Nugget: Portable Program Snippets

Zhantong Qiu, Mahyar Samani, Jason Lowe-Power

University of California, Davis

DArchR
DAVIS ARCHITECTURE RESEARCH

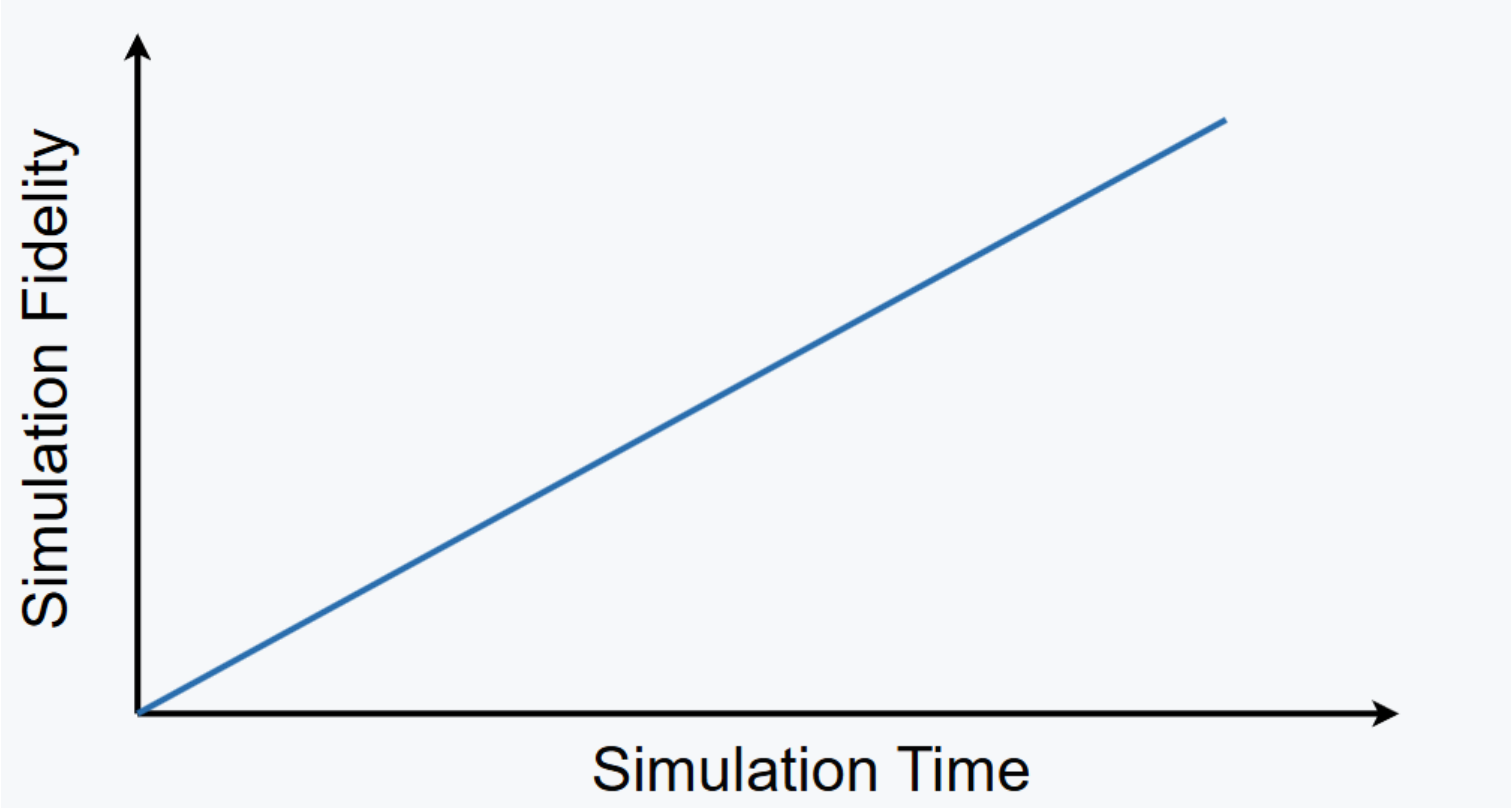
UCDAVIS
COMPUTER SCIENCE

Outline

1. Background and Motivation
2. The Nugget Framework
3. Evaluation
4. Conclusion

Background and Motivation

Problem: High fidelity simulation implies long simulation time

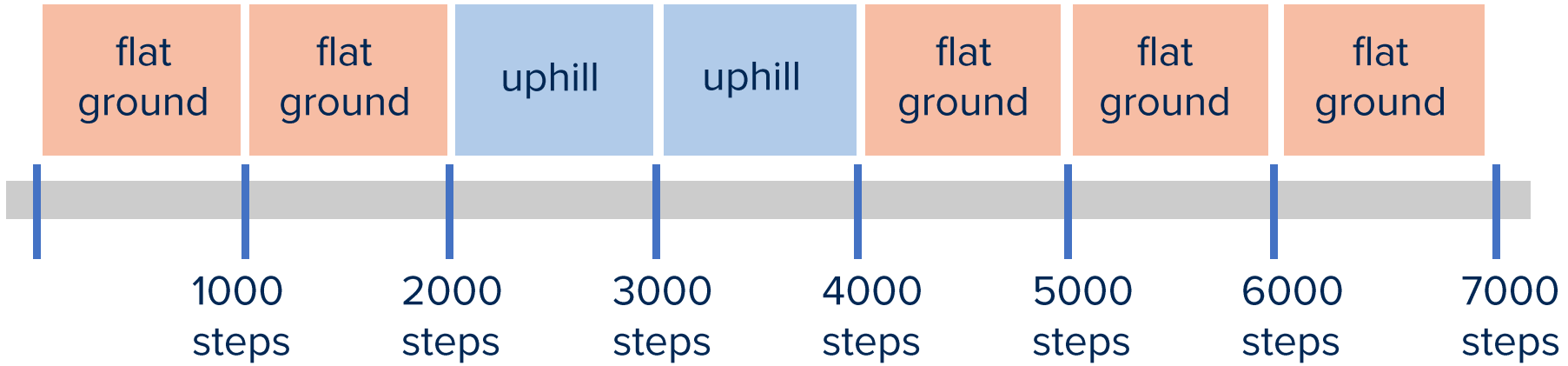


What are the solutions to long simulation time?

- Reduce simulation fidelity
- Reduce workload
  - Use sampling methodologies
    - Two major types:
      1. **Targeted sampling**
      2. Statistical sampling

Targeted Sampling

Representative methodologies: SimPoint, LoopPoint



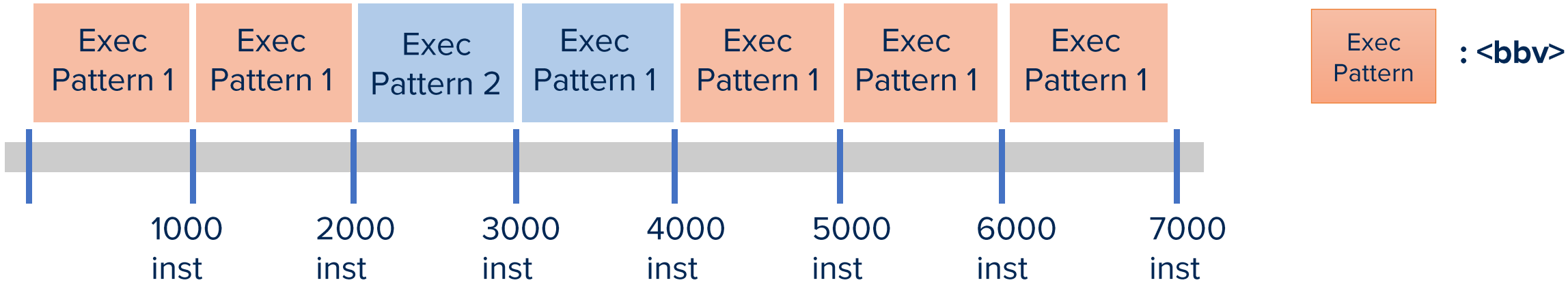
flat ground ⇨ 8 min/1000 steps

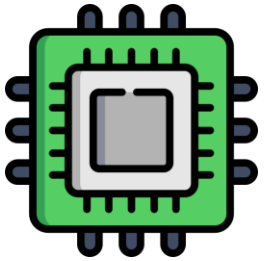
uphill ⇨ 10 min/1000 steps

Their predicted runtime for this path is
5 x 8 min + 2 x 10 min = 60 min

Targeted Sampling

Representative methodologies: **SimPoint**, LoopPoint





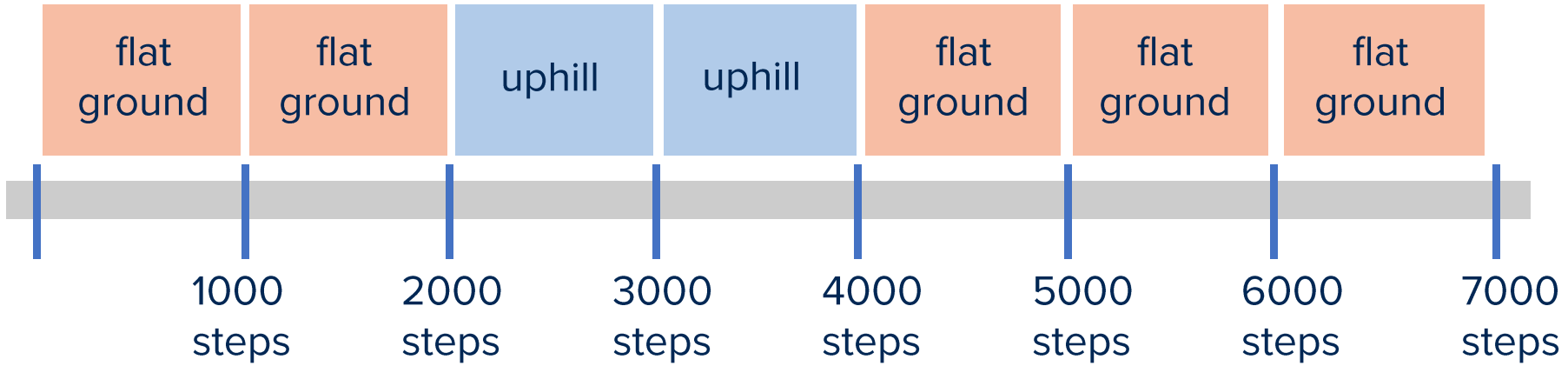
Exec Pattern 1 ⇨ Runtime: X = CPI_1 * 1000 * 1/f

Exec Pattern 2 ⇨ Runtime: Y = CPI_2 * 1000 * 1/f

Their predicted runtime for this program is
5 x X + 2 x Y

Targeted Sampling

Representative methodologies: SimPoint, LoopPoint



Targeted sampling selects samples based on specific characteristics that are discovered by **analysis**.

Drawbacks of prior works

Drawbacks	Prior Works: SimPoint, LoopPoint
Samples are expensive to find	Rely on **simulation/tool** to analyze program ✗
Samples are tied to a single executable binary	Rely on **machine code instruction** to define interval ✗
Validating the selected samples is infeasible	Rely on **slow simulation** to validate samples ✗

Drawbacks of prior works

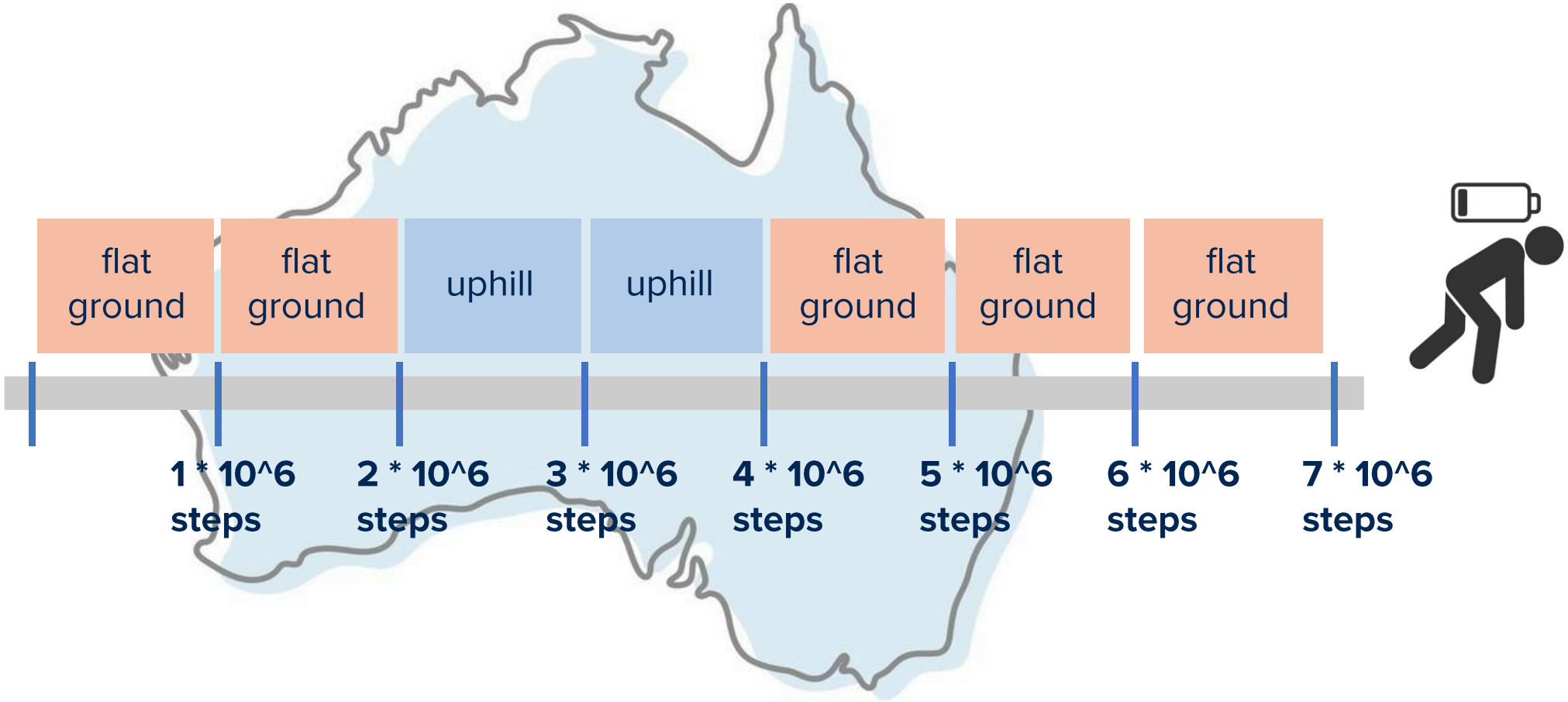
Drawbacks	Prior Works: SimPoint, LoopPoint
Samples are expensive to	Rely on **tool / simulation to**
validating the selected samples is infeasible	Rely on **slow simulation** to validate samples ✗

These drawbacks make the sampling process **slow and effectively a black box.**

The Nugget framework's goals

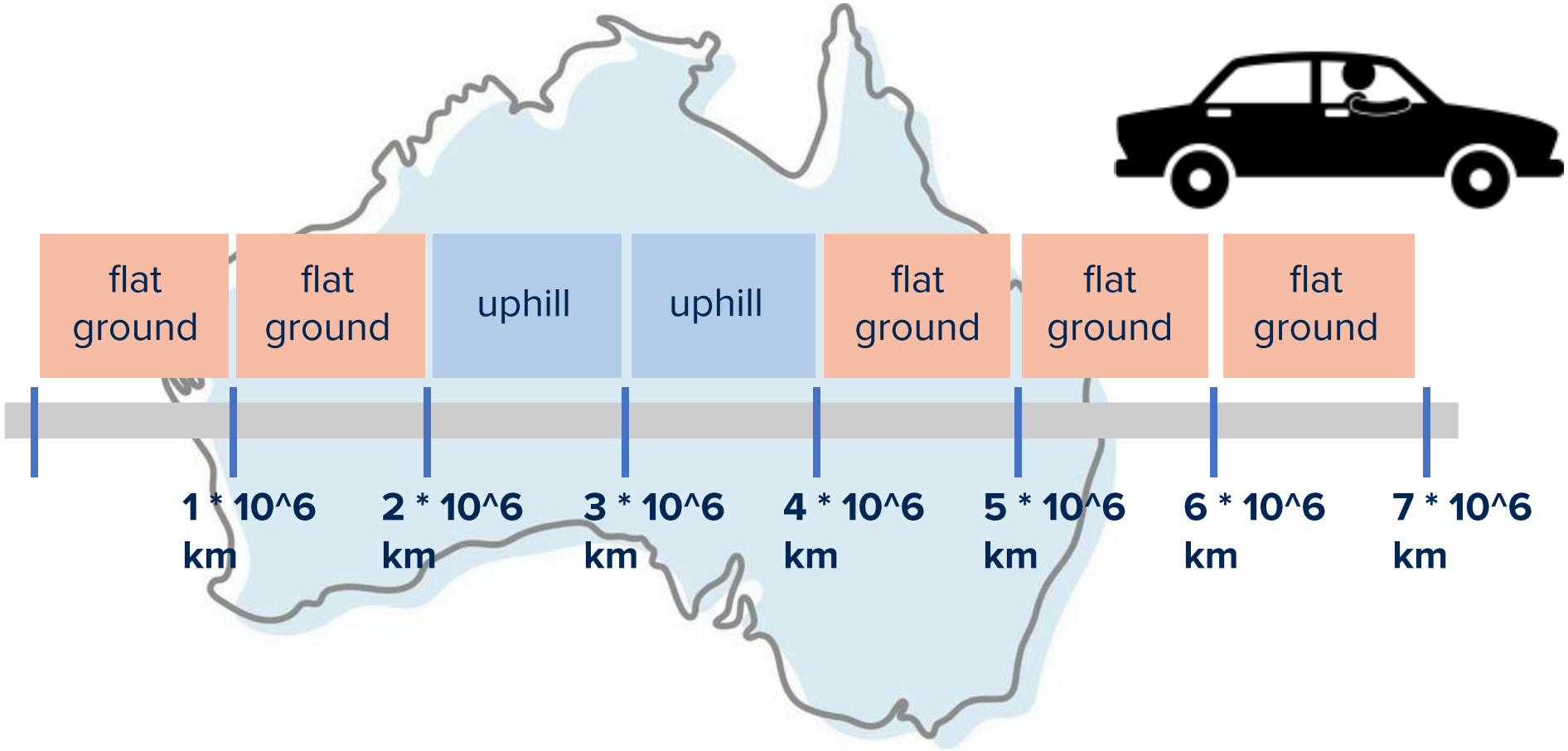
1. **Make finding samples fast**
   - Analyzing program execution quickly and without restrictions on architecture or host machine
2. **Make samples decoupled from a single executable binary**
   - Automatically generating samples that are independent of the binary
3. **Make validating the selected samples feasible**
   - Quickly validate the selected samples for the targeted benchmarks and input size

Connect the example with Nugget



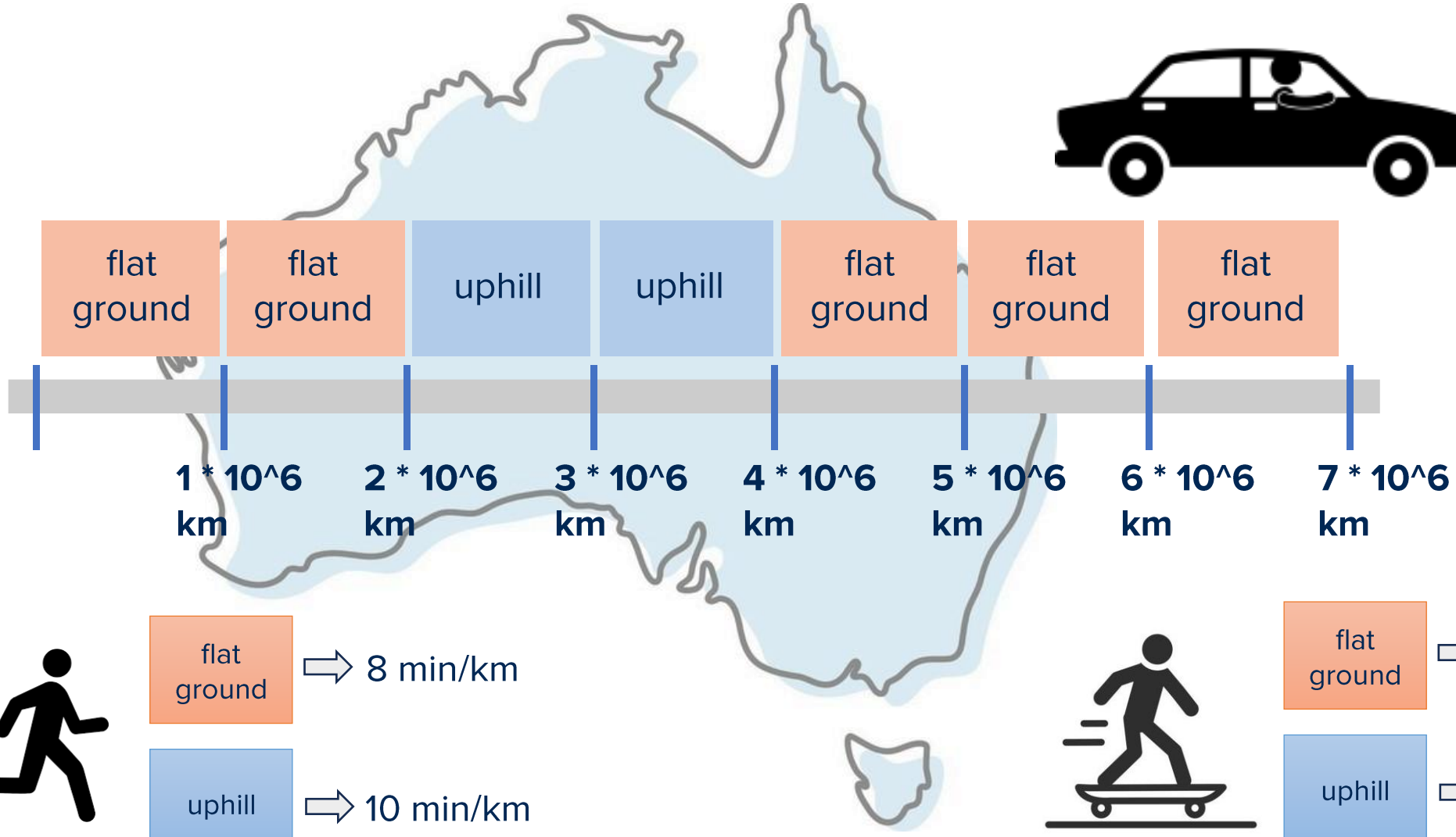
If the unit is **step**,
they'll need to step through the path

Universal unit enables faster analysis method



If the unit is **km** instead of step,
they can use faster method to measure km

Universal unit enables cross platform samples



flat ground ⇨ 1 min/km

uphill ⇨ 1.2 min/km

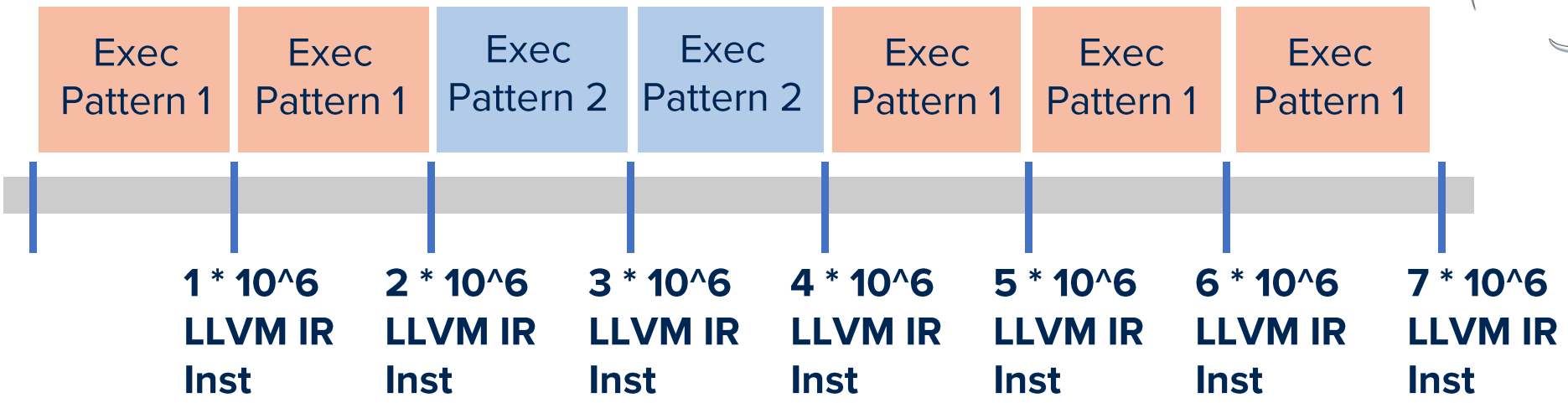
flat ground ⇨ 8 min/km

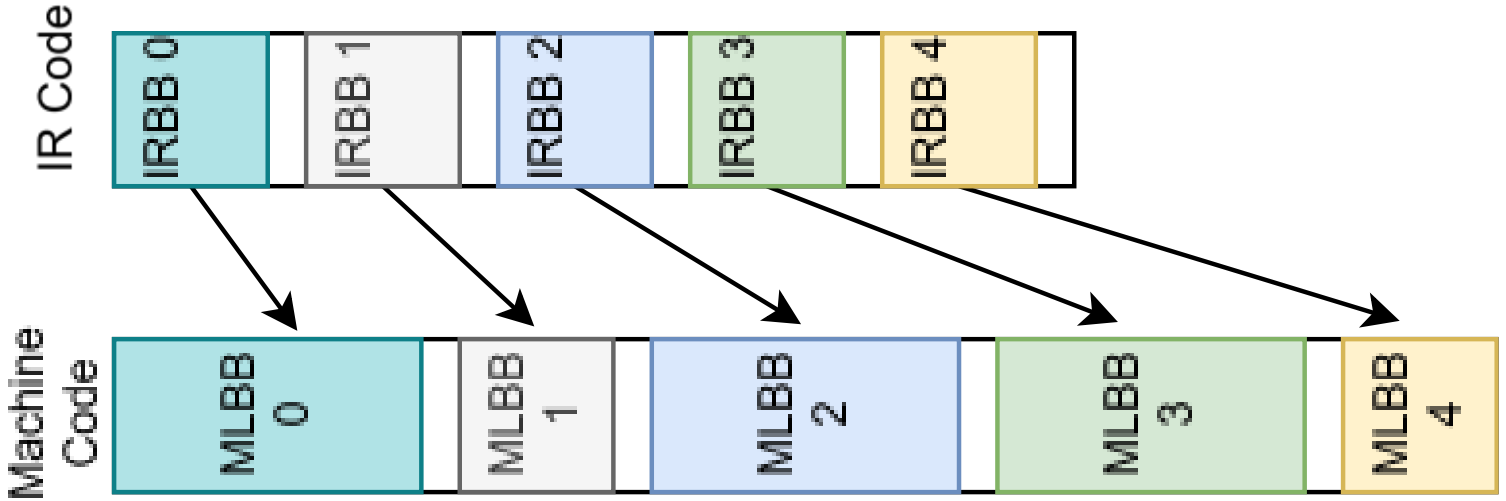
uphill ⇨ 10 min/km

flat ground ⇨ 4 min/km

uphill ⇨ 5 min/km

In Nugget's case



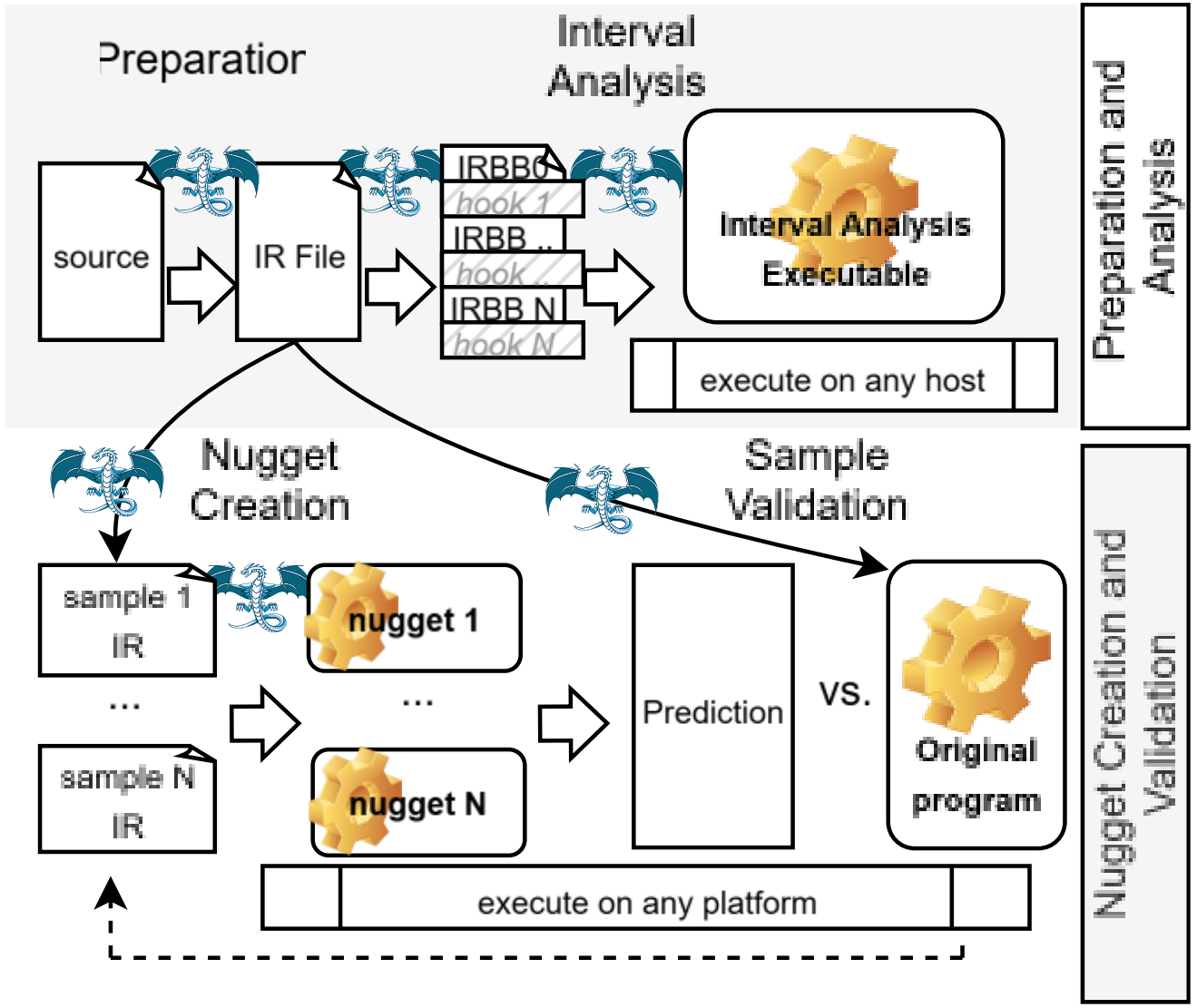


LLVM Intermediate representation (IR) is a compiler level representation.

The Nugget framework

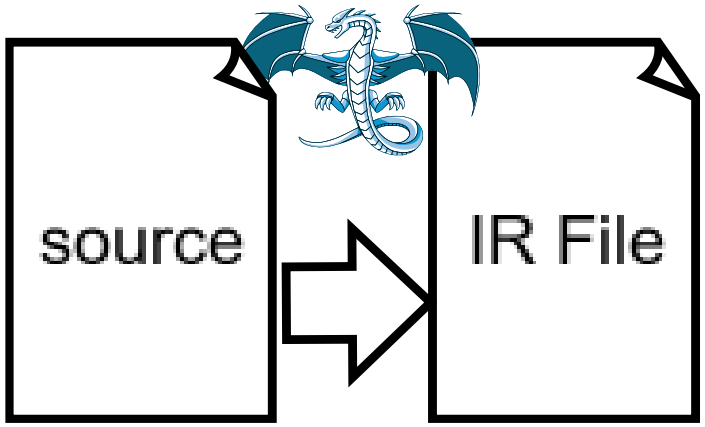
Drawbacks	Prior Works: SimPoint, LoopPoint	Nugget framework
Samples are expensive to find	Rely on simulation/tool to analyze program ✗	Create interval analysis program to **analyze on real hardware** ✓
Samples are tied to a single executable binary	Rely on machine code instruction to define interval ✗	Use **LLVM IR** to define interval ✓
Validating the selected samples is infeasible	Rely on simulation to validate samples ✗	Run samples on real hardware to **validate on real hardware** ✓

The Nugget Framework Pipeline

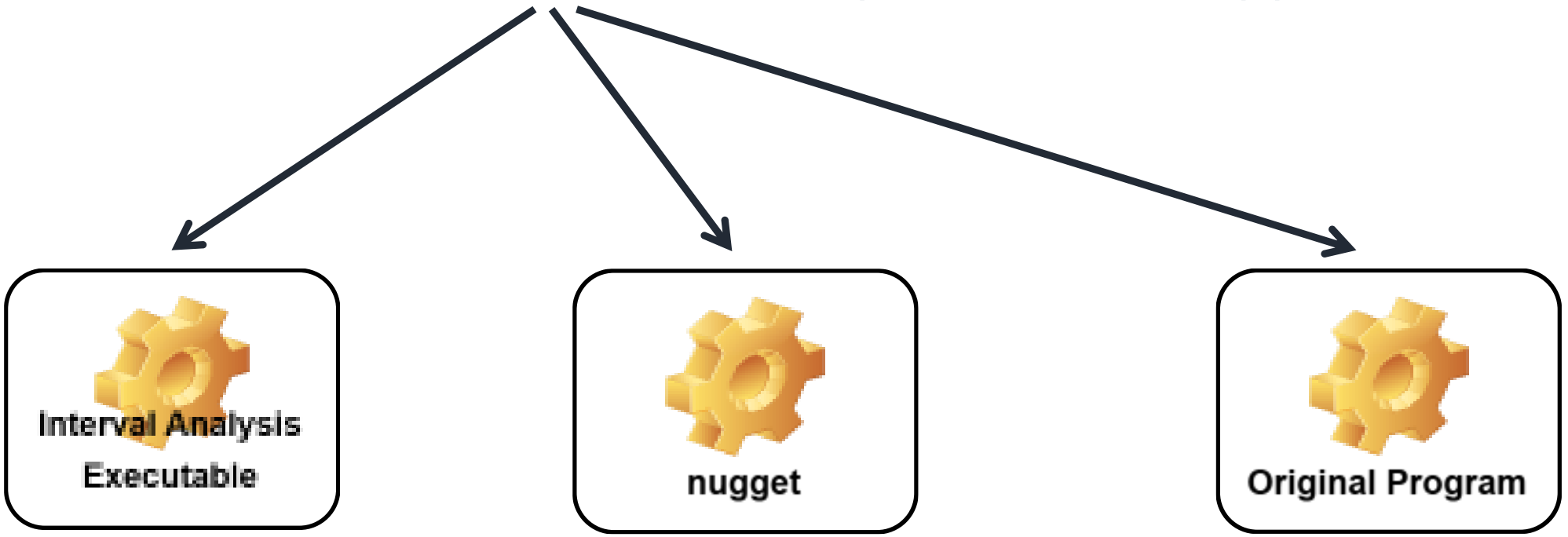


1. Preparation
2. Interval Analysis
3. Nugget Creation
4. Sample Validation

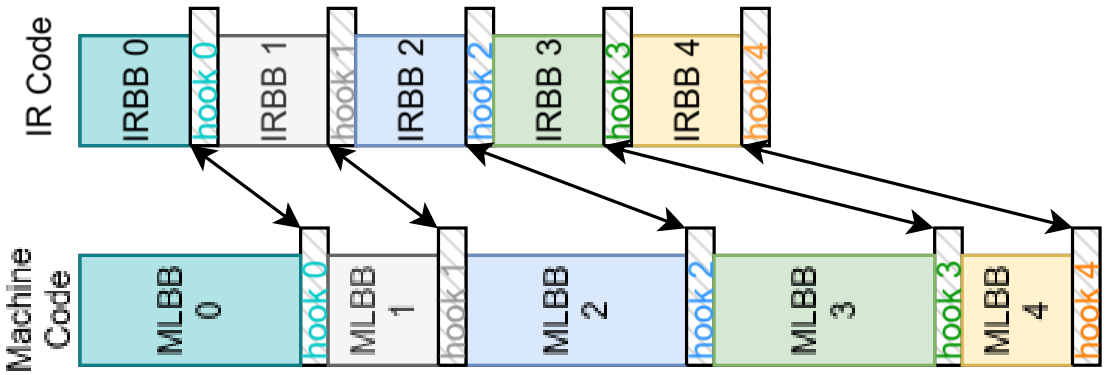
Preparation

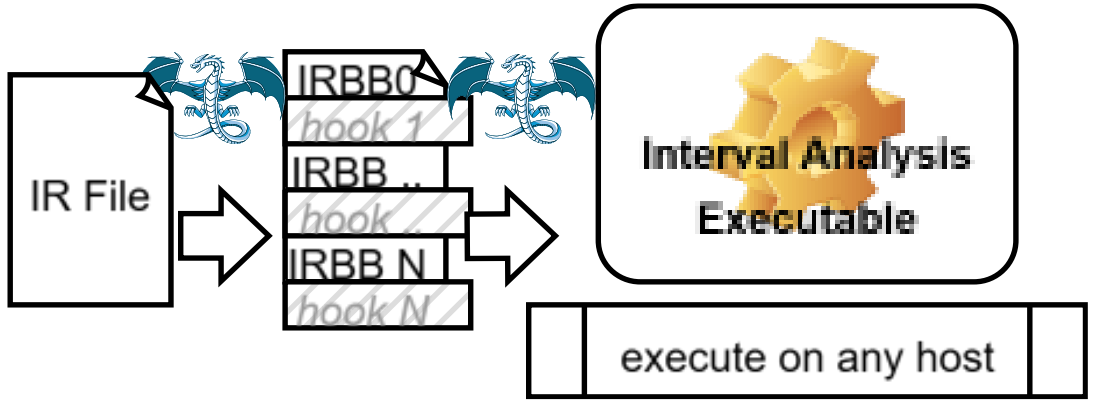


- **One base IR file only**: ensure the IR information is consistent across stages
- **Optimization:** frontend optimization is applied in this stage, and backend optimization is applied in other stages



```
hook:
  IR_bbv[IRBB_id] ++;
  IR_inst_counter += IRBB_inst;
  count_stamp_vector[IRBB_id] = IR_inst_counter
  if IR_inst_counter >= interval_length:
    call function();
```

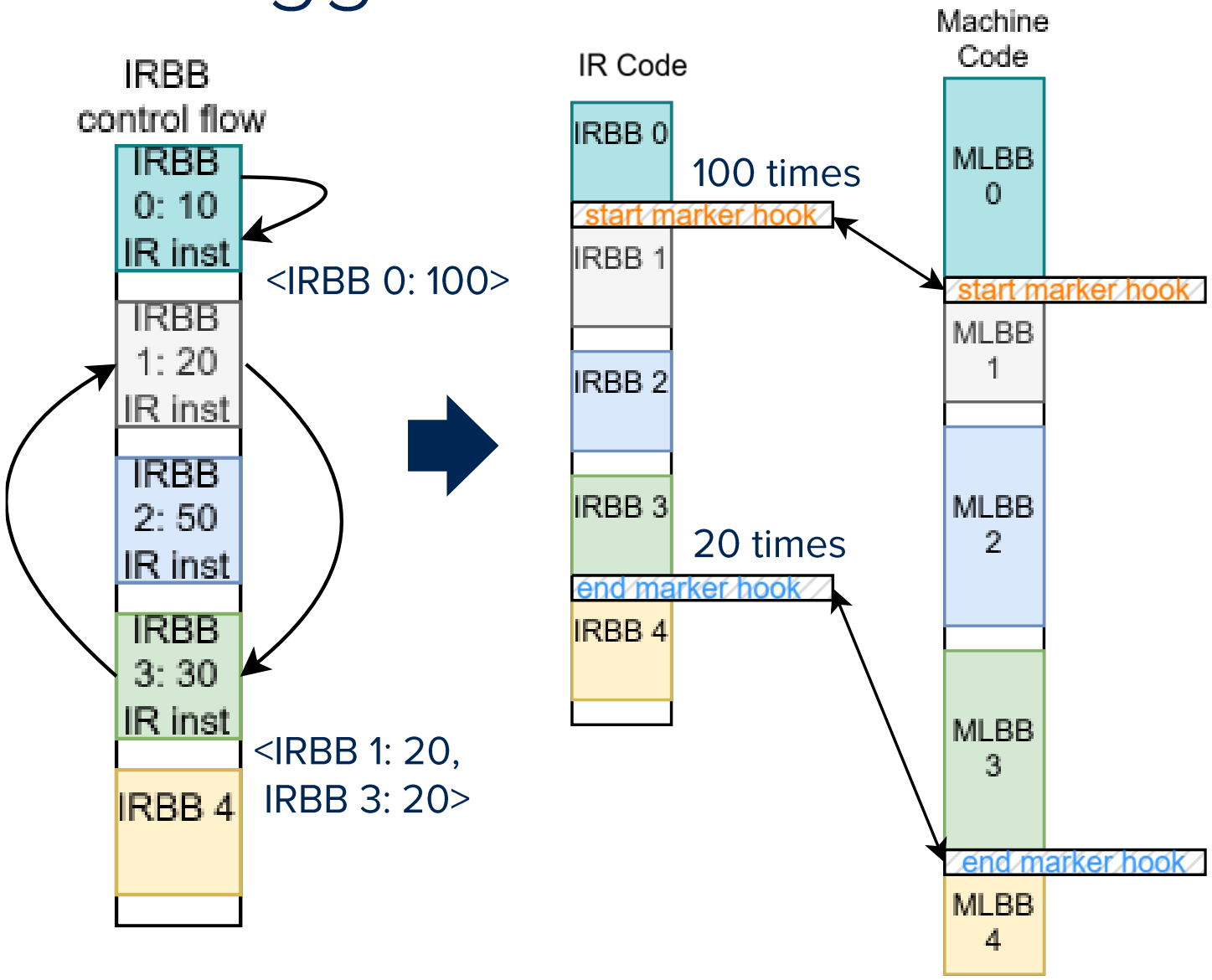




Interval Analysis

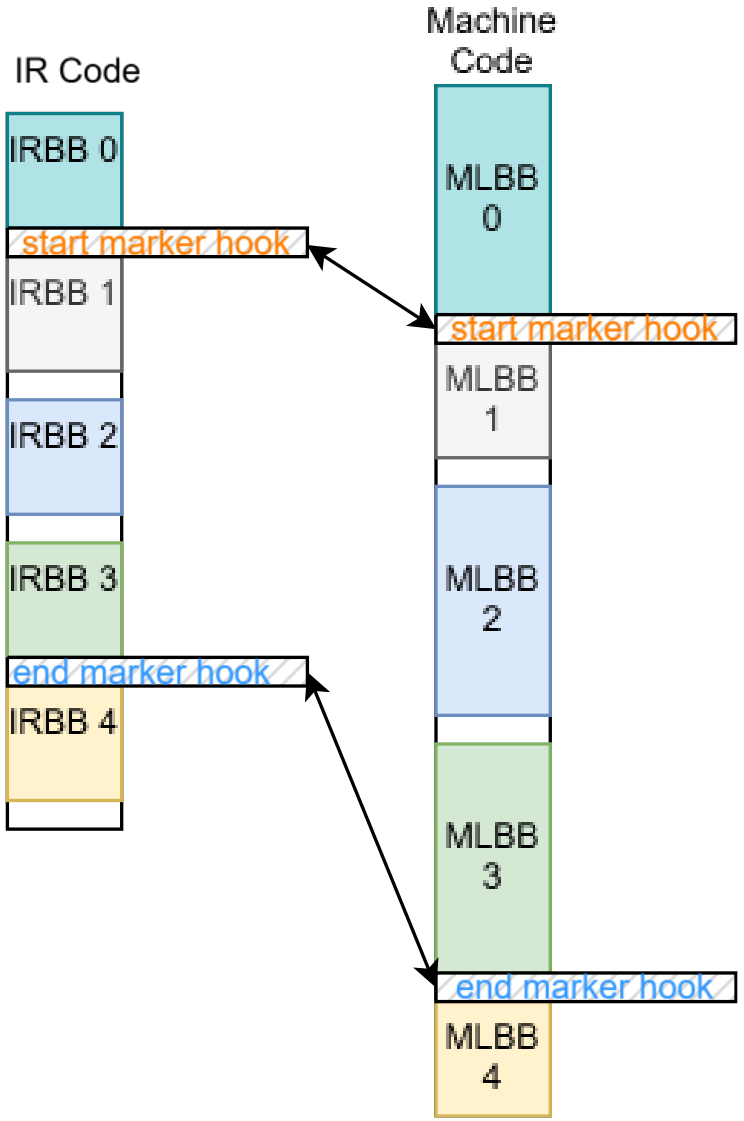
- **Unit of work:** # of IR instruction executed
- **IRBB** = LLVM IR basic block
- **Hook:** a lightweighted function
- Execution of the hook = execution of the IR basic block

Nugget Creation

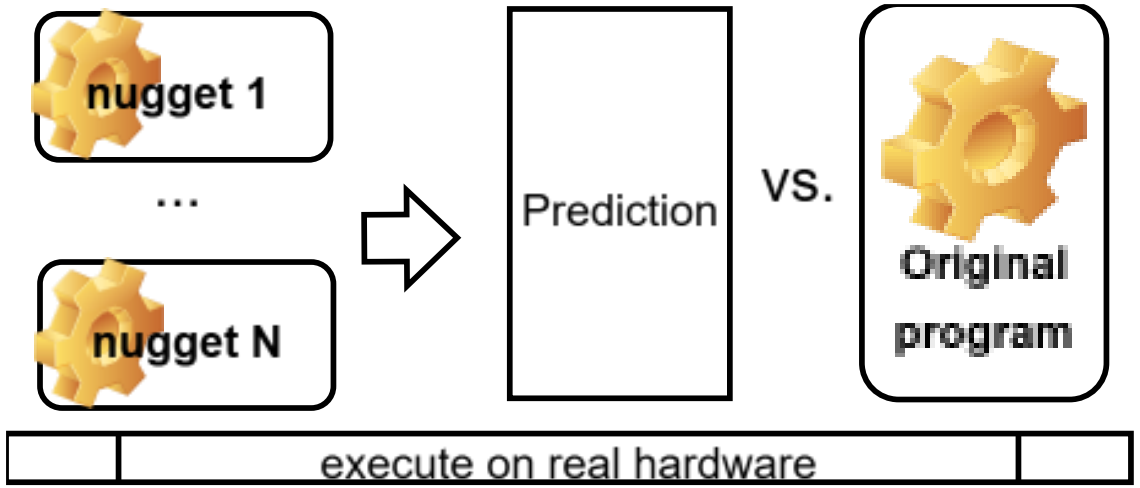


- **Marker:** identifies an execution point
- marker = (IRBB id, execution count)
- Markers bound the sample across platforms

Sample Validation



- Execute the samples and original program on the same real hardware to compare the predicted and measured metrics



Evaluation

1. Measure interval analysis overhead
2. Demonstrate portability across architecture (x86, aarch64)
3. Validate sample selection (prediction v.s. measurement)

We also did case studies on

1. Isolate ISA v.s. microarchitecture effects on program behavior by using nuggets in simulation
2. Evaluate simulator fidelity using nuggets (compare to real hardware)

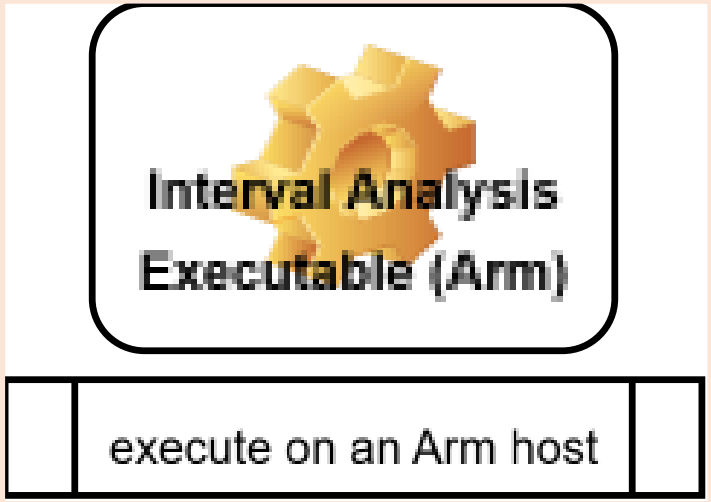
Evaluation Setup

- HW: 24-core Ryzen 3960X (x86) & 160-core Ampere Altra (Arm)
- Simulation: gem5 with full-system mode
- Suites: SPEC2017 (single-thread), NPB (OMP), LSMS (single-thread)
- Inputs: reference for SPEC2017, class A/C/D for NPB, and Fe for LSMS

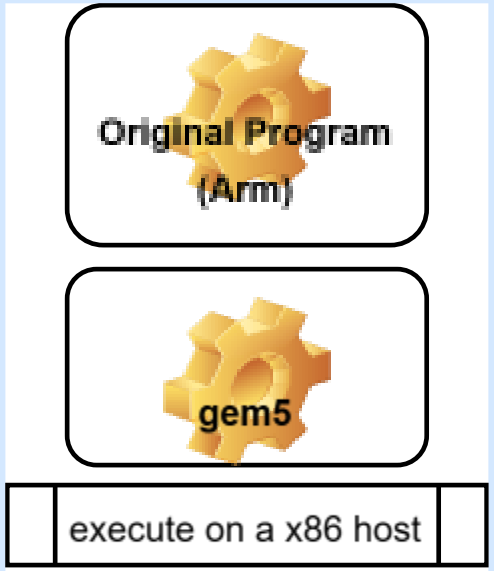
Measure interval analysis overhead

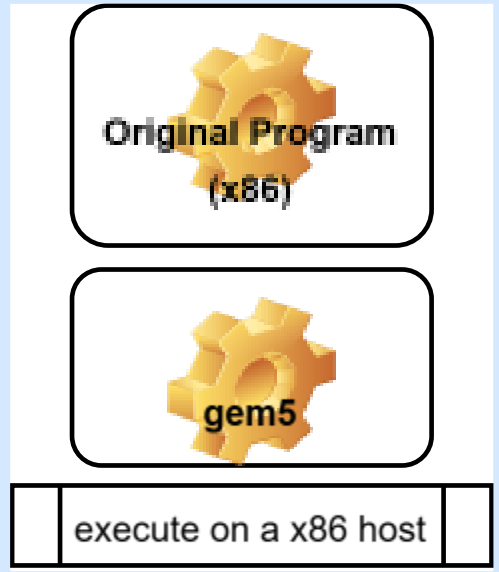
Compared to functional simulation in gem5, Nugget reduces analysis overhead by **578X** on 4-threaded NPB (Class A).

Overhead is small for single-threaded: **3X** (SPEC/LSMS); higher for multi-threaded: **34X** (NPB).



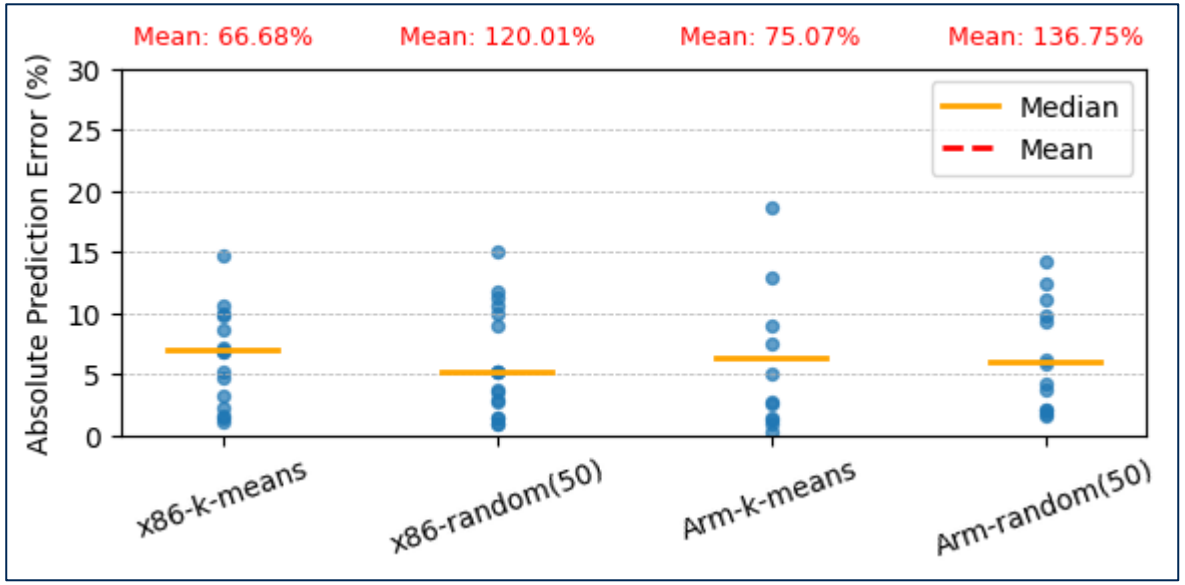
Nugget: average **54X** with a maximum of **118X**





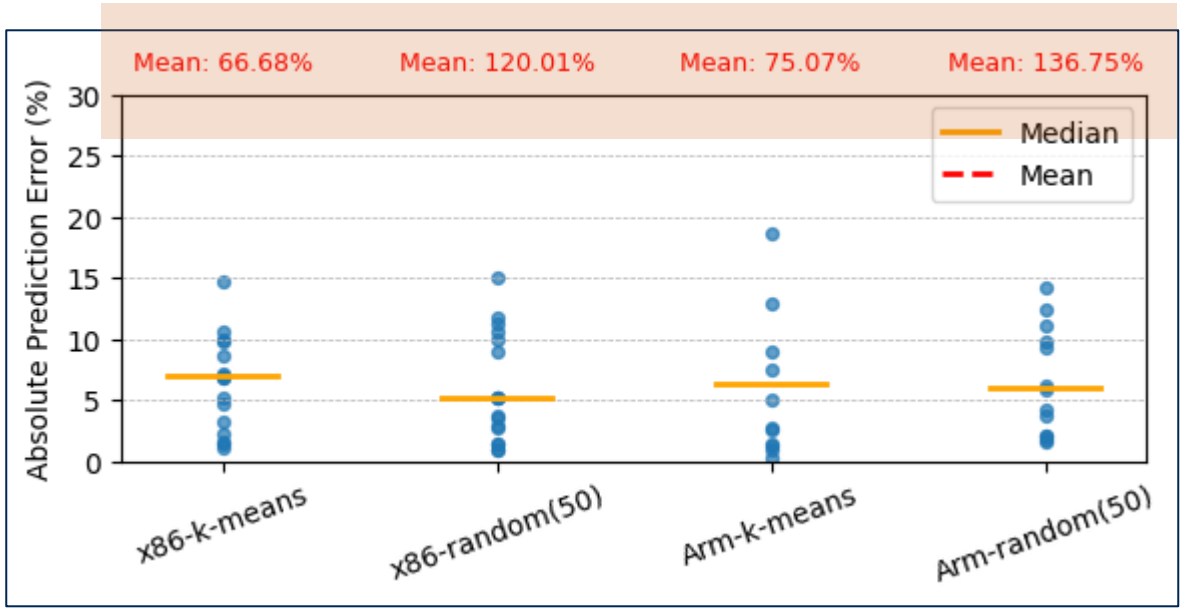
Functional simulation: average **31,343X** with maximum **55,497X**

Runing the same set of samples across machines with different architectures



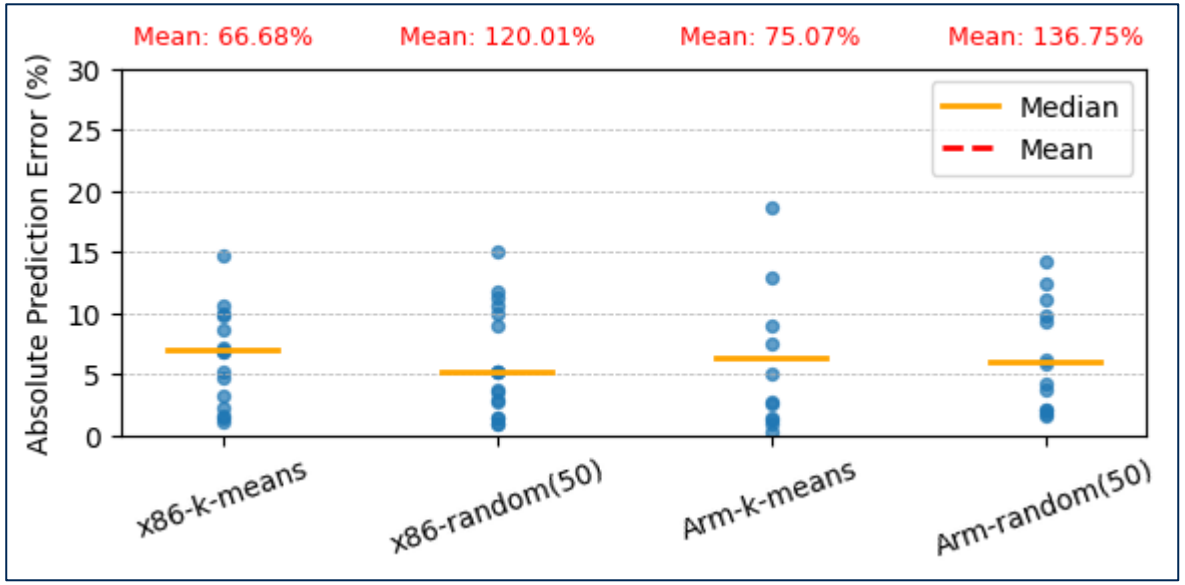
- We used two simple sample selection methods: **K-means** and **Random** with K=50
- We used the samples to predict the **runtime**
- “is” has a high prediction error up to 2157.43%

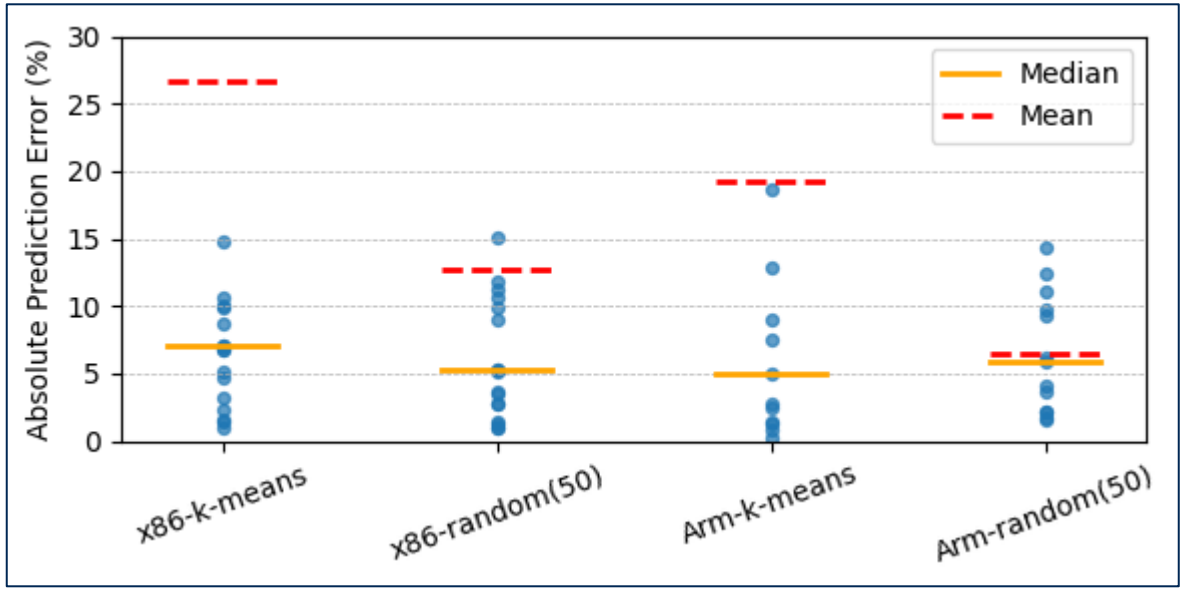
Runing the same set of samples across machines with different architectures



- We used two simple sample selection methods: **K-means** and **Random** with K=50
- We used the samples to predict the **runtime**
- “is” has a high prediction error up to 2157.43%

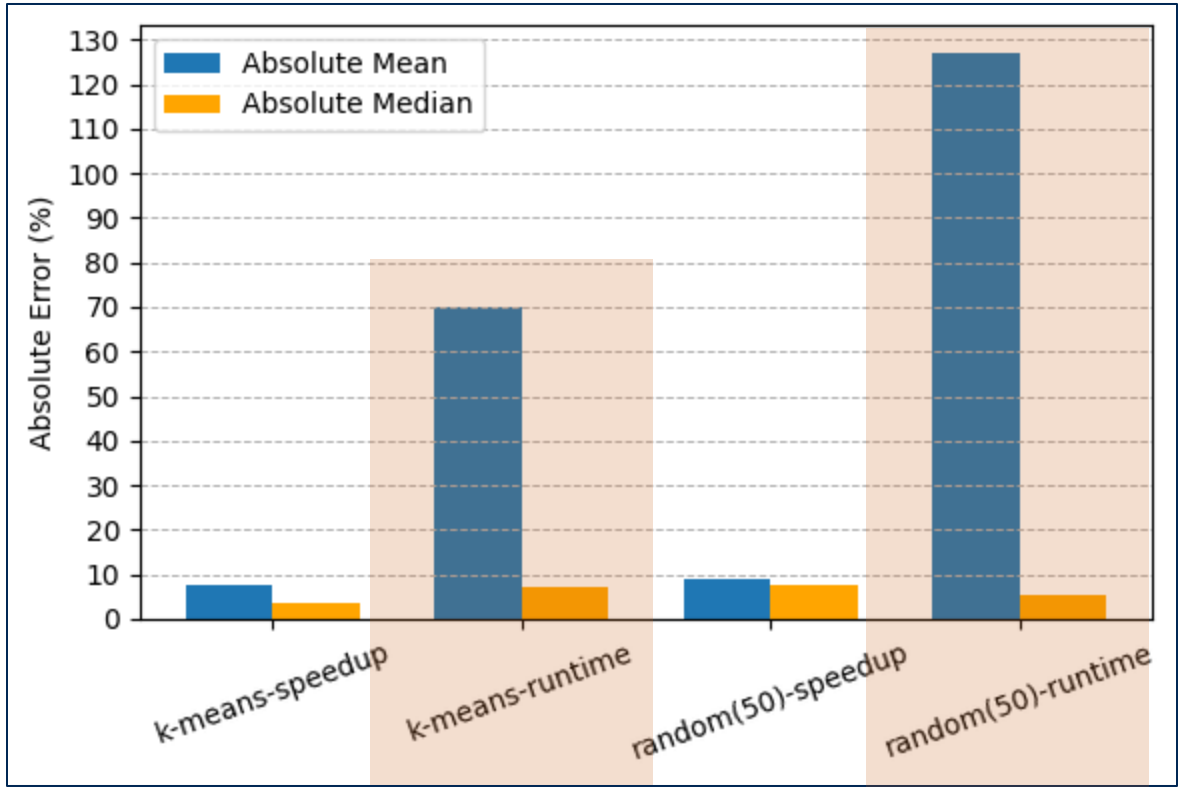
Runing the same set of samples across machines with different architectures





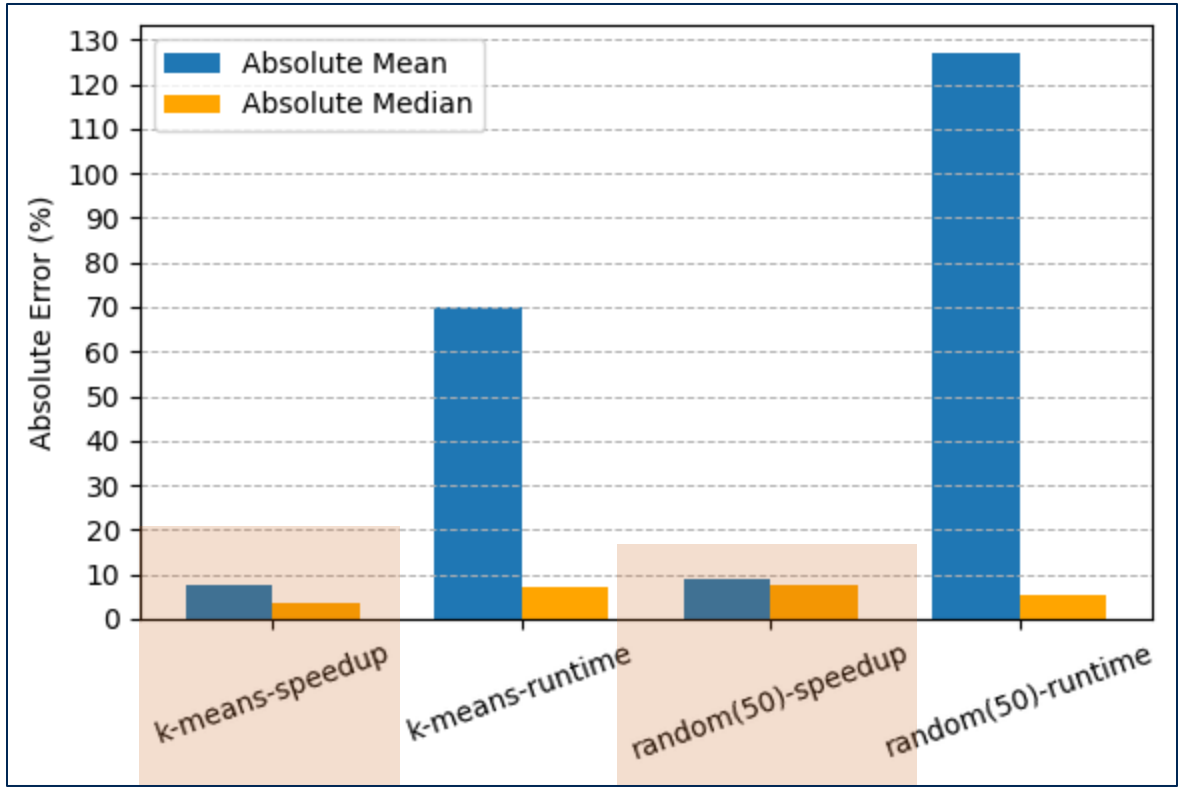
- We used two simple sample selection methods: **K-means** and **Random** with K=50
- We used the samples to predict the **runtime**
- “is” has a high prediction error up to 2157.43%

Look at predicting speedup



- Goal: predict the performance difference between designs
- Predicting speedup is more important than predicting runtime

Look at predicting speedup



- Goal: predict the performance difference between designs
- Predicting speedup is more important than predicting runtime

consistency across machines is a stronger indicator of sample quality than low error on one system

Case studies

1. Isolate ISA v.s. microarchitecture effects on program behavior by using nuggets in simulation

Takeaway: µarch impacts program behavior more than ISA

2. Evaluate simulator fidelity using nuggets (compare to real hardware)

Takeaway: nuggets can be used as realistic microbenchmarks

Nugget is just a start

1. What new sampling methodologies can be created using Nugget? What LLVM IR information that can represent hardware performance phases?
2. Can we skip the fast-forwarding to the start of an interval?
3. …

Conclusion

- Nugget enables fast program execution analysis on real hardware (**578X** faster than using function simulation).
- The analysis is a one-time cost—samples can be reused across different architectures.
- Selected samples can be validated quickly for the target benchmark and input size.
- Overall, Nugget increases confidence in the prediction/estimation results.

- GitHub repo: studyztp/Nugget-LLVM-passes