Coverage-Based Reduction of Test Execution Time: Lessons from a Very Large Industrial Project

Thomas Bach, Artur Andrzejak, Ralf Pannemans

Heidelberg University
http://pvs.ifi.uni-heidelberg.de

SAP SE
http://www.sap.de

Content

- Academic-industry collaboration details
- Test environment
- Challenges and gaps between research and practice
- Our results from coverage analysis

Collaboration Details

- Started in 2012
- Recurring student activities (> 10 theses, internships)
- PhD project: Testing in Very Large Software Projects
  - PhD student at Heidelberg University and SAP
- Success factors:
  - Good combination: Practical relevant & nontrivial research
  - Real, large scale software product as a use case
- Challenges:
  - Transfer research to production
  - Find interested persons in charge

Test Environment

- SAP HANA
  - In-memory database management system
  - Core product platform of SAP
  - Several million LOC C/C++, scales up to >600 cores
- Testing
  - More than 1000 test suites with more than 100 000 tests
  - Coverage is line based per test suite
  - Test framework in python
    - Test sends SQL to HANA and checks results

GAPS BETWEEN RESEARCH AND PRACTICE

Project goals and discovered gaps

- We want to
  - Reduce test runtime
  - Increase specificity of coverage based test characterization
- We encountered several issues with existing work

Evaluation with Small Projects

- Practitioners do not trust small evaluations

Work[1]	**Size**
Alspaugh et al. 2007	5 classes to 22 classes
Zhang et al. 2009	53 testcases to 209 testcases
Li et al. 2009	374 LOC to 11 kLOC
You et al. 2011	500 LOC to 10 kLOC
Zhang et al. 2013	2 kLOC to 80 kLOC
Do et al. 2008	7 kLOC to 80 kLOC
Elbaum et al. 2002	8 kLOC to 300 kLOC
Our work	> 3.50 MLOC

Related work comparing overlap-aware vs. non-overlap-aware solvers for TCS or TCP

[1] See paper for details

Flaky Tests

- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: Failed
- Execute test 1: OK

Investigate?

Ignore?

Test infrastructure?

Hardware Problems?

Memory leak?

Test dependencies?

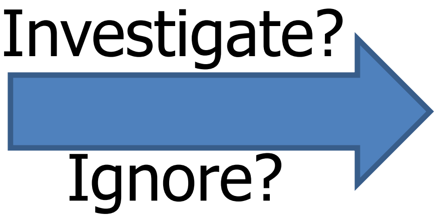
Real bug? (e.g. concurrency)

Performance?

and more …

Flaky Tests

- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: Failed
- Execute test 1: OK



Investigate?

Ignore?

Test infrastructure?

Hardware Problems?

Memory leak?

Test dependencies?

Real bug? (e.g. concurrency)

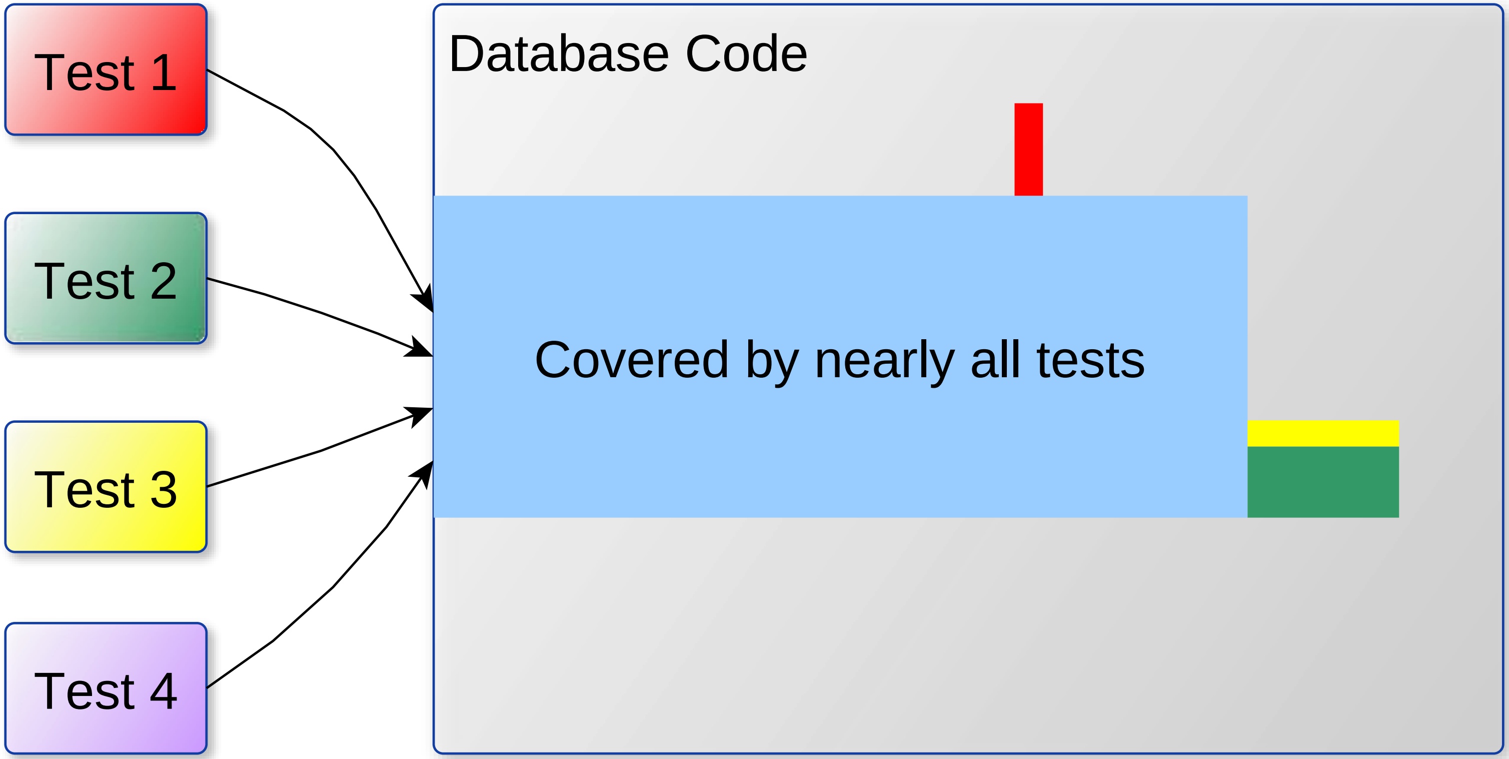
Performance?

and more …

Flaky test detection and handling is time consuming

Real world is not perfect and return of investment avoids perfection

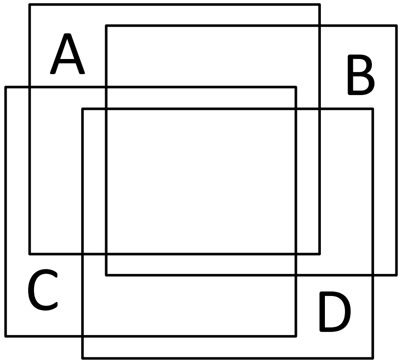
Shared coverage



Large part of coverage is not specific

Random Coverage

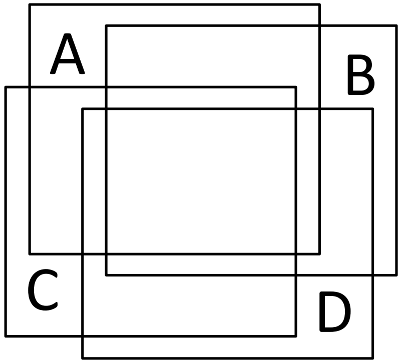
- Coverage A: 651 074 lines hit
- Coverage B: 651 845 lines hit
- Coverage C: 651 862 lines hit
- Coverage D: 652 015 lines hit



Venn diagram

Random Coverage

- Coverage A: 651 074 lines hit
- Coverage B: 651 845 lines hit
- Coverage C: 651 862 lines hit
- Coverage D: 652 015 lines hit

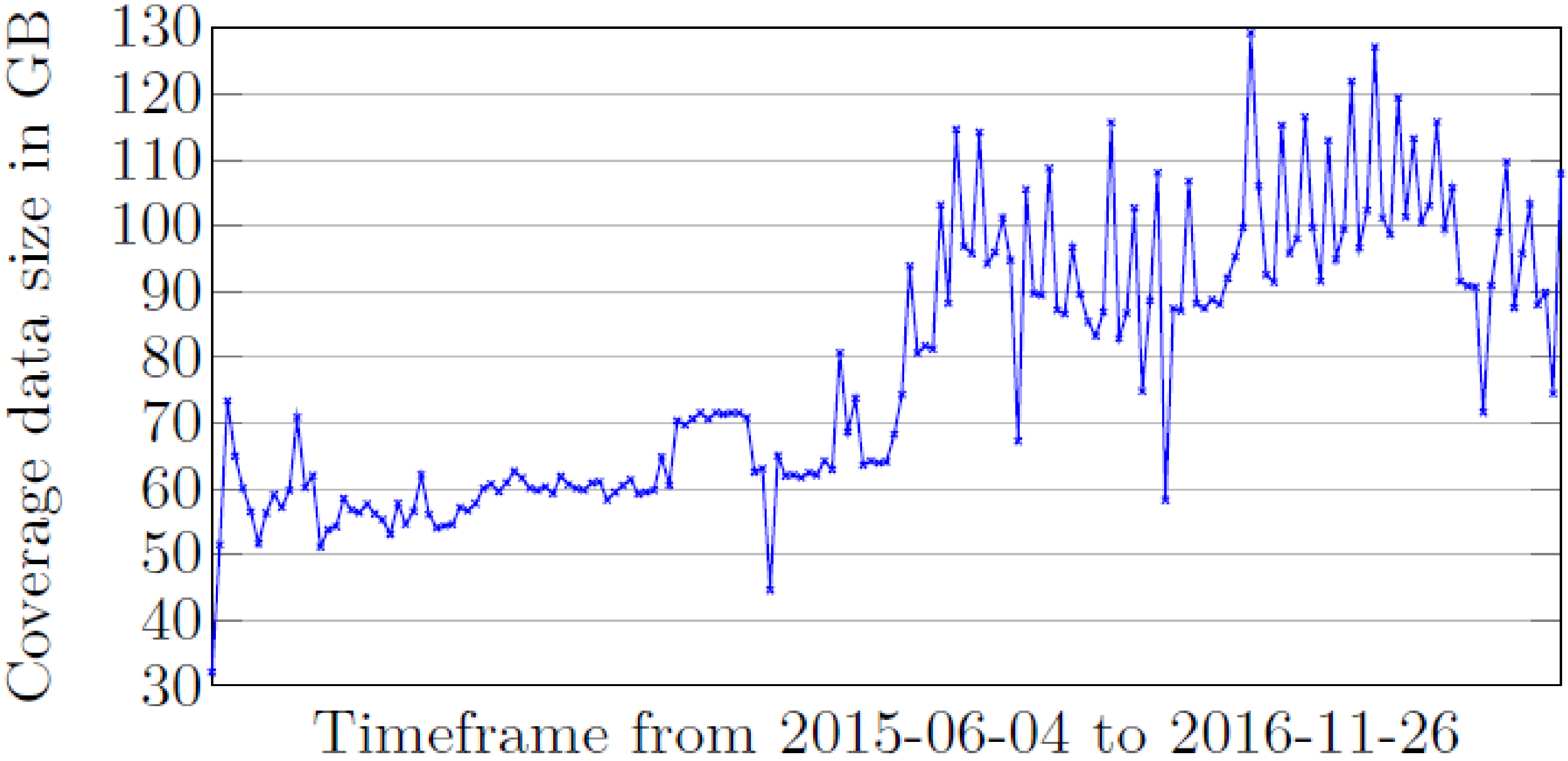


Venn diagram

Impossible to find exactly identical or included tests

In Fact:
A and B from same Test1
C and D from same Test2
Test2 contains Test1 + more

Size of Coverage Data



Size is nontrivial and increasing

OUR RESULTS ON COVERAGE ANALYSIS

Overlap-Aware Coverage Algorithms

- Test Case Selection
  - Time budget 1h: Which tests to run?
    - Objective: coverage – Maximum budgeted cov. problem
  - Which tests to run for full coverage?
    - Objective: cardinality – Set cover problem
    - Objective: runtime – Weighted set cover problem
- Test Case Prioritization
  - Which tests to run first? Objective: coverage (per time)

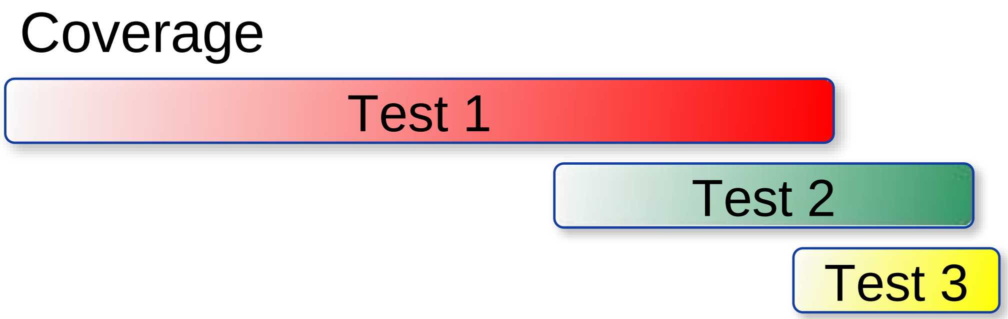
Unsafe algorithms,
we could miss functionality

Overlap-Aware Coverage Algorithms

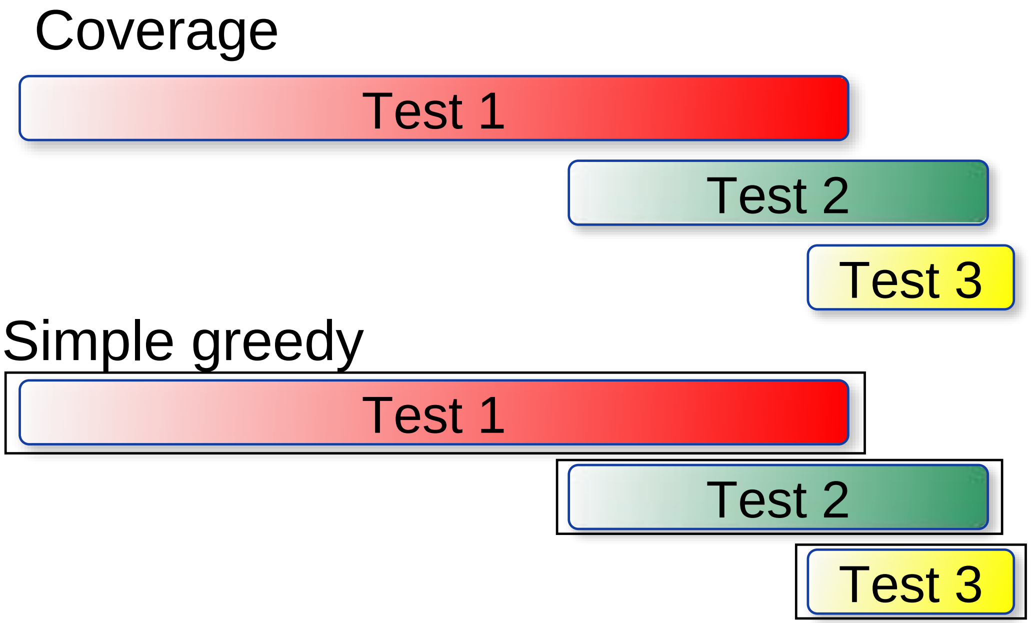
- Test Case Selection
  - Time budget 1h: Which tests to run?
    - Objective: coverage – Maximum budgeted cov. problem
  - Which tests to run for full coverage?
    - Objective: cardinality – Set cover problem
    - Objective: runtime – Weighted set cover problem
- Test Case Prioritization
  - Which tests to run first? Objective: coverage (per time)

Unsafe algorithms,
we could miss functionality

Overlap-Aware vs. Simple Greedy



Overlap-Aware vs. Simple Greedy



Coverage

Test 1

Test 2

Test 3

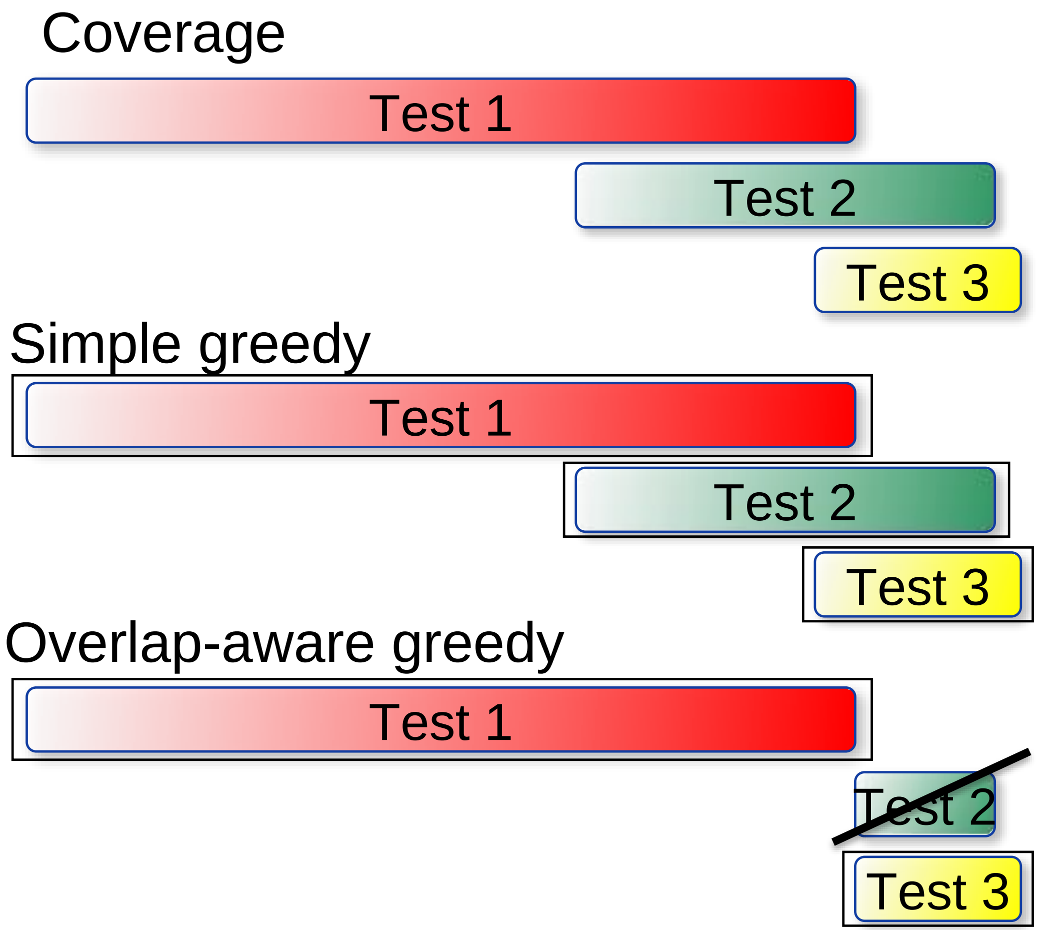
Simple greedy

Test 1

Test 2

Test 3

Overlap-Aware vs. Simple Greedy



Coverage

Test 1

Test 2

Test 3

Simple greedy

Test 1

Test 2

Test 3

Overlap-aware greedy

Test 1

~~Test 2~~

Test 3

Comparison Overlap-Aware

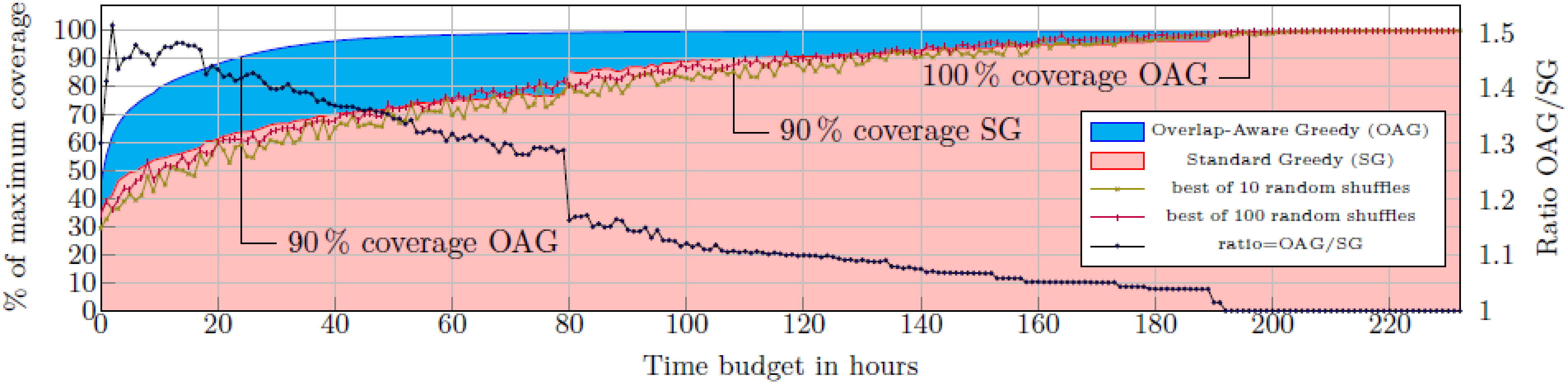
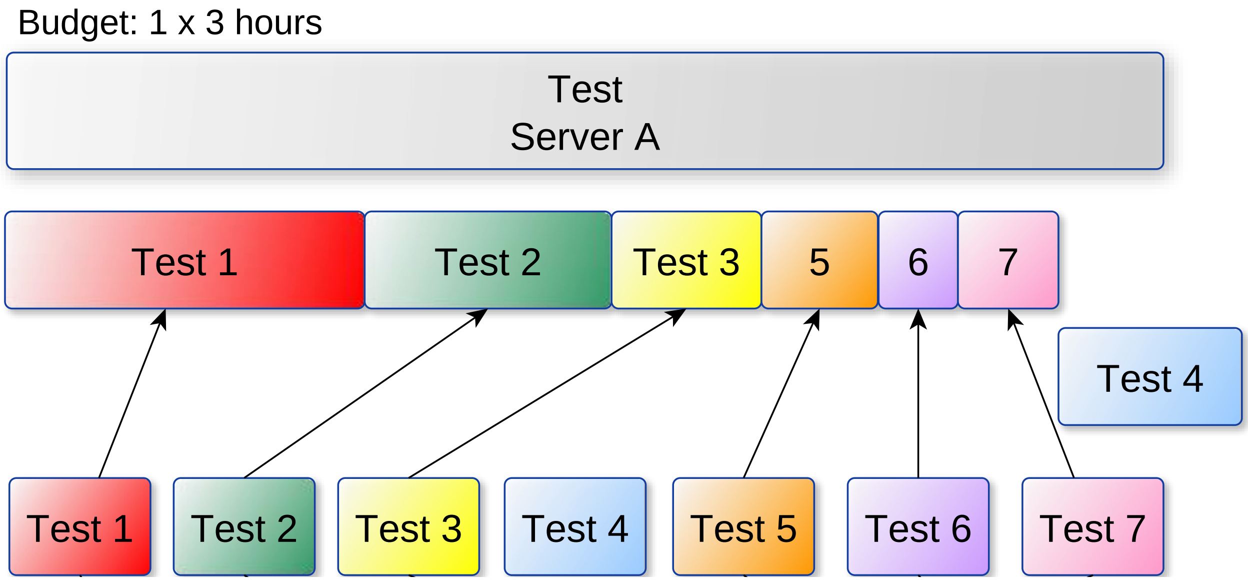


Figure 2. Exemplary comparison between different algorithms for maximum budgeted coverage problem. Higher is better

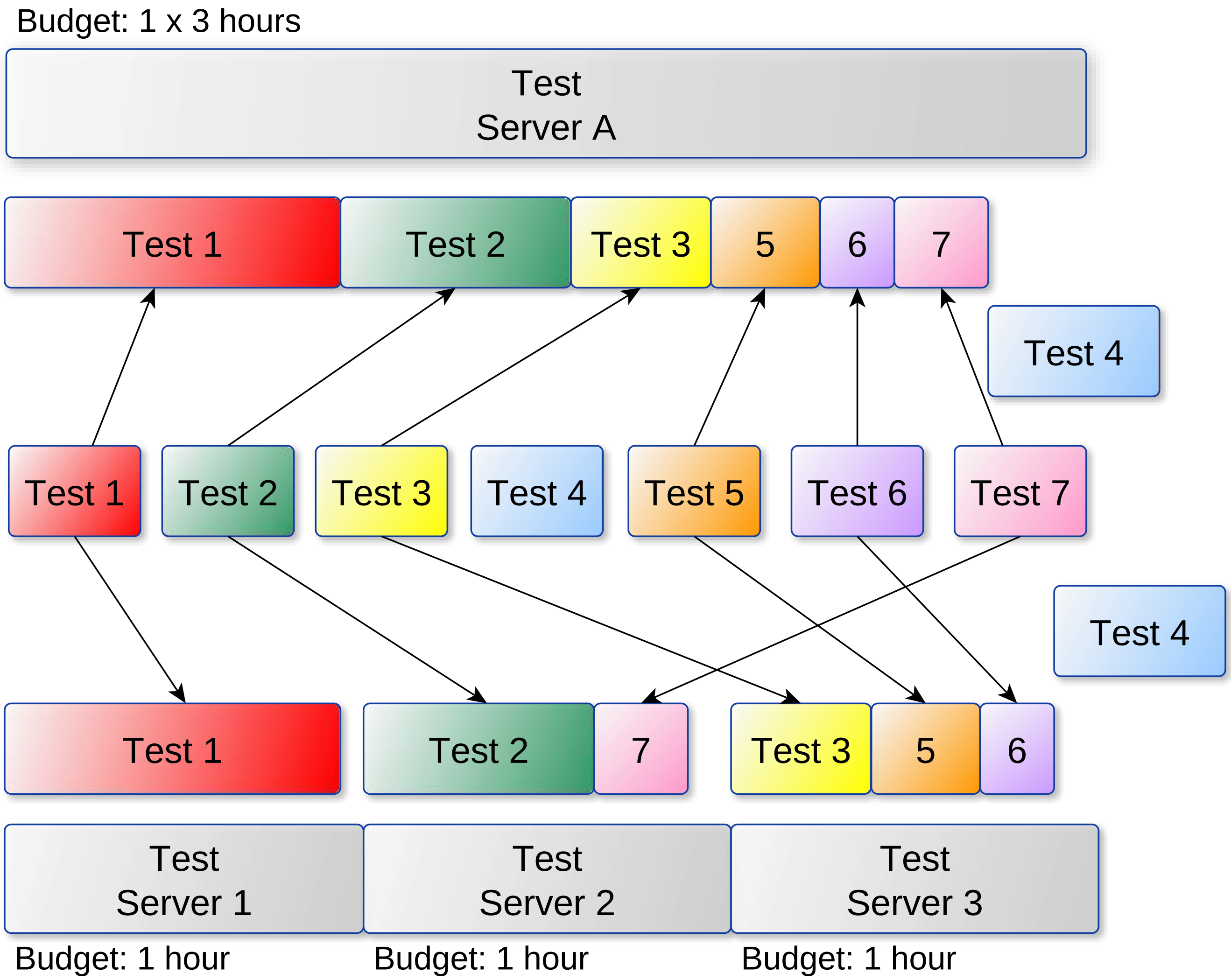
Overlap-aware greedy
reaches more coverage faster

Runtime for single run: <10s
Also works for test clusters with buckets

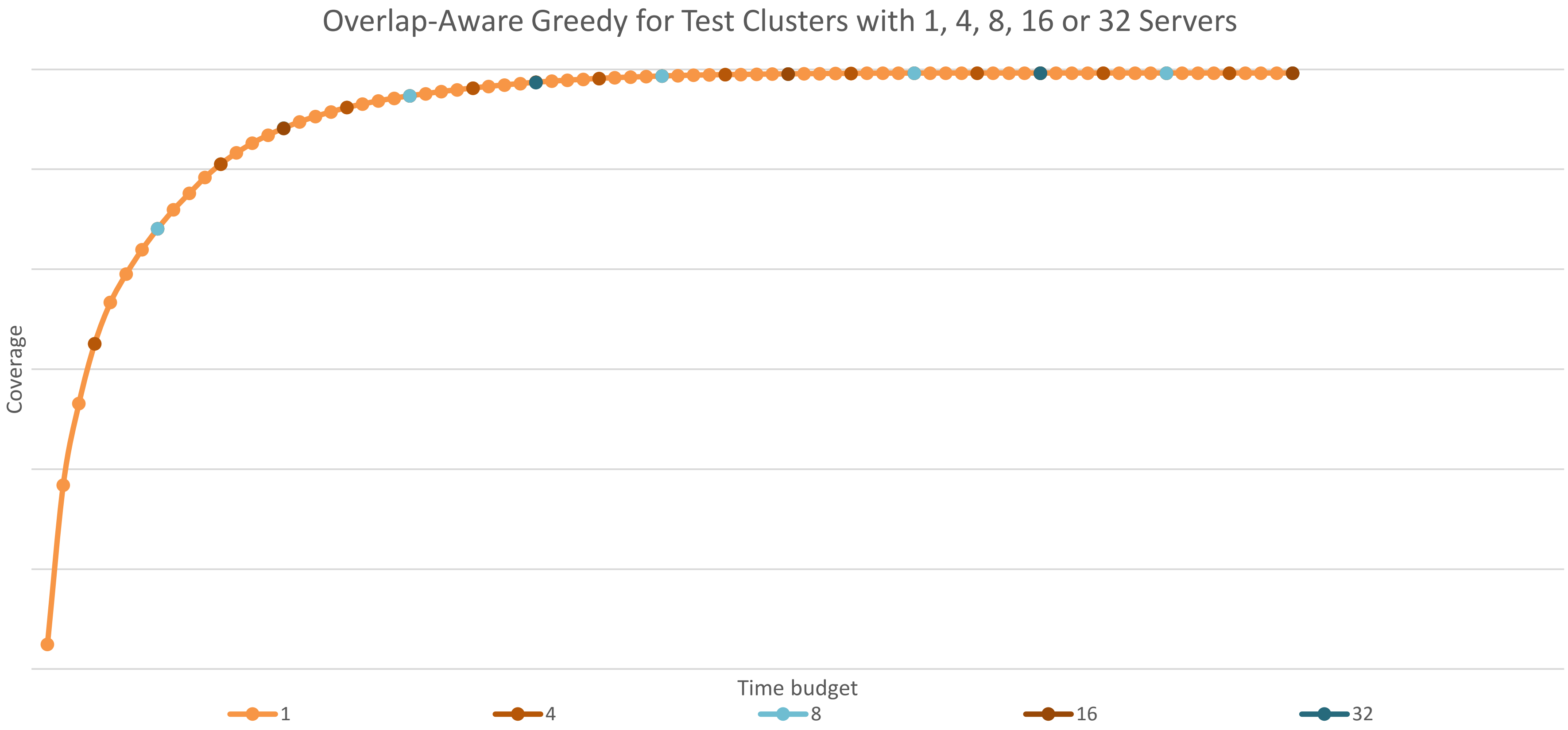
Parallel Variant for Test Clusters



Parallel Variant for Test Clusters



Overlap-Aware for Test Clusters



Coverage decrease < 0,01% -> works for test clusters

Coverage Redundancy

```c
int example_function(int a, int b) {
   int c = a + b;
   int d = a - b;
   return c*d;
}
```

Coverage Redundancy

```
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c*d;
}
```

	Test1	Test2	Test3
S1	x	x	
S2	x	x	
S3	x		x
S4		x	x
S5	x		x

Coverage Redundancy

```
1 int example_function(int a, int b) {
2     int c = a + b;
3     int d = a - b;
4     return c*d;
5 }
```

	Test1	Test2	Test3
S1	x	x	
S2	x	x	
S3	x		x
S4		x	x
S5	x		x

Coverage Redundancy

```
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c*d;
}
```

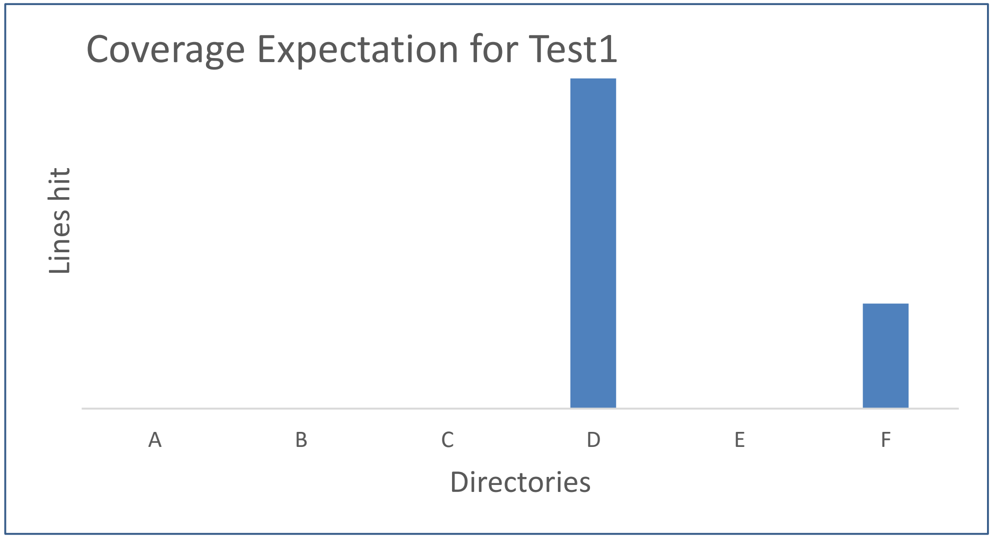
	Test1	Test2	Test3
S1	x	x	
S2	x	x	
S3	x		x
S4		x	x
S5	x		x

Coverage run	Lines hit	Line groups	Redundancy %
2015-11-15	2901575	79741	97.25
2016-05-19	3172337	93162	97.06
2016-08-04	3371109	97368	97.11
2016-10-25	3510727	104764	97.02
2016-11-01	3421780	104837	96.94
2016-11-15	3436853	106030	96.91

Large part of coverage data is redundant

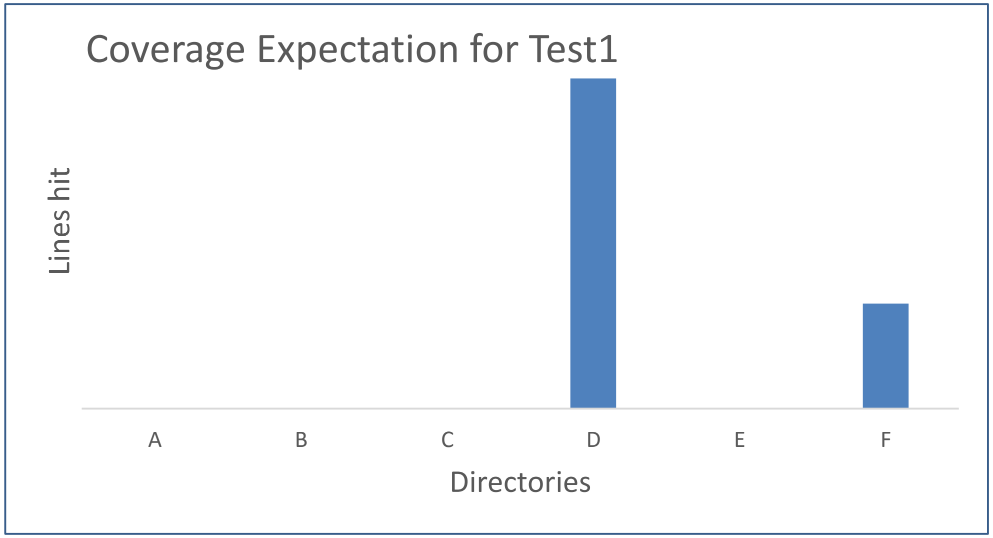
Shared Coverage Problem

- Ask SAP engineers where they expect coverage for Test1

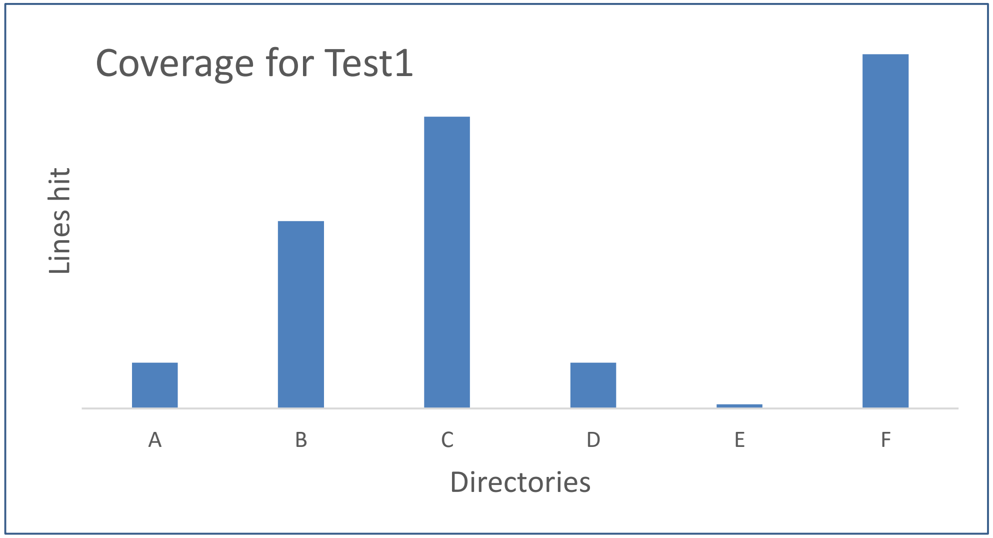


Shared Coverage Problem

- Ask SAP engineers where they expect coverage for Test1



- Measure Test1



Coverage does not characterize Test1

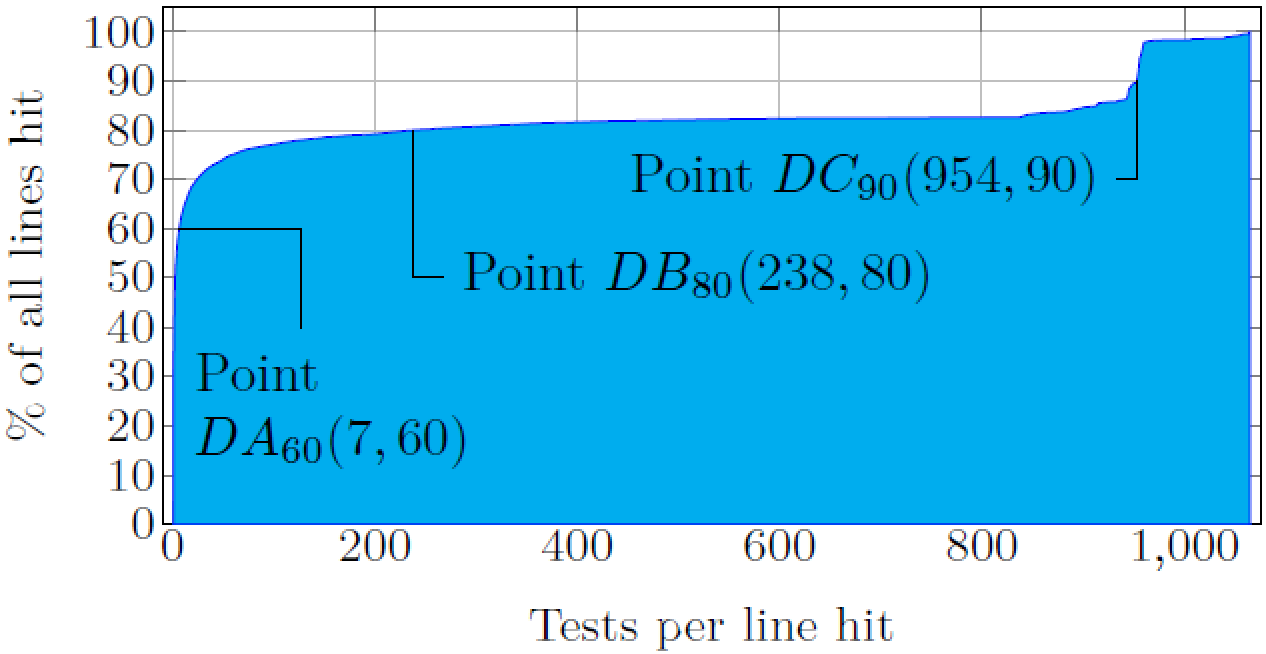
Filtering Shared Coverage Data

Considered two approaches:

a) Baseline approach
Define baseline test and remove baseline coverage from all other tests

b) Testcount approach
Remove all lines covered by more than e.g. 238 tests (of e.g. 1200 in total)

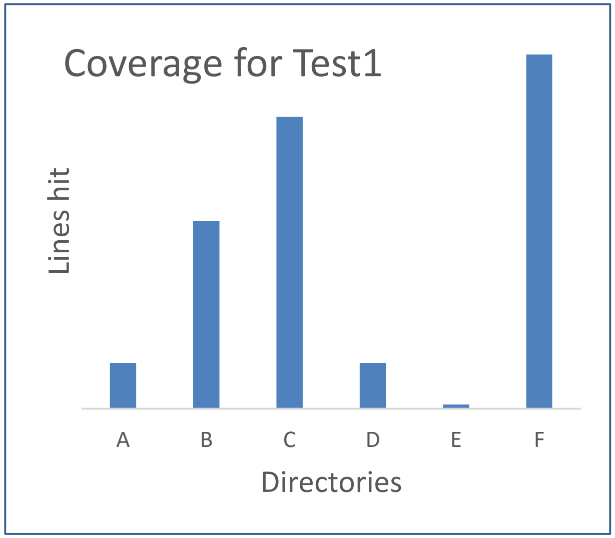
Testcount Approach



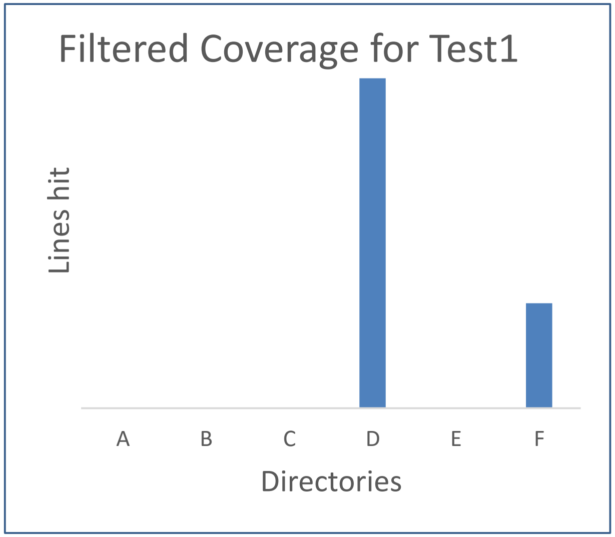
Distribution plot. E.g. 80% of all lines hit are covered by only 238 or less test suites and 31% of all lines are covered by only 1 test

Filtering Shared Coverage Evaluation

Measurement

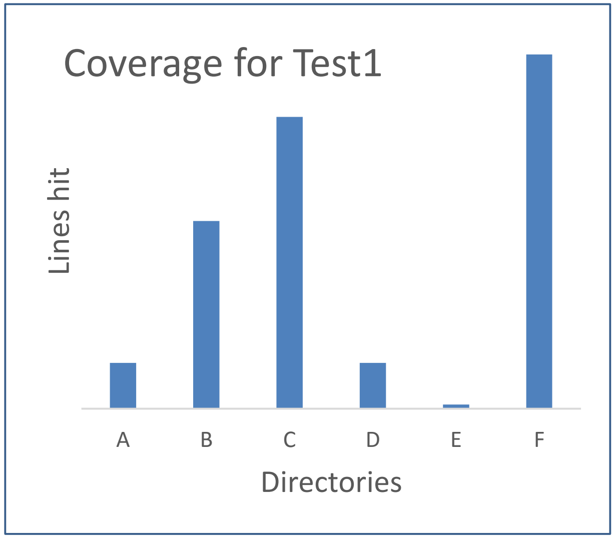


After Approach

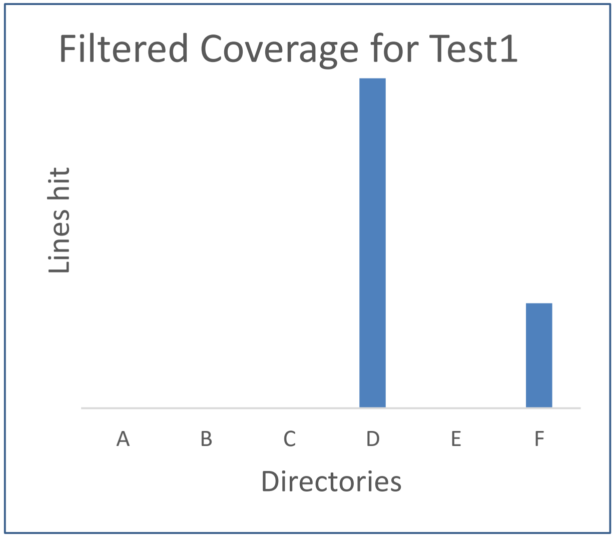


Filtering Shared Coverage Evaluation

Measurement



After Approach



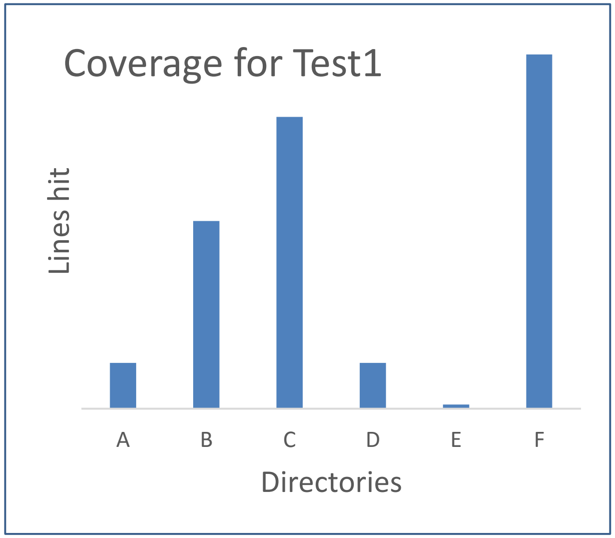
- List of top 5 directories ordered by lines hit:

F, C, B, D, A

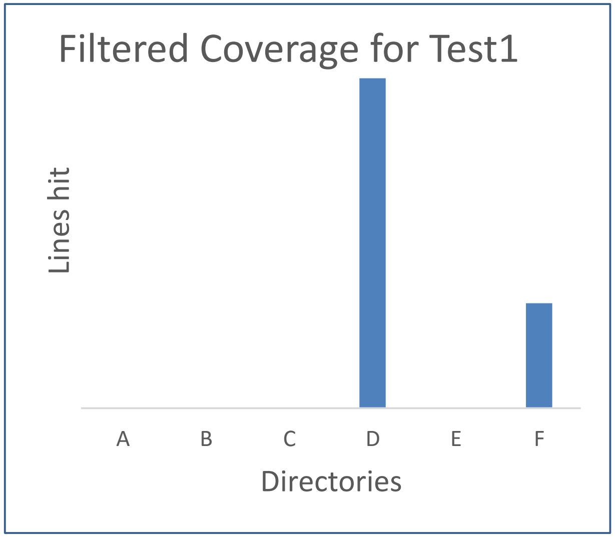
D, F, A, B, C

Filtering Shared Coverage Evaluation

Measurement



After Approach



- List of top 5 directories ordered by lines hit:

F, C, B, D, A   D, F, A, B, C

- Ask SAP engineers if this fits their expectations:

No   Yes

Filtering Shared Coverage Evaluation

	original				
Test1	🟩 x	🟥	🟥	🟥	🟥
…	🟩 x	🟥	🟥	🟥	🟥
	🟥	🟩 x	🟨	🟥	🟥
	🟩 x	🟥	🟩 x	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟩 x
Test10	🟥	🟥	🟥	🟥	🟨
	3/10				

🟩 correct=x
🟥 incorrect
🟨 unknown

T

Filtering Shared Coverage Evaluation

original

Test1	x	🟥	🟥	🟥	🟥
…	x	🟥	🟥	🟥	🟥
	🟥	x	🟨	🟥	🟥
	x	🟥	x	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	x
Test10	🟥	🟥	🟥	🟥	🟨

3/10

🟩	correct=x
🟥	incorrect
🟨	unknown

baseline approach

Test1	x	🟥	🟥	🟥	🟥
…	x	🟥	🟥	🟥	🟥
	x	🟥	🟥	🟥	🟥
	x	x	🟥	🟥	🟥
	🟥	🟥	x	🟥	🟥
	🟥	🟥	🟥	🟥	x
	🟥	x	🟥	🟥	🟥
	🟥	x	🟥	🟥	🟥
	🟥	x	x	x	🟥
Test10	🟥	🟥	🟨	🟥	🟥

4/10

testcount approach DB80

Test1	x	🟥	🟥	🟥	🟥
…	x	🟥	🟥	x	🟥
	x	🟨	🟥	🟥	🟥
	x	x	🟥	🟥	🟥
	x	🟥	🟨	🟥	🟥
	🟨	🟥	🟥	🟥	🟨
	x	🟥	🟥	🟥	x
	x	🟥	🟥	x	🟥
	x	x	🟥	x	🟨
Test10	🟥	🟥	x	🟥	🟥

8/10

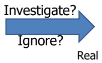
Specificity improved significantly

Summary

Gaps

Flaky Tests

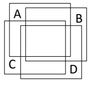
- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: OK
- Execute test 1: Failed
- Execute test 1: OK



Test infrastructure?
Hardware Problems?
Memory leak?
Test dependencies?
Real bug? (e.g. concurrency)
Performance?
and more …

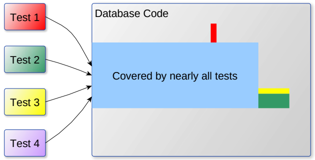
Random Coverage

- Coverage A: 651 074 lines hit
- Coverage B: 651 845 lines hit
- Coverage C: 651 862 lines hit
- Coverage D: 652 015 lines hit



Venn diagram

Shared coverage



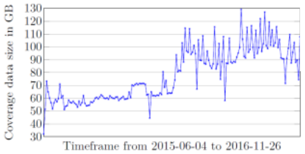
Evaluation with Small Projects

- Practitioners do not trust small evaluations

Work[1]	Size
Alspaugh et al. 2007	5 classes to 22 classes
Zhang et al. 2009	53 testcases to 209 testcases
Li et al. 2009	374 LOC to 11 kLOC
You et al. 2011	500 LOC to 10 kLOC
Zhang et al. 2013	2 kLOC to 80 kLOC
Do et al. 2008	7 kLOC to 80 kLOC
Elbaum et al. 2002	8 kLOC to 300 kLOC
Our work	> 3.50 MLOC

Related work comparing overlap-aware vs. non-overlap-aware solvers for TCS or TCP

Size of Coverage Data



Comparison Overlap-Aware

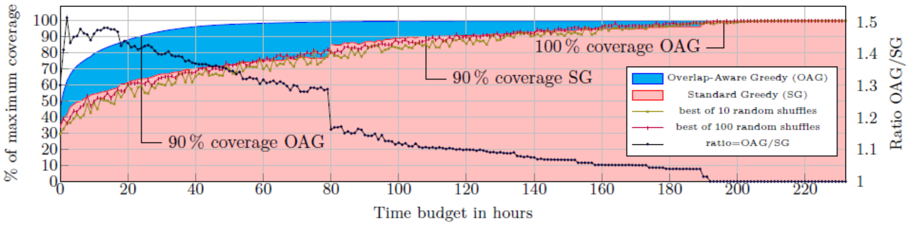


Figure 2. Exemplary comparison between different algorithms for maximum budgeted coverage problem. Higher is better

Coverage Redundancy

```
int example_function(int a, int b) {
    int c = a + b;
    int d = a - b;
    return c*d;
}
```

	t1	t2	t3
S1	x	x	
S2	x	x	
S3	x		x
S4		x	x
S5	x		x

Coverage run	Lines hit	Lines groups	Redundancy
2015-11-15	2901575	79741	97.25
2016-05-19	3172337	93162	97.06
2016-08-04	3371109	97368	97.11
2016-10-25	3510727	104764	97.02
2016-11-01	3421780	104837	96.94
2016-11-15	3436853	106030	96.91

Filtering Shared Coverage Evaluation

🟩 correct=x
🟥 incorrect
🟨 unknown

original

Test1	🟩 x	🟥	🟥	🟥	🟥
…	🟩 x	🟥	🟥	🟥	🟥
	🟥	🟩 x	🟨	🟥	🟥
	🟩 x	🟥	🟩 x	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟥
	🟥	🟥	🟥	🟥	🟩 x
Test10	🟥	🟥	🟥	🟥	🟨

3/10

baseline approach

Test1	🟩 x	🟥	🟥	🟥	🟥
…	🟩 x	🟥	🟥	🟥	🟥
	🟩 x	🟥	🟥	🟥	🟥
	🟩 x	🟩 x	🟥	🟥	🟥
	🟥	🟥	🟩 x	🟥	🟥
	🟥	🟥	🟥	🟥	🟩 x
	🟥	🟩 x	🟥	🟥	🟥
	🟥	🟩 x	🟥	🟥	🟥
	🟥	🟩 x	🟩 x	🟩 x	🟥
Test10	🟥	🟥	🟨	🟥	🟥

4/10

testcount approach DB80

Test1	🟩 x	🟥	🟥	🟥	🟥
…	🟩 x	🟥	🟥	🟩 x	🟥
	🟩 x	🟨	🟥	🟥	🟥
	🟩 x	🟩 x	🟥	🟥	🟥
	🟩 x	🟥	🟨	🟥	🟥
	🟨	🟥	🟥	🟥	🟨
	🟩 x	🟥	🟥	🟥	🟩 x
	🟩 x	🟥	🟥	🟩 x	🟥
	🟩 x	🟩 x	🟥	🟩 x	🟨
Test10	🟥	🟥	🟩 x	🟥	🟥

8/10

Backup Slides

Filtering Shared Coverage Evaluation

File	# lines hit
DirA\File1	2
DirB\File2	3
DirB\File3	2
DirB\File4	5
DirB\DirM\File5	7

Coverage result for Test1

Directory	# lines hit
DirA	2
DirB	17

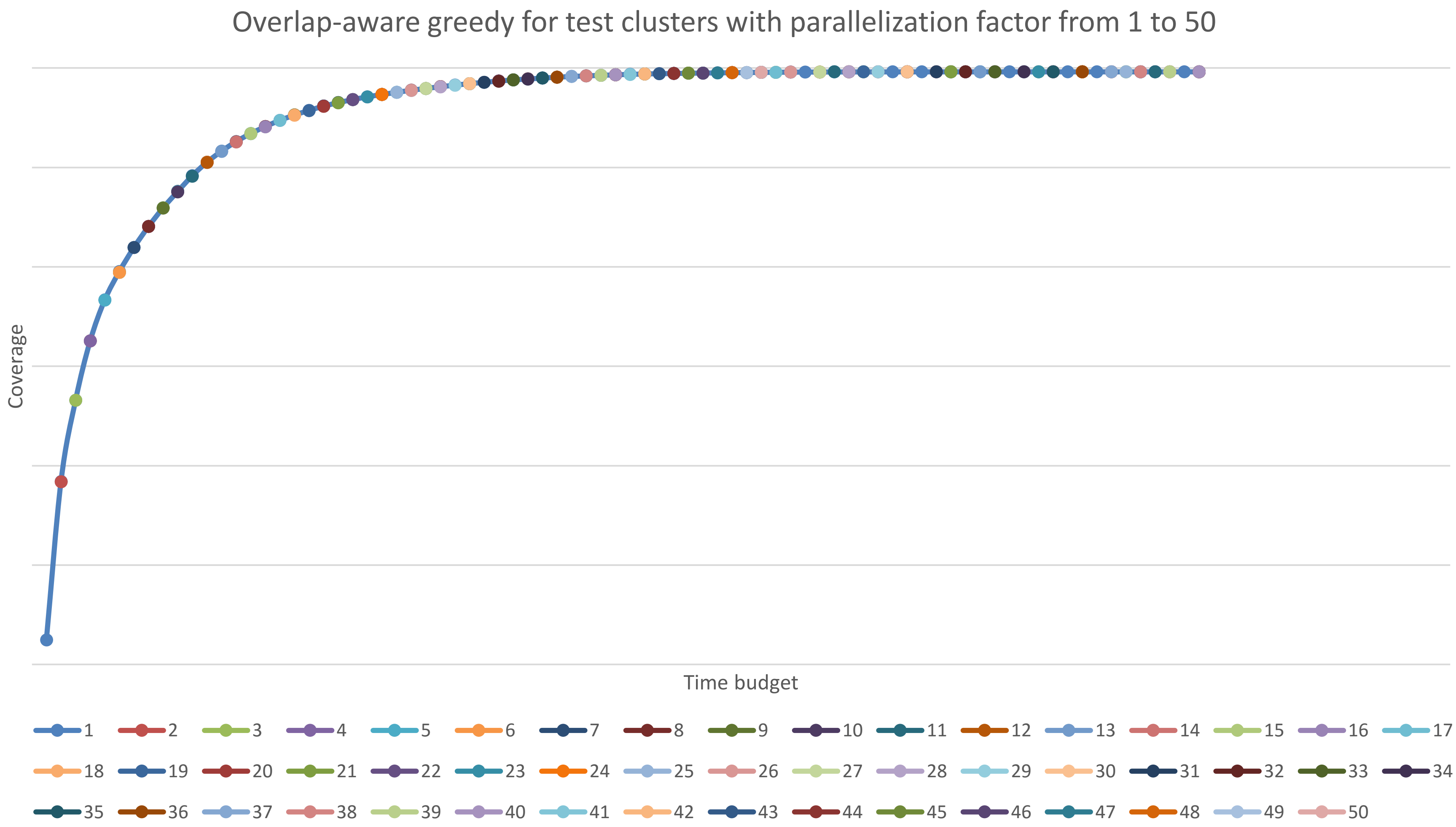
Coverage result for Test1 per directory

List of directories ordered by #lines hit:
DirB, DirA

Ask SAP engineers if DirA or DirB is expected for Test1

Top directory is wrong, coverage is not specific

Overlap-Aware for Test Clusters


Overlap-aware greedy for test clusters with parallelization factor from 1 to 50
Coverage
Time budget