Test Selection Safety Evaluation Framework

Claire Leong
Goal: to make a generic framework for evaluating test scheduling algorithms at scale from the historical record.
Project Overview

- Implementation:
  1. Determine safety information for historical changelists
  2. Evaluate the safety of test selection algorithms
  3. Implement optimistic, pessimistic and random test selection algorithms
Project Overview

- Used over 2 datasets:

<table>
<thead>
<tr>
<th>Small Dataset</th>
<th>Large Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 days of CL data (6-8 Dec 2017)</td>
<td>1 month of CL data (October 2017)</td>
</tr>
<tr>
<td>11k changelists</td>
<td>900k changelists</td>
</tr>
<tr>
<td>1k total targets</td>
<td>4m total targets</td>
</tr>
<tr>
<td>430k times targets were affected</td>
<td>16b times targets were affected</td>
</tr>
</tbody>
</table>
Determining safety

- Safety = would skipping this test target miss a transition?
- Transition = a change in target results, either from **failing->passing** or **passing->failing**
### Safe Targets

*skipping this target would not miss a transition*

<table>
<thead>
<tr>
<th>Time</th>
<th>Changelist</th>
<th>Target Result</th>
<th>Safety</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CL1</td>
<td>P</td>
<td>-</td>
<td>P-&gt;P</td>
</tr>
<tr>
<td></td>
<td>CL2</td>
<td>P</td>
<td>Safe</td>
<td></td>
</tr>
</tbody>
</table>

* = affected
Safe Targets *skipping this target would not miss a transition*

<table>
<thead>
<tr>
<th>Time</th>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Result</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td></td>
<td>Safe</td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td></td>
<td>F-&gt;F</td>
</tr>
</tbody>
</table>

* = affected
### Safe Targets

*skipping this target would not miss a transition*

<table>
<thead>
<tr>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
<th>CL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Result</td>
<td>P</td>
<td>*</td>
<td>P</td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td>Safe</td>
<td>Safe</td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td>P-&gt;P</td>
<td>P-&gt;P</td>
</tr>
</tbody>
</table>

* = affected
**Safe Targets** *skipping this target would not miss a transition*

<table>
<thead>
<tr>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
<th>CL3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target Result</strong></td>
<td>F</td>
<td>*</td>
<td>F</td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td>Safe</td>
<td>Safe</td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td>F-&gt;F</td>
<td>F-&gt;F</td>
</tr>
</tbody>
</table>

* = affected
Unsafe Targets *skipping this target would definitely miss a transition*

<table>
<thead>
<tr>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Result</td>
<td>P</td>
<td>F</td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td>Unsafe</td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td>P-&gt;F</td>
</tr>
</tbody>
</table>

* = affected
## Unsafe Targets

*skipping this target would definitely miss a transition*

<table>
<thead>
<tr>
<th>Time</th>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Result</td>
<td>F</td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td></td>
<td>Unsafe</td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td></td>
<td>F-&gt;P</td>
</tr>
</tbody>
</table>
## Maybe Unsafe Targets

*skipping this target might miss a transition*

<table>
<thead>
<tr>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
<th>CL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Result</td>
<td>P</td>
<td>*</td>
<td>F</td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td>Maybe unsafe</td>
<td>Maybe unsafe</td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td>P-&gt;F</td>
<td>P-&gt;F</td>
</tr>
</tbody>
</table>

* = affected
Maybe Unsafe Targets *skipping this target might miss a transition*

<table>
<thead>
<tr>
<th>Time</th>
<th>Changelist</th>
<th>CL1</th>
<th>CL2</th>
<th>CL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Result</td>
<td>*</td>
<td>F</td>
<td>*</td>
<td>P</td>
</tr>
<tr>
<td>Safety</td>
<td>-</td>
<td>Maybe unsafe</td>
<td>Maybe unsafe</td>
<td></td>
</tr>
<tr>
<td>Transition</td>
<td>-</td>
<td>F-&gt;P</td>
<td>F-&gt;P</td>
<td></td>
</tr>
</tbody>
</table>

* = affected
Can we skip targets safely?

- This information is used to determine whether skipping a target is safe
- All non-definitive pass or fail results treated as affected
- We calculate the safety of skipping tests at rates from 0-100% for an algorithm
Input data

- Input taken from Spanner backup’s Result and Affected tables
- Used 3 methods to eliminate flakes from the data
  - Only take pass and fail results
  - Removing target results identified as flaky by Kellogs
  - Removing targets with over X transitions in the time period
Removing high transition count targets

Target Transition Count Histogram (for >10 transition targets)

Remove targets with > 30 transitions (~3k targets)
Targets per CL Distribution

Stats:
- Median 38 tests!
- 90th percentile 2,604
- 95th percentile 4,702
- 99th percentile 55,730
Implementation: Safety Data Builder

This package creates safety data given the historical changelist data as input.
Pipeline

- Generate Input Data & Filter Flakes
- Read Target Results to PTable
- Read Affected Targets to PTable
- Join Tables
- Build Target Table Stage
- Build Target Safeties Stage
- Build Safety Records Stage
- Write out Safety Records
Pipeline

Output: PTable<Target, Collection<ClAndResult>>

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;//target_name&quot;</td>
<td>&lt;(CL10, PASS), (CL2, FAIL), (CL42, NONE)&gt;</td>
</tr>
</tbody>
</table>

Generate Input Data & Filter Flakes

Read Target Results to PTable

Join Tables

Build Target Table Stage

Build Target Safeties Stage

Build Safety Records Stage

Write out Safety Records

Read Affected Targets to PTable
Generate Input Data & Filter Flakes

Read Target Results to PTable

Join Tables

Build Target Table Stage

Build Target Safeties Stage

Build Safety Records Stage

Write out Safety Records

Read Affected Targets to PTable

Output: PTable<CL, TargetSafety>

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL42</td>
<td>(&quot;target_name&quot;, SAFE, PP)</td>
</tr>
</tbody>
</table>
Pipeline

Generate Input Data & Filter Flakes

Read Target Results to PTable

Read Affected Targets to PTable

Join Tables

Build Target Table Stage

Build Target Safeties Stage

Build Safety Records Stage

Write out Safety Records

Output PCollection<SafetyRecord>

CL42,
safe_targets:<("target_name", SAFE, PP)>,
unsafe_targets:<>,
maybe_unsafe_targets:<>
## Safety Data Results

<table>
<thead>
<tr>
<th></th>
<th>Small Data Set</th>
<th>Large Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total CLs</strong></td>
<td>10170</td>
<td>891,621</td>
</tr>
<tr>
<td>CLs with only safe targets</td>
<td>96.4% (9801)</td>
<td>90.2% (804,160)</td>
</tr>
<tr>
<td>CLs with maybe unsafe</td>
<td>3.4% (346)</td>
<td>8.3% (73,897)</td>
</tr>
<tr>
<td>CLs with unsafe</td>
<td>0.2% (25)</td>
<td>1.5% (13,564)</td>
</tr>
<tr>
<td><strong>Total target affecteds</strong></td>
<td>428,938</td>
<td>15,931,019,923</td>
</tr>
<tr>
<td>Safe target affecteds</td>
<td>99.9% (428,547)</td>
<td>99.98% (15,927,853,638)</td>
</tr>
<tr>
<td>Maybe unsafe target affecteds</td>
<td>0.09% (365)</td>
<td>0.019% (3,054,667)</td>
</tr>
<tr>
<td>Unsafe target affecteds</td>
<td>0.01% (26)</td>
<td>0.0001% (111,618)</td>
</tr>
</tbody>
</table>
Culprit finding works!

With culprit finding:

We don’t do fix finding
Culprit finding works!

Unsafe Target Transitions

FP 37.1%
PF 62.9%

Maybe Unsafe Target Transitions

FP 61.4%
PF 38.6%
Implementation: Algorithm Evaluator

This package evaluates the safety of using an algorithm to select tests to skip for a changelist.
Evaluator Implementation

- For every changelist in the safety data, it will call an algorithm with skip rates from 0 to 100%
- Using the targets returned by the algorithm, determines if that selection was safe or not
  - Safe = no unsafe or maybe unsafe tests were skipped
  - Maybe unsafe = maybe unsafe tests were skipped but no unsafe tests
  - Unsafe = unsafe tests were skipped
Algorithms

- Algorithms are implementations of the interface TestSelectionAlgorithm which contains the method

```java
ImmutableSet<Target> skipTargets(long cl, Iterable<Target> targets, int numToSkip)
```
Algorithms - Random

Changelist’s Affected Targets

1 Safe target
1 Maybe unsafe target
1 Unsafe target

Num to skip = 0
Random Algorithm

Safety = safe
Algorithms - **Random**

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

**Random Algorithm**

Num to skip = 1

Safety = unsafe
Algorithms - **Random**

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

Random Algorithm:

- Num to skip = 2

Safety = maybe unsafe
Algorithms - Random

Changelist’s Affected Targets

1 Safe target
1 Maybe unsafe target
1 Unsafe target

Num to skip = 3
Random Algorithm

Safety = unsafe
Algorithms - Optimistic

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

Num to skip = 0

Optimistic Algorithm

Safety = safe
Algorithms - Optimistic

Changelist’s Affected Targets

1 Safe target
1 Maybe unsafe target
1 Unsafe target

Num to skip = 1
Optimistic Algorithm

Safety = safe
Algorithms - Optimistic

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

Optimistic Algorithm

Num to skip = 2

Safety = maybe unsafe
Algorithms - Optimistic

Changelist’s Affected Targets

1 Safe target
1 Maybe unsafe target
1 Unsafe target

Optimistic Algorithm

Num to skip = 3

Safety = unsafe
Algorithms - **Pessimistic**

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

Num to skip = 0

Pessimistic Algorithm

Safety = safe
Algorithms - Pessimistic

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

Num to skip = 1

Pessimistic Algorithm

Safety = unsafe
**Algorithms - Pessimistic**

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

- Num to skip = 2

Safety = unsafe
**Algorithms - Pessimistic**

Changelist’s Affected Targets

- 1 Safe target
- 1 Maybe unsafe target
- 1 Unsafe target

Num to skip = 3

Pessimistic Algorithm

Safety = unsafe
Pipeline performance

- Safety data builder ran in 35 mins
- Algorithm evaluator
  - Optimistic ran in 2h 40m
  - Pessimistic ran in 3h 5m
  - Random ran in 4h 40m
Small dataset results

Safe Changelists

% of total CLs vs Skip Rate (%)

- Random
- Optimistic
- Pessimistic

floor of changelists with only safe affected targets
Small dataset results

Maybe Unsafe Changelists

ceiling of changelists with maybe unsafe but no unsafe affected targets

% of total CLs vs. Skip Rate (%)

Random
Optimistic
Pessimistic
Small dataset results

Unsafe Changelists

ceiling of changelists with unsafe affected targets

% of total CLs vs. Skip Rate (%)

- Random
- Optimistic
- Pessimistic
Large dataset results

Safe Changelists

- Random
- Optimistic
- Pessimistic

% of total CLs

floor of changelists with only safe affected targets

Skip Rate (%)
Why is random a curve?

- Previously we had predicted a straight line for random
- Small data set has a straight line
Probability Distribution

\[ \binom{n}{k} = \text{number of ways to select } k \text{ items from } n \text{ total items} \]

\[ P(\text{select } k \text{ only safe targets}) = \frac{\text{number of ways to select } k \text{ safe targets}}{\text{number of ways to select any } k \text{ targets}} \]

\[ = \frac{\binom{n}{Np}}{\binom{N}{Np}} \]

Where

- \( n \) = number of safe affected targets
- \( N \) = total number of affected targets
- \( p \) = % of targets being selected
Probability Distribution where \( N = 1000, n = 995 \)
Random Algorithm Safe Changelists

N = number of affected targets for a CL
Large dataset results

Maybe Unsafe Changelists

ceiling of changelists with maybe unsafe but no unsafe affected targets
Large dataset results

Unsafe Changelists

ceiling of changelists with unsafe affected targets
Conclusions

● The project was completed!
● We now have an offline method to evaluate test scheduling algorithms and a baseline for future comparison
Continuing the project

● Better flake exclusion
  ○ Filter using ratio transitions:results
  ○ Find the point where Kellogs doesn’t identify the target as flaky

● Rerunning Elbaum experiments
  ○ An algorithm which prioritizes targets based on the number of transitions in some previous window of time

● Evaluating Efficacy machine learning model
Questions?
Creating safeties

for all targets {
    for (result in sorted target results) {
        if (result = affected) {
            add result to pending results
            continue;
        }

        if (previous result = result) {
            mark this result and all pending results as safe
        } else (if no pending results) {
            mark this result as unsafe
        } else {
            mark this result and all pending results as maybe safe
        }

        previous result = result;
    }
}
Evaluating algorithms

for (changelist) {
    retrieve affected targets for changelist
    for (skip rate = 0..100%) {
        skipped targets = algorithm.skipTargets(affected targets, skip rate * num affected targets);

        if (skipped targets contain unsafer targets) {
            mark this test selection as unsafe
        }
        else if (skipped targets contains maybe unsafe targets) {
            mark this test selection as maybe unsafe
        }
        else {
            mark this test selection as safe
        }
    }
}
safety_record.proto

// Represents the safety information for all affected targets at a CL.
message SafetyRecord {
  optional int64 changelist = 1;
  repeated TargetSafety safe_targets = 2;
  repeated TargetSafety unsafe_targets = 3;
  repeated TargetSafety maybe_unsafe_targets = 4;
}

// Represents the safety of skipping a test target.
message TargetSafety {
  optional string target_name = 1;
  optional Safety safety = 2;
  optional Transition transition = 3;
}
// Represents the safety information for using a test selection algorithm on a
// CL's test targets with a given target skip rate.
message SafetyResult {
  optional int64 changelist = 1;
  optional string algorithm_name = 2;
  optional int32 skip_rate = 3;

  optional tko.testselectionevaluation.TargetSafety.Safety safety = 4;

  repeated string unsafe_skipped = 5;
  repeated string maybe_unsafe_skipped = 6;

  optional int32 total_skipped = 7;
}
Optimistic Algorithm Implementation

// Constructor takes a SafetyRecord as input
private final SafetyRecord record;

skipTargets(changelist, targetList, numToSkip) {
    remainingToSkip = numToSkip;

    skip targets from record.safe_targets with limit remainingToSkip
    remainingToSkip -= safe skipped targets

    skip targets from record.maybe_safe_targets with limit remainingToSkip
    remainingToSkip -= maybe unsafe skipped targets

    skip targets from record.unsafe_targets with limit remainingToSkip
    remainingToSkip -= unsafe skipped targets

    return skipped targets
}