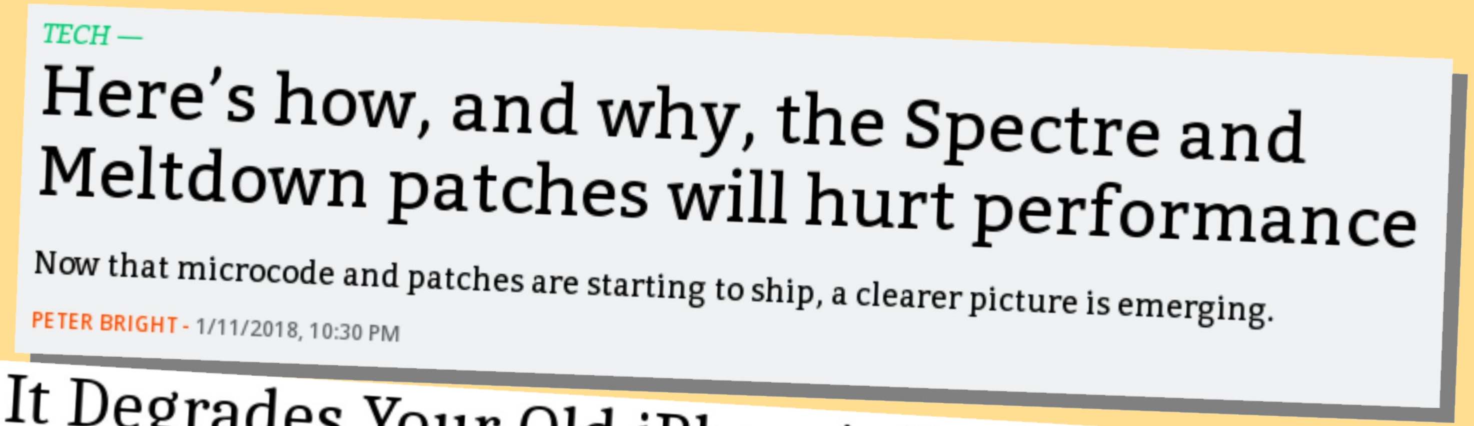
Assessing Performance Evolution for Configurable Systems

A Methodology

Advisors: Prof. Dr.-Ing. Ina Schaefer, Prof. Dr.-Ing. Norbert Siegmund

January 23, 2018

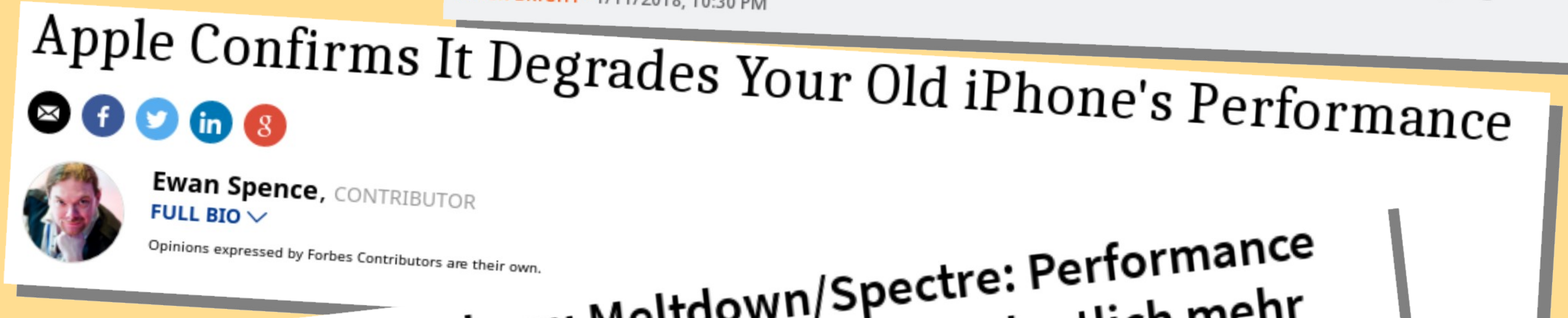
Stefan Mühlbauer


TECH —

Here's how, and why, the Spectre and Meltdown patches will hurt performance

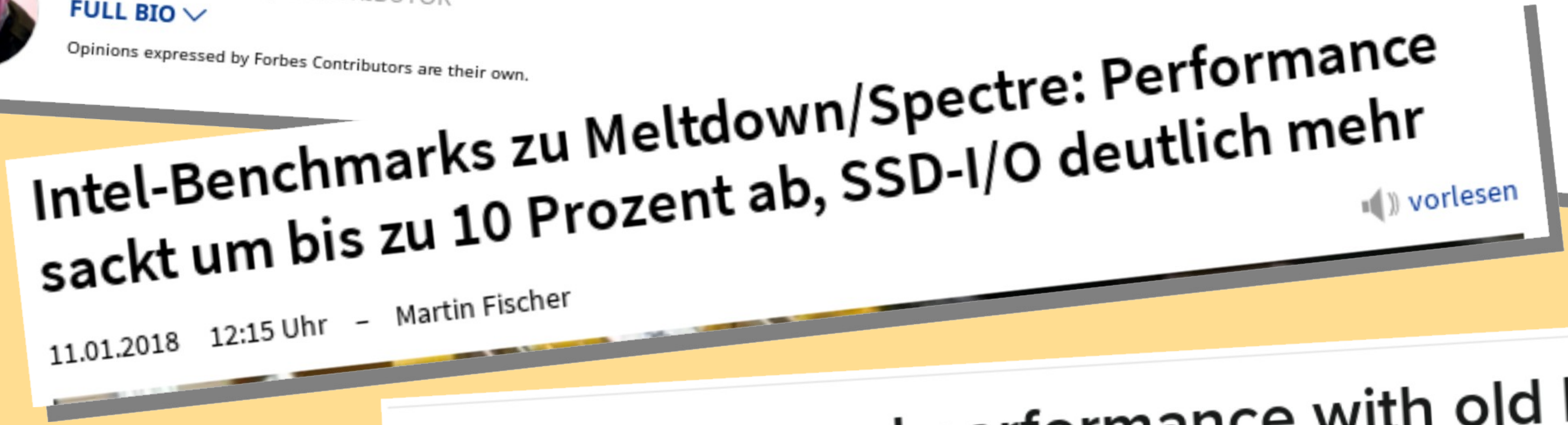
Now that microcode and patches are starting to ship, a clearer picture is emerging.

PETER BRIGHT - 1/11/2018, 10:30 PM


Apple Confirms It Degrades Your Old iPhone's Performance

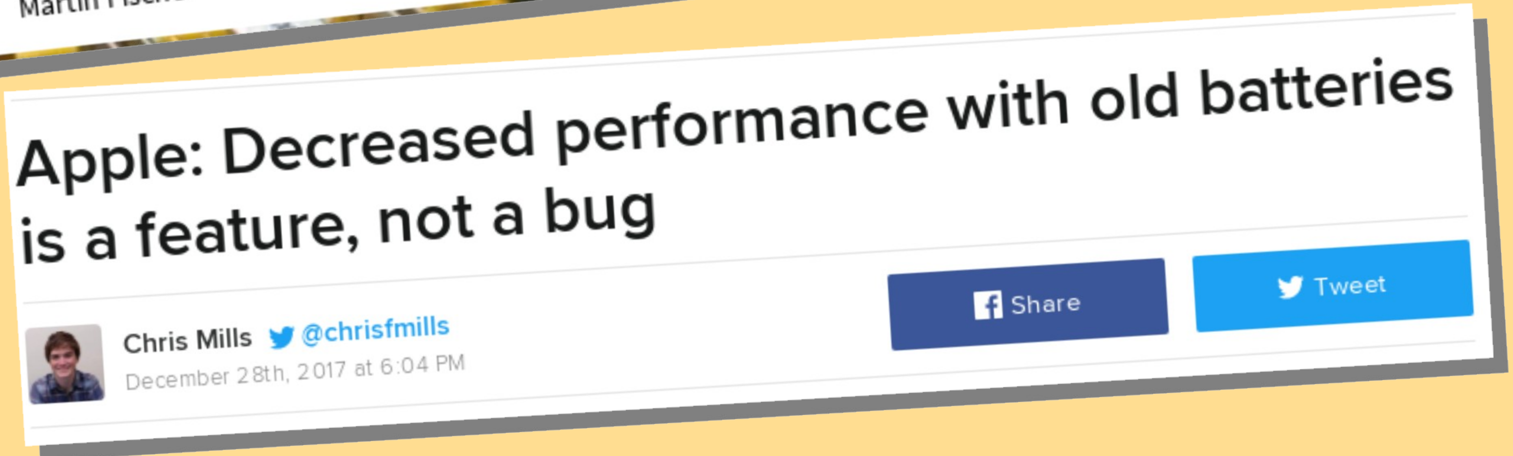
Ewan Spence, CONTRIBUTOR
FULL BIO

Opinions expressed by Forbes Contributors are their own.


Intel-Benchmarks zu Meltdown/Spectre: Performance sackt um bis zu 10 Prozent ab, SSD-I/O deutlich mehr

vorlesen

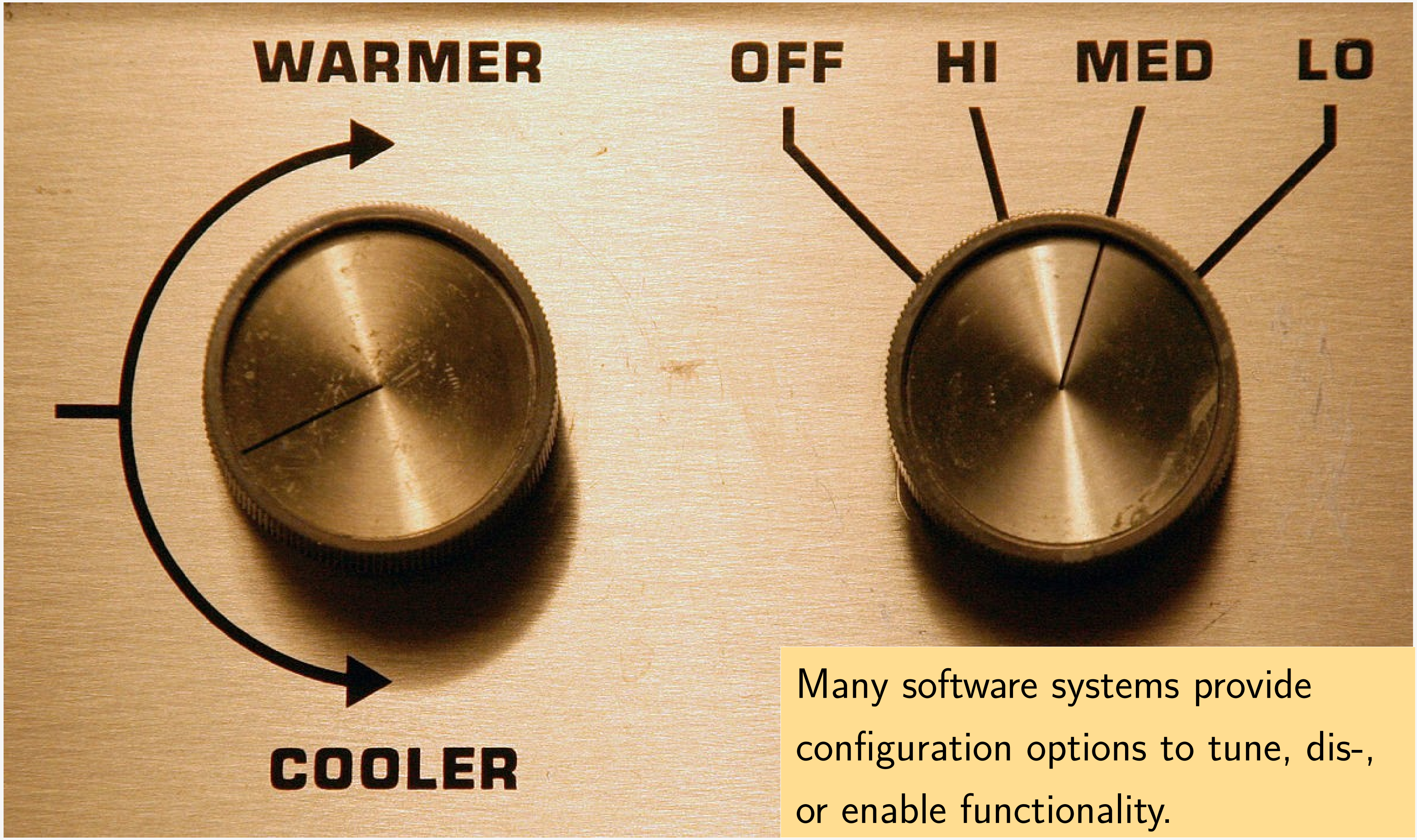
11.01.2018 12:15 Uhr – Martin Fischer


Apple: Decreased performance with old batteries is a feature, not a bug

Share
Tweet

Chris Mills @chrisfmills
December 28th, 2017 at 6:04 PM

Configurable Software Systems



Many software systems provide configuration options to tune, dis-, or enable functionality.

Configurable Software Systems

- Configurability: Compile-time vs load-time

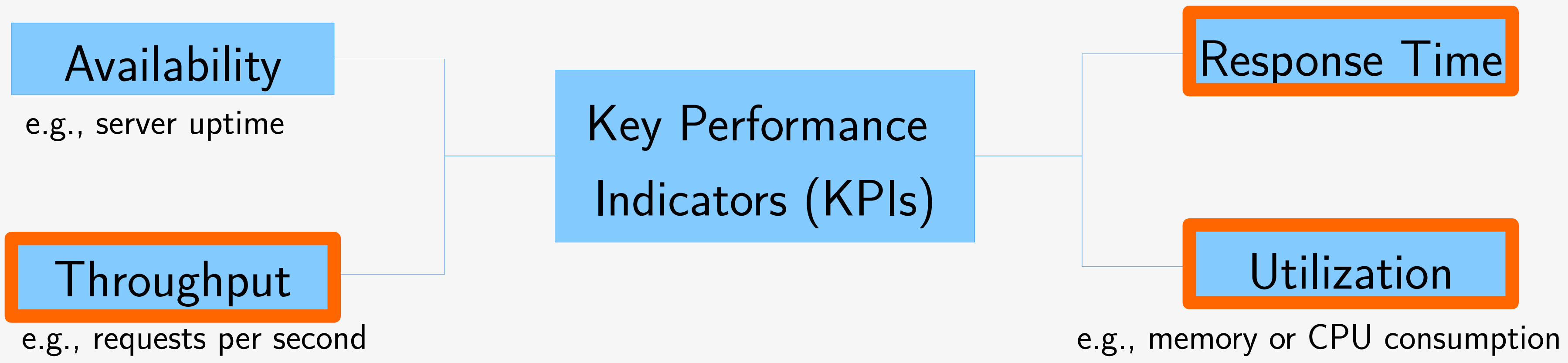
```
~$ zip -e -9 file.txt
```

- Unanticipated behavior can emerge with selections of multiple features (feature interaction)
  - Example: Compression and Encryption

    *Encrypting compressed data can be **faster** than encrypting raw data, since compressed data is already more compact.*

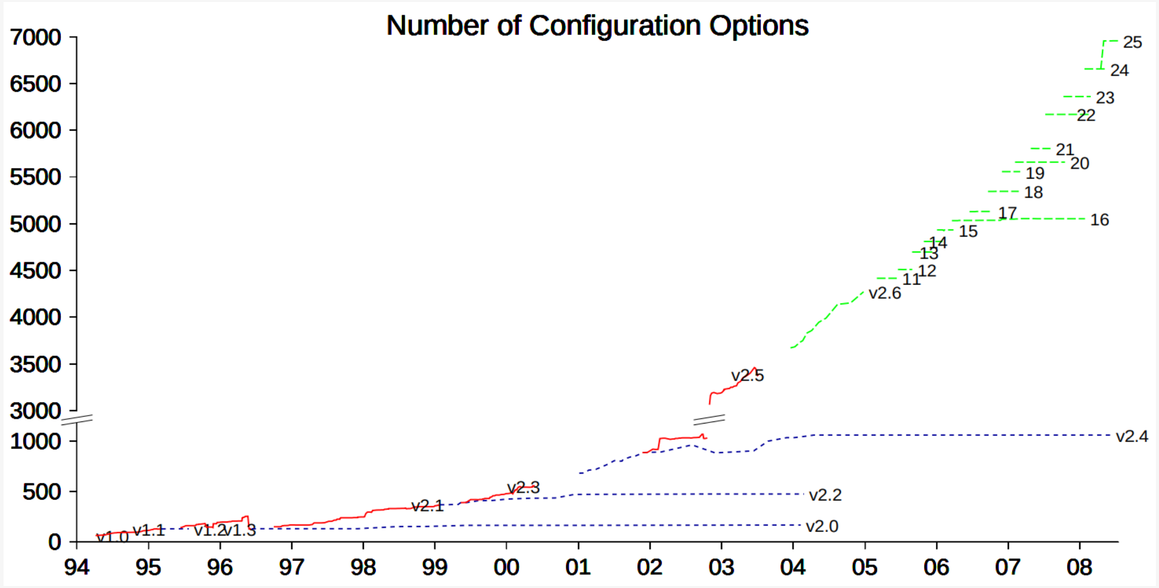
Software Performance Metrics

- Software performance: How **efficiently** is a task executed?
- Performance footprints can be outlined by key performance indicators:


Availability
e.g., server uptime
Throughput
e.g., requests per second
Key Performance Indicators (KPIs)
Response Time
Utilization
e.g., memory or CPU consumption

Software Evolution

- Evolution: adaption to changing contexts and requirements
- Software can *grow more complex* over time as it evolves:



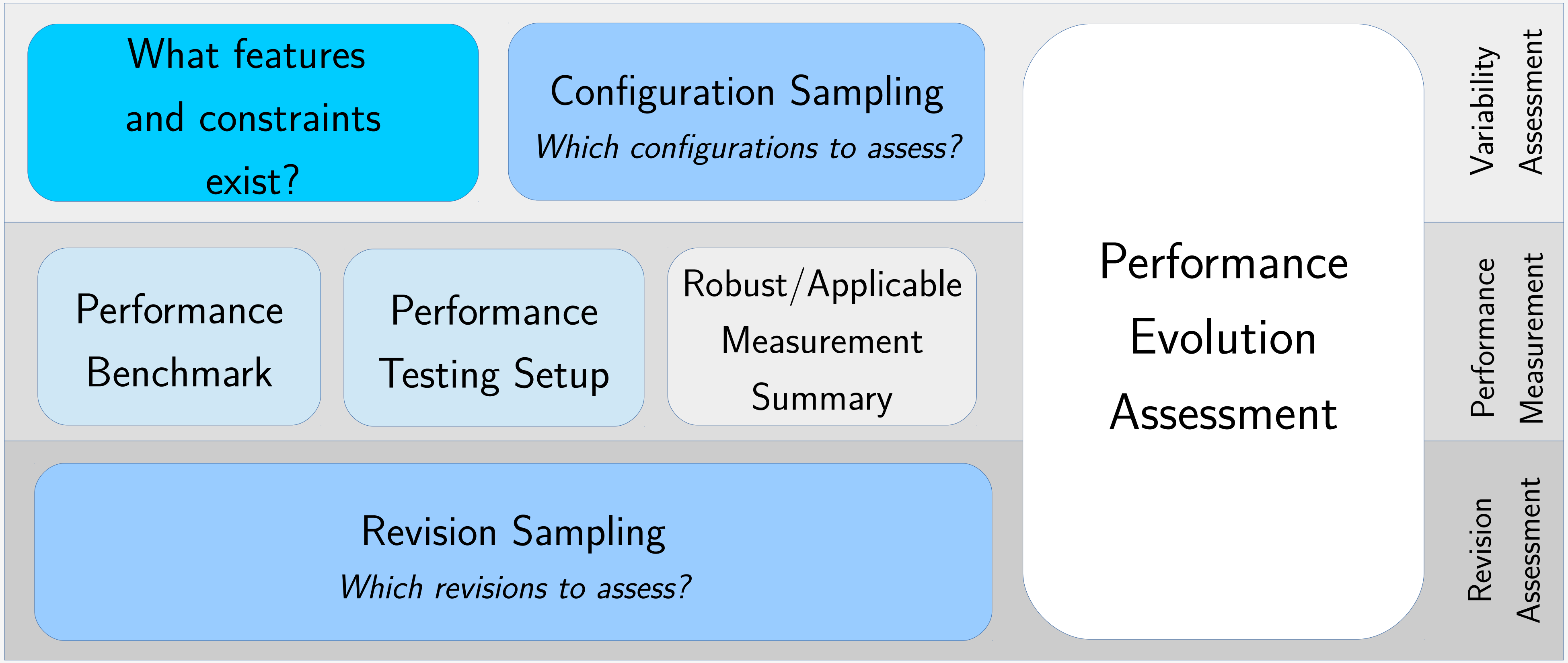
Adopted: Fig. 1 from Israeli, A., & Feitelson, D. G. (2010). *The Linux kernel as a case study in software evolution*. Journal of Systems and Software, 83(3), 485-501.

- Software can "erode" as it evolves, leading to degradation of overall software quality

Problem: Performance Evolution Assessment

- Performance evolution: assessment of performance for multiple versions *required*
- Performance Evolution Assessment (PEA) – How to do that?
- Problem space is outlined by three intertwined dimensions:
  - Variability: infeasibly high numbers of variants, feature interactions
  - Performance Assessment: suitable performance indicators/measures
  - Diachrony: incremental code revisions, software development history

Goal: Methodology for PEA


What features and constraints exist?
Configuration Sampling
Which configurations to assess?
Performance Benchmark
Performance Testing Setup
Robust/Applicable Measurement Summary
Revision Sampling
Which revisions to assess?
Performance Evolution Assessment
Variability Assessment
Performance Measurement
Revision Assessment

Variability Model Synthesis (1)

What features and constraints exist?

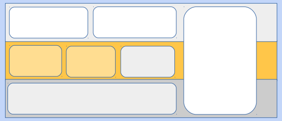
Variability Model Synthesis

- Automated
  - Natural Language Processing — *Cluster product descriptions, extract common features.*
  - Machine Learning — *Learn variability model from given configurations.*
  - Static Analyses — *Exploitation of config. APIs, or constraint violations*
- Manual
  - load-time vs. run-time variability
  - configuration encoding
  - features, types, and domains
  - feature dependencies
  - variability model evolution

Variability Model Synthesis (2)

- Automated approaches are only applicable under **preconditions**:
  - NLP: Textual description of a product required ($\sim$ domain analysis)
  - ML: Sufficiently large number of valid feature selections required
  - Static analyses: highly-specialized use cases
- General purpose strategy: manual assessment based on documentation
- Configuration sampling with respect to feature interactions: t-wise sampling

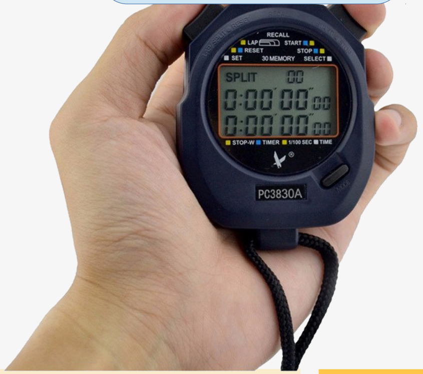
Performance Assessment (1)



- Performance benchmark selection
  - Expressive: measure desired performance metrics/indicators
  - Reproducible: obtain similar results under equal circumstances
  - Cost-efficient: reasonable cost of benchmark testing
- Profiling: dynamic assessment of performance metrics
  - No general purpose profiler, depending on test setup
  - e.g., VisualVM for Java, AOP, network sniffers, ...
- Timing statistics on host-machine: /usr/bin/time

Performance Benchmark

Performance Testing Setup



Performance Assessment (2)

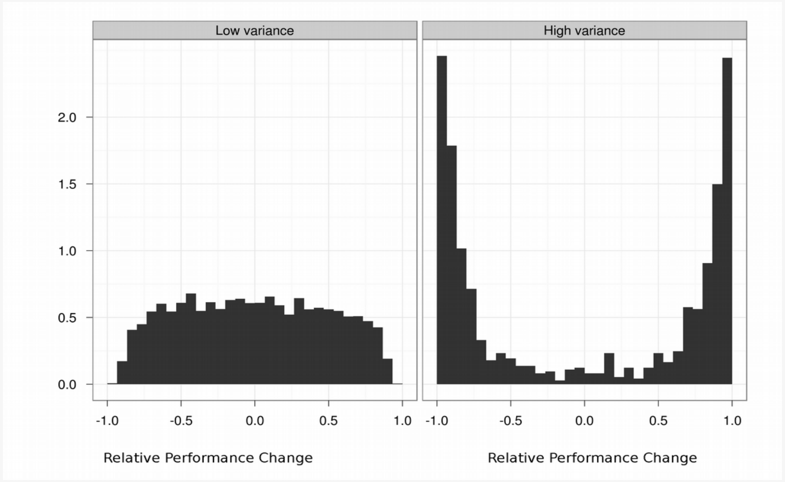
Robust/Applicable Measurement Summary

- Robust measures: median and interquartile range
  - Robust statistical measures are not distorted by extreme measurements

- Performance change magnitude: Relative change per variant

V1	0.5 %
...	...
V10	0.45 %
...	...
V20	-0.3 %

- Performance change range: variance of change per version



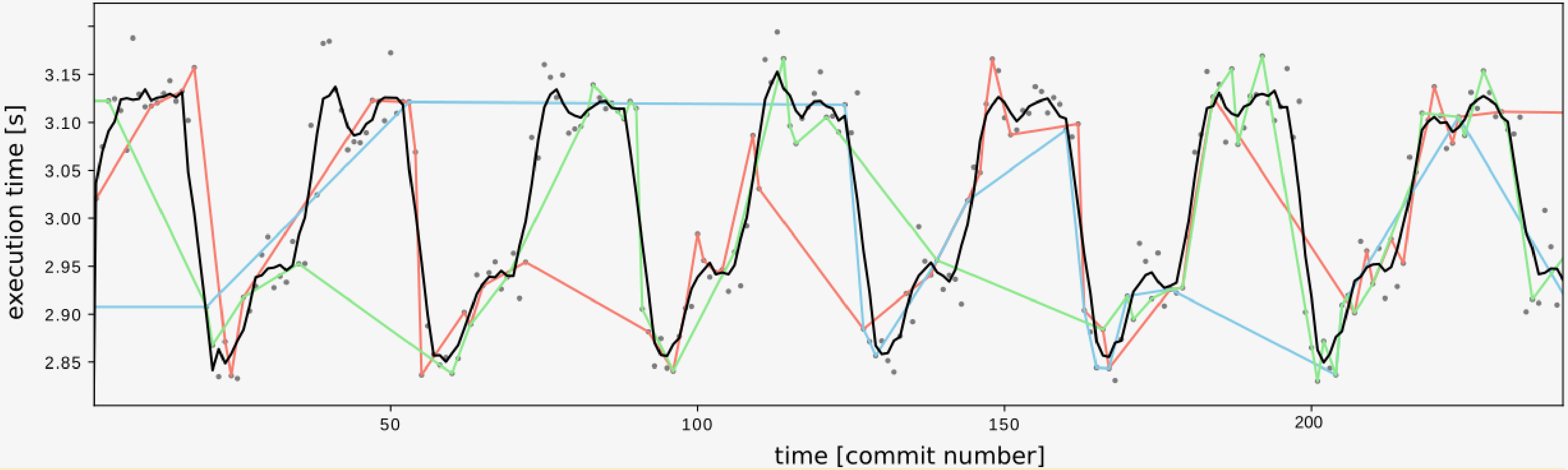
Evaluation & Case Study

RQ: What revisions induce significant performance changes?

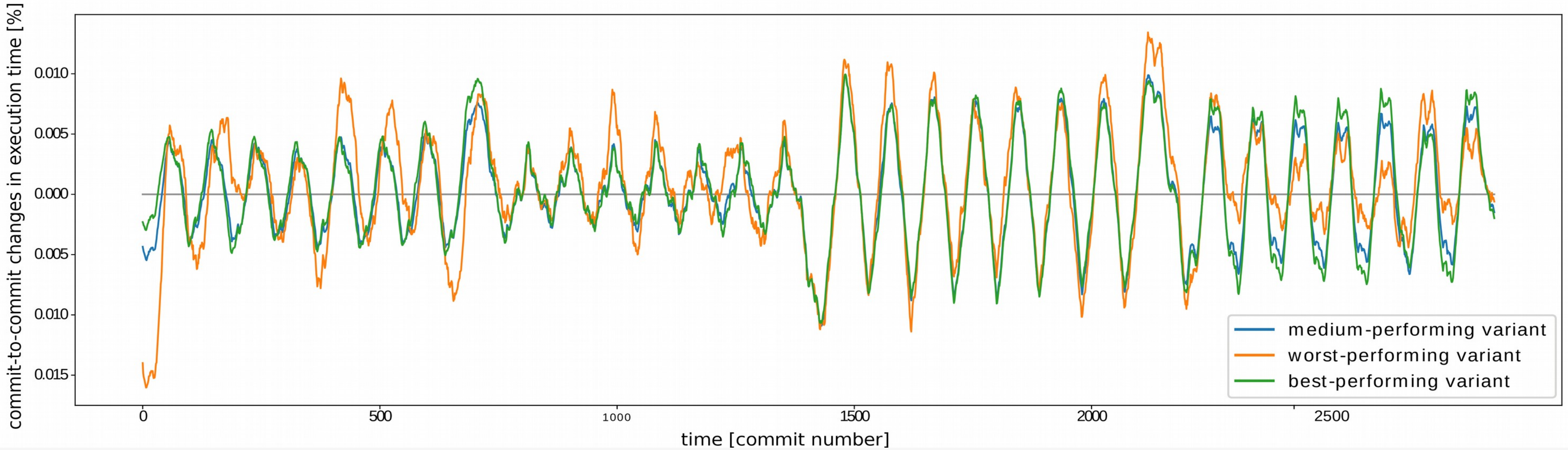
- RQ: Does performance evolve for configurable software systems?
- GNU XZ: file compression tool
  - configurable at load-time, 9B/4N features
  - 36 variants sampled, $\sim$ 1.100 versions assessed
- X264: video encoder
  - configurable at load-time, 8B/12N features
  - 8 variants sampled, $\sim$ 2.800 versions assessed

Revision Assessment

- What revisions induce significant performance changes?
- Case study suggests 2 sampling strategies:
  - Revision size: more code revised → more likely performance changes
  - Significant files: certain files revised → likely impact on performance

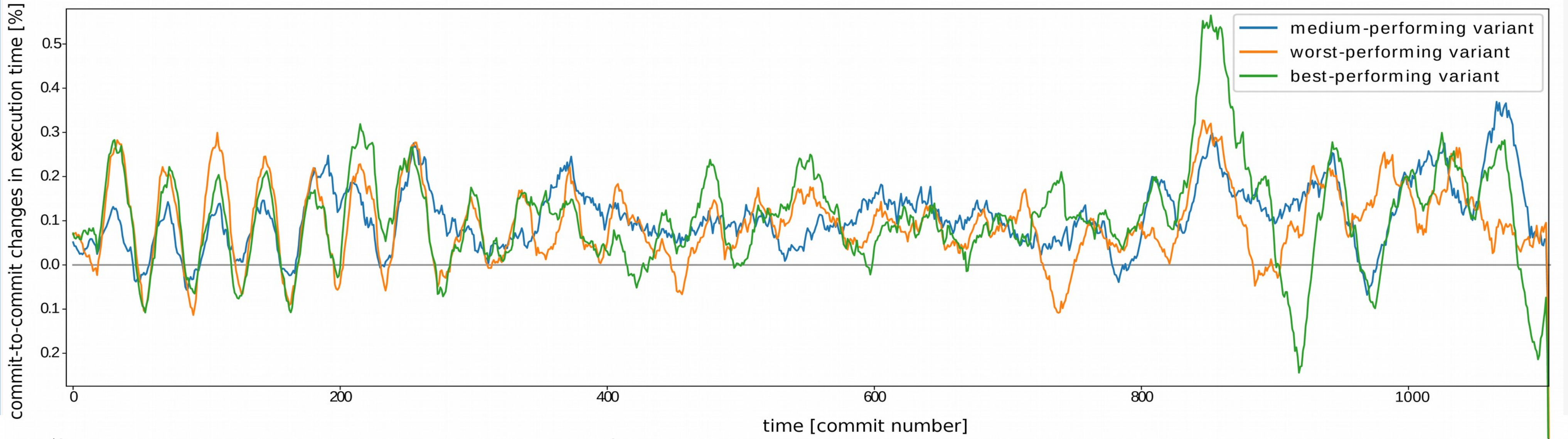


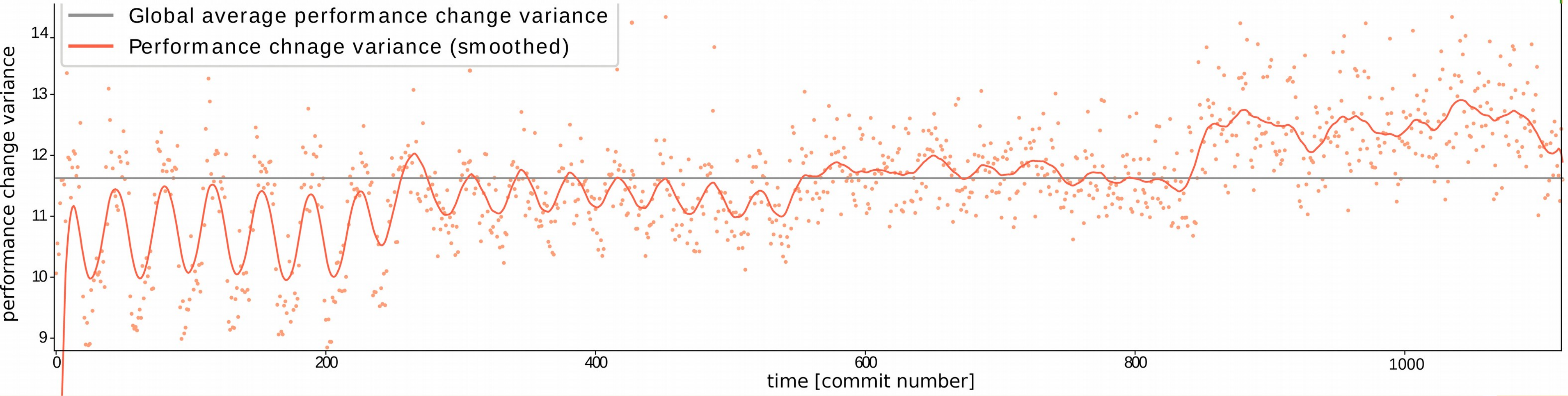
Performance evolution: x264



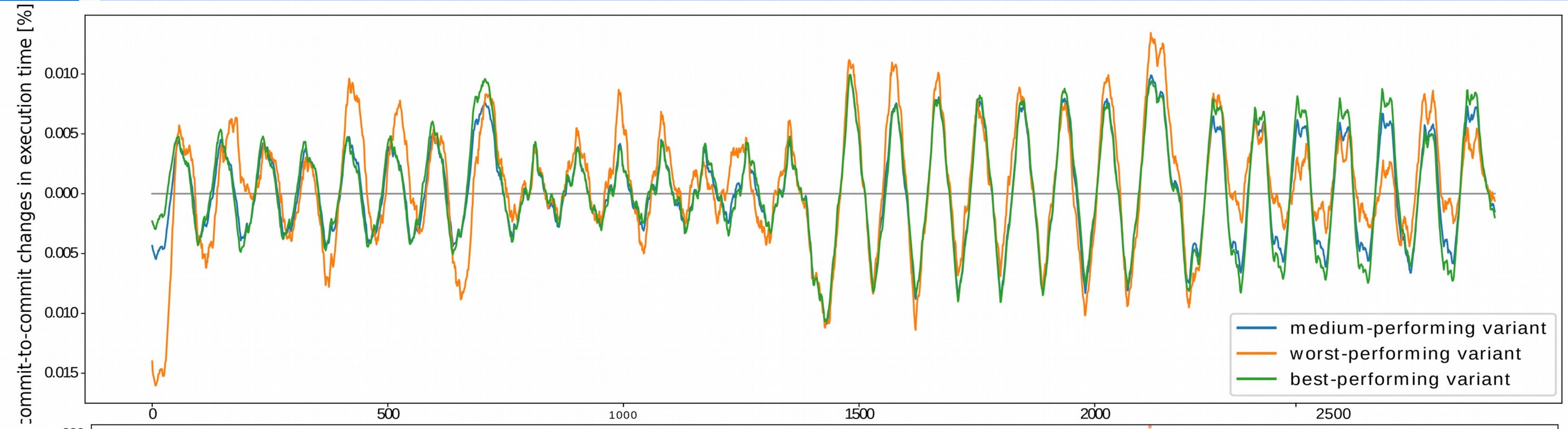
- Effect magnitude: oscillations throughout all versions
  - small range of performance changes: - 0.015 % to 0.01 %
- Effect range: homogeneous evolution throughout all versions
  - X264 appears to be more mature than GNU XZ

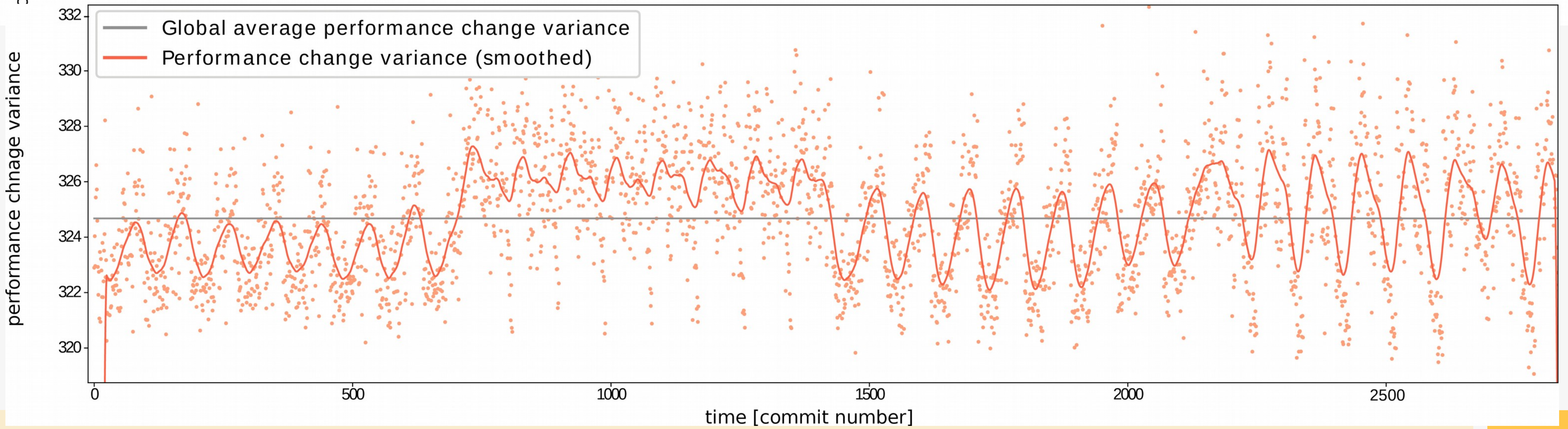
GNU XZ: effect magnitude and range





X264: effect magnitude and range





Methodology overview

- Variability Assessment
  - automated vs. manual assessment
- Performance Assessment
  - performance metrics,
  - benchmarks,
  - profiler
  - robust measures, effect magnitude and range
- Revision Assessment
  - Sampling strategy: largest revisions or learn significant code units

Conclusion & Future Work

- Methodology description for configurable software systems
- Feasibility evaluation with a minor case study (2 systems)

- Insights obtained: performance evolution history
  - Possible indicator for software maturity and quality

- Use cases and further directions:
  - Performance prediction for future revisions and variants

Resources

- Articles
  - https://www.forbes.com/sites/ewanspence/2017/12/20/apple-iphone-kill-switch-ios-degrade-cripple-performance-battery/
  - https://arstechnica.com/gadgets/2018/01/heres-how-and-why-the-spectre-and-meltdown-patches-will-hurt-performance/
  - https://www.heise.de/newsticker/meldung/Intel-Benchmarks-zu-Meltdown-Spectre-Performance-sackt-um-bis-zu-10-Prozent-ab-SSD-I-O-deutlich-mehr-3938747.html
  - http://bgr.com/2017/12/28/apple-batterygate-explainer-why-iphones-slow-down/