Carnegie Mellon University

Interference-Aware Edge Runtime Prediction

with Conformal Matrix Completion

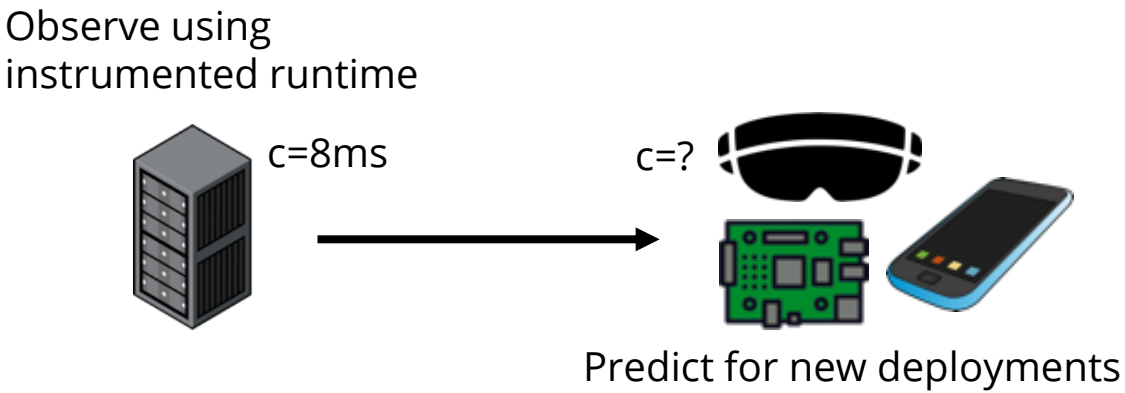
Tianshu Huang, Arjun Ramesh, Emily Ruppel,
Nuno Pereira, Anthony Rowe, Carlee Joe-Wong

BOSCH

Carnegie Mellon University
WebAssembly Research Center

Why **Runtime Prediction?**

- Roadmap new hardware devices
- App store deployment
- Edge orchestration



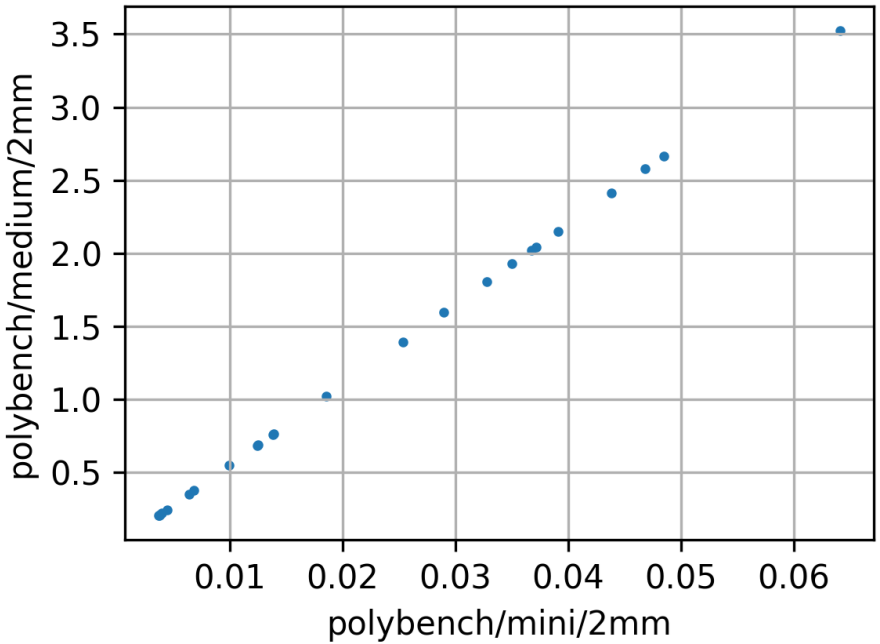
Why **Runtime Prediction?**

PassMark - CPU Mark

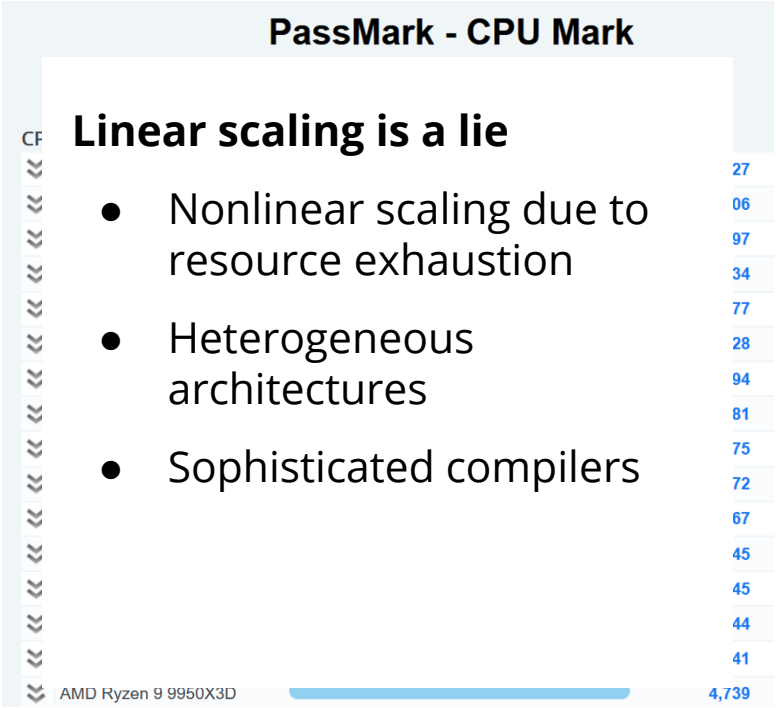
Single Thread Performance

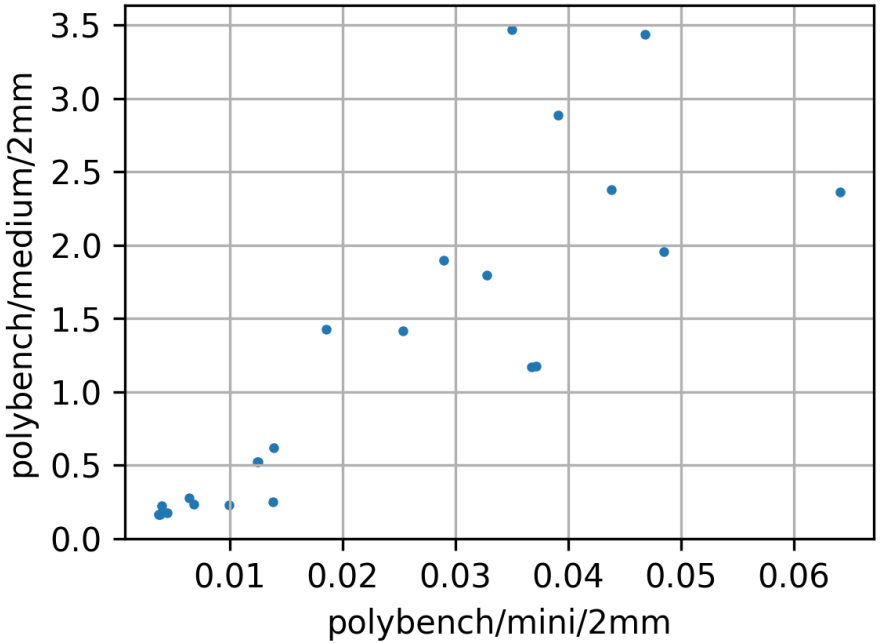
Updated 8th of May 2025

CPU	CPU Mark
Apple M3 Ultra 28 Core	5,127
Apple M3 Ultra 32 Core	5,106
Intel Core Ultra 9 285K	5,097
Intel Core Ultra 7 265KF	4,934
Intel Core Ultra 7 265K	4,877
Intel Core i9-14900KS	4,828
Apple M3 Max 16 Core	4,794
Apple M3 Max 14 Core	4,781
Intel Core Ultra 9 285	4,775
Intel Core Ultra 9 285HX	4,772
Apple M3 Pro 11 Core	4,767
Intel Core Ultra 7 255HX	4,745
Apple M3 8 Core	4,745
Intel Core Ultra 9 285T	4,744
AMD Ryzen 9 9950X	4,741
AMD Ryzen 9 9950X3D	4,739



Why **Runtime Prediction?**



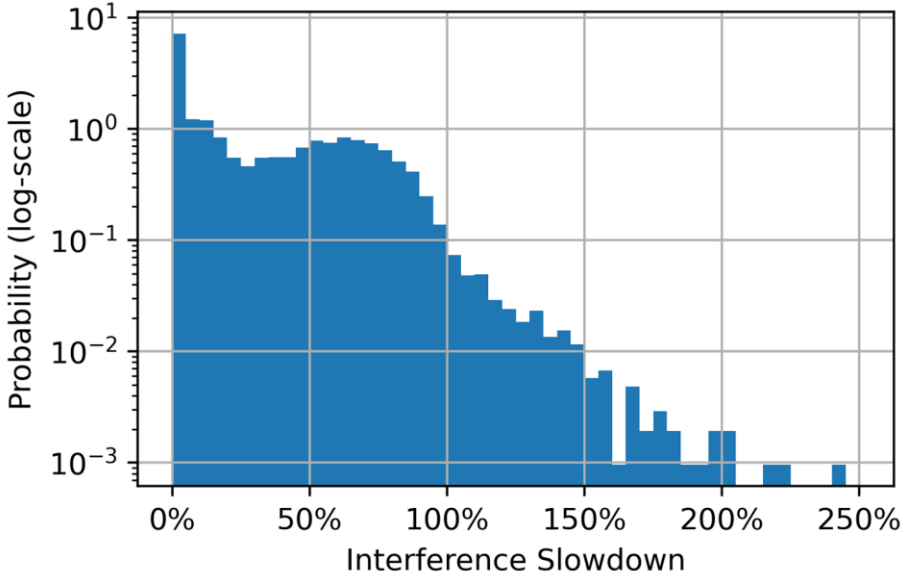


Why **Runtime Prediction?**

Interference can be crippling!

- **Resource contention:** memory, disk, SMT / functional units, …
- **Alignment:** the order that tasks are scheduled can affect interference

Interference is **hard to quantify and characterize.**

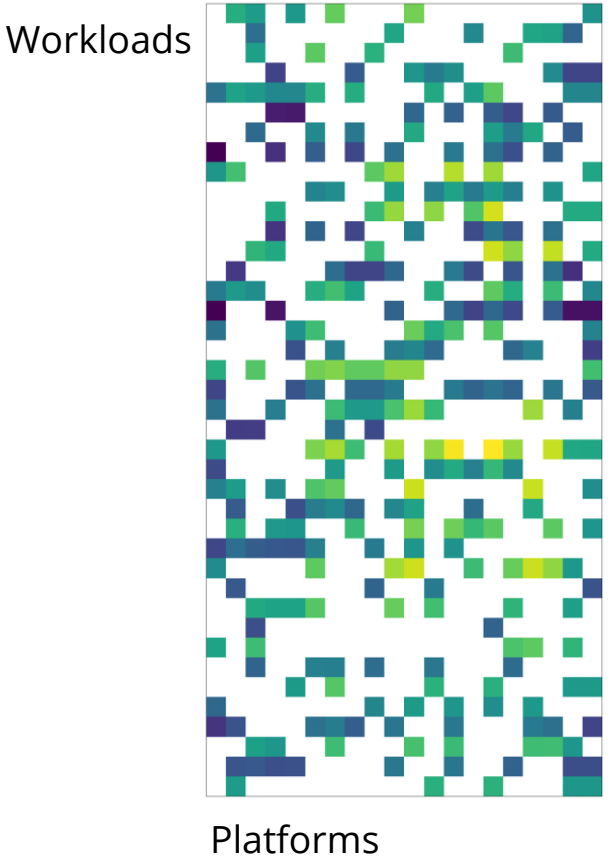


Matrix Completion

Runtime Prediction is **Matrix Completion**

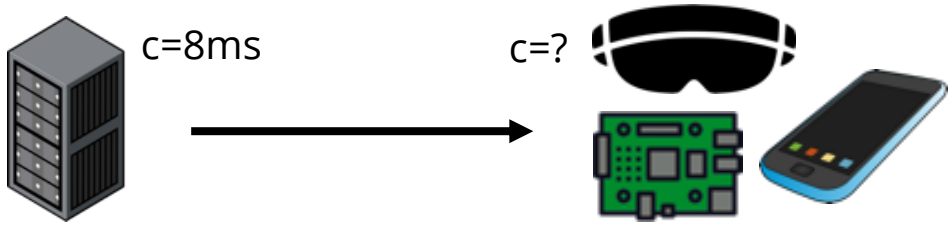
- Rows × Columns: Workloads × Platforms
- Observe runtime of a subset of (workload, platforms) pairs, predict unobserved entries

Not a new insight: Paragon (SIGPLAN ‘13), Quasar (SIGPLAN ‘14)



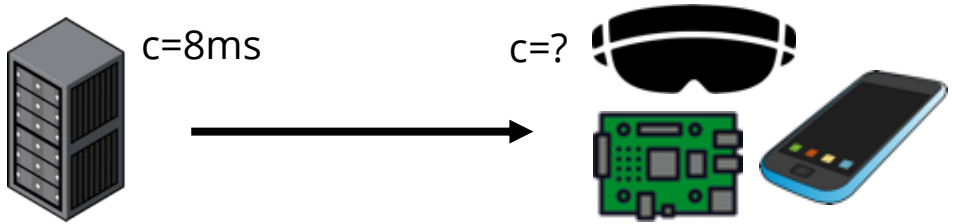
Isn’t this a solved problem?

… and why hasn’t matrix completion solved it?



Why isn’t Matrix Completion more popular?

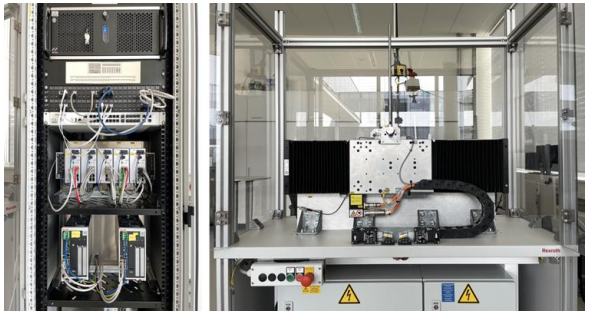
- The cloud isn’t so heterogeneous;
- You can just do exhaustive benchmarking…
- … so performance prediction is not so critical.

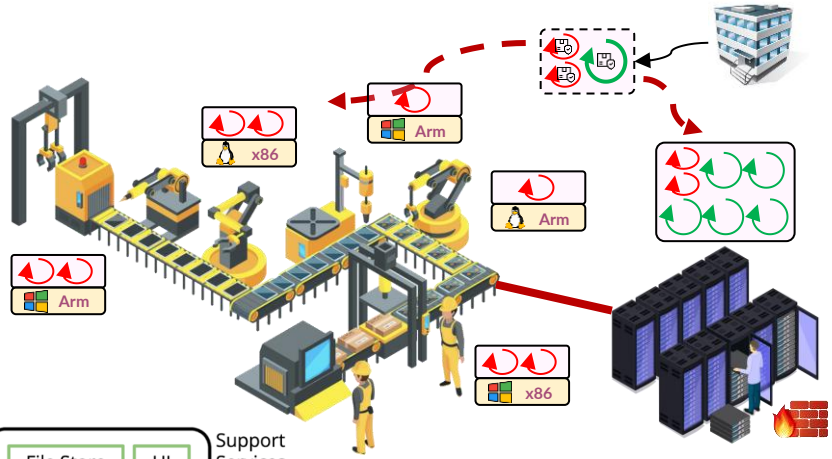


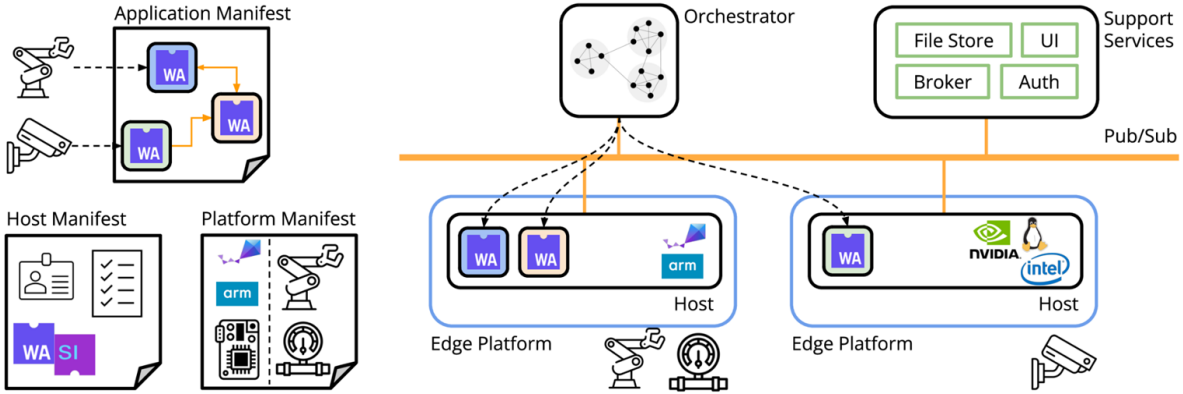
Why isn’t Matrix Completion more popular?

- The cloud isn’t so heterogeneous;
- You can just do exhaustive benchmarking…
- … so performance prediction is not so critical.

Why **Runtime Prediction?**







The (Cyber-Physical) Edge

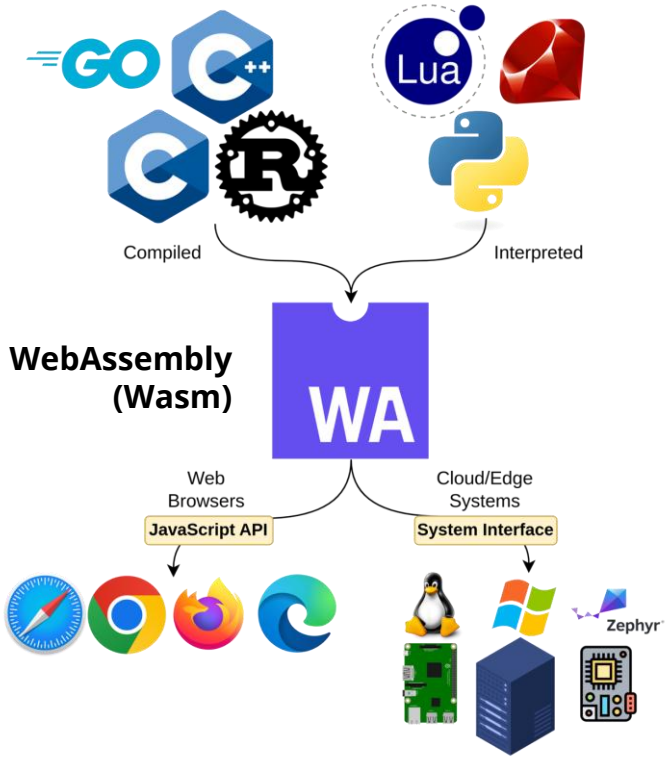
- **Highly heterogeneous**: ARM, x86, RISC-V, Accelerators, …
- **Resource constrained**: interference effects can be significant
- **Real time**: deadlines, not efficiency

Dataset: N=53,637 + 357,333

- **24 unique devices**: x86 Intel & AMD, ARM A & M class, and RISC-V
- **8 Wasm runtimes**: interpreted, JIT, & AOT
- **176 benchmarks** from <1ms to 30s long

Aside: **Why Webassembly?**

- **Capture runtime diversity**: like having different operating systems, without having to actually install different operating systems
- **Cross-platform**: fix compilation issues M+N instead of M×N times
- **Easy to instrument**: capture program-specific but cross-platform “side information” by looking at how much the program uses various instructions



Runtime Prediction **at the Cyber-Physical Edge**

Static Analysis with **per-platform model**

- Very tricky (halting problem)
- Hard to predict performance other than worst-case

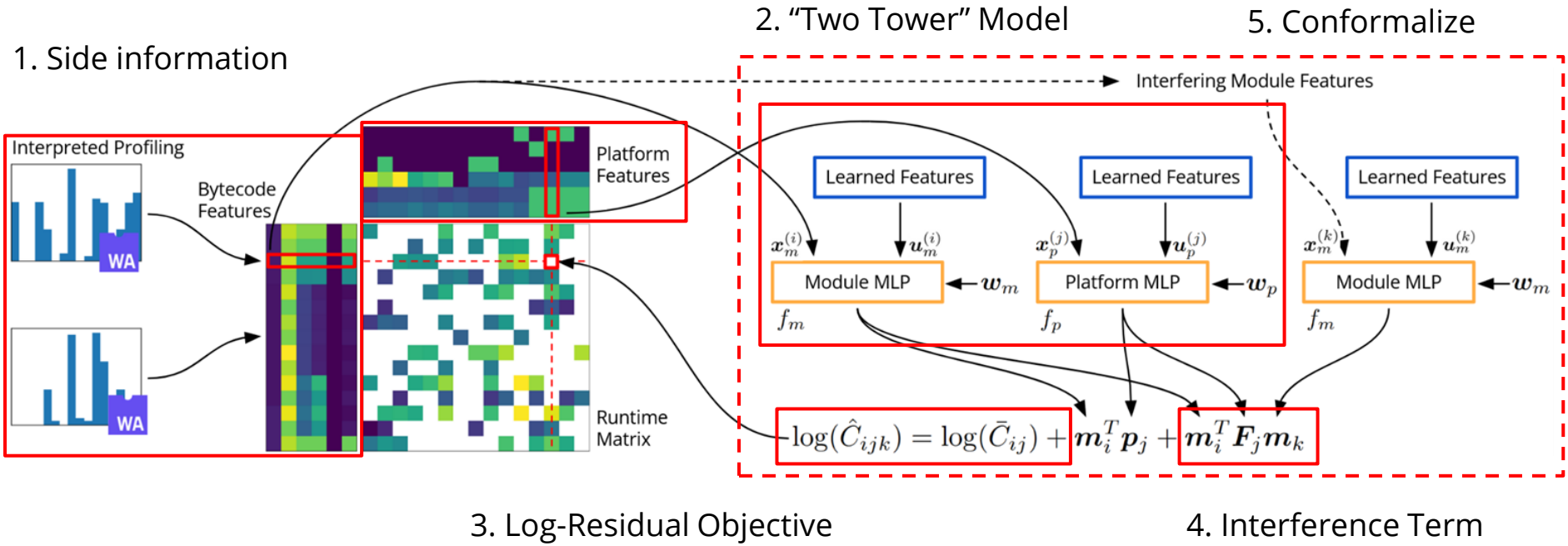
Dynamic Analysis, with **device transfer model**

- Invasive instrumentation required
- Different model for each pair of platforms

Dynamic Analysis, with **simulator-device transfer model**

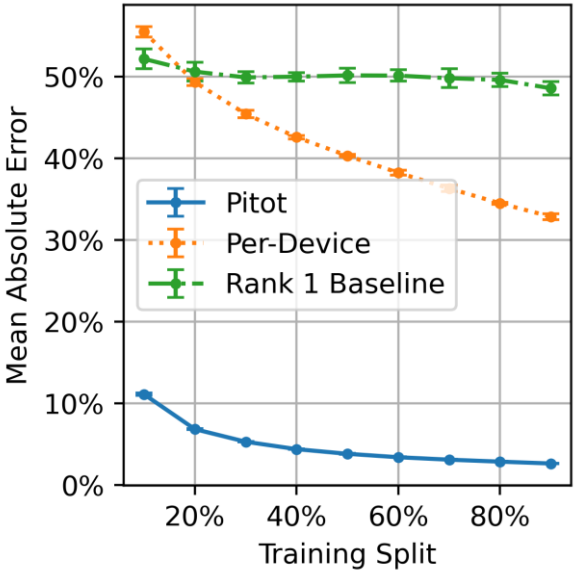
- Need to build a simulator for each platform

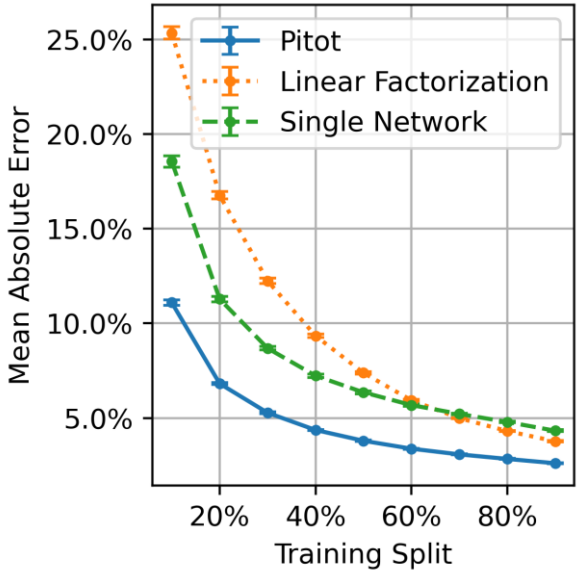
Method: **Matrix Factorization** (with Side Information)



It works

Pitot vs Baselines:

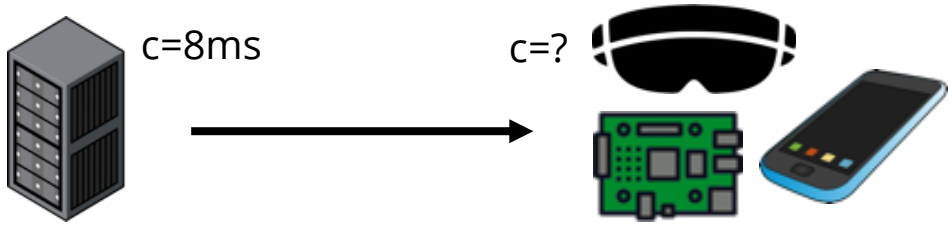




(2.6%)

Isn’t this a solved problem?

… and why hasn’t our method we solved it?



What’s needed for this to work?

- Can’t just “download a model”: need to benchmark!
- Data availability & data sharing is limited
- True cross-platform portability is not quite here yet

Interference-Aware Edge Runtime Prediction

with Conformal Matrix Completion

Dataset & code:
github.com/WiseLabCMU/pitot

Correspondence to:
tianshu2@andrew.cmu.edu