Pacific Northwest NATIONAL LABORATORY

QuaLM: Learning Quantitative Performance of Latency-Sensitive Code

IWAPT 2022, 06/03/2022

Presenter: Arun V Sathanur

Senior Scientist, Data Sciences,

Pacific Northwest National Laboratory (PNNL)

Co-authors:

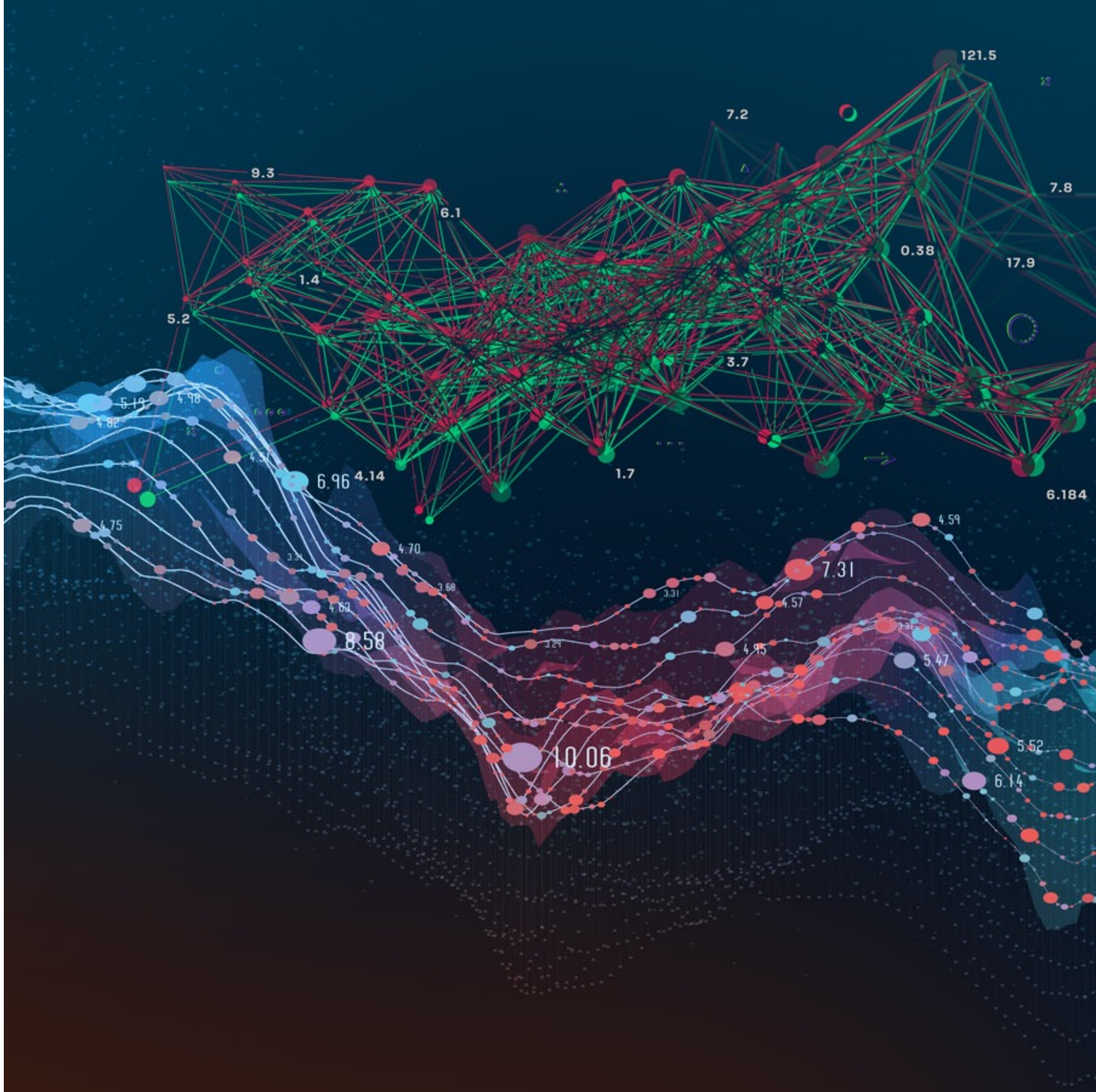
Nathan Tallent (PNNL),

Patrick Konsor, Ken Koyanagi, Ryan Mclaughlin (Intel)

Joseph Olivas, Michael Chynoweth (Intel)

U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy



Introduction

- Diagnosing bottlenecks in CPUs and hence optimizing applications for a given microarchitecture is becoming increasingly challenging

- Good predictive models can lead to more effective performance introspection
  - Better JIT and runtime decisions
  - Informed application steering

- In this work we consider the problem of learning to predict the performance of an application on Intel Skylake CPU and other emerging architectures
  - Resolution: dynamic superblock (dynamic instructions between two branches)

Related work

- Best current methods are based on top-down analysis and cycle accounting [1]. Useful but,
  - Time-consuming for human to construct
  - Possibly misleading: assumes penalties do not overlap – may be false

- Recent approaches in this direction include deep neural network models such as Long Short-Term Memory (LSTM) units [2] and Graph Neural Networks (GNN) [3].

- Unrealistic assumptions:
  - Throughput performance (best case instruction parallelism) [2]
  - Data is L1-resident (fastest cache) [2] or data layout is irrelevant [3]
  - Ignores pipeline stalls/hazards (TLB walks, store blocking, memory fences) [2,3]
  - Basic block only [2]

[1] Yasin A. A top-down method for performance analysis and counters architecture. In2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) 2014
[2] Mendis C, Renda A, Amarasinghe S, Carbin M. Ithemal: Accurate, portable and fast basic block throughput estimation using deep neural networks. ICML 2019
[3] Shi Z, Swersky K, Tarlow D, Ranganathan P, Hashemi M. Learning Execution through Neural Code Fusion. ICLR, 2020

Our approach and contributions

- Learn bottlenecks through combination static and dynamic sources.
- Static features (MCA): static dependencies, instruction scheduling limitations
  - LLVM Machine Code Analyzer
- Dynamic features (PMU): execution stalls/penalties via perf. monitoring units
  - Leverage *precise* timings of super-block execution (new feature on Intel PMUs)
- Modeling lessons for performance-based data sets
  - wide variability and heavy tails
- Compare against predictions from the state-of-the-art top-down methodology
- Open-source implementation of QuaLM's modeling pipeline

https://gitlab.pnnl.gov/perf-lab/qualm

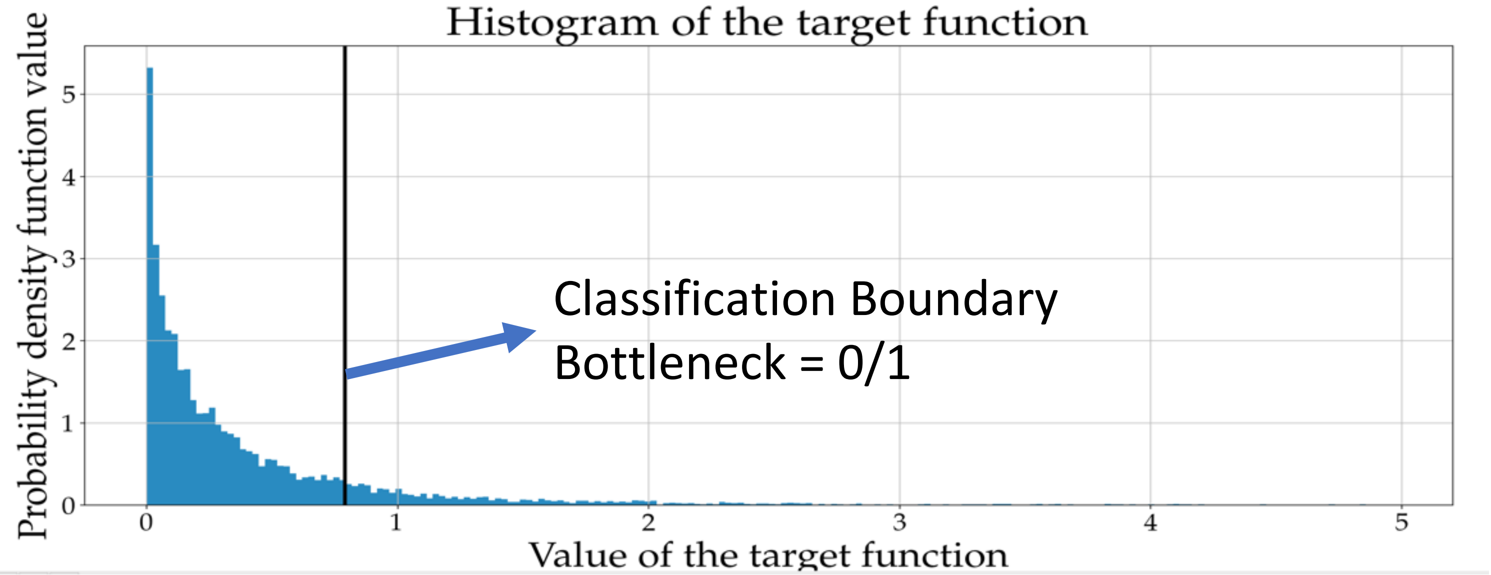
Model description

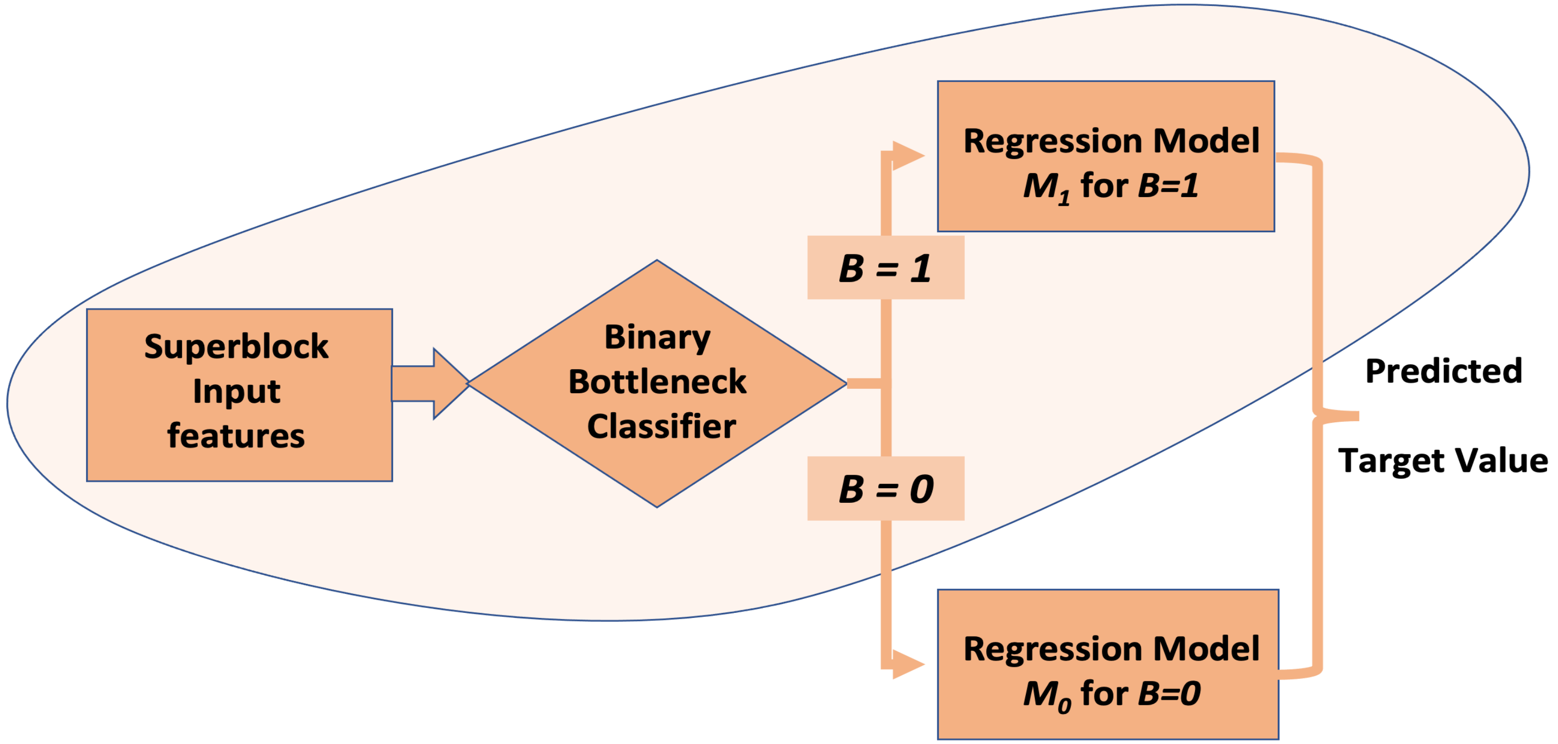
A 2-stage model combining a classifier and a regression model

Classifier and Regressor can be implemented by any model

Current implementations comprise Linear Models, Ensemble Decision Trees, and Fully Connected Neural Networks

We predict both Mean CPI and Extra CPI;





Model Performance

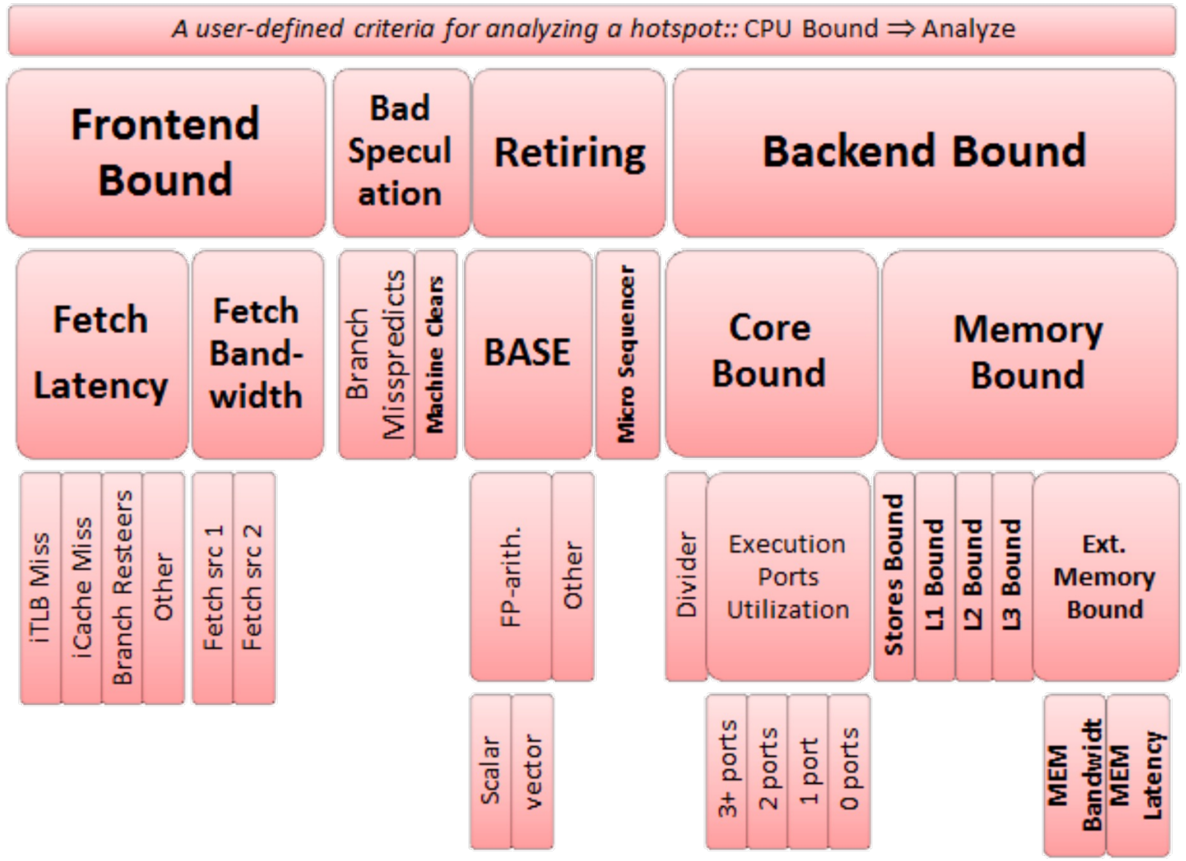
- Ensemble Decision Trees perform better than linear models and deep learning models
- Significantly improve prediction accuracy vs. state-of-the-art Top Down (TD) modeling

Datasets	Best Model	R^2	RMSE
Sm - Uniq	RF / XGB	0.70	0.71
Lg - Uniq	RF / RF	0.66	2.50
Lg - Mult	XGB /XGB	0.84	1.79

TD predictor: linear combination of events

TD.Shallow: R^2 of 0.39 (Large-Multiple)

TD.Deep: R^2 of 0.37 (Large-Multiple)



Top-down methodology from [1]

[1] Yasin A. A top-down method for performance analysis and counters architecture. In2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) 2014

Dataset description

- End-user commercial apps: latency-sensitive
  - web browsers, word processors, spreadsheets, audio/video, development environment
- Two methods: Benchmark suites, Long term (6 months)
- Three different data sets:
  - ***Small-Unique***: Benchmark method. Each superblock has ≥1000 LBR samples for very high-quality data.
  - ***Large-Unique***: Long-term method. Each superblock has ≥100 samples.
  - ***Large-Multi***: Long-term method. Same method as Lg-Uniq, except that superblock data is retained as distinct over all collection windows.

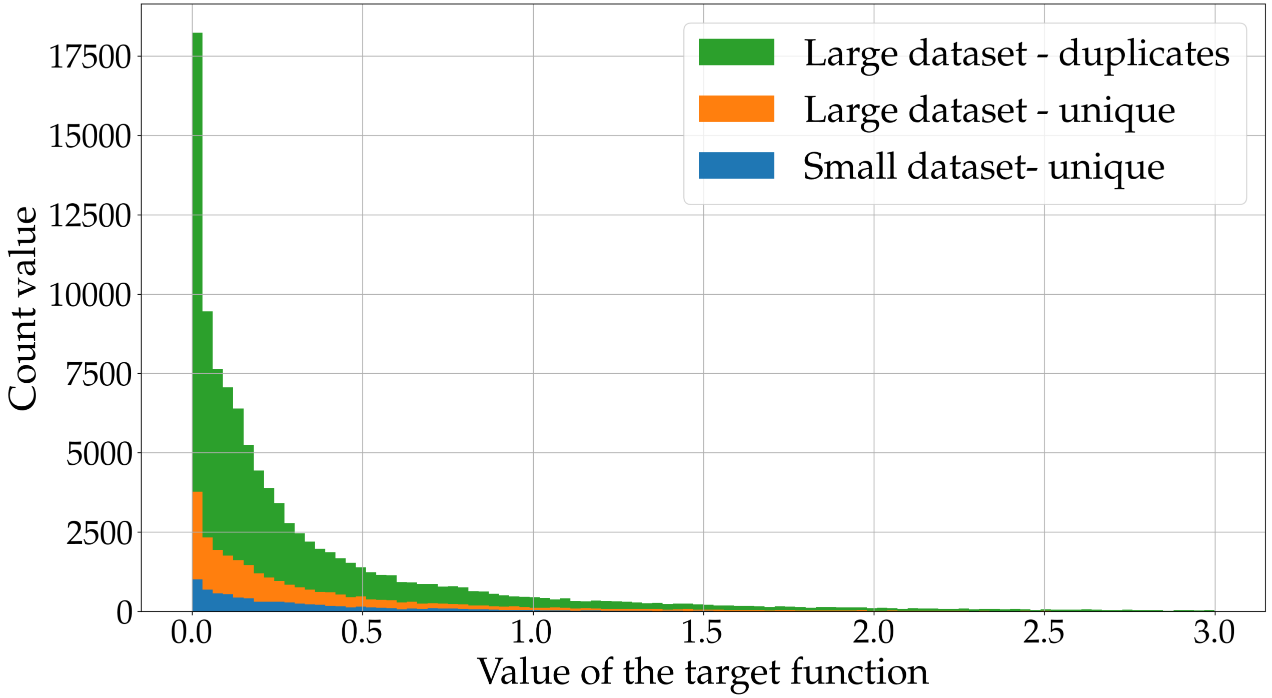
	Initial			*Pruned*		
Dataset	S-blocks	PMU	MCA	S-blocks	PMU	MCA
Sm-Uniq	12k	147	25	10k	23	21
Lg-Uniq	44k	147	25	31k	23	21
Lg-Multi	149k	147	25	122k	23	21

Statistical description of the data

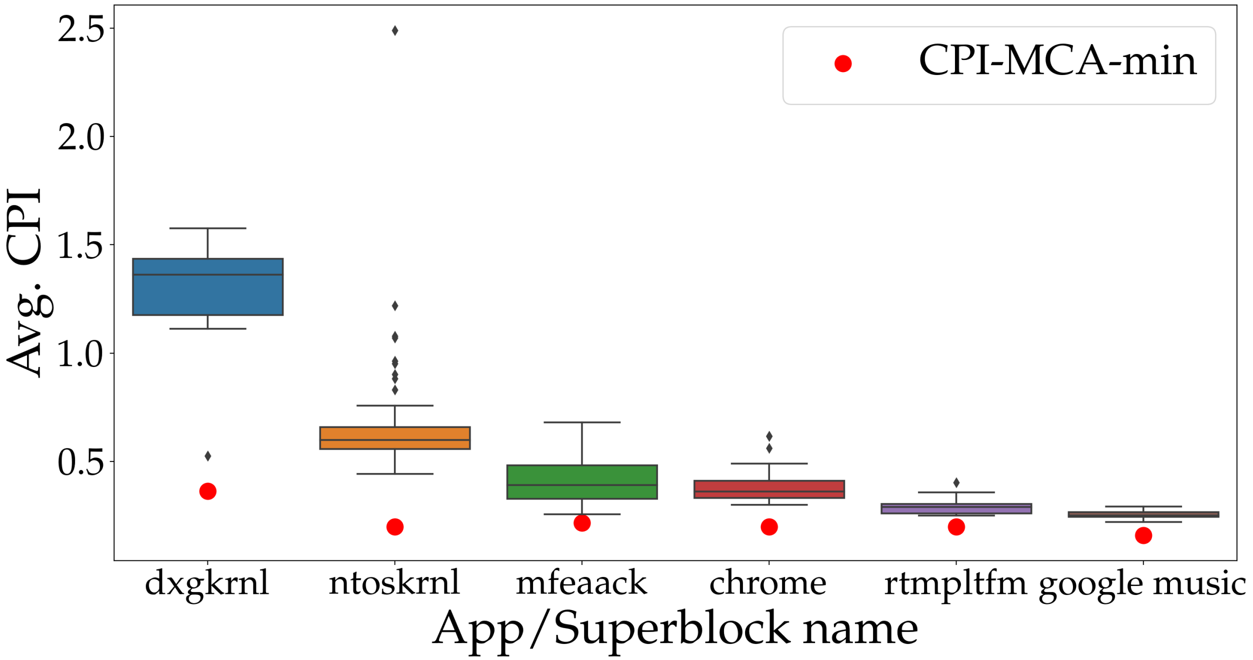
Basic stats for Extra CPI and Mean CPI

Dataset	Min.	Med.	Max.	Mean	Std. Dev
Sm-Uniq	0.0, 0.006	0.24, 0.5	19.3, 28.6	0.49, 0.74	0.92, 0.98
Lg-Uniq	0.0, 0.0016	0.22, 0.5	139.26, 242.9	0.65, 1.00	2.45, 3.57
Lg-Multi	0.0, 0.00157	0.19, 0.46	215.8, 242.9	0.66, 1.04	2.77, 3.39

Distribution viz. for Extra CPI



Understanding the variability



Feature descriptions

- Dynamic ('PMU') vs static ('MCA') features: actual vs best-case execution behavior

- Dynamic features critical for capturing microarchitectural behavior throughout the core, uncore, and memory system that characterize latency sensitive execution.
  - Pruned features (expert knowledge): identified 23 'most predictive' of performance
    - ✓ frontend, speculation, backend, and retiring bottlenecks
  - Independent selections by two analysts disagreed by only two metrics

- Static features use LLVM-MCA, a static analyzer, that models throughput performance of basic blocks at the instruction level. Convert MCA reports into metrics that decompose each superblock's execution, like cycle accounting.
  - CPI (min and expected), CPI waiting for data (min and expected)
  - A total of 21 MCA features are incorporated in the modeling pipeline

Feature correlations

- Used Pearson correlation coefficient between the target (X.CPI) and the features
- Top-10 features (out of 44) were consistently replicated across the three datasets.

Sm-Uniq		Lg-Uniq		Lg-Multi	
Features	ρ_{xy}	Features	ρ_{xy}	Features	ρ_{xy}
mem-stalls-ld	0.44	CPI-LBR-med	0.7	CPI-LBR-med	0.6
l1-miss	0.42	l3-stall-ld	0.44	mem-stalls-ld	0.47
l2-miss	0.41	mem-stalls-ld	0.43	l2-miss	0.42
fb-hit	0.38	l2-miss	0.41	ms-uops	0.39
CPI-LBR-med	0.36	l3-hitm	0.39	l3-stall-ld	0.39
stlb-miss-ld	0.34	l1-miss	0.38	l1-miss	0.34
l1-stall-ld	0.33	ms-uops	0.36	l3-miss	0.32
l3-miss	0.32	CPI-LBR-min	0.33	fb-hit	0.27
l3-hitm	0.31	Xcpi-Wd-min	0.33	l3-hitm	0.27
l2-stall-ld	0.29	l2-stall-ld	0.28	rat-stalls	0.26
l3-stall-ld	0.29	rat-stalls	0.26	baclear	0.23
ms-uops	0.23	stlb-miss-ld	0.26	l1-stall-ld	0.22

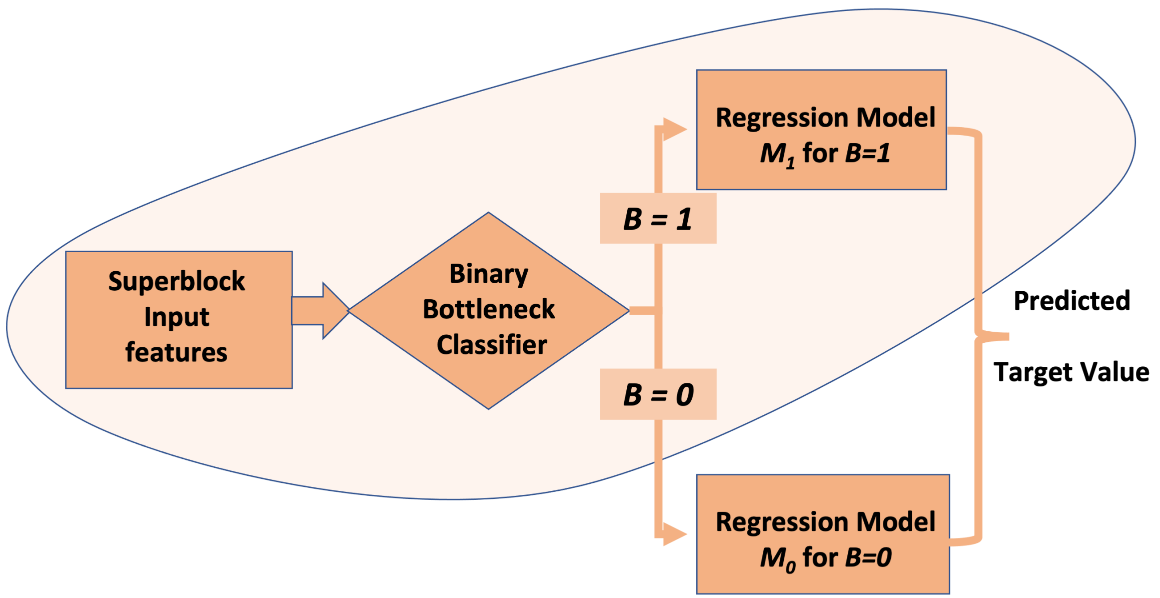
Lessons learnt from single stage models

- Single stage models with PMU events only resulted in low prediction accuracy
- Next, we separated the superblocks into non-bottleneck (B=0) and bottleneck (B=1) categories via thresholding
- Adding MCA features help improve the performance of the models for the severe bottleneck case noticeably
- Although the large datasets have more ‘noise’ (less environmental control) than the small one, the models are able to learn better
- Of all tested methods, ensemble decision tree-based models, especially the random forest and the XGBoost models performed the best

Dataset	PMU Only		MCA only		PMU + MCA	
	B=0	B=1	B=0	B=1	B=0	B=1
Sm-Uniq	0.23	0.41	0.35	0.18	0.44	0.46
Lg-Uniq	0.16	0.5	0.28	0.42	0.31	0.68
Lg-Multi	0.15	0.59	0.63	0.44	0.44	0.73

Considerations with staged classification-regression

- Being able to do good predictions on more severe bottlenecks is useful
- This also means we need to develop a classification model that'll predict whether a superblock constitutes a severe bottleneck or not
- Thus, we need a pre-regression classification stage with good performance



- Overall model can be written as:

$$y = (1 - C(\mathbf{X}))f_0(\mathbf{X}) + C(\mathbf{X})f_1(\mathbf{X})$$

- Note the differences with a traditional piece-wise model. *C(**X**)* provides the boundary in this case.

Mis-classifications (false positives and false negatives) will result in using the wrong regression model thereby reducing the accuracy.

Since we are predicting the target values for the severe bottlenecks only (shaded area in the figure), false negatives won't have a quantitative prediction associated with them.

Optimizing the multi-stage model

- As a direct consequence of the no-free-lunch theorem, the final prediction pipeline must have a composite model that is optimized jointly.
- Thus, for each of the datasets, we explore all possible classifier- regressor combinations to find the optimal assignments.

- Used two training strategies.
  - S1 consists of splitting the available training data into training and validation sets for each stage.
  - S2 uses all the available training data for training and validation steps of both stages.
- The second strategy turns out to be a better one due to two reasons.
  - The individual classifiers and regressors learn better due to more data being available
  - The classifier is able to pre-select superblocks which can be better modeled by the bottleneck regression model

- The B=0/1 threshold is essentially a hyperparameter. Optimal value was found to be the 70^{th} percentile

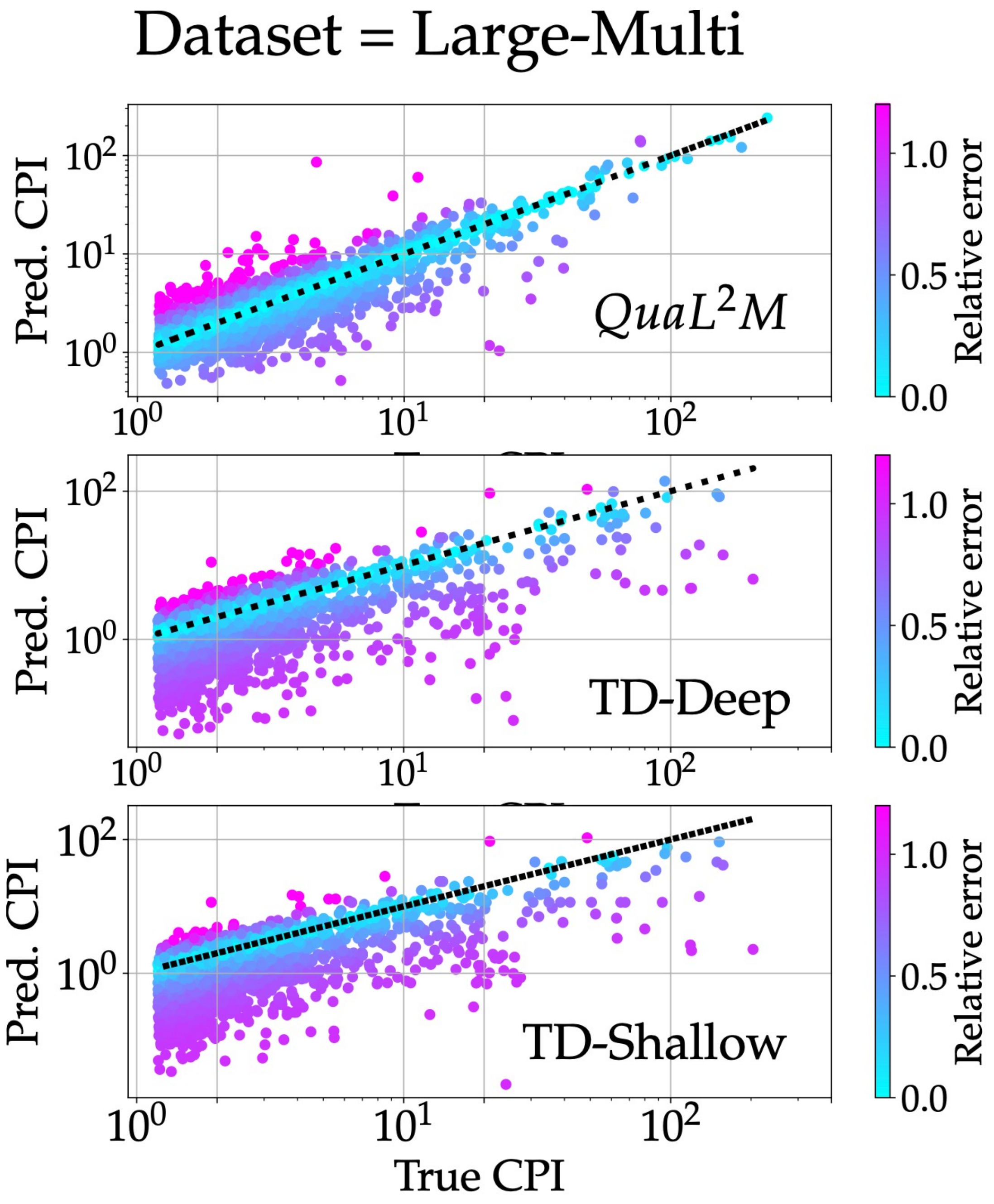
Performance of the multi-stage model

- Best model R2 values achieved by the combined pipeline are shown in the table.

Dataset	Extra CPI	Mean CPI
Sm-Uniq	0.7	0.6
Lg-Uniq	0.66	0.71
Lg-Multi	0.84	0.87

- Error metric ϵ is defined as the median relative error
- Comparison of the error metric for different prediction methods is shown below

Dataset	ϵ(MCPI.L)	ϵ(MCPI.TDS)	ϵ(MCPI.TDD)	ϵ(XCPI.L)
Sm-Uniq	0.2	0.45	0.48	0.17
Lg-Uniq	0.22	0.48	0.52	0.26
Lg-Multi	0.15	0.47	0.5	0.23



Conclusions and future work

- Conclusions
  - Combination of dynamic PMU features and static features can provide good predictions of the real-world superblock CPI measures when combined with ML models
  - These ML models turn out to be ensemble decision trees that include random forests and gradient boosted decision trees
  - Regime of interest (most severe bottlenecks) is best modeled by a composite classifier-regression

- Future work
  - Extending our predictions to accurate bottleneck quantification, like top-down cycle accounting.
  - Improve modeling accuracy by exploring other classes of composite models, and strategies for end-to-end training.