Fraunhofer AISEC

Evaluating the performance of continuous test-based cloud service certification

Philipp Stephanow, Christian Banse

Fraunhofer AISEC

{philipp.stephanow, christian.banse}@aisec.fraunhofer.de

May 14, 2017

Fraunhofer AISEC

Overview

Motivation

- Certification of Cloud Services
  - satisfaction of a set of control (or requirements)

Motivation

- Certification of Cloud Services
  - satisfaction of a set of control (or requirements)
  - Exemplary source: Cloud Control Matrix (CCM) upon which CSA STAR Certificate is based

Motivation

- Certification of Cloud Services
  - satisfaction of a set of control (or requirements)
  - Exemplary source: Cloud Control Matrix (CCM) upon which CSA STAR Certificate is based
  - Problem: Traditional certification is a discrete task producing results valid for some interval, e.g. one year

Motivation

- Certification of Cloud Services
  - satisfaction of a set of control (or requirements)
  - Exemplary source: Cloud Control Matrix (CCM) upon which CSA STAR Certificate is based
  - Problem: Traditional certification is a discrete task producing results valid for some interval, e.g. one year
  - Implication: Audit results are stable during that interval

Motivation

- Certification of Cloud Services
  - satisfaction of a set of control (or requirements)
  - Exemplary source: Cloud Control Matrix (CCM) upon which CSA STAR Certificate is based
  - Problem: Traditional certification is a discrete task producing results valid for some interval, e.g. one year
  - Implication: Audit results are stable during that interval
  - Incorrect assumption: Cloud services may change during this period where changes are hard to predict

Motivation

- Certification of Cloud Services
  - satisfaction of a set of control (or requirements)
  - Exemplary source: Cloud Control Matrix (CCM) upon which CSA STAR Certificate is based
  - Problem: Traditional certification is a discrete task producing results valid for some interval, e.g. one year
  - Implication: Audit results are stable during that interval
  - Incorrect assumption: Cloud services may change during this period where changes are hard to predict
  - Popular example: Location of cloud service components (e.g. migrating VMs between different AWS regions)

Fraunhofer AISEC

Motivation

- Continuous Cloud Service Certification

Motivation

- Continuous Cloud Service Certification
  - automatically and repeatedly check if cloud service satisfies a set of requirements

Motivation

- Continuous Cloud Service Certification
  - automatically and repeatedly check if cloud service satisfies a set of requirements
  - *Monitoring-based certification techniques*: use monitoring data as evidence collected from components involved in service delivery during the execution of the cloud service

Motivation

- Continuous Cloud Service Certification
  - automatically and repeatedly check if cloud service satisfies a set of requirements
  - *Monitoring-based certification techniques*: use monitoring data as evidence collected from components involved in service delivery during the execution of the cloud service
  - *Test-based certification techniques*: produce evidence by controlling some input to the cloud service and evaluating the output, e.g., calling a cloud service's RESTful API and comparing responses with expected results

Motivation

- Continuous Cloud Service Certification
  - automatically and repeatedly check if cloud service satisfies a set of requirements
  - *Monitoring-based certification techniques*: use monitoring data as evidence collected from components involved in service delivery during the execution of the cloud service
  - ***Test-based certification techniques*: produce evidence by controlling some input to the cloud service and evaluating the output, e.g., calling a cloud service's RESTful API and comparing responses with expected results**

Problem

- Accuracy and precision of produced evidence (i.e., test results)

Problem

- Accuracy and precision of produced evidence (i.e., test results)
  - False positive test result: A test result *incorrectly* indicates that a control is *not* violated

Problem

- Accuracy and precision of produced evidence (i.e., test results)
  - False positive test result: A test result *incorrectly* indicates that a control is *not* violated
  - *False positive test results erodes cloud service customer's trust in test-based evidence*

Problem

- Accuracy and precision of produced evidence (i.e., test results)
  - False positive test result: A test result *incorrectly* indicates that a control is *not* violated
  - *False positive test results erodes cloud service customer's trust in test-based evidence*
  - False negative test results: A test result *incorrectly* indicates that a control is violated

Problem

- Accuracy and precision of produced evidence (i.e., test results)
  - False positive test result: A test result *incorrectly* indicates that a control is *not* violated
  - *False positive test results erodes cloud service customer's trust in test-based evidence*
  - False negative test results: A test result *incorrectly* indicates that a control is violated
  - *False negative test results undermines cloud service provider's trust in test-based evidence*

Problem

- Accuracy and precision of produced evidence (i.e., test results)
  - False positive test result: A test result *incorrectly* indicates that a control is *not* violated
  - *False positive test results erodes cloud service customer's trust in test-based evidence*
  - False negative test results: A test result *incorrectly* indicates that a control is violated
  - *False negative test results undermines cloud service provider's trust in test-based evidence*
- How well do continuously executed tests perform?

Problem

- Accuracy and precision of produced evidence (i.e., test results)
  - False positive test result: A test result *incorrectly* indicates that a control is *not* violated
  - *False positive test results erodes cloud service customer's trust in test-based evidence*
  - False negative test results: A test result *incorrectly* indicates that a control is violated
  - *False negative test results undermines cloud service provider's trust in test-based evidence*
- How well do continuously executed tests perform?
- Put differently: How close are produced test results to their true values?

Approach

- Goal: Evaluate performance of test-based cloud service certification techniques when executed continuously

Approach

- Goal: Evaluate performance of test-based cloud service certification techniques when executed continuously
  1. define four test metrics universally applicable to any test-based certification technique

Approach

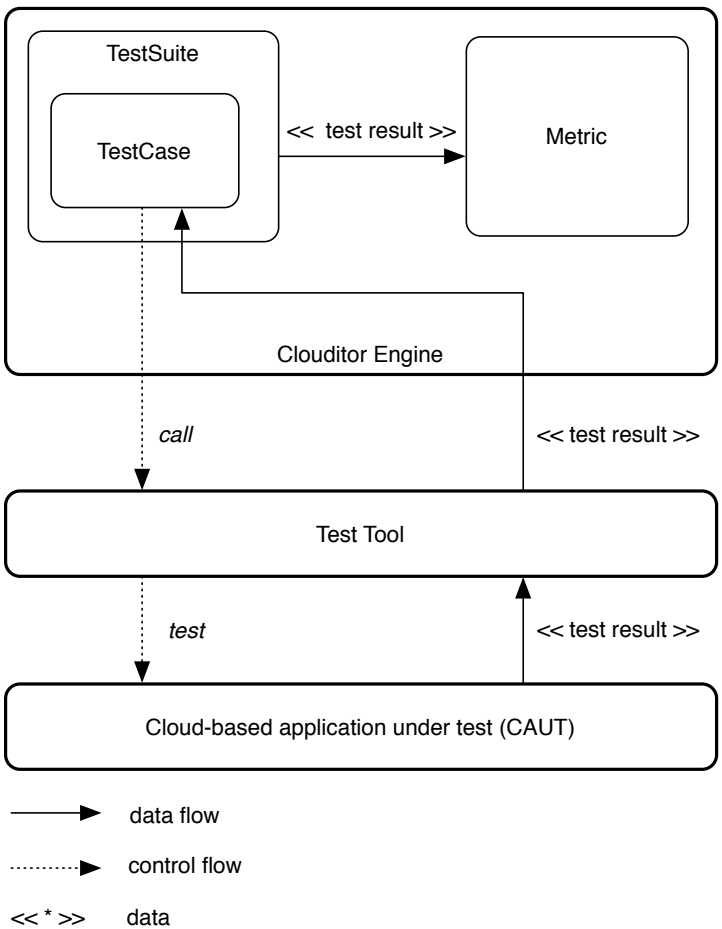
- Goal: Evaluate performance of test-based cloud service certification techniques when executed continuously
  1. define four test metrics universally applicable to any test-based certification technique
  2. derive performance measures to evaluate and compare alternative test-based techniques as well as alternative configurations

Approach

- Goal: Evaluate performance of test-based cloud service certification techniques when executed continuously
  1. define four test metrics universally applicable to any test-based certification technique
  2. derive performance measures to evaluate and compare alternative test-based techniques as well as alternative configurations
  3. provide a process to analyze performance of continuous test-based certification techniques

Framework

Building blocks to continuously produce test-based evidence



- Test cases: primitive
- Test suites: combine test cases, singular test
- Workflow: which test suite is run next?
- **Test metrics: produce evidence based on test (suite) results**

Universal continuous test metrics

- Basic-Result-Counter (brC)

Universal continuous test metrics

- Basic-Result-Counter (brC)
  - Basic result (br): indicates whether a test failed (f) or passed (p), i.e. $br \in \{f, p\}$

Universal continuous test metrics

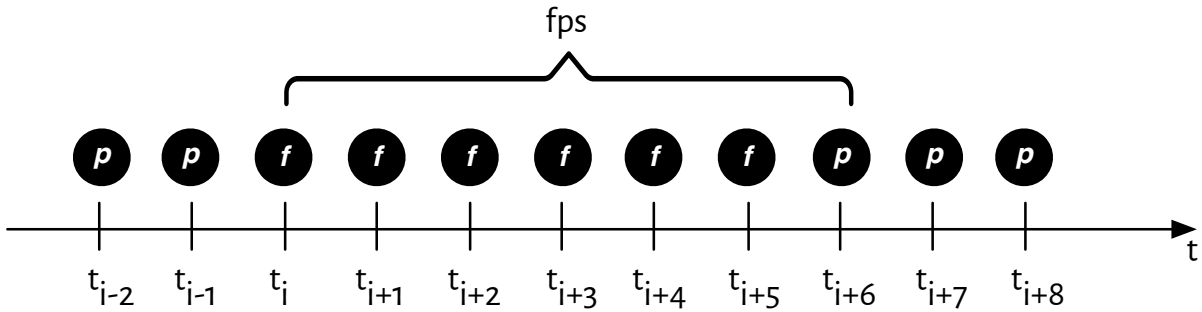
- Basic-Result-Counter (brC)
  - Basic result (br): indicates whether a test failed (f) or passed (p), i.e. $br \in \{f, p\}$
  - brC: Counts the times a test (suite) failed or passed over time

Universal continuous test metrics

- Basic-Result-Counter (brC)
  - Basic result (br): indicates whether a test failed (f) or passed (p), i.e. $br \in \{f, p\}$
  - brC: Counts the times a test (suite) failed or passed over time
- Failed-Passed-Sequence-Counter ($fpsC$)

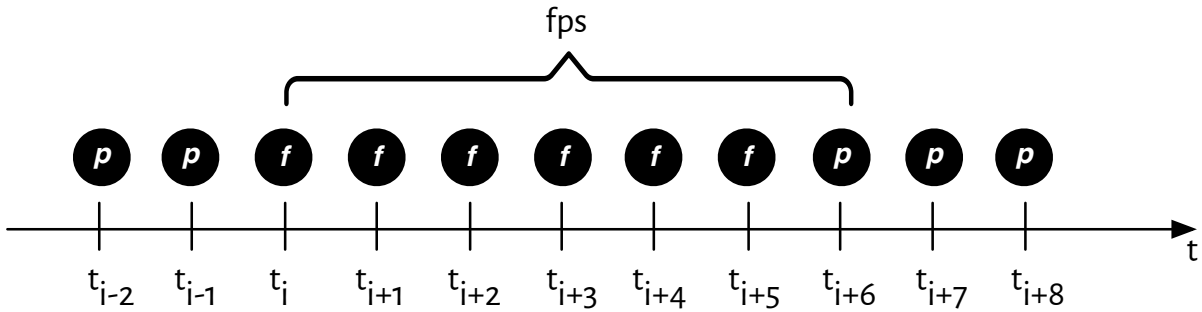
Universal continuous test metrics

- Basic-Result-Counter (*brC*)
  - Basic result (*br*): indicates whether a test failed (*f*) or passed (*p*), i.e. $br \in \{f, p\}$
  - *brC*: Counts the times a test (suite) failed or passed over time
- Failed-Passed-Sequence-Counter (*fpsC*)
  - Failed-Passed-Sequence (*fps*): special type of sequence of basic test results



Universal continuous test metrics

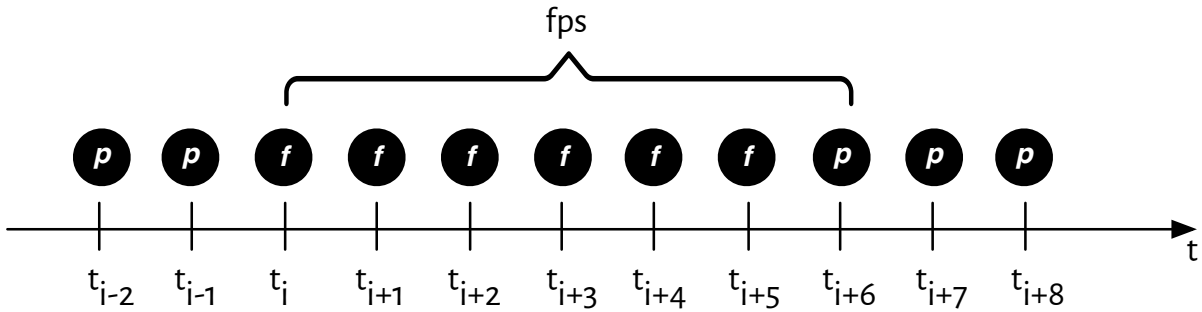
- Basic-Result-Counter (*brC*)
  - Basic result (*br*): indicates whether a test failed (*f*) or passed (*p*), i.e. $br \in \{f, p\}$
  - *brC*: Counts the times a test (suite) failed or passed over time
- Failed-Passed-Sequence-Counter (*fpsC*)
  - Failed-Passed-Sequence (*fps*): special type of sequence of basic test results



  - starts with a failed test at t_i provided that the previous test at t_{i-1} passed

Universal continuous test metrics

- Basic-Result-Counter (brC)
  - Basic result (br): indicates whether a test failed (f) or passed (p), i.e. $br \in \{f, p\}$
  - brC: Counts the times a test (suite) failed or passed over time
- Failed-Passed-Sequence-Counter ($fpsC$)
  - Failed-Passed-Sequence (fps): special type of sequence of basic test results



  - starts with a failed test at t_i provided that the previous test at t_{i-1} passed
  - ends with next occurrence of a passed test

Universal continuous test metrics

- Failed-Passed-Sequence-Duration (*fpsD*)

Universal continuous test metrics

- Failed-Passed-Sequence-Duration (*fpsD*)
  - builds on the definition of a failed-passed-sequence (*fps*)

Universal continuous test metrics

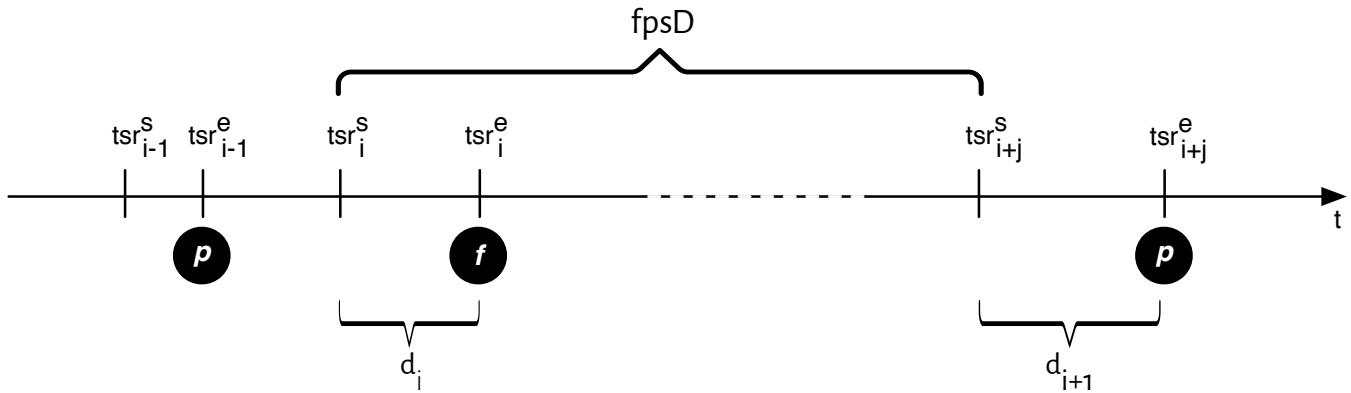
- Failed-Passed-Sequence-Duration (*fpsD*)
  - builds on the definition of a failed-passed-sequence (*fps*)
  - reason about properties over individual periods of time, e.g. time to fix TLS setup, downtime etc.

Universal continuous test metrics

- Failed-Passed-Sequence-Duration (*fpsD*)
  - builds on the definition of a failed-passed-sequence (*fps*)
  - reason about properties over individual periods of time, e.g. time to fix TLS setup, downtime etc.
  - time between the first failed test of a *fps* and its last basic test result which passes, i.e. $fps = \langle \boldsymbol{f_i}, f_{i+1}, f_{i+2}, \ldots, \boldsymbol{p_{i+j}} \rangle$.

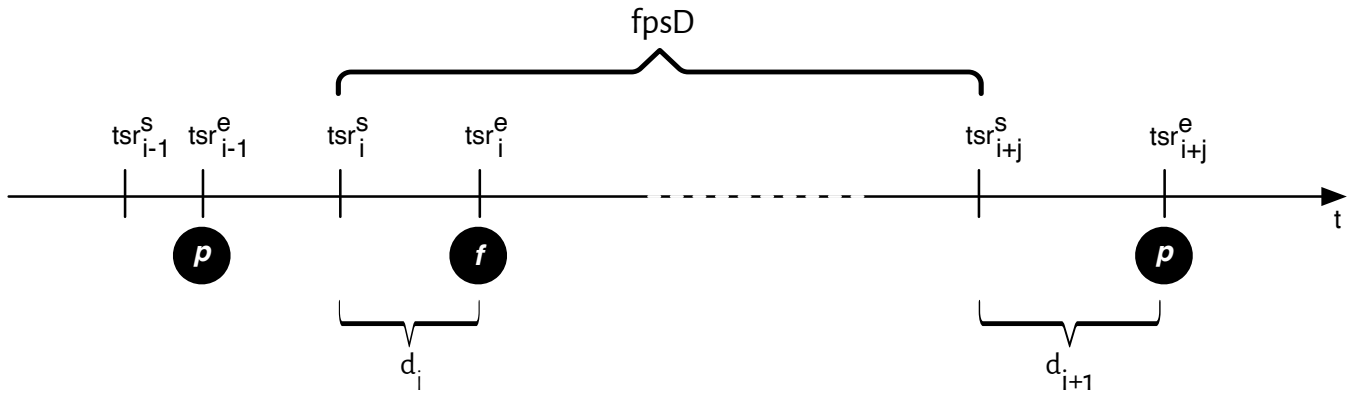
Universal continuous test metrics

- Failed-Passed-Sequence-Duration (*fpsD*)
  - builds on the definition of a failed-passed-sequence (*fps*)
  - reason about properties over individual periods of time, e.g. time to fix TLS setup, downtime etc.
  - time between the first failed test of a *fps* and its last basic test result which passes, i.e. $fps = \langle \boldsymbol{f_i}, f_{i+1}, f_{i+2}, \ldots, \boldsymbol{p_{i+j}} \rangle$.
  - bounds of a *fpsD*: start of the first test (suite) that failed and start of next test (suite) that passed



Universal continuous test metrics

- Failed-Passed-Sequence-Duration (*fpsD*)
  - builds on the definition of a failed-passed-sequence (*fps*)
  - reason about properties over individual periods of time, e.g. time to fix TLS setup, downtime etc.
  - time between the first failed test of a *fps* and its last basic test result which passes, i.e. $fps = \langle \boldsymbol{f_i}, f_{i+1}, f_{i+2}, \ldots, \boldsymbol{p_{i+j}} \rangle$.
  - bounds of a *fpsD*: start of the first test (suite) that failed and start of next test (suite) that passed



  - this definition of *fpsD* is agnostic to test durations

Universal continuous test metrics

- Cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)

Universal continuous test metrics

- Cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
  - builds on the definition of a Failed-Passed-Sequence-Duration (*fpsD*)

Universal continuous test metrics

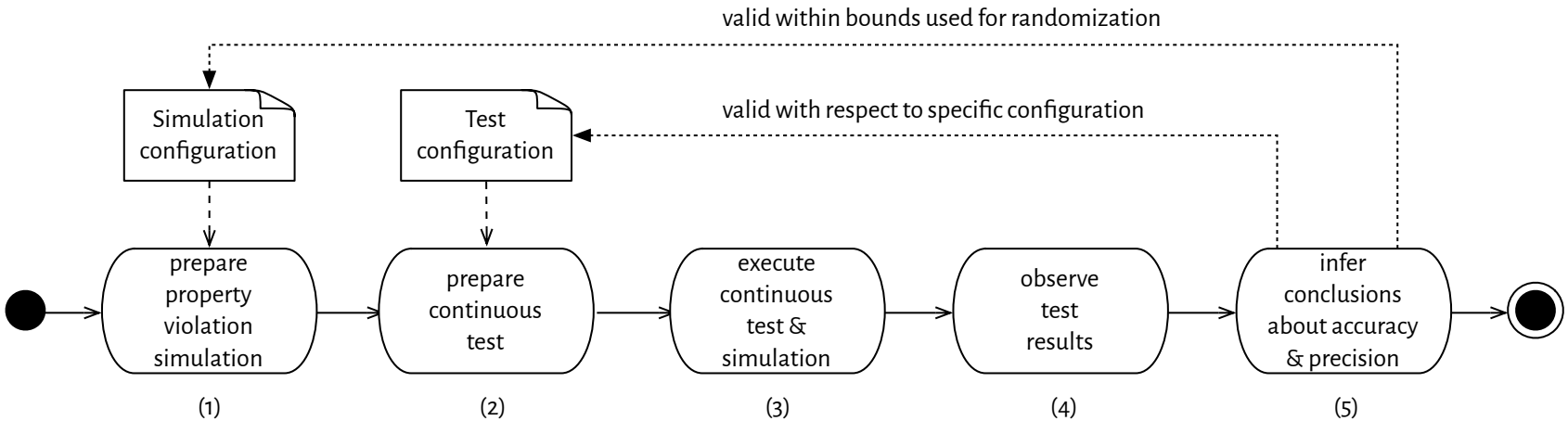
- Cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
  - builds on the definition of a Failed-Passed-Sequence-Duration (*fpsD*)
  - takes as input the sequence which contains any *fpsD* observed during a continuous test, and returns their accumulated value

Universal continuous test metrics

- Cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
  - builds on the definition of a Failed-Passed-Sequence-Duration (*fpsD*)
  - takes as input the sequence which contains any *fpsD* observed during a continuous test, and returns their accumulated value
  - permits us to reason about the satisfaction of cloud service properties within a predefined period of time, e.g. measure yearly downtime

Evaluating test-based certification techniques

Overview process



Property violation simulation

- manipulates properties of cloud service under test to mock violations a specific test-based technique aims to detect

Property violation simulation

- manipulates properties of cloud service under test to mock violations a specific test-based technique aims to detect
- establish the ground truth against which we evaluate specific test-based certification techniques

Property violation simulation

- manipulates properties of cloud service under test to mock violations a specific test-based technique aims to detect
- establish the ground truth against which we evaluate specific test-based certification techniques
- simulating properties' violations need to be executed repeatedly

Property violation simulation

- manipulates properties of cloud service under test to mock violations a specific test-based technique aims to detect
- establish the ground truth against which we evaluate specific test-based certification techniques
- simulating properties' violations need to be executed repeatedly
- $V = \langle pve_1, pve_2, \ldots, pve_i \rangle$ with $pve = (pveD, pveW)$

Property violation simulation

- manipulates properties of cloud service under test to mock violations a specific test-based technique aims to detect
- establish the ground truth against which we evaluate specific test-based certification techniques
- simulating properties' violations need to be executed repeatedly
- $V = \langle pve_1, pve_2, \ldots, pve_i \rangle$ with $pve = (pveD, pveW)$
- define bounds $[pveD^L, pveD^R]$ and $[pveW^L, pveW^R]$ randomize duration and waiting time of each $pve \in V$

Performance measures

- comparing results produced for universal test metrics with simulations

Performance measures

- comparing results produced for universal test metrics with simulations
- universal test metrics: *brC*, *fpsC*, *fpsD* and *cfpsD*

Fraunhofer AISEC

Performance measures

- comparing results produced for universal test metrics with simulations
- universal test metrics: *brC*, *fpsC*, ***fpsD*** and ***cfpsD***

Fraunhofer AISEC

Performance measures

- Performance based on Failed-Passed-Sequence-Duration (*fpsD*)

Fraunhofer AISEC

Performance measures

- Performance based on Failed-Passed-Sequence-Duration (*fpsD*)
- comparing produced *fpsD* with simulation:

Performance measures

- Performance based on Failed-Passed-Sequence-Duration (*fpsD*)
- comparing produced *fpsD* with simulation:
  - performance measures: $\boldsymbol{efpsD^{TN}}$, $efpsD^{TN}_{pre}$, $efpsD^{TN}_{post}$, $efpsD^{FN}$, and $efpsD^{FP}$

Performance measures

- Performance based on Failed-Passed-Sequence-Duration (*fpsD*)
- comparing produced *fpsD* with simulation:
  - performance measures: $\boldsymbol{efpsD^{TN}}$, $efpsD^{TN}_{pre}$, $efpsD^{TN}_{post}$, $efpsD^{FN}$, and $efpsD^{FP}$
  - computing standard descriptive statistics for each observed distribution, i.e. mean, sd, median, min, max

Performance measures

- Performance based on Failed-Passed-Sequence-Duration (*fpsD*)
- comparing produced *fpsD* with simulation:
  - performance measures: $\boldsymbol{efpsD^{TN}}$, $efpsD^{TN}_{pre}$, $efpsD^{TN}_{post}$, $efpsD^{FN}$, and $efpsD^{FP}$
  - computing standard descriptive statistics for each observed distribution, i.e. mean, sd, median, min, max
  - compute precision using confidence intervals for sample mean, allowing for conclusion such as *We are 95% confident that the mean error a true negative fps makes when estimating a property violation event is 2sec%±0.2.*

Performance measures

- Performance based on cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)

Performance measures

- Performance based on cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
- comparing produced *cfpsD* with simulation:

Performance measures

- Performance based on cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
- comparing produced *cfpsD* with simulation:
  - cumulative duration error of true negative *cfpsD*: $\boldsymbol{ecfpsD^{TN}}$

Performance measures

- Performance based on cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
- comparing produced *cfpsD* with simulation:
  - cumulative duration error of true negative *cfpsD*: $\boldsymbol{ecfpsD^{TN}}$
  - cumulative duration error of false negative *cfpsD*: $ecfpsD^{FN}$

Performance measures

- Performance based on cumulative-Failed-Passed-Sequence-Duration (*cfpsD*)
- comparing produced *cfpsD* with simulation:
  - cumulative duration error of true negative *cfpsD*: $\boldsymbol{ecfpsD^{TN}}$
  - cumulative duration error of false negative *cfpsD*: $ecfpsD^{FN}$
  - cumulative duration error of false positive *cfpsD*: $ecfpsD^{FP}$

Application

Setup & Environment

- Exemplary cloud services under test
  - $IaaS^{OS}$: IaaS instance provided by OpenStack Mitaka
  - $SaaS^{OS}$: $IaaS^{OS}$ instance plus Apache Webserver
  - $IaaS^{OS}$ and $SaaS^{OS}$ both attached to same tenant network

Setup & Environment

- Exemplary cloud services under test
  - $IaaS^{OS}$: IaaS instance provided by OpenStack Mitaka
  - $SaaS^{OS}$: $IaaS^{OS}$ instance plus Apache Webserver
  - $IaaS^{OS}$ and $SaaS^{OS}$ both attached to same tenant network
- Simulation framework
  - separate custom tool to simulate continuous property violations
  - deployed on designated instance but in the same tenant network as $IaaS^{OS}$ and $SaaS^{OS}$

Setup & Environment

- Exemplary cloud services under test
  - $IaaS^{OS}$: IaaS instance provided by OpenStack Mitaka
  - $SaaS^{OS}$: $IaaS^{OS}$ instance plus Apache Webserver
  - $IaaS^{OS}$ and $SaaS^{OS}$ both attached to same tenant network
- Simulation framework
  - separate custom tool to simulate continuous property violations
  - deployed on designated instance but in the same tenant network as $IaaS^{OS}$ and $SaaS^{OS}$
- Test-based certification framework
  - separate custom tool to implement continuous tests
  - runs on a different network, external to cloud infrastructure (but same building)

Setup & Environment

- Exemplary cloud services under test
  - $IaaS^{OS}$: IaaS instance provided by OpenStack Mitaka
  - $SaaS^{OS}$: $IaaS^{OS}$ instance plus Apache Webserver
  - $IaaS^{OS}$ and $SaaS^{OS}$ both attached to same tenant network
- Simulation framework
  - separate custom tool to simulate continuous property violations
  - deployed on designated instance but in the same tenant network as $IaaS^{OS}$ and $SaaS^{OS}$
- Test-based certification framework
  - separate custom tool to implement continuous tests
  - runs on a different network, external to cloud infrastructure (but same building)
- Evaluation engine
  - hosted locally (run after experiment)
  - statistics calculated using *Apache Commons Math*

Testing resource availability

- Scenario: Cloud service provider wants to select a test-based technique that overestimates actual downtimes (single events and cumulative duration) as little as possible

Testing resource availability

- Scenario: Cloud service provider wants to select a test-based technique that overestimates actual downtimes (single events and cumulative duration) as little as possible
- Exemplary control: *IVS-04* of the Cloud Control Matrix (CCM)

Testing resource availability

- Scenario: Cloud service provider wants to select a test-based technique that overestimates actual downtimes (single events and cumulative duration) as little as possible
- Exemplary control: *IVS-04* of the Cloud Control Matrix (CCM)
- Three alternative test-based techniques, all are executed every 60 seconds

Testing resource availability

- Scenario: Cloud service provider wants to select a test-based technique that overestimates actual downtimes (single events and cumulative duration) as little as possible
- Exemplary control: *IVS-04* of the Cloud Control Matrix (CCM)
- Three alternative test-based techniques, all are executed every 60 seconds
  - *PingTest*: Pings $IaaS^{OS}$ ten times to check availability, test passes if *average rtt* and *sd rtt* are smaller than 20ms and 10ms

Testing resource availability

- Scenario: Cloud service provider wants to select a test-based technique that overestimates actual downtimes (single events and cumulative duration) as little as possible
- Exemplary control: *IVS-04* of the Cloud Control Matrix (CCM)
- Three alternative test-based techniques, all are executed every 60 seconds
  - *PingTest*: Pings $IaaS^{OS}$ ten times to check availability, test passes if *average rtt* and *sd rtt* are smaller than 20ms and 10ms
  - *TCPTest*: Uses *Nping* to send TCP segments to determine whether $IaaS^{OS}$ is available, test passes if the maximum average response time and the maximum response time of probes are not greater than 75 and 100 ms.

Testing resource availability

- Scenario: Cloud service provider wants to select a test-based technique that overestimates actual downtimes (single events and cumulative duration) as little as possible
- Exemplary control: *IVS-04* of the Cloud Control Matrix (CCM)
- Three alternative test-based techniques, all are executed every 60 seconds
  - *PingTest*: Pings $IaaS^{OS}$ ten times to check availability, test passes if *average rtt* and *sd rtt* are smaller than 20ms and 10ms
  - *TCPTest*: Uses *Nping* to send TCP segments to determine whether $IaaS^{OS}$ is available, test passes if the maximum average response time and the maximum response time of probes are not greater than 75 and 100 ms.
  - *SSHTest*: Connects to $IaaS^{OS}$ via SSH and then test the session using a SSH_MSG_CHANNEL_REQUEST, a test passes if no I/O exception is thrown

Testing resource availability

- Simulation configuration
  - 1000 downtimes of $IaaS^{OS}$ for each of the three candidate tests

Testing resource availability

- Simulation configuration
  - 1000 downtimes of $IaaS^{OS}$ for each of the three candidate tests
  - Each downtime event lasted at least 60 seconds plus selecting [0,30] seconds at random ($pveD$)

Testing resource availability

- Simulation configuration
  - 1000 downtimes of $IaaS^{OS}$ for each of the three candidate tests
  - Each downtime event lasted at least 60 seconds plus selecting [0,30] seconds at random ($pveD$)
  - interval between two downtime events is at least 120 seconds plus selecting [0,60] seconds at random ($pveW$)

Testing resource availability

- Simulation configuration
  - 1000 downtimes of $IaaS^{OS}$ for each of the three candidate tests
  - Each downtime event lasted at least 60 seconds plus selecting [0,30] seconds at random ($pveD$)
  - interval between two downtime events is at least 120 seconds plus selecting [0,60] seconds at random ($pveW$)

Simulation statistic (sec)	$\mathbf{V}^{\mathbf{PingTest}}$	$\mathbf{V}^{\mathbf{TCPTest}}$	$\mathbf{V}^{\mathbf{SSHTest}}$
$cpveD$	75102.11	75544.07	75706.71
$\bar{x}_{pveD}$	75.11	75.54	75.71
sd_{pveD}	8.97	8.70	8.82
min_{pveD}	60.17	60.38	60.39
max_{pveD}	90.33	90.6	92.58

Fraunhofer AISEC

Testing resource availability

- Test statistics

Test statistic		PingTest	TCPTest	SSHTest
	$tsrC$	3153	3546	2491
tsr (sec)	$\bar{x}_{tsr}$	12.04	4.38	30.91
	sd_{tsr}	4.51	0.47	44.90
	min_{tsr}	9.01	4.02	0.54
	max_{tsr}	20.03	5.22	127.34
fpsD (sec)	$\bar{x}_{TN}$	81.22	75.13	176.74
	$sd_{fpsD\,TN}$	13.54	23.51	61.04
	$median_{fpsD\,TN}$	79.1	65.2	187.24
	$min_{fpsD\,TN}$	15.83	64.1	123.09
	$max_{fpsD\,TN}$	69.02	130.287	561.76
cfpsD (sec)	TN	78378.36	73481.13	82362.03
	FN	296.37	1099.5	1872.9
	FP	2103.27	818.69	38319.32

Fraunhofer AISEC

Testing resource availability

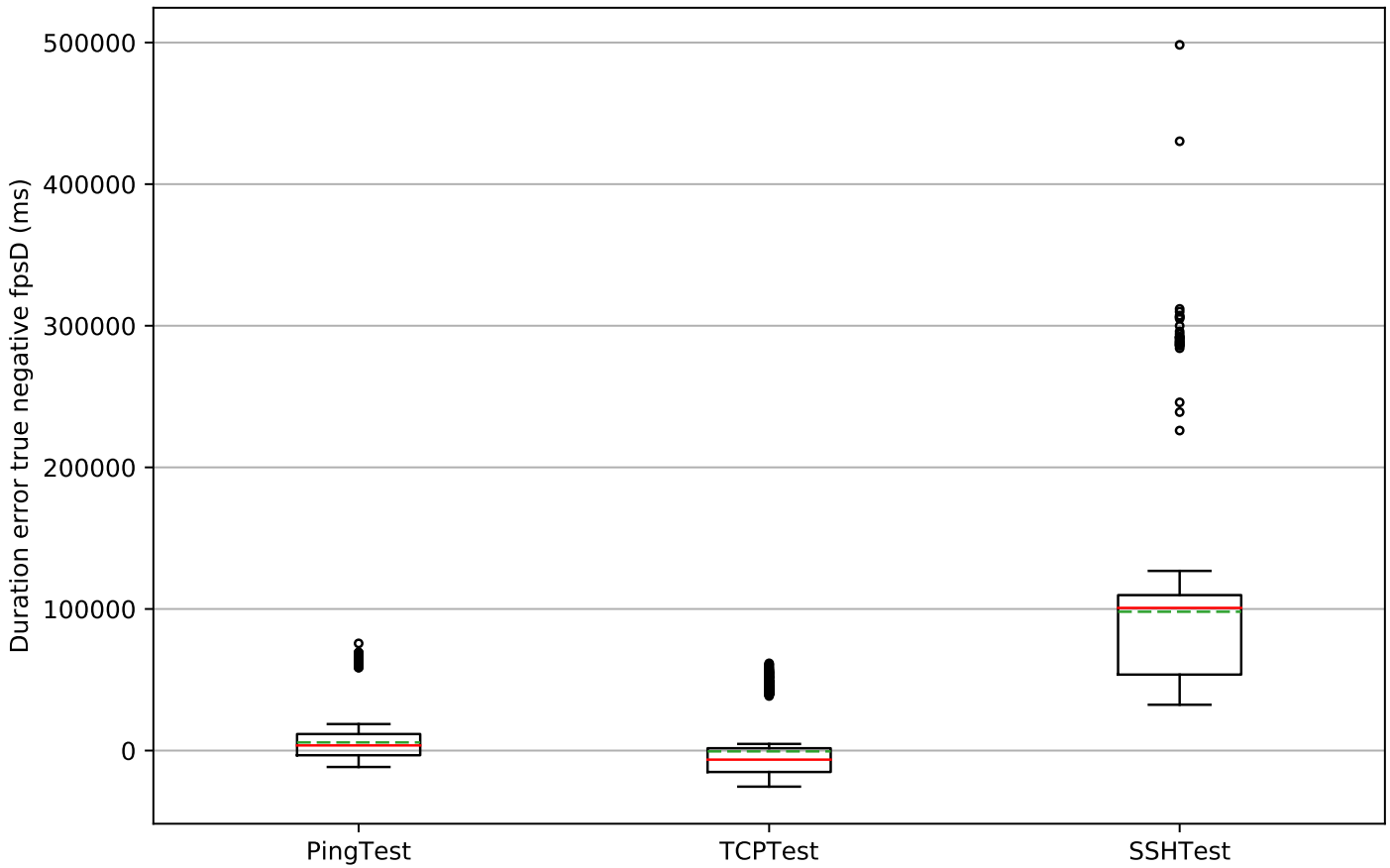
- Evaluation of PingTest, TCPTest and SSHTest to test availability of $IaaS^{OS}$ based on the failed-passed-sequence Duration ($fpsD$) test metric

Accuracy & Precision measure		PingTest	TCPTest	SSHTest
$efpsD^{TN}$ (ms)	$\bar{x}$	5749	-569	98050
	$\widehat{x}$	3643	-6386.5	100754
	sd	14143	21996	61586
	min	-11586	-25467	32372
	max	75692	61613	498360
	$E^{95\%}$	895	1378	5606
$efpsD^{TN}_{rel}$ (%)	$\bar{x}$	13.22	19.77	126.27
	$\widehat{x}$	9.22	14.94	116.49
	sd	14.74	18.41	83.69
	min	0.26	0.32	35.66
	max	119.39	91.26	786.01
	$E^{95\%}$	0.93	1.16	7.62

Fraunhofer AISEC

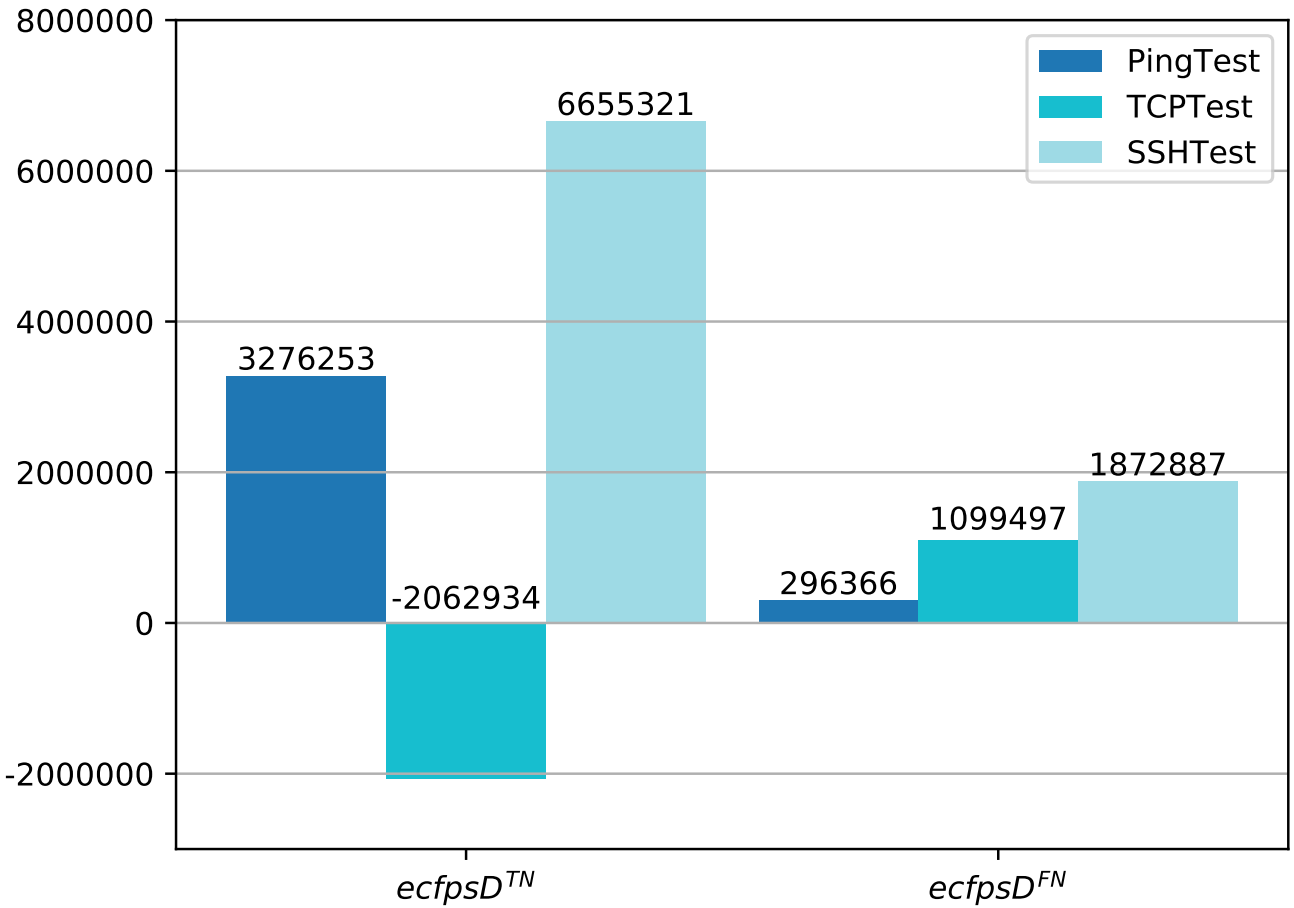
Testing resource availability

- Duration error of true negative fps ($efpsD^{TN}$) of PingTest, TCPTest, and SSHTest



Testing resource availability

- cumulative duration error of true negative fps ($ecfpsD^{TN}$) of PingTest, TCPTest, and SSHTest



Summary

Summary

- Conclusion

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques

Fraunhofer AISEC

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test
  - derived performance measures based on these four universal test metrics

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test
  - derived performance measures based on these four universal test metrics
  - demonstrated application of the method in three exemplary scenarios

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test
  - derived performance measures based on these four universal test metrics
  - demonstrated application of the method in three exemplary scenarios
- Next steps

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test
  - derived performance measures based on these four universal test metrics
  - demonstrated application of the method in three exemplary scenarios
- Next steps
  - a scenario driven approach which allow to select the assurance and precision measure most suited for that particular scenario

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test
  - derived performance measures based on these four universal test metrics
  - demonstrated application of the method in three exemplary scenarios
- Next steps
  - a scenario driven approach which allow to select the assurance and precision measure most suited for that particular scenario
  - simulating property violation events approximating those observed in the wild

Fraunhofer AISEC

Summary

- Conclusion
  - a method to experimentally evaluate the accuracy and precision of test-based cloud service certification techniques
  - proposed four universal test metrics applicable to any continuous test
  - derived performance measures based on these four universal test metrics
  - demonstrated application of the method in three exemplary scenarios
- Next steps
  - a scenario driven approach which allow to select the assurance and precision measure most suited for that particular scenario
  - simulating property violation events approximating those observed in the wild
  - consider other characteristics determining which test-based technique is most suitable for a given scenario, e.g., overhead incurred

Questions?