# Predicting ARM64 Serverless Functions Runtime: Leveraging function profiling for generalized performance models

Xinghan Chen, Ling-Hong Hung, Robert Cordingly, Wes Lloyd
kirito20@uw.edu

School of Engineering and Technology
University of Washington Tacoma

December 16, 2024



# Outline

- Background and Motivation
- Research Questions
- Methodology
- Results
- Conclusions

# Serverless Computing

Function-as-a-Service

Serverless function-as-a-service (FaaS) platforms offer many desirable features:

- Rapid elastic scaling
- Scale to zero
- No infrastructure management
- Fine grained billing
- Fault tolerance
- No up front cost to deploy an application



# X86 vs. ARM64

Computing architecture

Switch to ARM64:

- Simplicity
- Power efficiency
- Customization and Flexibility
- Open Ecosystem
- High Compute Density
- Low cost

Stay on X86:

- No migration cost
- Widely supported
- Performance optimization
- Rely on platform specific abilities

# Outline





# Research Questions

- **RQ-1: (Function-Specific Performance Modeling)**: What is the accuracy of ARM64 function runtime predictions for FaaS functions based on profiling on x86_64 processors where training data includes functions being predicted?
- **RQ-2: (Generalized Function Performance Modeling):** What is the accuracy of ARM64 function runtime predictions for unseen FaaS functions not included as training data for models, where models are trained using carefully selected workloads having a range of resource utilization characteristics?

# Research Questions

- **RQ-3: (ARM Performance Classification)**: How accurate are ARM64 serverless function runtime performance classifications using classifiers trained with x86 64 profiling data?
- **RQ-4: (ARM Performance Modeling without FaaS)**: Outside a FaaS platform, what is the accuracy of ARM64 function runtime predictions using models trained by running functions on x86 64 VMs?



# Outline

- Background and Motivation
- Research Questions
- ➤ Methodology
- Results
- Conclusions

# Workloads

**AWS Lambda Functions**
Region: us-west-2 (Oregon),
Memory size: 3008MB(3GB) with 2 vCPU cores,
5GB ephemeral disk for I/O related tests

| Group | Function Name | Source | Description |
|---|---|---|---|
| cpuUser | chacha20*† | openssl | Repeatedly perform openssl encryption of 8MB file n times |
| cpuUser | graph-bfs† | sebs | Breadth-first search (BFS) implementation with igraph. |
| cpuUser | graph-mst† | sebs | Minimum spanning tree (MST) implementation with igraph. |
| cpuUser | graph-pagerank† | sebs | PageRank implementation with igraph. |
| cpuUser | primenumber*† | sysbench | Prime number generator |
| cpuUser | chameleon | FunctionBench | Create HTML table of n rows and M columns |
| cpuUser | csv | Cordingly [9] | Generates a large CSV file and performs calculates on columns. |
| cpuUser | float | FunctionBench | Perform sin, cos, sqrt ops |
| cpuUser | json_dumps | FunctionBench | JSON deserialization using a downloaded JSON-encoded string dataset |
| cpuUser | sqlite | original | Execute n random SELECT queries on a 10*1000 SQLite database |
| cpuUser | video-processing* | sebs | Convert PNG to GIF n times |
| cpuKernel | filehandle† | original | Open and close file handles |
| cpuKernel | socket† | original | Open and close socket n times |
| cpuKernel | thread† | sysbench | Create thread, put locks and release thread |
| Memory | readmemory*† | sysbench | N sequential reads of 1GB memory block |
| Memory | readwritememory† | original | Allowcate 1MByte of memory, write 0x42 into it and release |
| I/O | readdisk*† | fio | Test random read speed on a 1GB block |
| I/O | compression | sebs | Create a .gz file for a file |

cpuUser group: Runtime dominated by CPU user time (blue), cpuKernel group: Runtime with higher CPU kernel time (yellow), Memory group: Workload is memory intensive (orange), and I/O group: Workload is I/O intensive (grey).
*: Function executes external binary program (non-Python)
†: Function used to train models



# Predicting ARM64 FaaS Performance

Methodology for Predicting ARM64 FaaS Performance

**Objective:**
Develop and evaluate models to predict ARM64 serverless function runtime using x86 profiling data.

**Key Approaches:**
- ✦ Function-specific performance modeling
- ✦ Generalized performance modeling for unseen workloads
- ✦ ARM64 runtime classification for optimized predictions

# Model Development

Linear Regression and Random Forest

- **Simple Linear Regression (SLR, SLR-RF) - BASELINE**

  **Runtime - > Runtime**
- **Multi-Regression Analysis (MLR, MLR-RF)**

  **CPU User, CPU Kernel, ….. - > Runtime**
- **Linux CPU Time Accounting (LTA, LTA-RF)**

  **CPU User, CPU Kernel, ….. - > CPU User**

  **CPU User, CPU Kernel, ….. - > CPU Kernel**

  **……**

  **CPU User + CPU Kernel + …… => Runtime**



# Model Development

Types of Generalized Models for Unseen Workloads

- **All-in-One:** Single model for all data
- **Resource-Bound:** Separate models for CPU-user and CPU-kernel intensive tasks
- **ARM-Speed:** Models grouped by ARM64 runtime relative to x86 (faster, slower, similar)

# Methodology Overview

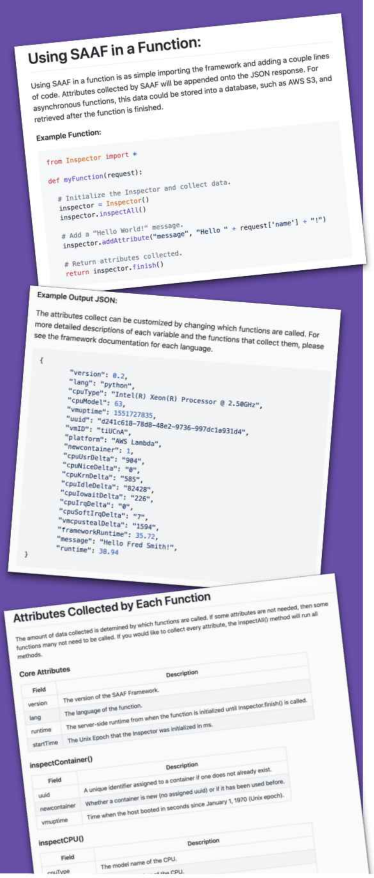
Classification Models for ARM-Speed Selection

- **Challenge:** Identify the best ARM-speed model (ARM-faster, ARM-slower, ARM-similar) for unseen workloads.
- **Solution:** Classification models using x86 profiling data to categorize ARM performance.

| | |
|---|---|
| **ARM-faster** | ARM64 runtime ≥ 15% faster than x86 |
| **ARM-slower** | ARM64 runtime ≤ 15% slower than x86 |
| **ARM-similar** | ARM64 and x86_64 runtime within +/-15% |

- **Features Used:**
  - 21 features, including Linux CPU metrics, memory utilization, and page faults.
- **Classification Algorithms Tested:**
  - Random Forest
  - AdaBoost, MLP(Multi-layer Perceptron), Decision Tree, KNeighbors, Gaussian Process, Quadratic Discriminant Analysis.






# Supporting Tools - SAAF

We utilize the Serverless Application Analytics Framework to collect metrics from serverless functions.

Metrics including CPU time accounting metrics (CPU User, CPU Kernel, CPU Idle), runtime, latency, and more

SAAF Gathers data during function execution provides inputs for training performance models.

SAAF and our other tools are is available here: [https://github.com/wlloyduw/SAAF](https://github.com/wlloyduw/SAAF)

# Outline

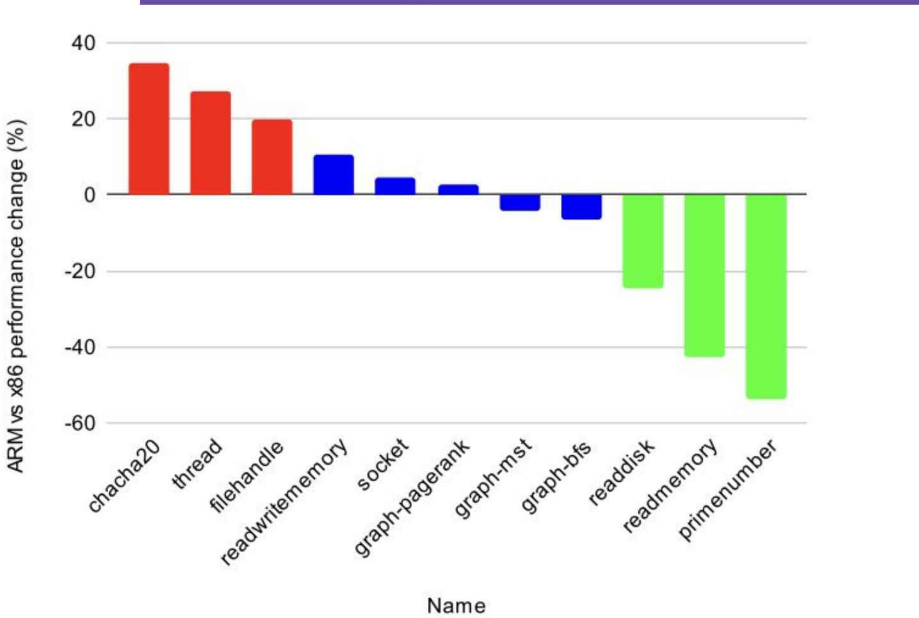
- Background and Motivation
- Research Questions
- Methodology
- Results
- Conclusions



# Research Question 1

Function-Specific Performance Modeling

What is the accuracy of ARM64 function runtime predictions for FaaS functions based on profiling on x86 64 processors where training data includes functions being predicted?



# Research Question 2

## Generalized Function Performance Modeling

- What is the accuracy of ARM64 function runtime predictions for unseen FaaS functions not included as training data for models, where models are trained using carefully selected workloads having a range of resource utilization characteristics?

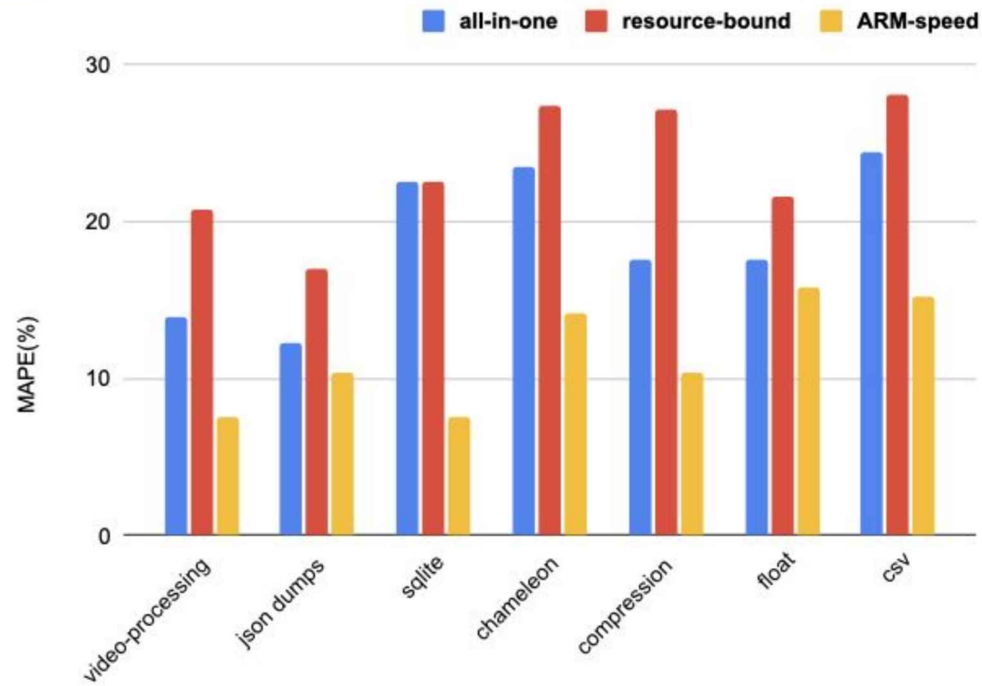





TABLE III

TRAINING AND TESTING FUNCTION'S RUNTIME, COEFFICIENT OF VARIATION (CV), AND MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

| **Function name** | **Min runtime** x86_64 (sec) | **Min runtime** ARM64 (sec) | **Max runtime** x86_64 (sec) | **Max runtime** ARM64 (sec) | **CV**(%) x86_64 | **CV**(%) ARM64 | **MAPE** fn-specific[1 2] | **MAPE** All-in-One[1] | **MAPE** ARM-speed[1] |
|---|---|---|---|---|---|---|---|---|---|
| primenumber | 6.00 | 5.27 | 120.92 | 108.73 | 0.72 | 0.58 | 0.83 | 28.55 | 0.18 |
| readmemory | 3.15 | 3.85 | 132.68 | 106.40 | 2.17 | 3.51 | 1.2 | 7.02 | 2.15 |
| readdisk | 6.77 | 7.46 | 135.01 | 114.11 | 2.05 | 1.79 | 2.17 | 16.47 | 1.76 |
| chacha20 | 4.70 | 4.53 | 118.90 | 144.54 | 0.73 | 0.23 | 0.2 | 27.93 | 7.42 |
| readwritememory | 5.08 | 3.89 | 123.16 | 134.82 | 1.28 | 2.32 | 1.44 | 8.93 | 5.41 |
| filehandle | 4.69 | 8.87 | 109.33 | 132.41 | 1.88 | 0.95 | 2.84 | 5.26 | 2.49 |
| thread | 4.46 | 5.38 | 128.17 | 135.75 | 0.63 | 0.56 | 0.96 | 18.82 | 1.75 |
| graph-pagerank | 5.58 | 6.15 | 58.69 | 61.45 | 0.60 | 0.57 | 0.98 | 9.32 | 2.15 |
| graph-mst | 6.83 | 3.40 | 65.03 | 56.15 | 0.63 | 0.56 | 0.96 | 3.05 | 2.46 |
| graph-bfs | 4.25 | 8.77 | 64.10 | 67.49 | 0.94 | 0.84 | 0.39 | 4.02 | 3.94 |
| socket | 7.82 | 6.91 | 125.99 | 130.18 | 2.31 | 3.08 | 0.97 | 1.51 | 3.72 |
| video-processing | 3.01 | 3.17 | 139.54 | 135.75 | 0.42 | 1.07 | 1.79 | 25.26 | 8.32 |
| json dumps | 5.30 | 8.71 | 128.80 | 134.02 | 1.59 | 1.45 | 0.64 | 5.23 | 7.83 |
| sqlite | 6.28 | 4.25 | 134.92 | 121.42 | 1.06 | 0.82 | 0.97 | 18.79 | 6.96 |
| chameleon | 5.12 | 8.29 | 112.96 | 101.62 | 1.09 | 0.74 | 1.13 | 13.07 | 10.60 |
| compression | 8.21 | 7.48 | 135.76 | 122.41 | 1.80 | 0.46 | 0.52 | 15.26 | 11.93 |
| float | 4.19 | 8.63 | 122.40 | 135.99 | 3.26 | 2.14 | 0.85 | 24.04 | 14.30 |
| csv | 8.87 | 8.90 | 136.81 | 124.68 | 1.22 | 0.94 | 2.17 | 29.72 | 12.10 |
| **Avg-training** | 5.39 | 5.86 | 107.45 | 108.37 | 1.27 | 1.36 | 1.17 | 11.90 | 3.04 |
| **Avg-unseen** | 5.85 | 7.06 | 130.17 | 125.13 | 1.49 | 1.09 | 1.15 | 18.77 | 10.29 |
| **Average** | 5.57 | 6.33 | 116.29 | 114.88 | 1.35 | 1.26 | 1.16 | 14.57 | 5.86 |

[1]-random forest regression w/ multi-features, [2]-evaluated w/ 2nd independent 4k sample/fn dataset

TABLE III
TRAINING AND TESTING FUNCTION'S RUNTIME, COEFFICIENT OF VARIATION (CV), AND MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

| **Function name** | **Min runtime** x86_64 (sec) | **Min runtime** ARM64 (sec) | **Max runtime** x86_64 (sec) | **Max runtime** ARM64 (sec) | **CV(%)** x86_64 | **CV(%)** ARM64 | **MAPE** fn-specific[1,2] | **MAPE** All-in-One[1] | **MAPE** ARM-speed[1] |
|---|---|---|---|---|---|---|---|---|---|
| primenumber | 6.00 | 5.27 | 120.92 | 108.73 | 0.72 | 0.58 | 0.83 | 28.55 | 0.18 |
| readmemory | 3.15 | 3.85 | 132.68 | 106.40 | 2.17 | 3.51 | 1.2 | 7.02 | 2.15 |
| readdisk | 6.77 | 7.46 | 135.01 | 114.11 | 2.05 | 1.79 | 2.17 | 16.47 | 1.76 |
| chacha20 | 4.70 | 4.53 | 118.90 | 144.54 | 0.73 | 0.23 | 0.2 | 27.93 | 7.42 |
| readwritememory | 5.08 | 3.89 | 123.16 | 134.82 | 1.28 | 2.32 | 1.44 | 8.93 | 5.41 |
| filehandle | 4.69 | 8.87 | 109.33 | 132.41 | 1.88 | 0.95 | 2.84 | 5.26 | 2.49 |
| thread | 4.46 | 5.38 | 128.17 | 135.75 | 0.63 | 0.56 | 0.96 | 18.82 | 1.75 |
| graph-pagerank | 5.58 | 6.15 | 58.69 | 61.45 | 0.60 | 0.57 | 0.98 | 9.32 | 2.15 |
| graph-mst | 6.83 | 3.40 | 65.03 | 56.15 | 0.63 | 0.56 | 0.96 | 3.05 | 2.46 |
| graph-bfs | 4.25 | 8.77 | 64.10 | 67.49 | 0.94 | 0.84 | 0.39 | 4.02 | 3.94 |
| socket | 7.82 | 6.91 | 125.99 | 130.18 | 2.31 | 3.08 | 0.97 | 1.51 | 3.72 |
| video-processing | 3.01 | 3.17 | 139.54 | 135.75 | 0.42 | 1.07 | 1.79 | 25.26 | 8.32 |
| json dumps | 5.30 | 8.71 | 128.80 | 134.02 | 1.59 | 1.45 | 0.64 | 5.23 | 7.83 |
| sqlite | 6.28 | 4.25 | 134.92 | 121.42 | 1.06 | 0.82 | 0.97 | 18.79 | 6.96 |
| chameleon | 5.12 | 8.29 | 112.96 | 101.62 | 1.09 | 0.74 | 1.13 | 13.07 | 10.60 |
| compression | 8.21 | 7.48 | 135.76 | 122.41 | 1.80 | 0.46 | 0.52 | 15.26 | 11.93 |
| float | 4.19 | 8.63 | 122.40 | 135.99 | 3.26 | 2.14 | 0.85 | 24.04 | 14.30 |
| csv | 8.87 | 8.90 | 136.81 | 124.68 | 1.22 | 0.94 | 2.17 | 29.72 | 12.10 |
| **Avg-training** | 5.39 | 5.86 | 107.45 | 108.37 | 1.27 | 1.36 | 1.17 | 11.90 | 3.04 |
| **Avg-unseen** | 5.85 | 7.06 | 130.17 | 125.13 | 1.49 | 1.09 | 1.15 | 18.77 | 10.29 |
| **Average** | 5.57 | 6.33 | 116.29 | 114.88 | 1.35 | 1.26 | 1.16 | 14.57 | 5.86 |

[1]-random forest regression w/ multi-features, [2]-evaluated w/ 2nd independent 4k sample/fn dataset

TABLE III
TRAINING AND TESTING FUNCTION'S RUNTIME, COEFFICIENT OF VARIATION (CV), AND MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

| **Function name** | **Min runtime** x86_64 (sec) | **Min runtime** ARM64 (sec) | **Max runtime** x86_64 (sec) | **Max runtime** ARM64 (sec) | **CV(%)** x86_64 | **CV(%)** ARM64 | **MAPE** fn-specific[1,2] | **MAPE** All-in-One[1] | **MAPE** ARM-speed[1] |
|---|---|---|---|---|---|---|---|---|---|
| primenumber | 6.00 | 5.27 | 120.92 | 108.73 | 0.72 | 0.58 | 0.83 | 28.55 | 0.18 |
| readmemory | 3.15 | 3.85 | 132.68 | 106.40 | 2.17 | 3.51 | 1.2 | 7.02 | 2.15 |
| readdisk | 6.77 | 7.46 | 135.01 | 114.11 | 2.05 | 1.79 | 2.17 | 16.47 | 1.76 |
| chacha20 | 4.70 | 4.53 | 118.90 | 144.54 | 0.73 | 0.23 | 0.2 | 27.93 | 7.42 |
| readwritememory | 5.08 | 3.89 | 123.16 | 134.82 | 1.28 | 2.32 | 1.44 | 8.93 | 5.41 |
| filehandle | 4.69 | 8.87 | 109.33 | 132.41 | 1.88 | 0.95 | 2.84 | 5.26 | 2.49 |
| thread | 4.46 | 5.38 | 128.17 | 135.75 | 0.63 | 0.56 | 0.96 | 18.82 | 1.75 |
| graph-pagerank | 5.58 | 6.15 | 58.69 | 61.45 | 0.60 | 0.57 | 0.98 | 9.32 | 2.15 |
| graph-mst | 6.83 | 3.40 | 65.03 | 56.15 | 0.63 | 0.56 | 0.96 | 3.05 | 2.46 |
| graph-bfs | 4.25 | 8.77 | 64.10 | 67.49 | 0.94 | 0.84 | 0.39 | 4.02 | 3.94 |
| socket | 7.82 | 6.91 | 125.99 | 130.18 | 2.31 | 3.08 | 0.97 | 1.51 | 3.72 |
| video-processing | 3.01 | 3.17 | 139.54 | 135.75 | 0.42 | 1.07 | 1.79 | 25.26 | 8.32 |
| json dumps | 5.30 | 8.71 | 128.80 | 134.02 | 1.59 | 1.45 | 0.64 | 5.23 | 7.83 |
| sqlite | 6.28 | 4.25 | 134.92 | 121.42 | 1.06 | 0.82 | 0.97 | 18.79 | 6.96 |
| chameleon | 5.12 | 8.29 | 112.96 | 101.62 | 1.09 | 0.74 | 1.13 | 13.07 | 10.60 |
| compression | 8.21 | 7.48 | 135.76 | 122.41 | 1.80 | 0.46 | 0.52 | 15.26 | 11.93 |
| float | 4.19 | 8.63 | 122.40 | 135.99 | 3.26 | 2.14 | 0.85 | 24.04 | 14.30 |
| csv | 8.87 | 8.90 | 136.81 | 124.68 | 1.22 | 0.94 | 2.17 | 29.72 | 12.10 |
| **Avg-training** | 5.39 | 5.86 | 107.45 | 108.37 | 1.27 | 1.36 | 1.17 | 11.90 | 3.04 |
| **Avg-unseen** | 5.85 | 7.06 | 130.17 | 125.13 | 1.49 | 1.09 | 1.15 | 18.77 | 10.29 |
| **Average** | 5.57 | 6.33 | 116.29 | 114.88 | 1.35 | 1.26 | 1.16 | 14.57 | 5.86 |

-random forest regression w/ multi-features, -evaluated w/ 2nd independent 4k sample/fn dataset

# Research Question 3

ARM Performance Classification

- How accurate are ARM64 serverless function runtime performance classifications using classifiers trained with x86_64 profiling data?

| TARGET / OUTPUT | ARM-faster | ARM-similar | ARM-slower | SUM |
|---|---|---|---|---|
| ARM-faster | 11972<br>16.63% | 27<br>0.04% | 1<br>0.00% | 12000<br>99.77%<br>0.23% |
| ARM-similar | 2027<br>2.82% | 40030<br>55.60% | 1936<br>2.69% | 43993<br>90.99%<br>9.01% |
| ARM-slower | 93<br>0.13% | 707<br>0.98% | 15200<br>21.11% | 16000<br>95.00%<br>5.00% |
| SUM | 14092<br>84.96%<br>15.04% | 40764<br>98.20%<br>1.80% | 17137<br>88.70%<br>11.30% | 67202 / 71993<br>93.35%<br>6.65% |



Best single sample prediction result

# Select Classifier

Classifier accuracy comparison

| Classifier | Accuracy |
|---|---|
| Random Forest | 93.35% |
| DecisionTree | 91.65% |
| Gaussian Process | 83.63% |
| AdaBoost | 78.78% |
| KNeighbors | 74.55% |
| MLP | 65.83% |
| Quadratic Discriminant Analysis | 62.05% |

ARM Faster: ARM64 runtime 15% Faster
ARM Slower: ARM64 runtime 15% Slower

Training Set: 40 Steps x 100 runs/step x 11 functions x 2 architectures = 88,000 Samples
Testing Set: 40 Steps x 100 runs/step x 7 functions = 28,000 Samples

# Research Question 4

ARM Performance Modeling without FaaS

- Outside a FaaS platform, what is the accuracy of ARM64 function runtime predictions using models trained by running functions on x86 64 VMs?



# Outline

- Background and Motivation
- Research Questions
- Methodology
- Results
- Conclusions

# Conclusions - RQ-1

We executed experiments using 18 functions on AWS to compare X86 vs. ARM64 FaaS and generate models to predict the performance.

**RQ-1: (Function-Specific Performance Modeling):**

Function-specific models = very high accuracy for ARM64 runtime predictions.

Average MAPE with Random Forest achieving the best results (1.17 MAPE).

Models trained on x86 profiling data successfully predicted ARM64 performance with minimal error, validating their reliability for known workloads.



# Conclusions - RQ-2

**RQ-2: (Generalized Function Performance Modeling):**

Generalized models effectively predicted runtime for unseen workloads using diverse training sets.

ARM-speed models achieved the best accuracy by grouping workloads into ARM-faster, ARM-slower, and ARM-similar categories.

Generalized models had an average MAPE of 10.29 for unseen functions and 5.86 for all functions, highlighting their potential for broader applicability.

# Conclusions - RQ-3

## RQ-3: (ARM Performance Classification):

ARM runtime classification into ARM-faster, ARM-slower, and ARM-similar was highly accurate.

Random Forest achieved 93.35% classification accuracy for a single prediction, with 10 prediction we could accumulate 99.75% accuracy, significantly reducing misclassification risks for unseen workloads.

Performance classification supports reliable pairing of workloads with the appropriate ARM-speed model.



# Conclusions - RQ-4

## RQ-4: (ARM Performance Modeling without FaaS):

ARM64 runtime predictions were successfully validated on AWS EC2 VMs, extending the approach beyond serverless platforms.

The models maintained strong accuracy, with an average MAPE of 1.41 for function-specific predictions.

This demonstrates that x86-to-ARM64 modeling is robust and adaptable for non-serverless applications.

?



# Thank You!