Efficient Performance Modeling for Evolving Software

FOSD Meeting 2019

Stefan Mühlbauer
March 11, 2019
Poor Software Performance
Poor Software Performance
Poor Software Performance
Poor Software Performance

Has anyone ever experienced this?
Poor Software Evolution

Architectural drift:
Insensitivty about and violations of original architecture
Poor Software Evolution

Architectural drift:
Insensitivity about and violations of original architecture

Architectural erosion:
Missing coherence and adaptability makes software brittle (technical debt)
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific
  [Han and Yu, 2016]
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)
Performance in the Presence of Variability

- Performance bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)

r957: liblzma: Add lzma_memcmplen() for fast memory comparison. This commit just adds the function. Its uses will be in separate commits.
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)

r957: liblzma: Add lzma_memcmplen() for fast memory comparison. This commit just adds the function. Its uses will be in separate commits.

r958: liblzma: Use lzma_memcmplen() in the match finders.
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

lrzip

GNU XZ
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264
Irzip
GNU XZ
NumPy
SciPy
astropy
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264
NumPy
SciPy
Irzip
GNU XZ
astropy

Efficient Performance Modeling for Evolving Software | Stefan Mühlbauer
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

- x264
- Irzip
- GNU XZ
- NumPy
- SciPy
- astropy

Efficient Performance Modeling for Evolving Software | Stefan Mühlbauer
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

- x264
- lrzip
- GNU XZ
- NumPy
- SciPy
- astropy
Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

- x264
- NumPy
- SciPy
- GNU XZ
- astropy
RQ$_1$: What are characteristics of performance evolution?

Reasons to (de-)select prediction models
RQ$_1$: What are characteristics of performance evolution?

Reasons to (de-)select prediction models

- Time-series characteristics?
  - Stationarity
  - Trends
  - Change-points

![Graph showing execution time against revision with categories]

- effi cient Performance Modeling for Evolving Software | Stefan Mühlbauer
RQ₁: What are characteristics of performance evolution?

Reasons to (de-)select prediction models

- Time-series characteristics?
  - Stationarity
  - Trends
  - Change-points

- On what scales do patterns exist?
  - Release-to-release version
  - Feature-model iterations
  - Merges, Bugfixes, …

![Graph showing execution time over revisions categorised as stationary, trendy, and disruptive.]
RQ₂: Can we model performance evolution efficiently?

Disruptive changes are hard to pinpoint, can we search for them?
RQ$_2$: Can we model performance evolution efficiently?

Disruptive changes are hard to pinpoint, can we search for them?

- Gaussian Process Regression for time-series [Roberts et al., 2012]
- Framework for actively learning time-series when obtaining samples is expensive
RQ₂: Adaptive Learning with Gaussian Processes (GP)

(1) Initial random sampling of $k$ revisions.
(2) Training of GP regressor with sample set (MLE).
(3) Prediction
RQ₂: Adaptive Learning with Gaussian Processes (GP)

(1) Initial random sampling of $k$ revisions.
(2) Training of GP regressor with sample set (MLE).
(3) Prediction
(4) Acquisition of next sample with maximum prediction variance
RQ₂: Adaptive Learning with Gaussian Processes (GP)

(1) Initial random sampling of $k$ revisions.

(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met
RQ₂: Adaptive Learning with Gaussian Processes (GP)

(1) Initial random sampling of $k$ revisions.
(2) Training of GP regressor with sample set (MLE).
(3) Prediction
(4) Acquisition of next sample with maximum prediction variance
(5) Repeat (2-4) until termination criteria is met
**RQ₂: Adaptive Learning with Gaussian Processes (GP)**

1. Initial random sampling of \( k \) revisions.
2. Training of GP regressor with sample set (MLE).
3. Prediction
4. Acquisition of next sample with maximum prediction variance
5. Repeat (2-4) until termination criteria is met
RQ₂: Adaptive Learning with Gaussian Processes (GP)

(1) Initial random sampling of $k$ revisions.
(2) Training of GP regressor with sample set (MLE).
(3) Prediction
(4) Acquisition of next sample with maximum prediction variance
(5) Repeat (2-4) until termination criteria is met
RQ$_2$: Adaptive Learning with Gaussian Processes (GP)

1. Initial random sampling of $k$ revisions.
2. Training of GP regressor with sample set (MLE).
3. Prediction
4. Acquisition of next sample with maximum prediction variance
5. Repeat (2-4) until termination criteria is met
RQ₂: Adaptive Learning with Gaussian Processes (GP)

(1) Initial random sampling of $k$ revisions.
(2) Training of GP regressor with sample set (MLE).
(3) Prediction
(4) Acquisition of next sample with maximum prediction variance
(5) Repeat (2-4) until termination criteria is met
RQ₂: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline
RQ$_2$: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline

- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold
RQ₂: Evaluation Plan

■ Which acquisition strategy minimizes prediction error?
  ■ Uncertainty-aware [Roberts et al., 2012]
  ■ Bisection/binary search [Heger et al., 2013]
  ■ Random/uniform sampling as baseline

■ Termination criteria: How robust is adaptive learning?
  ■ Uniform prediction variance threshold
RQ$_2$: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline

- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold
  - Minimum variance-change threshold
RQ$_2$: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline

- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold
  - Minimum variance-change threshold
Summary

- performance evolves heterogeneously
- classification of performance-changes
- active-learning and estimation of performance evolution
Thank you for your kind attention!

Any questions or suggestions?
References

**Han, X. and Yu, T. (2016).**

**Heger, C., Happe, J., and Farahbod, R. (2013).**

Gaussian processes for time-series modelling.
Backup: Performance Assessment

¯\_(ツ)_/¯
Backup: Case Study Metrics