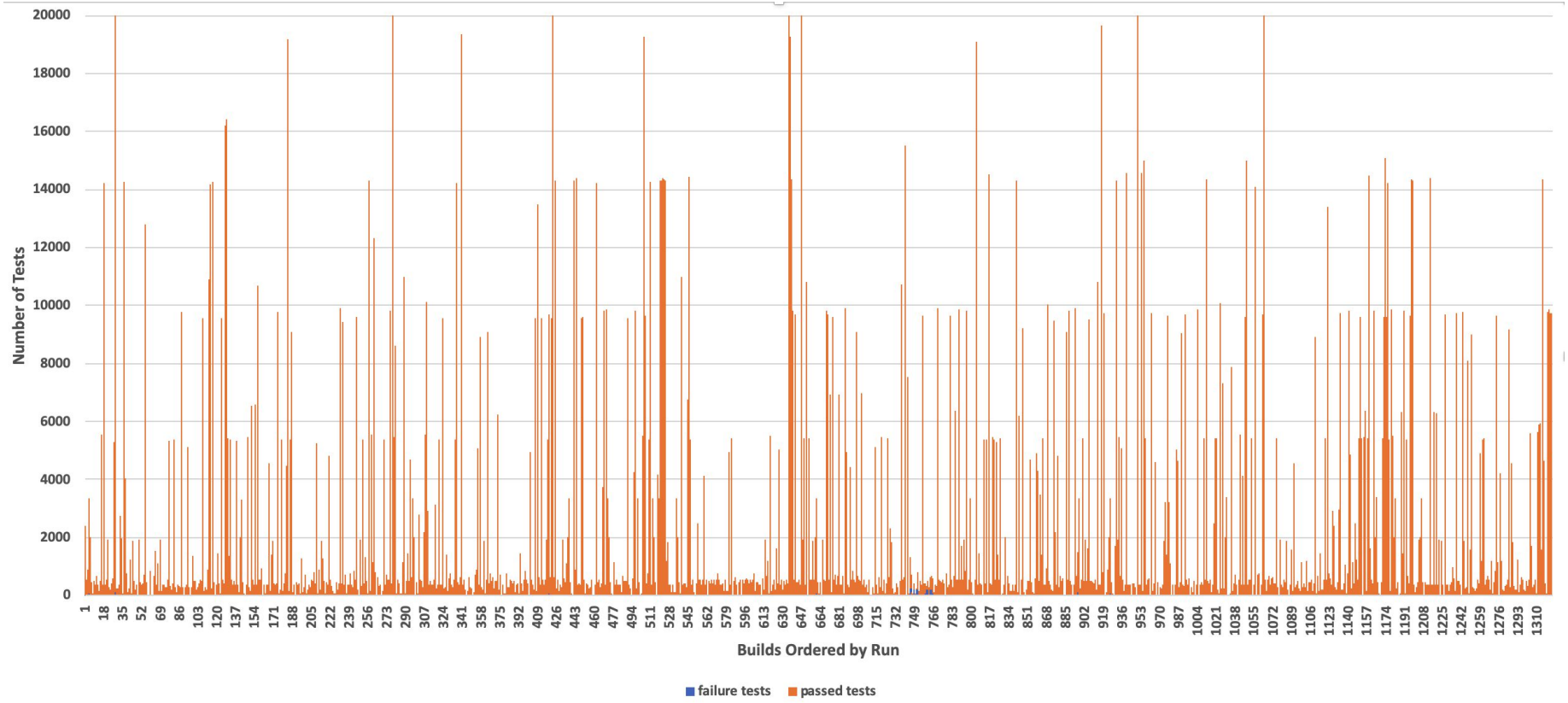
Google test failures time series



Chrome test failures time series



Google Dataset Failures to Passes



Chrome Dataset Failures to Passes