OSNABRÜCK UNIVERSITY

Embedded Software Systems

Understanding Product Line Runtime Performance with Behaviour Models and Regression Model Trees

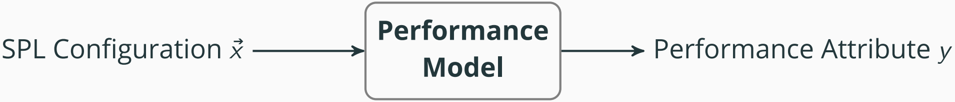
Birte Friesel, Olaf Spinczyk

September 4th, 2025

`ess.cs.uos.de/~bf`

birte.friesel@uos.de

The State of Runtime Performance Models

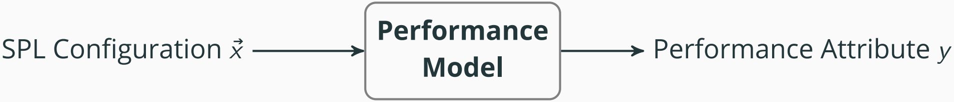


The State of Runtime Performance Models

SPL Configuration $\vec{x}$ → **Performance Model** → Performance Attribute y

- x264 video encoder [Zha+15; Guo+18; Sie+15; Sie+13; DAS21]
  - runtime flags → latency, output file size
- Database management systems systems [Guo+13; Sar+15; Nai+17; Per+21]
  - Static features → latency, throughput, . . .

The State of Runtime Performance Models



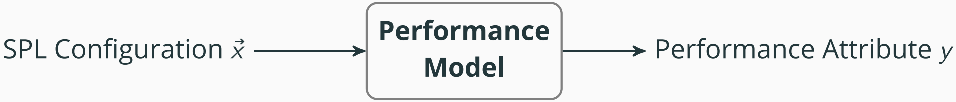
- x264 video encoder [Zha+15; Guo+18; Sie+15; Sie+13; DAS21]
  - runtime flags → latency, output file size of fixed input file
- Database management systems systems [Guo+13; Sar+15; Nai+17; Per+21]
  - Static features → latency, throughput, . . . of fixed reference query

The State of Runtime Performance Models

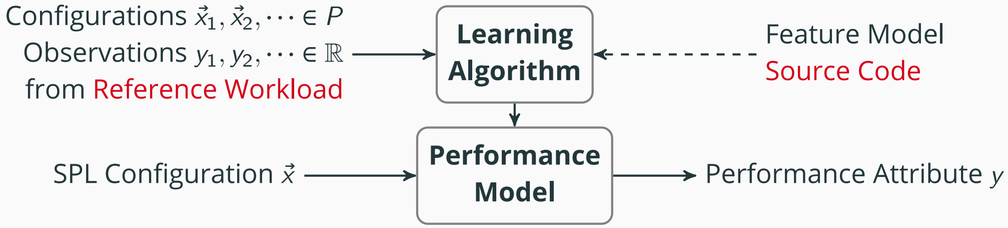
SPL Configuration $\vec{x}$ → **Performance Model** → Performance Attribute y

- x264 video encoder [Zha+15; Guo+18; Sie+15; Sie+13; DAS21]
  - runtime flags → latency, output file size of fixed input file
  - Input file length, resolution $\overset{?}{\to}$ latency, output file size
- Database management systems systems [Guo+13; Sar+15; Nai+17; Per+21]
  - Static features → latency, throughput, . . . of fixed reference query
  - Database size, query sequence $\overset{?}{\to}$ latency, throughput, . . .

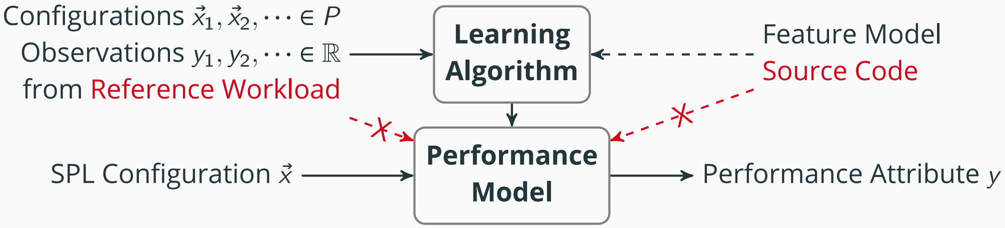
Performance Models: Workload as a Black Box



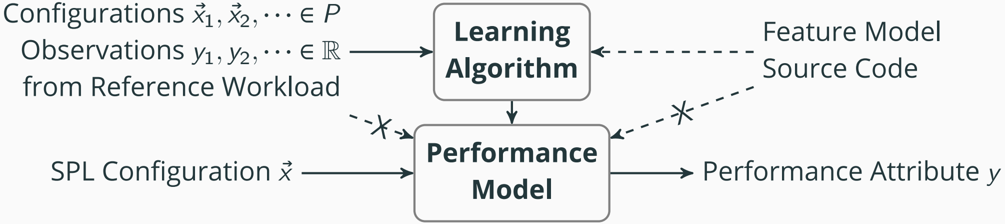
Performance Models: Workload as a Black Box



Performance Models: Workload as a Black Box

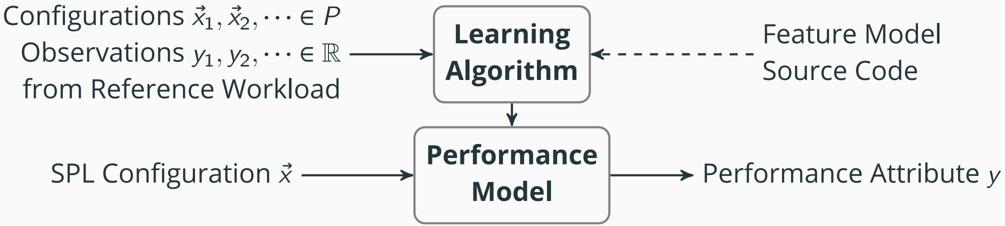


Performance Models: Workload as a Black Box



- Workload changes → re-run benchmarks and re-build model
- Performance bottlenecks → no link to workload / source code

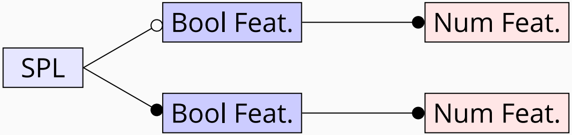
Performance Models: Workload as a Black Box



- Workload changes → re-run benchmarks and re-build model
- Performance bottlenecks → no link to workload / source code
- Proposal: Workload-aware and interpretable performance models

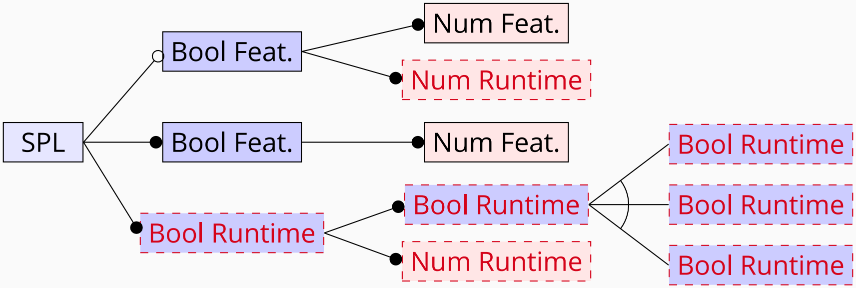
→ ① runtime variability, ② workload model, ③ performance annotations

① Runtime Variability Model



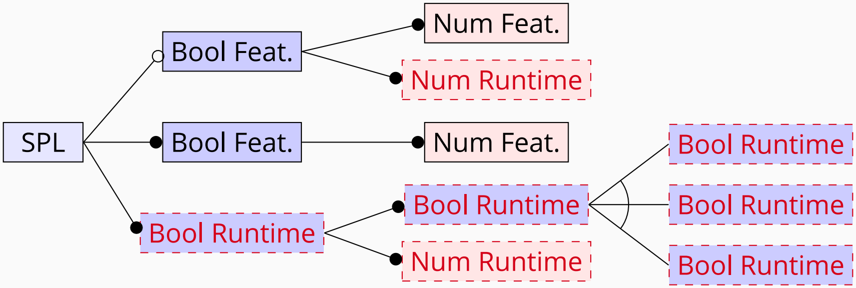
- Feature model: static features only, unaware of runtime variability

① Runtime Variability Model



- Feature model + runtime-only variability (e.g. input file length, table size)

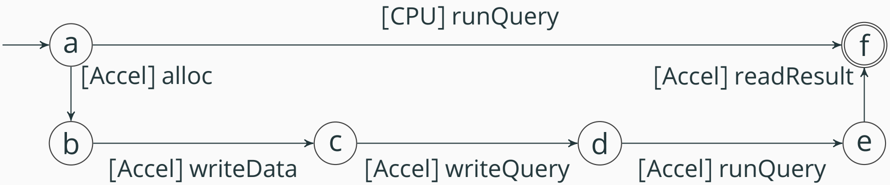
① Runtime Variability Model



- Feature model + runtime-only variability (e.g. input file length, table size)
- Extension of Dynamic Software Product Lines (DSPLs) [Hal+08]
  - Compile-time defaults can be changed at runtime
  - DSPLs: no support for runtime variability ∉ product line features

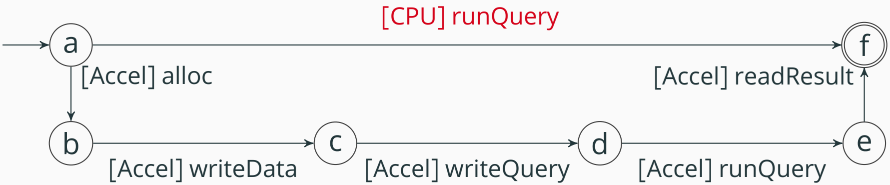
② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)



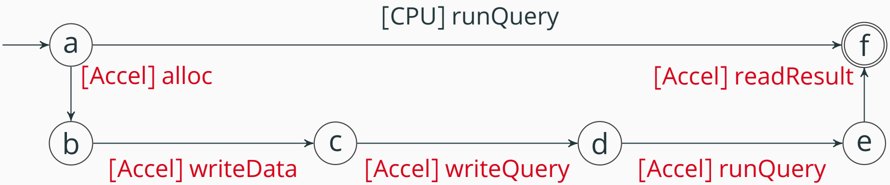
② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)



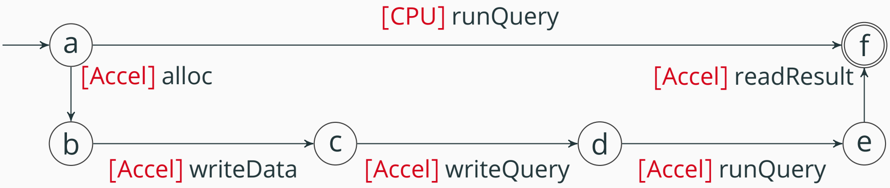
② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)



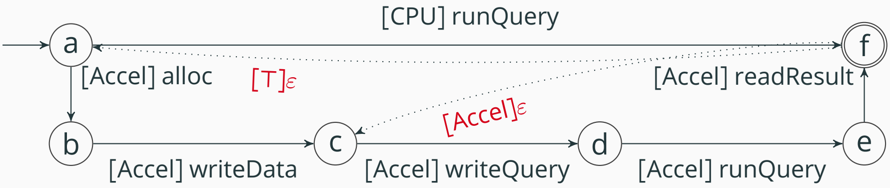
② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)
  - State machine; transitions ≙ runtime steps
  - Feature guards: transitions may depend on feature configuration



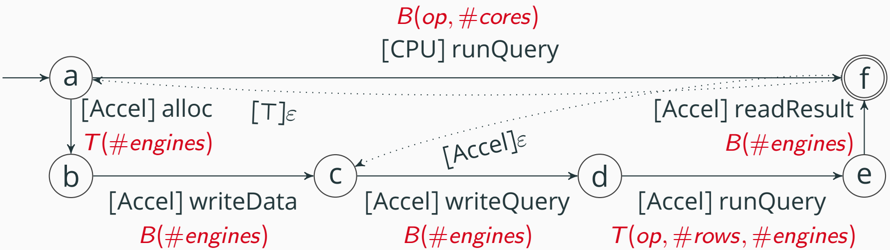
② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)
  - State machine; transitions ≙ runtime steps or loops (consecutive queries)
  - Feature guards: transitions may depend on feature configuration



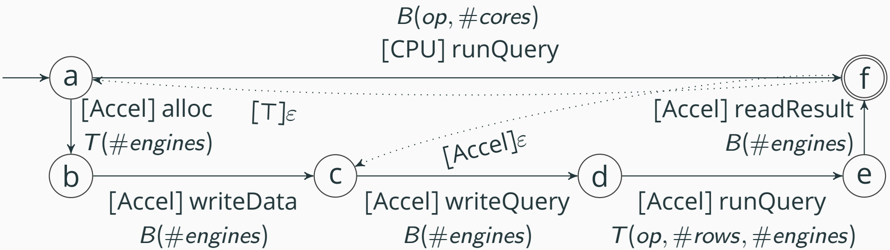
② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)
  - State machine; transitions ≙ runtime steps or loops (consecutive queries)
  - Feature guards: transitions may depend on feature configuration
  - Transitions annotated with performance models (bandwidth or latency)

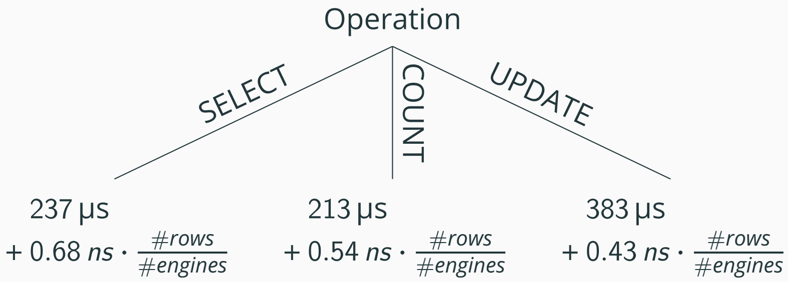


② Workload Model

- Example: DBMS with optional offloading engines (query accelerators)
  - State machine; transitions ≙ runtime steps or loops (consecutive queries)
  - Feature guards: transitions may depend on feature configuration
  - Transitions annotated with performance models (bandwidth or latency)
- Extension of featured transition systems [AFL15; Cla+13; Cla+14]



③ Regression Model Trees

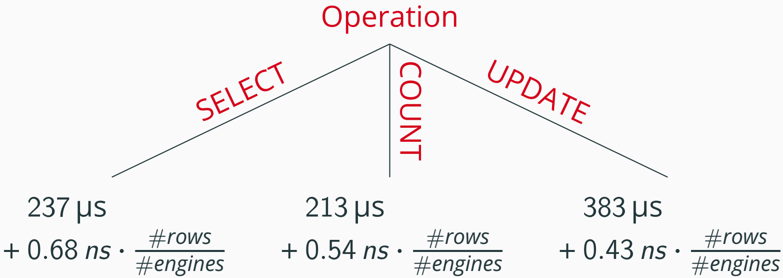


Regression model trees [FS22]

= regression trees [Bre+84]

\+ unsupervised least-squares [FBS18]

③ Regression Model Trees

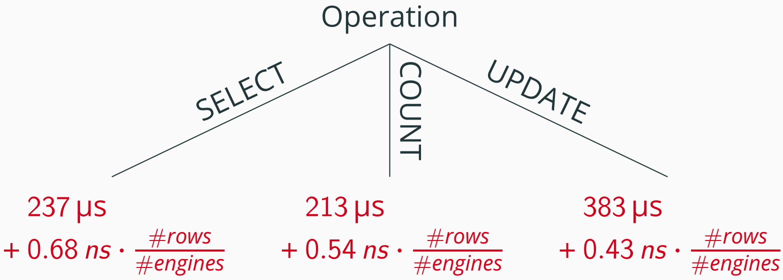


Regression model trees [FS22]

= regression trees [Bre+84]

+ unsupervised least-squares [FBS18]

③ Regression Model Trees

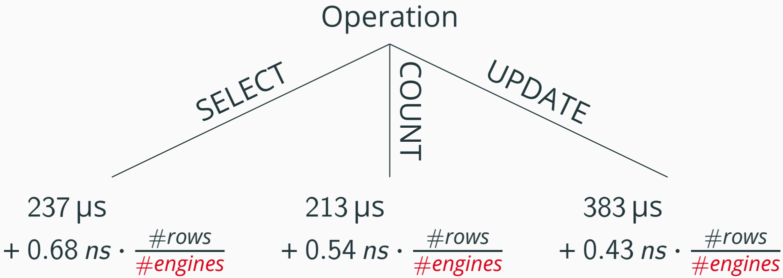


Regression model trees [FS22]

= regression trees [Bre+84]

\+ unsupervised least-squares [FBS18]

③ Regression Model Trees



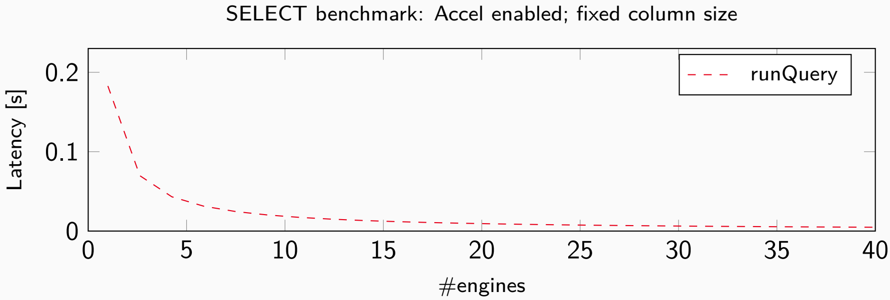
Regression model trees [FS22]

= regression trees [Bre+84]

+ unsupervised least-squares [FBS18]

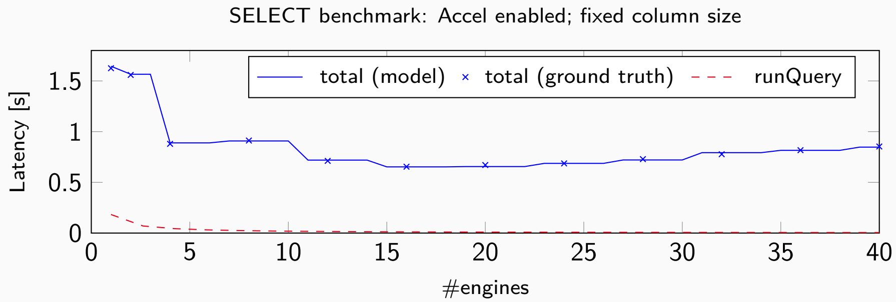
- Accurate and interpretable
- runQuery example: linear scaling with # accelerator engines

Understanding Product Line Behaviour



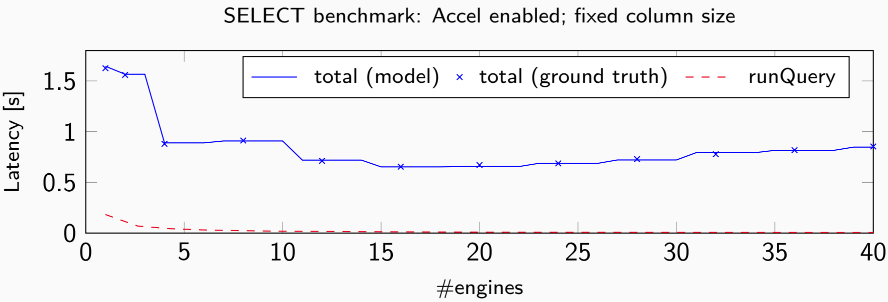
- runQuery scales linearly with #engines

Understanding Product Line Behaviour



- Reference benchmark does not scale linearly with #engines

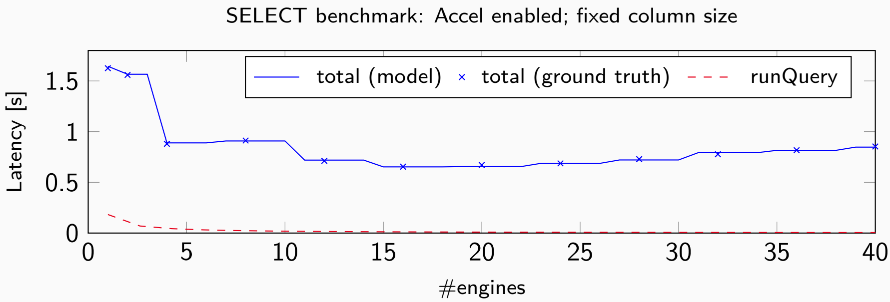
Understanding Product Line Behaviour



- Reference benchmark does not scale linearly with #engines
- Neither minimum nor maximum are optimal

→ Why?

Understanding Product Line Behaviour



- Reference benchmark does not scale linearly with #engines
- Neither minimum nor maximum are optimal

→ Why? (explanation in the paper)

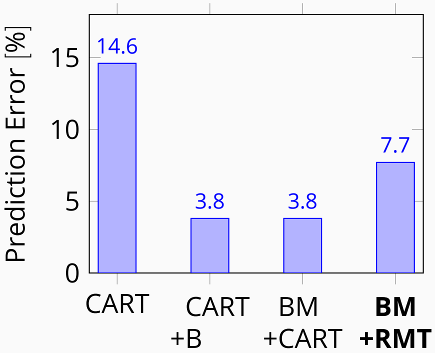
Quantitative Evaluation

- Case study: DBMS with query accelerators
- Four models:
  - CART: conventional performance model
  - CART+B: ① CART with runtime variability
  - BM+CART: ① ② behaviour model with CART annotations
  - BM+RMT: ① ② ③ behaviour model with regression model trees

Quantitative Evaluation

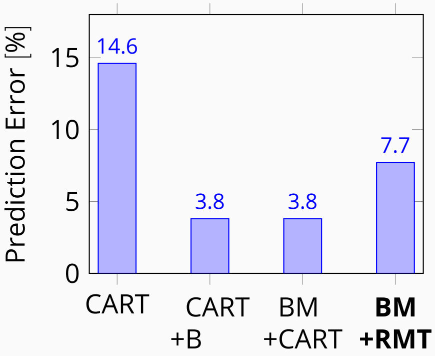
- Case study: DBMS with query accelerators
- Four models:
  - CART: conventional performance model
  - CART+B: ① CART with runtime variability
  - BM+CART: ① ② behaviour model with CART annotations
  - BM+RMT: ① ② ③ behaviour model with regression model trees
- Evaluation metrics:
  - Latency prediction error: variable configuration and query sequences (10-fold cross validation)
  - Model complexity (# tree nodes + # regression weights)

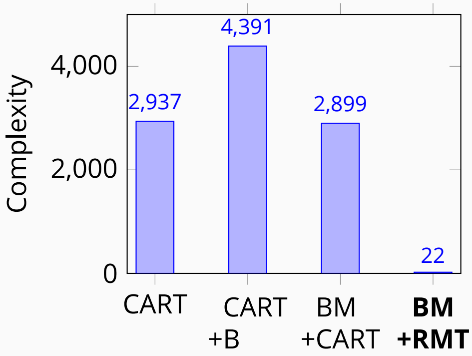
Evaluation Results



⇒ Sufficient accuracy for reasoning about runtime performance

Evaluation Results



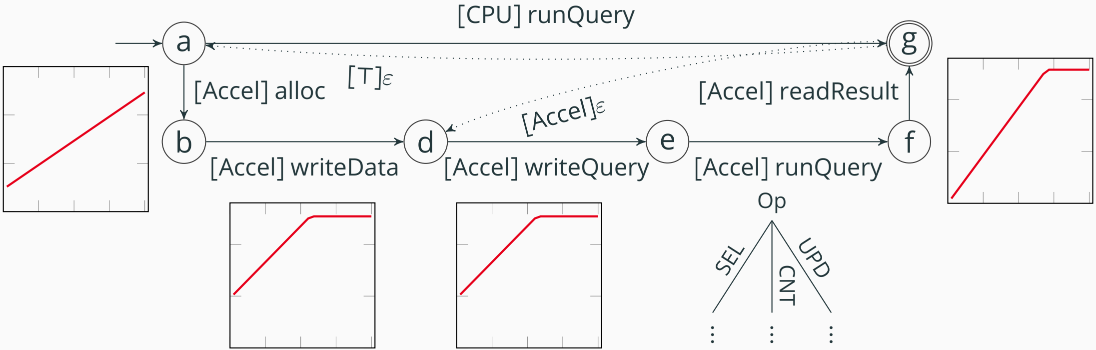


⇒ Sufficient accuracy for reasoning about runtime performance

⇒ Two orders of magnitude lower complexity → interpretable models

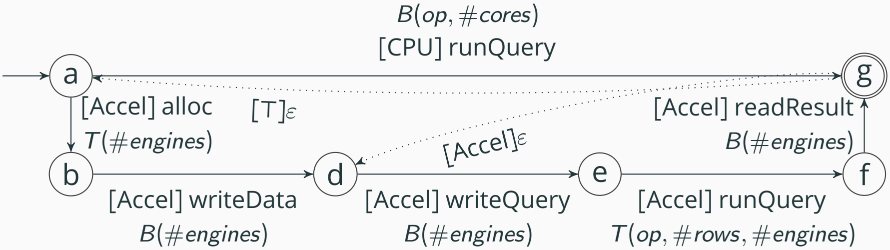
Conclusion

- Behaviour Models and Regression Model Trees:
  flexible, interpretable, workload-independent performance models



Conclusion

- Behaviour Models and Regression Model Trees:
  flexible, interpretable, workload-independent performance models
  → Understanding performance issues and bottlenecks
  → Predicting runtime performance of arbitrary workloads



Conclusion

- Behaviour Models and Regression Model Trees:
  flexible, interpretable, workload-independent performance models
  - → Understanding performance issues and bottlenecks
  - → Predicting runtime performance of arbitrary workloads
- Definitions, algorithms and case study in the paper
  - Case study: DBMS application on real query accelerators
  - Artifacts available and functional: `zenodo.org/records/15827230` [Fri25]

Conclusion

- Behaviour Models and Regression Model Trees:
  flexible, interpretable, workload-independent performance models
  → Understanding performance issues and bottlenecks
  → Predicting runtime performance of arbitrary workloads
- Definitions, algorithms and case study in the paper
  – Case study: DBMS application on real query accelerators
  – Artifacts available and functional: `zenodo.org/records/15827230` [Fri25]
- Learning behaviour models from application traces:
  work in progress; proof of concept to appear @ CCMCC'25 [FS25]

References i

[AFL15] Joanne M. Atlee, Uli Fahrenberg, and Axel Legay. **“Measuring Behaviour Interactions between Product-Line Features”.** In: Proceedings of the 3rd FME Workshop on Formal Methods in Software Engineering. FormaliSE ’15. Florence, Italy: IEEE, May 2015, pp. 20–25. DOI: 10.1109/FormaliSE.2015.11.

[Bre+84] Leo Breiman et al. **Classification and Regression Trees.** 1st ed. Routledge, 1984. ISBN: 978-1-3151-3947-0. DOI: 10.1201/9781315139470.

[Cla+13] Andreas Classen et al. **“Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking”.** In: IEEE Transactions on Software Engineering 39.8 (2013), pp. 1069–1089. DOI: 10.1109/TSE.2012.86.

References ii

[Cla+14] Andreas Classen et al. **“Formal semantics, modular specification, and symbolic verification of product-line behaviour”.** In: Science of Computer Programming 80.PB (Feb. 2014), pp. 416–439. ISSN: 0167-6423. DOI: `10.5555/2748144.2748397`.

[DAS21] Johannes Dorn, Sven Apel, and Norbert Siegmund. **“Mastering Uncertainty in Performance Estimations of Configurable Software Systems”.** In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. ASE ’20. Melbourne, Australia: Association for Computing Machinery, Sept. 2021, pp. 684–696. ISBN: 978-1-4503-6768-4. DOI: `10.1145/3324884.3416620`.

References iii

[FBS18] Birte Friesel, Markus Buschhoff, and Olaf Spinczyk. **“Parameter-Aware Energy Models for Embedded-System Peripherals”.** In: Proceedings of the 13th International Symposium on Industrial Embedded Systems. SIES ’18. Graz, Austria: IEEE, June 2018. DOI: 10.1109/SIES.2018.8442096.

[Fri25] Birte Friesel. **Understanding Product Line Runtime Performance with Behaviour Models and Regression Model Trees (Artefact).** 2025. DOI: https://doi.org/10.5281/zenodo.15827230.

References iv

[FS22] Birte Friesel and Olaf Spinczyk. **“Regression Model Trees: Compact Energy Models for Complex IoT Devices”.** In: Proceedings of the Workshop on Benchmarking Cyber-Physical Systems and Internet of Things. CPS-IoTBench ’22. Milan, Italy: IEEE, May 2022, pp. 1–6. DOI: `10.1109/CPS-IoTBench56135.2022.00007`.

[FS25] Birte Friesel and Olaf Spinczyk. **“Overhead Prediction for PIM-Enabled Applications with Performance-Aware Behaviour Models”.** In: Proceedings of the 1st IEEE Cross-disciplinary Conference on Memory-Centric Computing. CCMCC ’25. to appear. Dresden, Germany, Oct. 2025.

References v

[Guo+13] Jianmei Guo et al. **"Variability-Aware Performance Prediction: A Statistical Learning Approach".** In: Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering. ASE '13. IEEE, 2013, pp. 301–311. DOI: `10.1109/ASE.2013.6693089`.

[Guo+18] Jianmei Guo et al. **"Data-Efficient Performance Learning for Configurable Systems".** In: Empirical Software Engineering 23.3 (June 2018), pp. 1826–1867. ISSN: 1382-3256. DOI: `10.1007/s10664-017-9573-6`.

[Hal+08] Svein Hallsteinsen et al. **"Dynamic Software Product Lines".** In: Computer 41.4 (2008), pp. 93–95. DOI: `10.1109/MC.2008.123`.

References vi

[Nai+17] Vivek Nair et al. **"Using bad learners to find good configurations".** In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ESEC/FSE 2017. Paderborn, Germany: Association for Computing Machinery, 2017, pp. 257–267. ISBN: 9781450351058. DOI: `10.1145/3106237.3106238`.

[Per+21] Juliana Alves Pereira et al. **"Learning software configuration spaces: A systematic literature review".** In: Journal of Systems and Software 182 (2021), p. 111044. ISSN: 0164-1212. DOI: `https://doi.org/10.1016/j.jss.2021.111044`. URL: `https://www.sciencedirect.com/science/article/pii/S0164121221001412`.

[Sar+15] Atrisha Sarkar et al. **"Cost-Efficient Sampling for Performance Prediction of Configurable Systems".** In: Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). ASE '15. IEEE, 2015, pp. 342–352. DOI: 10.1109/ASE.2015.45.

[Sie+13] Norbert Siegmund et al. **"Scalable prediction of non-functional properties in software product lines: Footprint and memory consumption".** In: Information and Software Technology 55.3 (Mar. 2013), pp. 491–507. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2012.07.020.

References viii

[Sie+15] Norbert Siegmund et al. **“Performance-Influence Models for Highly Configurable Systems”.** In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering. ESEC/FSE ’15. Bergamo, Italy: Association for Computing Machinery, Aug. 2015, pp. 284–294. ISBN: 978-1-4503-3675-8. DOI: 10.1145/2786805.2786845.

[Zha+15] Yi Zhang et al. **“Performance Prediction of Configurable Software Systems by Fourier Learning”.** In: Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering. ASE ’15. Lincoln, NE, USA: IEEE, Nov. 2015, pp. 365–373. DOI: 10.1109/ASE.2015.15.