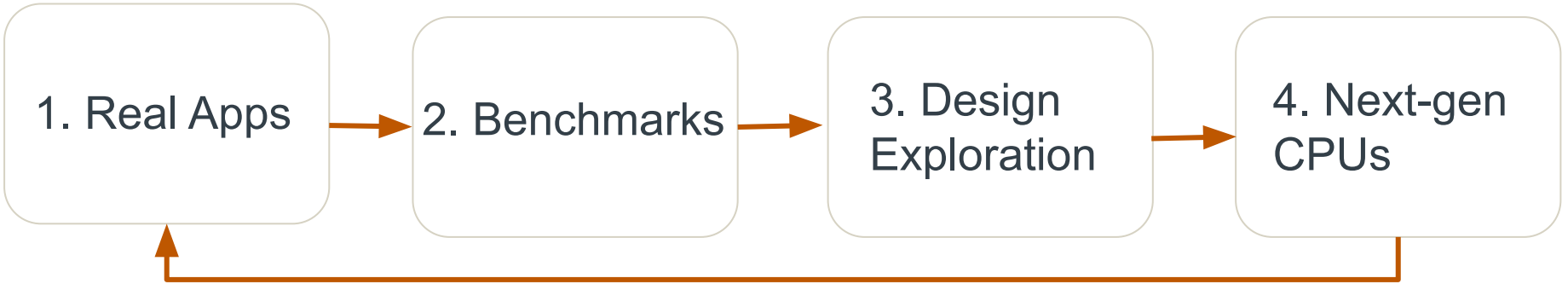
March 23, 2026
Architecture 2.0 - ASPLOS 2026

# Cloning the Unshareable: Agentic AI for Synthesizing Open, Production-Faithful Datacenter Benchmarks

Alan Andrade , Petar Acimovic, Wei Su, Jovan Stojkovic

# The Datacenter Hardware Design Loop

## Cloud computing drives server design




Performance signals: IPC, miss rates …

# No Good Benchmarks Exist

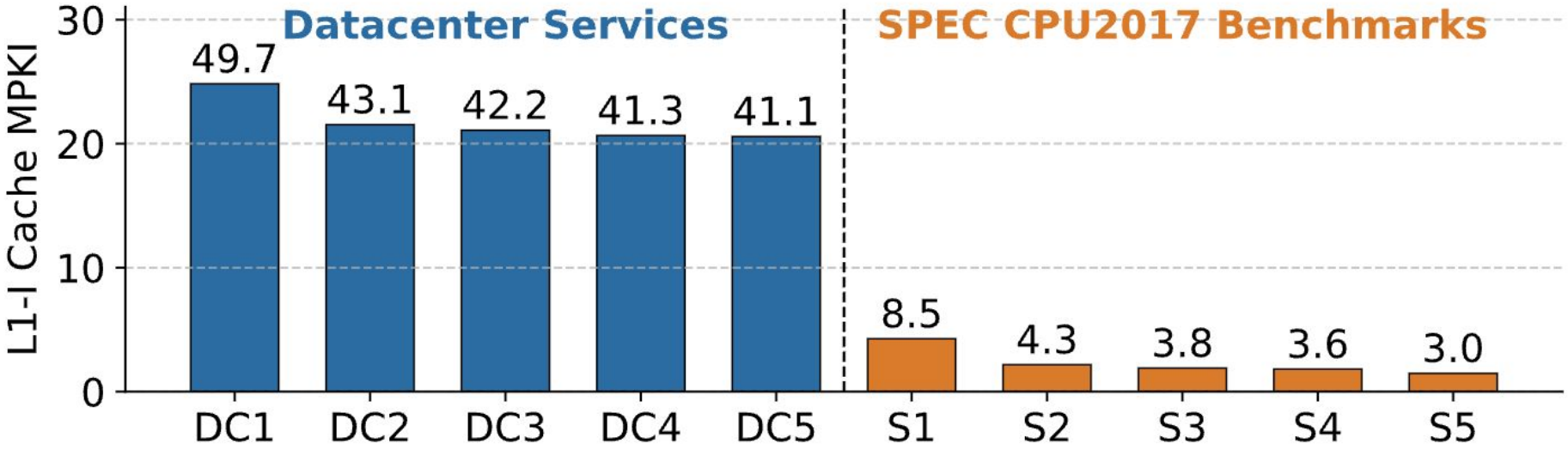
- Production services are confidential
- Vendors and academics lack access to real apps
- Gap between what architects optimize for vs. what actually runs

# WHY EXISTING BENCHMARKS FAIL

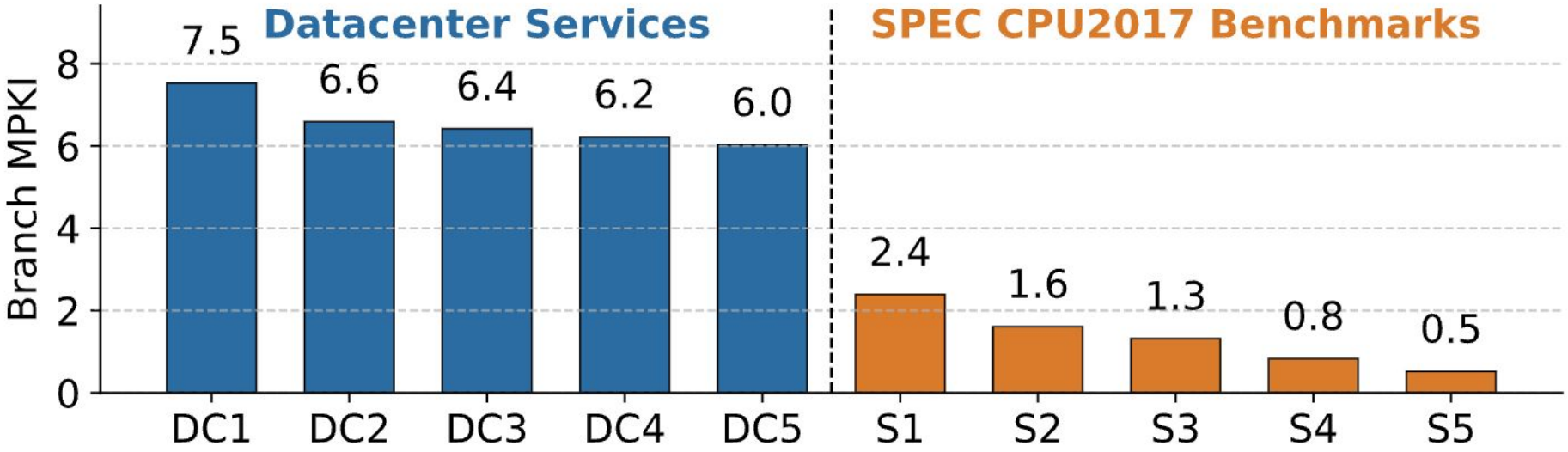
# SPEC CPU2017 != Datacenter

- SPEC dominates CPU evaluation for decades
- Datacenter workloads look fundamentally different
- Optimizing for SPEC can mislead datacenter processor design

# I-Cache Pressure: DC vs. SPEC



# Branch prediction: DC vs. SPEC



# Manual Proxy Engineering (DCPerf)

- Dedicated teams, domain expertise, maintenance
- Artificial tools (I-cache boosting) break real patterns
- Becomes stale as software stacks evolve
- Doesn't scale to pace/diversity of modern datacenter software

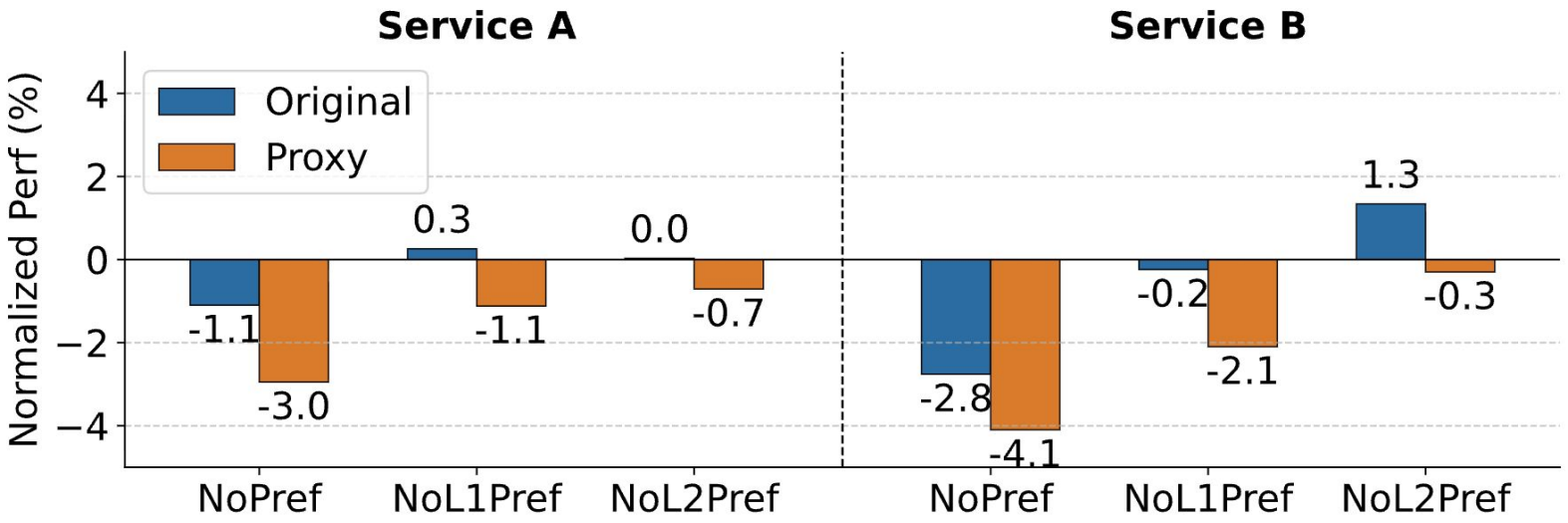
# The Sensitivity Problem

Matching counters on ONE configuration is not enough

**Benchmarks exist to evaluate design changes**

Does the proxy respond the same way when you change the HW?

# Experiment: Prefetcher Sensitivity



# Implications for Architects

For example, If your proxy says "disabling L1 prefetchers improves 1%" but the real service loses 5% ➡ we make the wrong design trade-off

**Sensitivity divergence undermines every downstream design decision**

# Same Counters, Different Sensitivity

Program A: strided array walk ➞ 5 L1D-load MPKI, prefetcher learns pattern ➞ large speedup

Program B: pointer chase ➞ 5 L1D-load MPKI, prefetcher can't help ➞ no speedup

**Same miss rate, opposite prefetcher sensitivity**

# WHY IS THIS HARD?

# 1. The Multi-Objective Challenge

- Must match many counters simultaneously
- Objectives can conflict
  - Raising branch MPKI may inflate I-cache MPKI
- Fixing one metric can break another

# 2. The Sensitivity Challenge

Beyond counters: must match response to HW design changes

**Second-order reasoning: not "does the benchmark miss?" but "does it miss for the right reasons?"**

# OUR APPROACH

# Agentic AI for Benchmark Cloning

Iterative, feedback-driven synthesis

Large search space and dense measurable reward signal is a natural fit for agents

Autonomous loop: measure -> diagnose -> transform -> re-measure

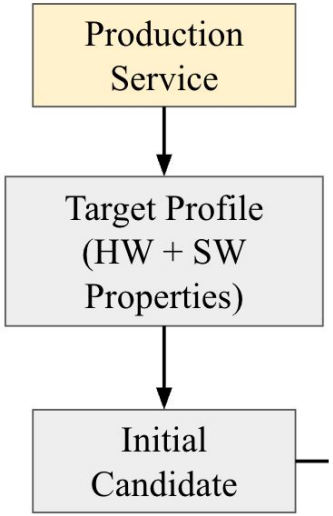
# Starting from a Knowledge Database

Cold-start problem: LLMs struggle generating code with a target IPC from scratch

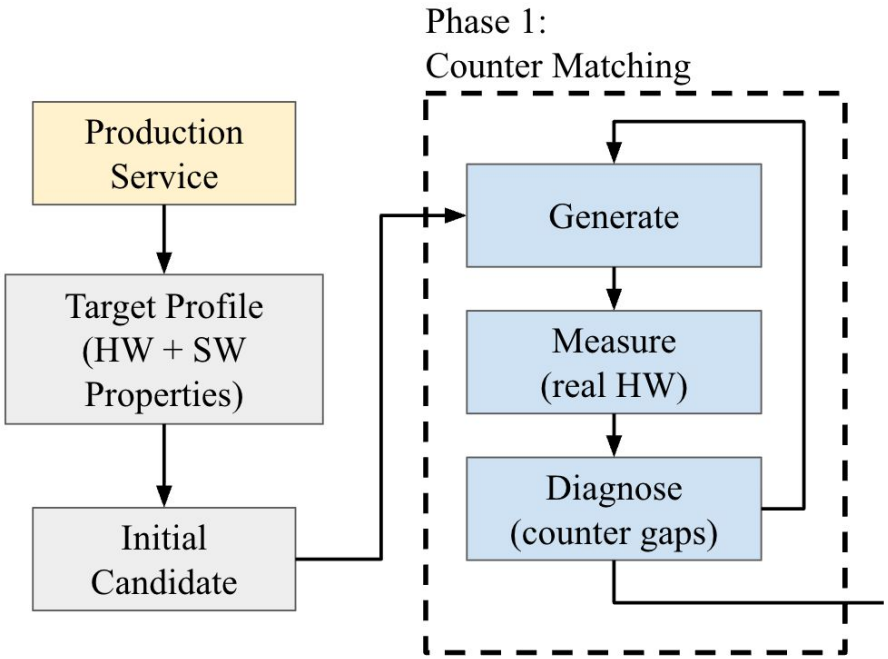
**Retrieve similar microbenchmarks from a pre-profiled corpus**

- Start from some similar accuracy instead of synthesizing from nothing

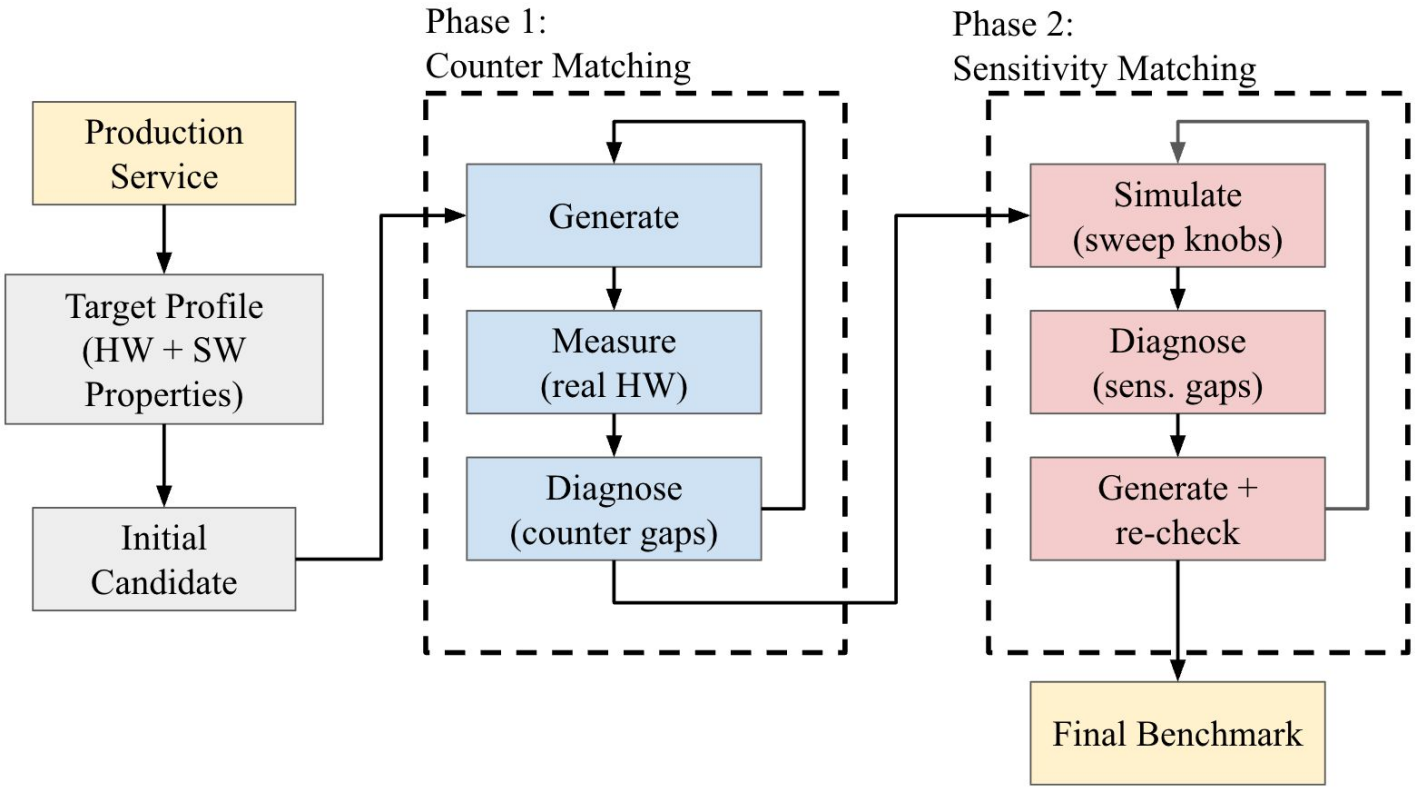
# Two-Phase Approach



# Two-Phase Approach



# Two-Phase Approach



# The Role of Diagnostic Reasoning

**Separating "what's wrong" from "how to fix it"**

Microarchitecture-aware reasoning maps gaps to code transformations

Example: "IPC too high" ➞ decompose via TMA ➞ "frontend too efficient" ➞ "inflate I-cache pressure"

# Tiered Fidelity: Not All Counters Are Equal

**IPC is primary; I-cache and branch MPKI are tier 1 (dominant DC bottleneck)**

- Lower tiers: BTB, D-cache, TLB, pipeline balance

# Sensitivity Matching: Second-Order Reasoning

**Not the "right miss rate" but the "right kind of misses"**

Example: benchmark benefits too much from advanced prefetcher:

- Access pattern too regular ➡ add indirection to defeat all Prefetchers equally
- Fix reshapes the pattern, not the miss rate

# Privacy by Design

- Never sees source code or binary
- Only reason from statistical profiles
- All generated code stems from open-source microbenchmark repositories or is newly LLM-generated
- Safe for open-source release

# EVALUATION

# Evaluation

- DCPerf benchmarks (Mediawiki, Django, FeedSim, TaoBench, etc.)
- Production workloads from a hyperscaler
- Our system matches the behavior of all applications across different counters and sensitivities with 80-100% accuracy

# Conclusions

- Lack of representative Datacenter benchmarks prevents hardware innovation
- Manually creating proxy benchmarks is expensive and not scalable
- We propose an agentic AI framework to automatically design benchmarks with high fidelity

# QUESTIONS?

# THANK YOU!

andrade@cs.utexas.edu