Efficient Dependency Detection for Safe Java Test Acceleration

Jonathan Bell, Gail Kaiser, Eric Melski and Mohan Dattatreya
Columbia University & Electric Cloud, Inc
Simplified Software Lifecycle

Make changes to code

Build & test

Commit

How long is too long of a build?

1 day? 6 hours? 10 minutes?
Simplified Software Lifecycle

- Compile sources
- Generate documentation
- Run tests
- Package

Make changes to code ➔ Build & test ➔ Commit
Testing Dominates Build Times

- Testing: 41%
- Compiling: 38%
- Other: 20%

351 projects from GitHub
Testing Dominates Build Times

- Testing: 60%
- Compiling: 26%
- Other: 14%

Projects taking > 10 minutes to build (69)
Testing Dominates Build Times

- Testing: 90%
- Compiling: 8%
- Other: 2%

Projects taking > 1 hour to build (8)
Faster tests = Faster builds
Test acceleration is well studied
Regression Test Selection

Gligoric et al. [ISSTA ’15], Orso et al. [FSE ’04], Harrold et al. [OOPSLA ’01]
Regression Test Selection

Gligoric et al. [ISSTA ’15], Orso et al. [FSE ’04], Harrold et al. [OOPSLA ’01]

Tests not relevant to changeset: skipped
Test Suite Minimization

Hao et al. [ICSE ’12]; Orso et al. [ICSE ’09]; Jeffrey et al. [TSE ’07]; Tallam et al. [PASTE ’05]; Jones et al. [TOSEM ’03]; Harrold et al. [TOSEM ’93]; Chen et al. [IST ’98]; Wong et al. [ICSE ’95] and more
Test Suite Minimization

Hao et al. [ICSE ’12]; Orso et al. [ICSE ’09]; Jeffrey et al. [TSE ’07]; Tallam et al. [PASTE ’05]; Jones et al. [TOSEM ’03]; Harrold et al. [TOSEM ’93]; Chen et al. [IST ’98]; Wong et al. [ICSE ’95] and more
Test Parallelization
Test Parallelization

Test 1
Test 2
Test 3
Test 4

Test 5
Test 6
Test 7
Test 8
Test 9

Test 10
Test 11
Test 12
Test 13
Test 14
Controlled Regression Testing Assumption

Tests

Code
Controlled Regression Testing Assumption

External Factors

Tests

Code

External Factors

External Factors

External Factors
Controlled Regression Testing Assumption

- External Factors
- External Factors
- External Factors
- Test 1
- Code
- External Factors
- External Factors
- External Factors
Controlled Regression Testing Assumption

External Factors

Test 2

Code

External Factors

External Factors

External Factors
Controlled Regression Testing Assumption

External Factors

Test 3

Code

External Factors

External Factors

External Factors
Not sound in practice
Test Dependencies

Test 1
Write, Value "A"

Test 2
Read

Test 3
Read

Test 4
Write, Value "B"

Shared File

Value: B

ESEC/FSE
@_jon_bell_
September 4, 2015
Test Dependencies

Test 1
Test 4
Test 2
Test 3

Shared File
Test Dependencies

- **Test 1**: Write, Value “A”
- **Test 4**: Write, Value “B”
- **Test 2**: Read, Expect Value “A”
- **Test 3**: Read

Shared File

Value: B
Test Dependencies

A manifest test dependency

Test 1
Write, Value “A”

Test 2
Write, Value “B”

Test 3
Read, Expect Value “A”

Shared File
Value: B
Test Dependencies: A Clear and Present Danger

- Really exist in practice (Zhang et al. found 96, Luo et al. found 14)
- Hard to specify - if we could specify, would be safe to accelerate
- Rarely: isolated (especially not in long building projects)
- Existing technique to detect: combinatorially run tests [Zhang, et al ’14]
Brute Force Dependency Detection

• Looked at feasibility on 10 large open source test suites

• Exhaustive approach: $> 10^{300}$ years to find all dependencies

• Pairwise approach: Average 31,882 executions of the entire test suite to find (incomplete) dependencies

• Problem: How do we safely accelerate test suites in the presence of unknown dependencies?
Manifest Test Dependencies

• Definition: a data dependence between tests T1, T2 that results in the outcome of T2 changing

• All manifest dependencies are data dependencies

• Not all data dependencies are manifest dependencies
Data Dependencies

Present Dependencies:
Test 1 must run before 2 and 3
Test 4 must run after 2 and 3

Test 1
Write, Value “A”

Test 2
Read

Test 3
Read

Test 4
Write, Value “B”

Shared File
Key Insight: Dependencies don’t need to be precise, but must be sound.
Intuition

Test 1 → Test 2 → Test 3 → Test 4 → Test 5 → Test 6 → Test 7

Test 8 → Test 9 → Test 10 → Test 11 → Test 12 → Test 13 → Test 14 → Test 15

Idle extra capacity
Intuition

A lot of dependencies, but still a 2x speedup

Idle extra capacity
ElectricTest - Detecting Data Dependencies in Java

- Tracks in-memory dependencies (JVMTI plugin)
- Tracks file and network dependencies (IO-Trace agent)
- Implemented entirely within the Oracle or OpenJDK JVM, no specialized drivers, etc required
- Captures stack traces when dependencies occur to support debugging
- Generates dependency trees to enable sound test acceleration
Identifying Heap Dependencies

After each test, garbage collect; traverse heap to map objects back to static fields.

Class A

End of test 1
Identifying Heap Dependencies

After each test, garbage collect; traverse heap to map objects back to static fields.

Class A

End of test 1
Identifying Heap Dependencies

After each test, garbage collect; traverse heap to map objects back to static fields.
Identifying Heap Dependencies

During test execution, monitor accesses to existing objects
Identifying Heap Dependencies

During test execution, monitor accesses to existing objects

Diagram:
- Class A
- Static fields: W1, W2
- During Test 2:
  - W2 writing to W1
  - W1 reading from W2

ESEC/FSE
@jon_bell_
Identifying Heap Dependencies

During test execution, monitor accesses to existing objects

During Test 2

Dependency!
Identifying External Dependencies

Application under test

Network

Log remote host address

Log path

Filesystem
ElectricTest enables sound exploitation of existing test acceleration techniques.
Safe Test Parallelization

Test 1 → Test 2 → Test 3 → Test 4 → Test 5 → Test 6 → Test 7

Test 8 → Test 9 → Test 10 → Test 11 → Test 12 → Test 13 → Test 14

Test 15
Safe Test Parallelization

Test 1 → Test 2 → Test 3 → Test 8 → Test 9 → Test 10 → Test 15

Test 4 → Test 5 → Test 6 → Test 7

Test 14 → Test 11 → Test 12 → Test 13

ESEC/FSE
Safe Test Selection

Single test selected to be executed
Safe Test Selection

Single test selected to be executed with its dependencies
Understanding Dependencies

• What should a developer do about test dependencies?

• Might be intentional (e.g. cache shared state)

• Might be unintentional but OK (e.g. loggers)

• Might be unintentional and bad (e.g. bug)
Assisting Debugging

Test 3 → Depends on → Test 1

Debugging information reported by the previous technique
Assisting Debugging

Value that is read

Test that wrote value

Stack trace shows use
Evaluation

• RQ1: Recall (accuracy)
• RQ2: Runtime overhead
• RQ3: Impact on acceleration
# RQ1: Recall

<table>
<thead>
<tr>
<th>Project</th>
<th>Dependencies Detected</th>
<th>Ground Truth</th>
<th>ElectricTest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Writers</td>
<td>Readers</td>
</tr>
<tr>
<td>Joda</td>
<td>2</td>
<td>15</td>
<td>121</td>
</tr>
<tr>
<td>XMLSecurity</td>
<td>4</td>
<td>3</td>
<td>103</td>
</tr>
<tr>
<td>Crystal</td>
<td>18</td>
<td>15</td>
<td>39</td>
</tr>
<tr>
<td>Synoptic</td>
<td>1</td>
<td>10</td>
<td>117</td>
</tr>
</tbody>
</table>
## RQ1: Recall

<table>
<thead>
<tr>
<th>Project</th>
<th>Dependencies Detected</th>
<th>ElectricTest Shared Resource Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ground Truth</td>
<td>ElectricTest Writers</td>
</tr>
<tr>
<td>Joda</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>XMLSecurity</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Crystal</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>Synoptic</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>
RQ2: Overhead

• Selected 10 projects with > 10 minutes of tests

• Also included projects studied by Zhang et al, averaging < 10 seconds of testing

• Previous exhaustive approach slowdown: >10^{300}X

• Previous heuristic approach slowdown: 31,882X

• ElectricTest slowdown: 36X (885X faster than previous approach)
RQ2: Overhead

On average, ElectricTest is 885X faster than running all tests pairwise.

Slowdown relative to a single test suite execution (lower is better)
RQ2: Overhead

A lot of fast running tests: Runtime dominated by pauses between tests (gc)

Average 36X

Slowdown relative to a single test suite execution (lower is better)
RQ3: Impact on Acceleration

Average (Unsafe) 19x

Average (Safe) 7x
Testing Dominates Build Times

- Compiling: 8%
- Other: 2%
- Testing: 90%

Projects taking > 1 hour to build (8)

Test Dependencies

- Test 1
- Test 4
- Test 2
- Test 3

- Write, Value “B”
- Write, Value “A”
- Read, Expect Value “A”
- Value: A

Intuition

- A lot of dependencies, but still a 2x speedup

Efficiently Identifying Dependencies

- During test execution, monitor accesses to existing objects

- Class A
- Static field
- Static field
- Static field

- During Test 2
- Write! W2
- W1
- Read!
- W1
- Write!
- W2
- W1
- W1
- W1
- W1
- W1
Efficient Dependency Detection for Safe Java Test Acceleration

Jonathan Bell, Gail Kaiser, Eric Melski and Mohan Dattatreya
Columbia University & Electric Cloud, Inc