Mimesys:
Generating Realistic Executable Testing Environments from Resource Usage Traces

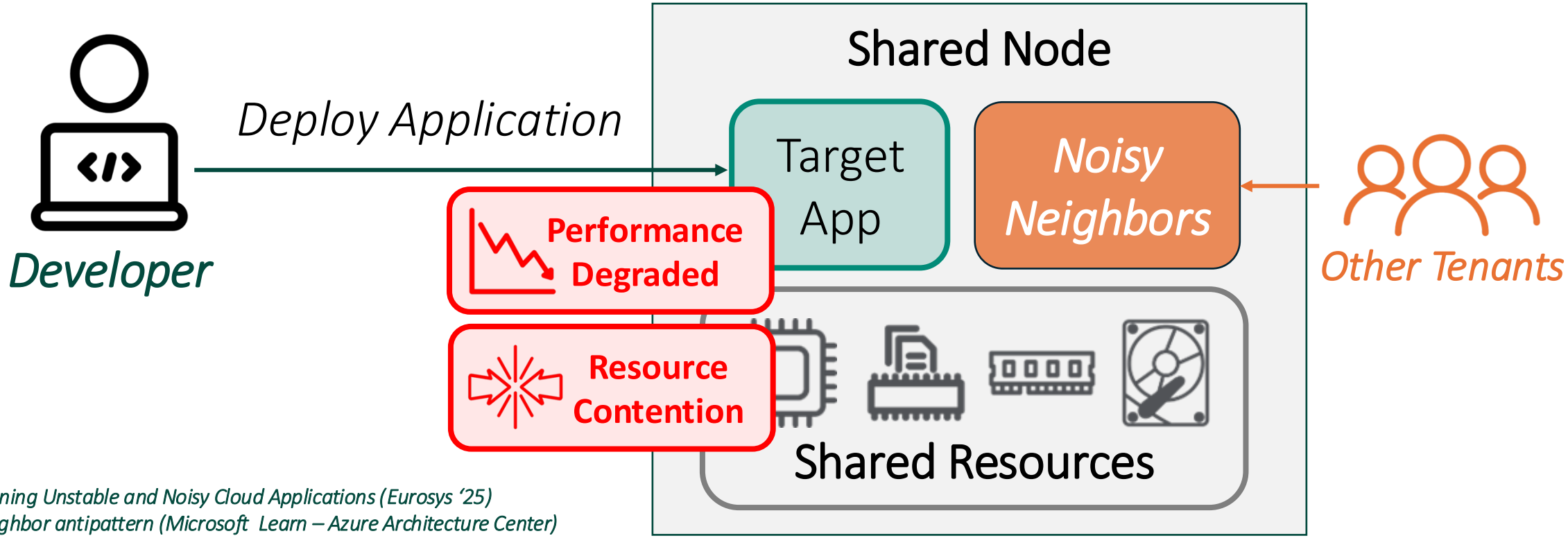
Donghyun Kim, Zichao Hu, Joydeep Biswas, Aditya Akella, Daehyeok Kim

The University of Texas at Austin

20th USENIX Symposium on Operating Systems Design and Implementation (OSDI '26)

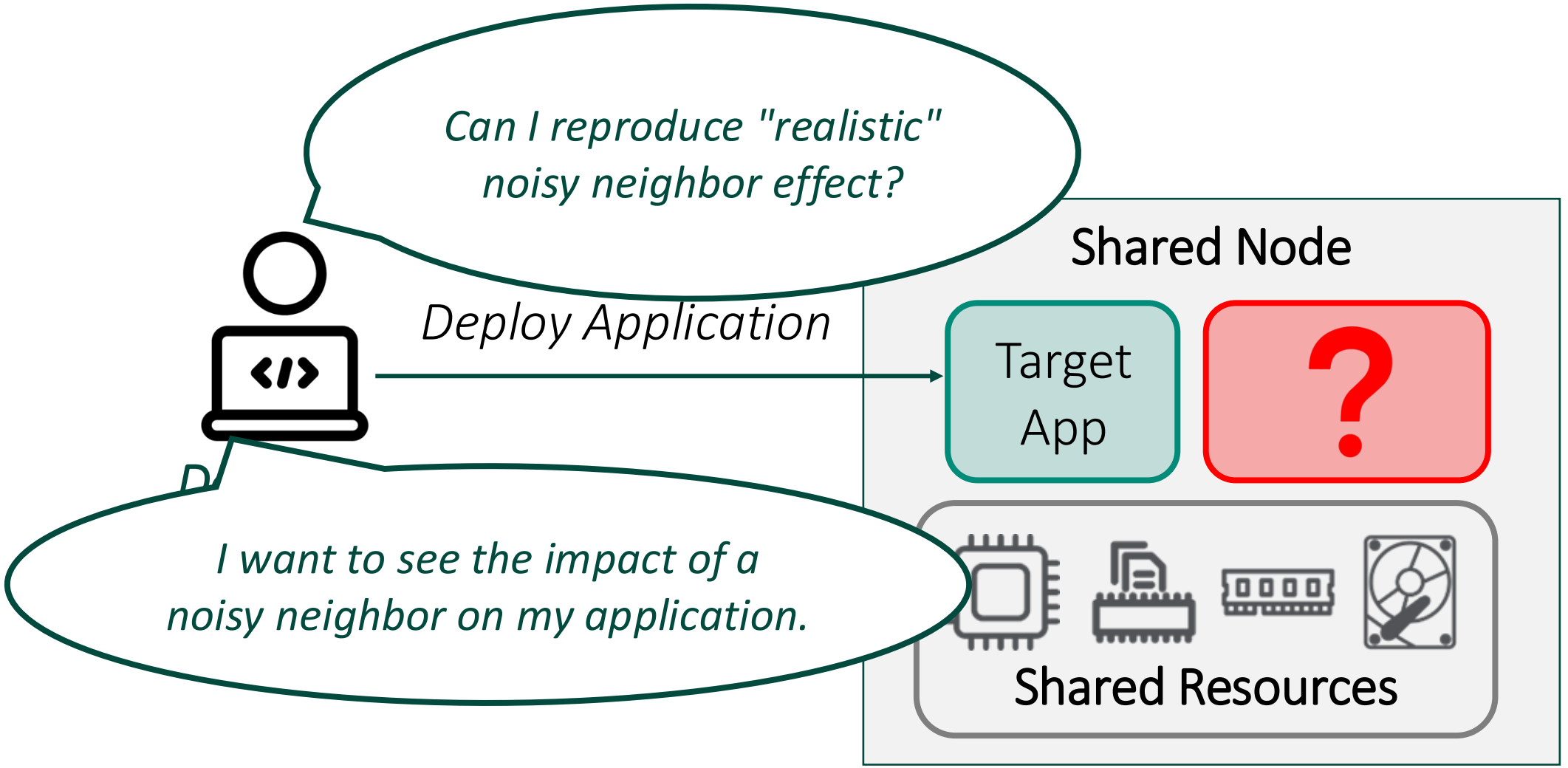
Noisy Neighbors Are Everywhere

In the cloud, co-located workloads can significantly impact application performance by over 30-70% in realistic deployments [1,2]



[1] TUNA: Tuning Unstable and Noisy Cloud Applications (Eurosys '25)
[2] Noisy Neighbor antipattern (Microsoft Learn – Azure Architecture Center)

Developers Need Realistic Noisy Neighbors



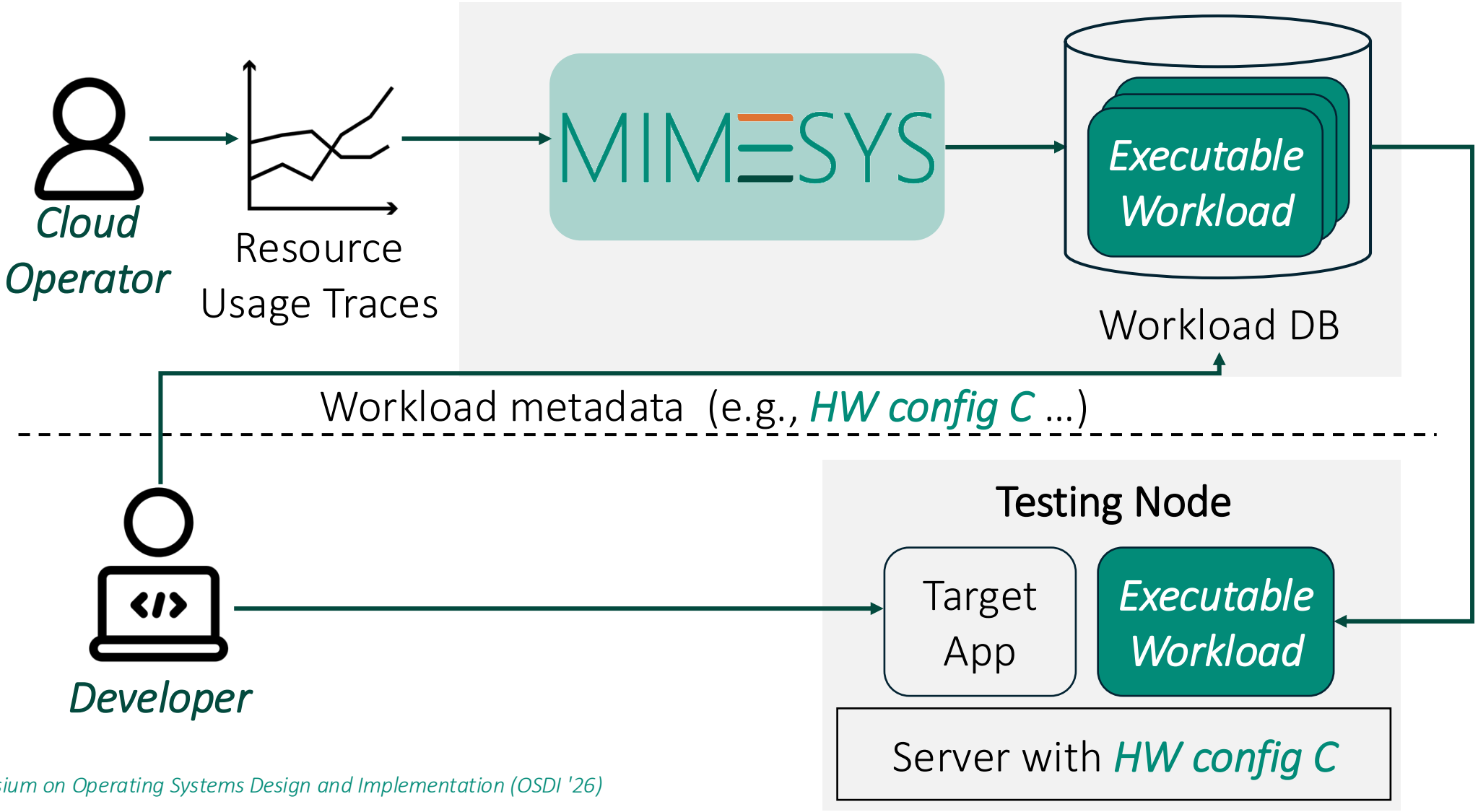
No Existing Approach Meets All Requirements

	Resource Usage Traces	Resource Stressors	Benchmarks	Application Cloning
Executable workloads	✗	✓	✓	✓
Multi-resource interactions	✓	✗	✓	✓
Temporal patterns	✓	✗	✓	✗
Ease of diversification	✓	✓	✗	✗

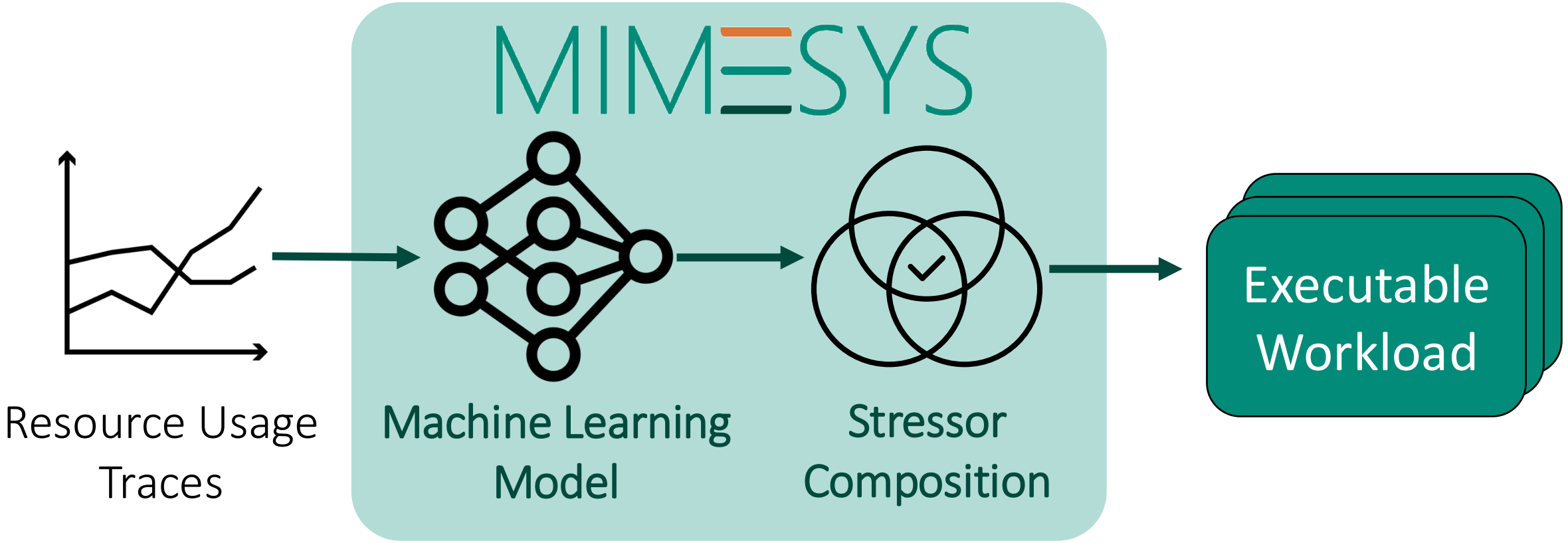
[1] Ditto: End-to-End Application Cloning for Networked Cloud Services (ASPLOS '23)

Can we synthesize "executable" workloads from resource usage traces?

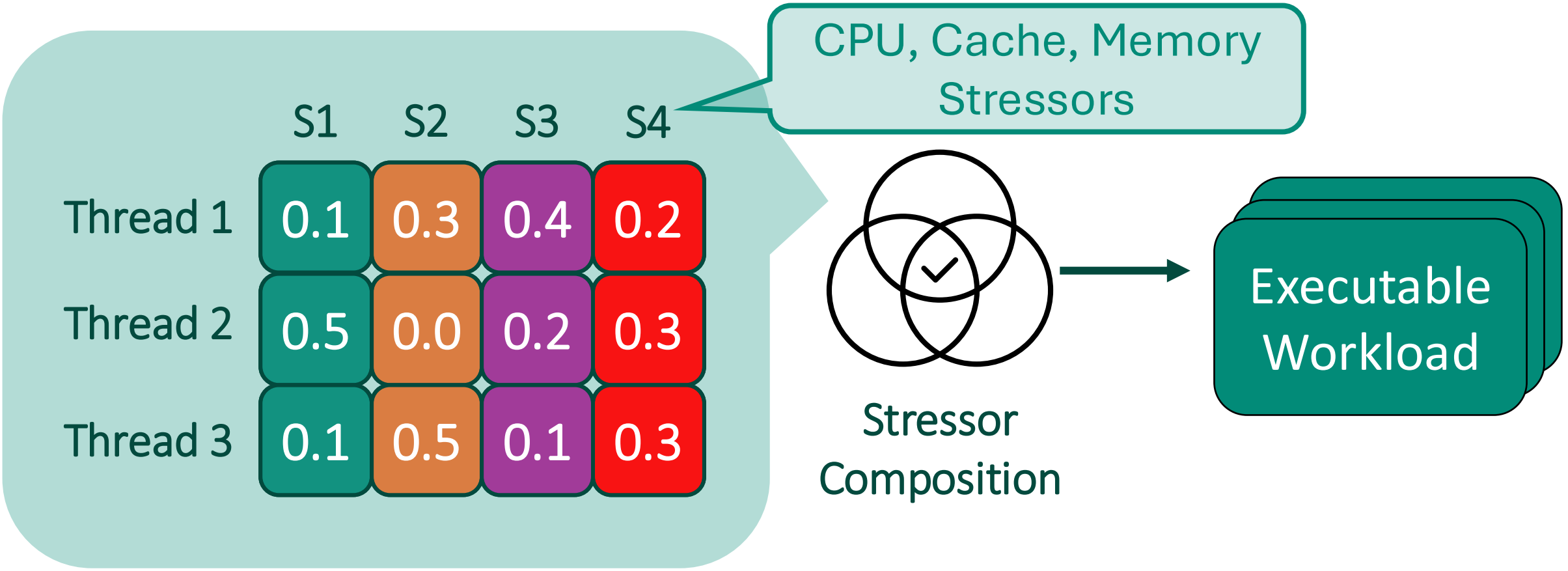
Our Vision: Generating Workloads From Traces



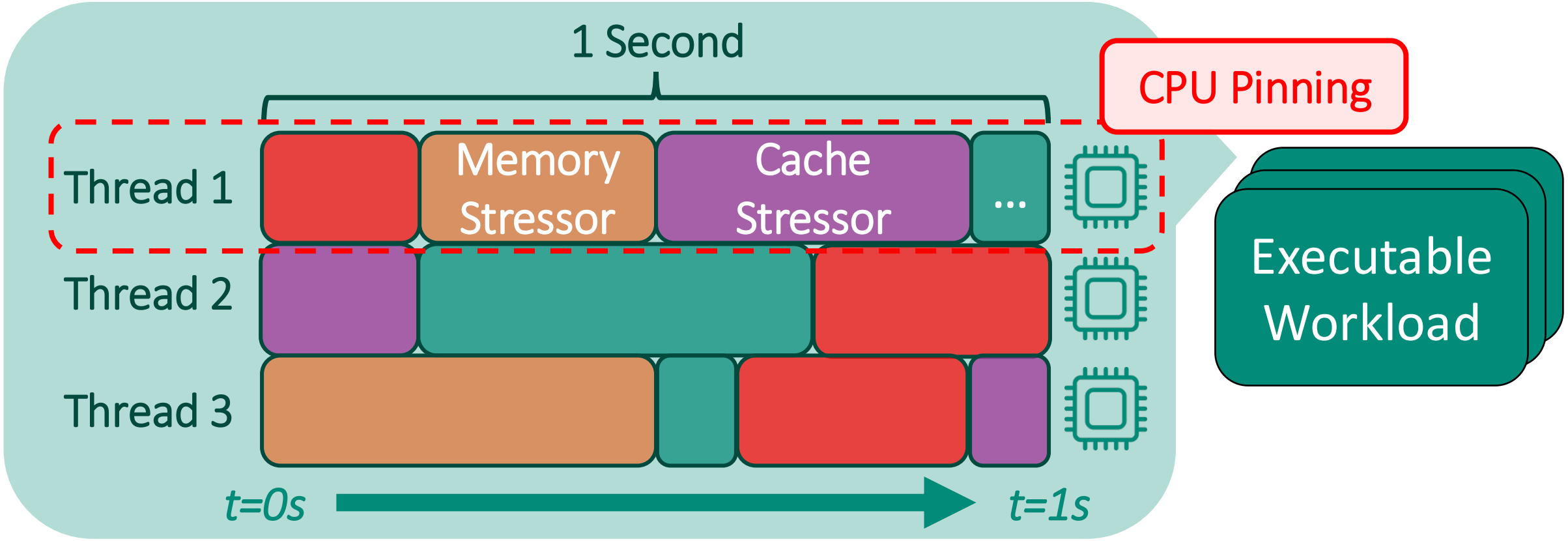
Our Approach



Stressor Composition



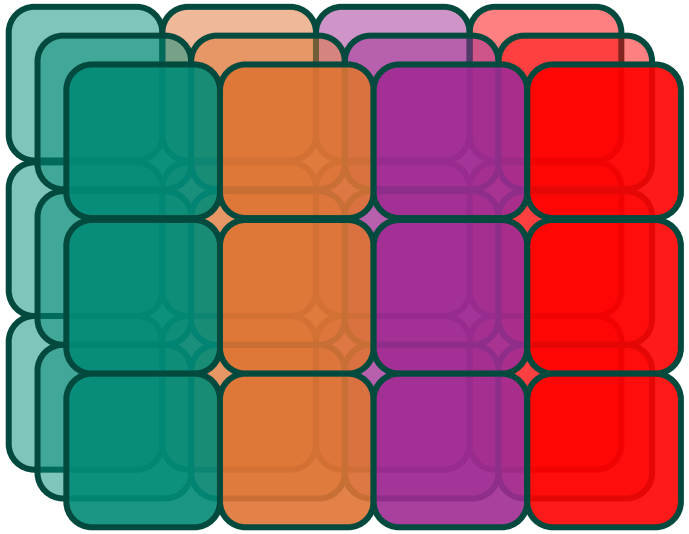
Stressor Composition

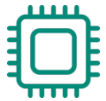


Challenge 1: Temporal & Cross-resource Relationships

The mapping from stressor compositions to resource usage is complex.

- Cross-resource: Memory hierarchy, Concurrency, …
- Temporal: Cache warming, Page cache, …





CPU: Per-core Utilization, Avg. IPC

Cache: LLC Occupancy, Cache traffic

Memory: Read/Write Bandwidth

Disk I/O: Read/Write Bandwidth

Resource- and State-Aware Diffusion Model

We train a Diffusion-based model to generate stressor compositions to match target resource usages

Why Diffusion?

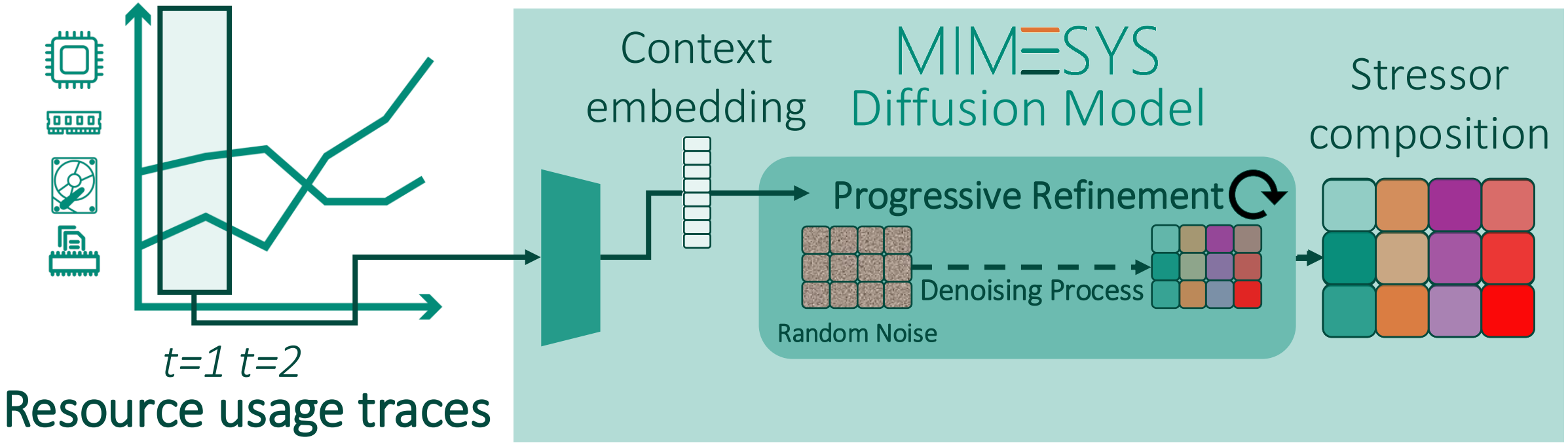
- Flexible conditioning
- One-to-many generation
- Strong generalization

MIMESYS
Diffusion Model

Resource- and State-Aware Diffusion Model

We train a **Diffusion-based model** to generate stressor compositions to match target resource usages

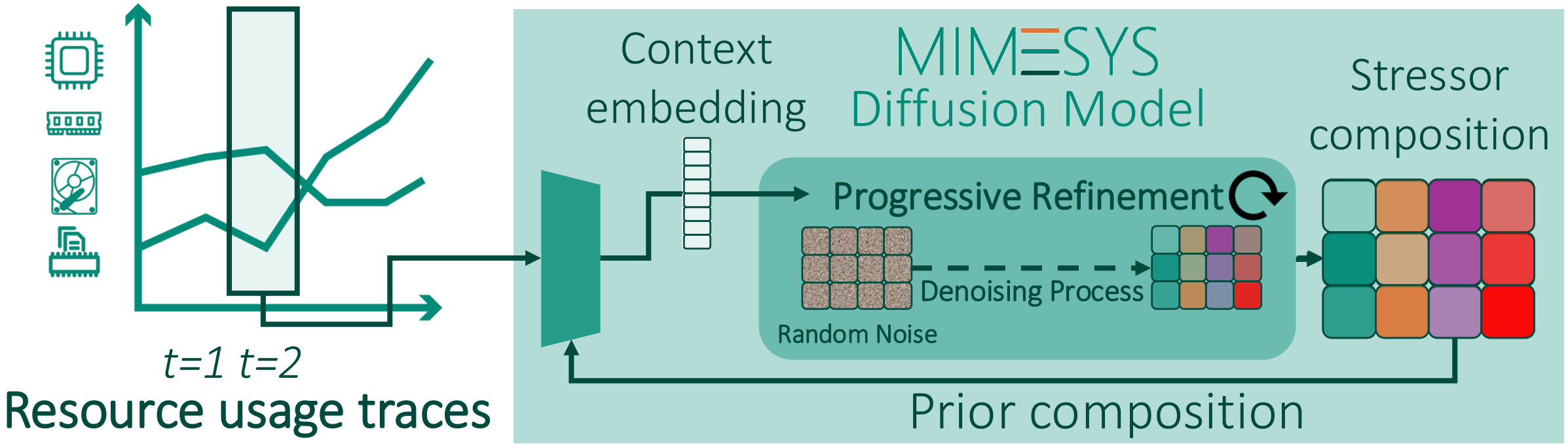
- **Conditioning on** 1) resource traces



Resource- and State-Aware Diffusion Model

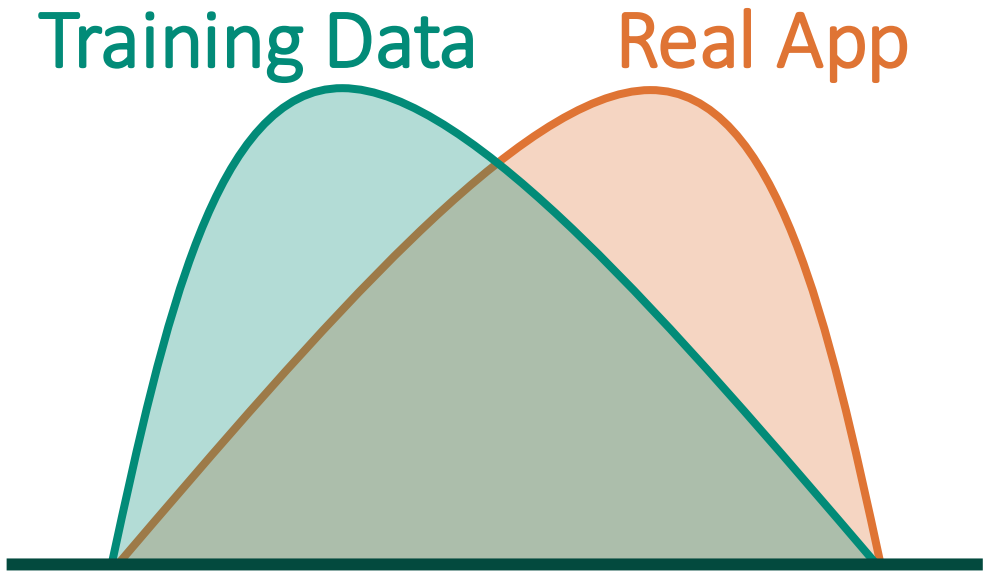
We train a **Diffusion-based model** to generate stressor compositions to match target resource usages

- **Conditioning on** 1) resource traces 2) prior stressor composition

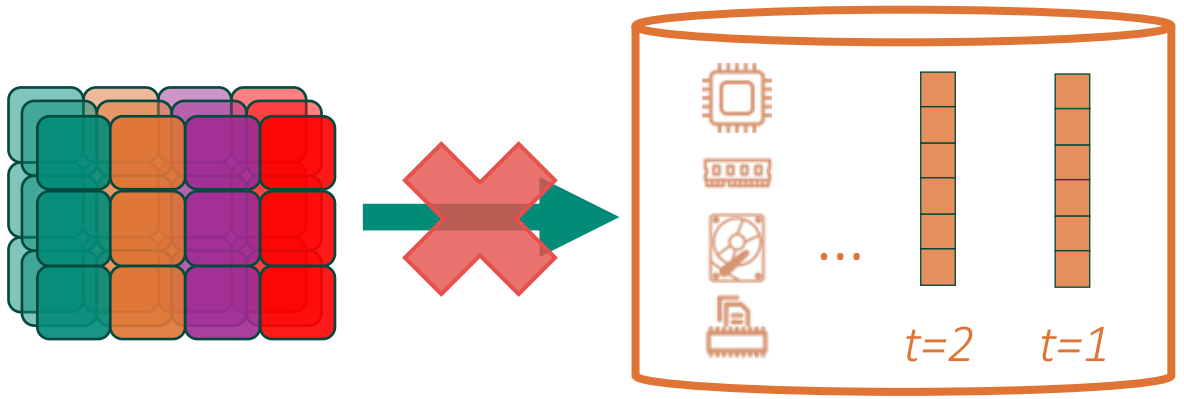


Challenge 2: Aligning with Real Applications

The Diffusion model is trained on synthetic data, not real application data



Resource usage distribution gap

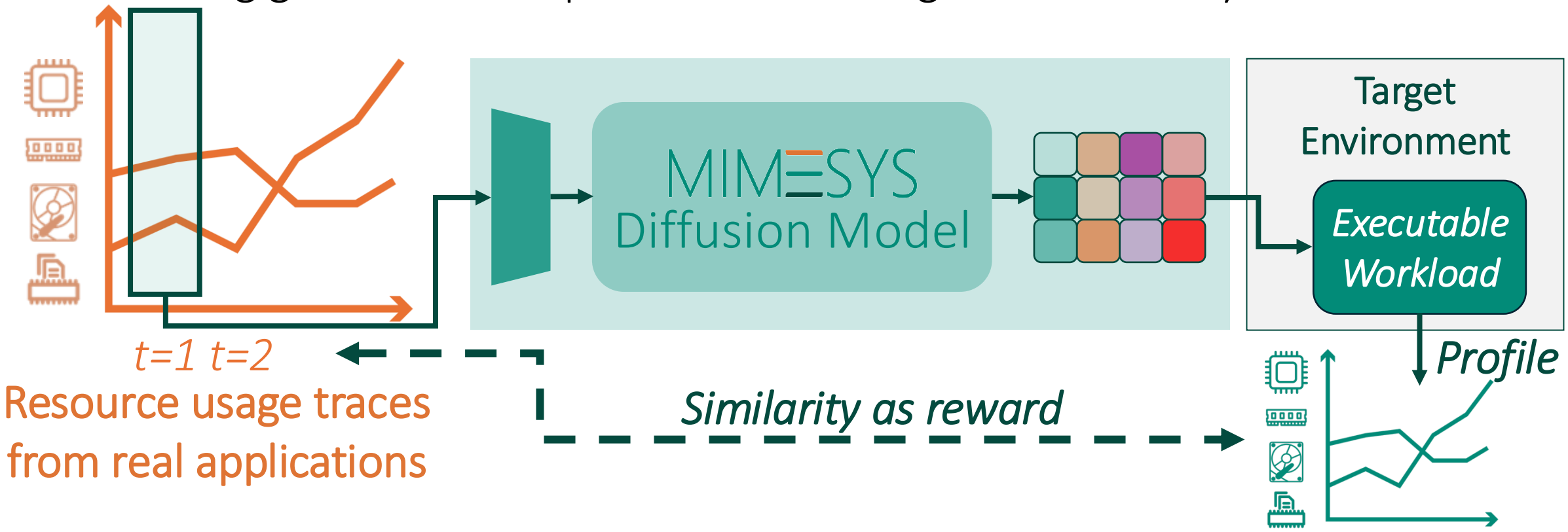


Resource usage from real app

No ground-truth stressor compositions for real app traces

RL-based Fine-tuning

Execution-driven alignment: Fine-tune the diffusion model with RL by executing generated compositions and using trace similarity as the reward



Evaluation Questions

- How accurately does Mimesys reproduce resource usage?
- Can Mimesys reproduce noisy-neighbor performance impact?
- How effective are Mimesys's techniques?
- How sensitive is Mimesys to input trace variations?

Evaluation Setup

Mimesys model training setup:

- 13 stressors from Google fleetbench, stress-ng
- 23 metrics from CPU, Memory, Cache, and Disk I/O
- Profiling: 8 x CloudLab c220g2 machines
- Training: 1 x Nvidia A100 GPU

Baselines:

- *Search-based*: the closest stressor composition from the training data
- *Linear interpolation*: Interpolating stressor compositions from the training data
- *Single stressor*: a single resource stressor matching per-core CPU utilization

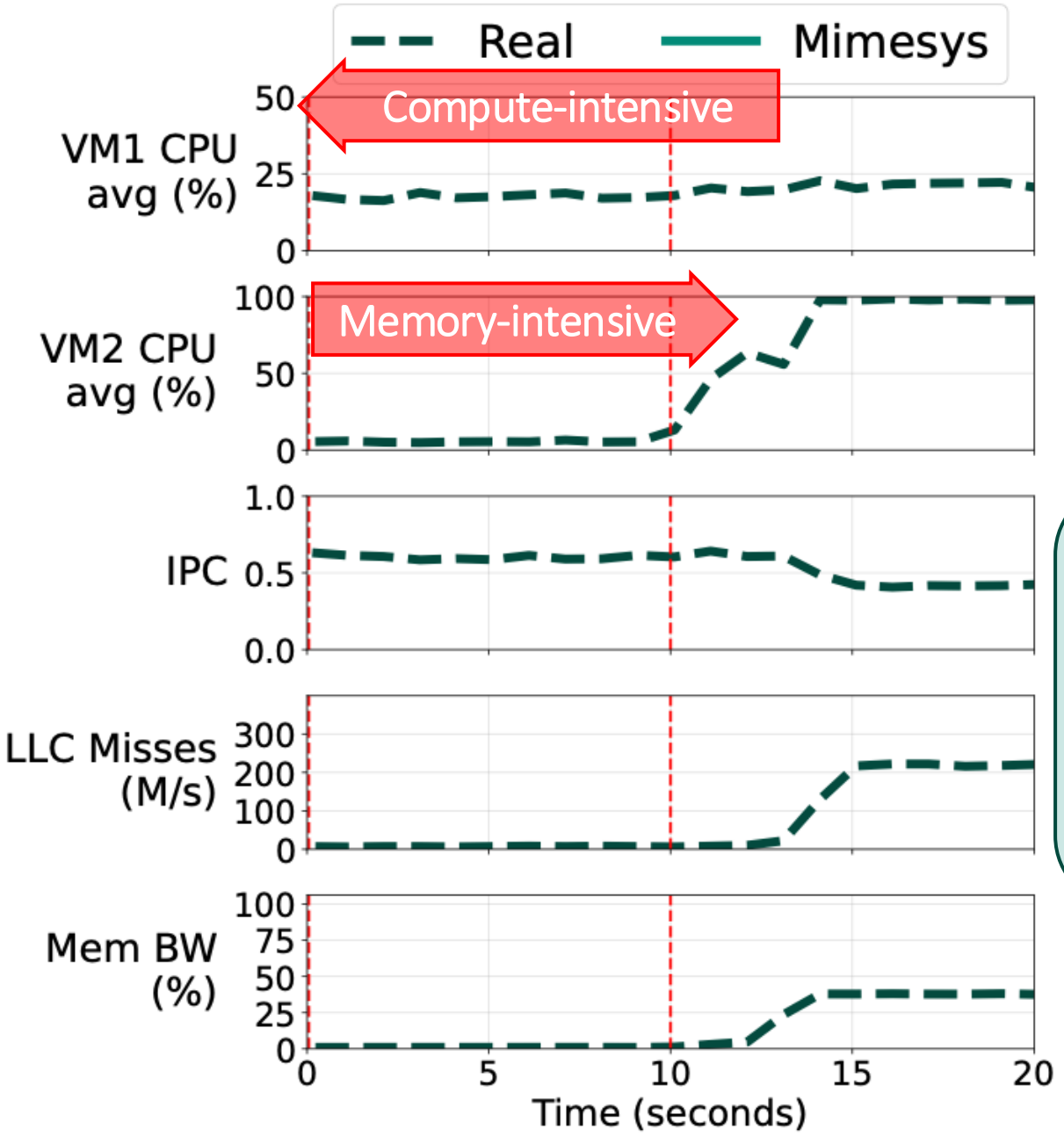
Evaluation Setup

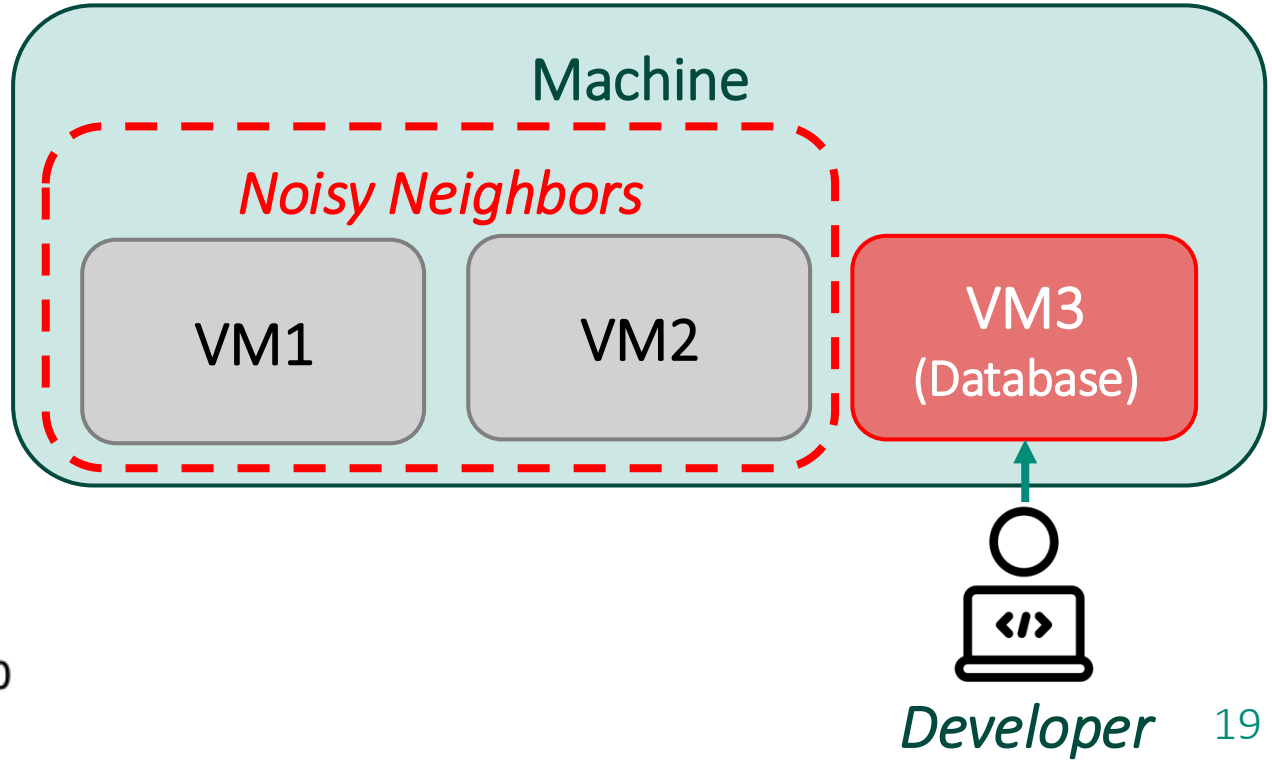
Workloads:

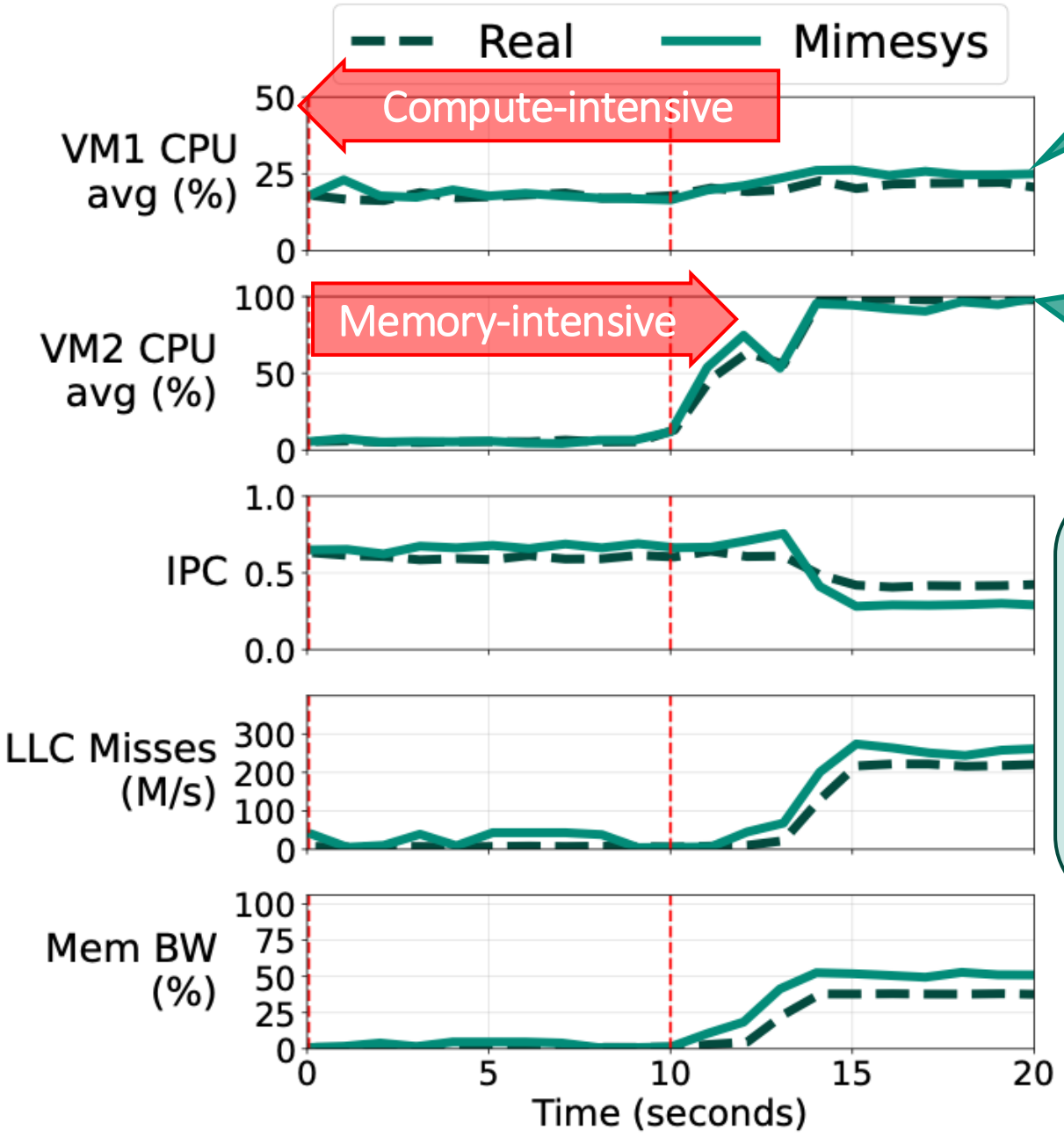
- We collect realistic contention scenarios by colocating multiple workloads following Azure's workload distribution [1]

	Applications	Ratio (%)
Web Serving	Dacapo, Renaissance	32
Big Data	HiBench on Spark	31
KV Store	YCSB on Redis, memcached, FASTER	14
ML Inference	MLPerf (ResNet50, StableDiffusion)	11
Database	TPC-C on Silo, TPC-H on PostgreSQL	7
Graph	GAP Benchmark	5

[1] Designing Cloud Servers for Lower Carbon (ISCA '24)


Real
Mimesys
Compute-intensive
Memory-intensive
VM1 CPU avg (%)
VM2 CPU avg (%)
IPC
LLC Misses (M/s)
Mem BW (%)
50
25
0
100
50
0
1.0
0.5
0.0
300
200
100
0
100
75
50
25
0
0
5
10
15
20
Time (seconds)


Machine
Noisy Neighbors
VM1
VM2
VM3
(Database)
Developer



Resource usage traces as inputs

Mimesys workloads show similar resource usage

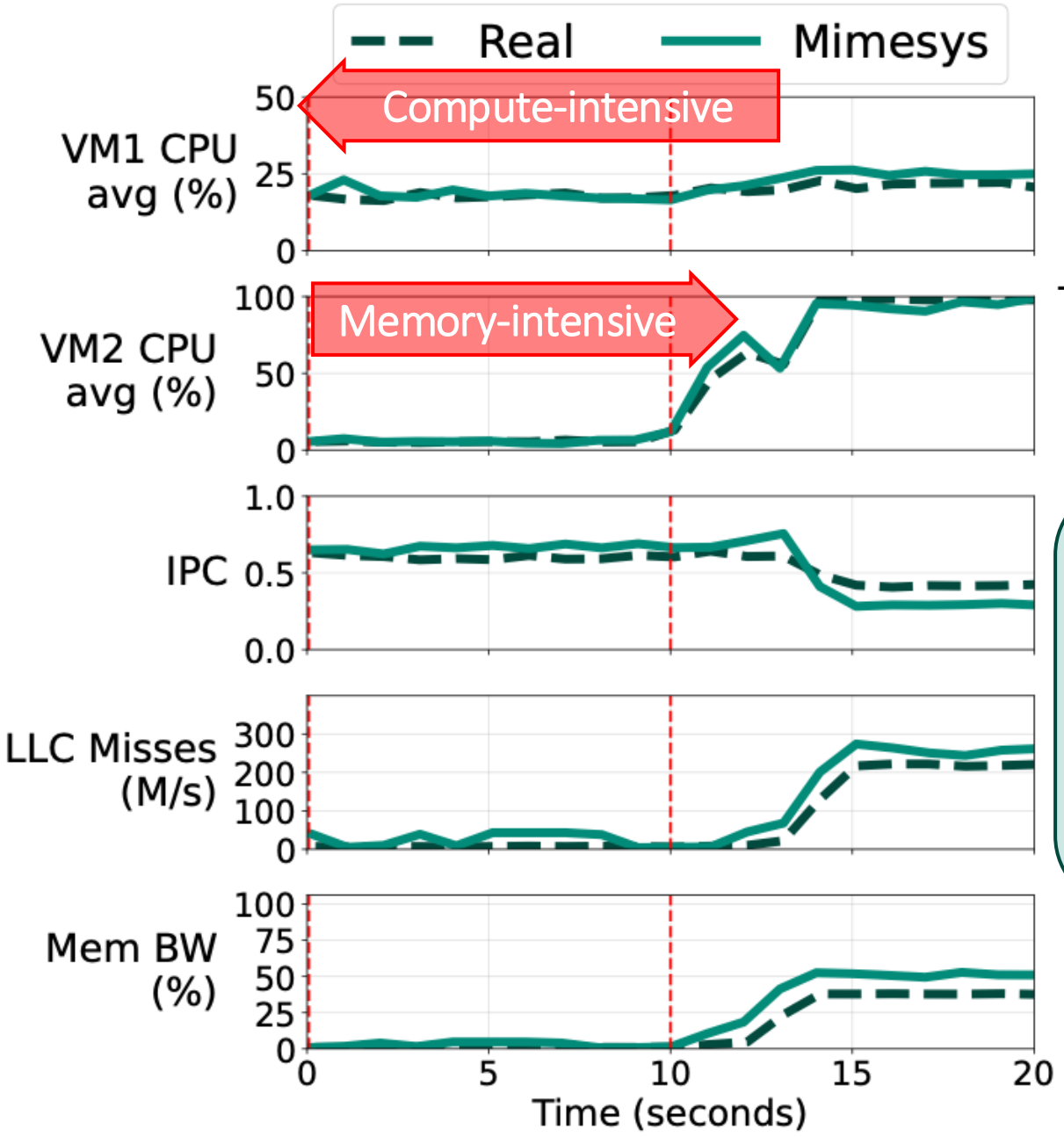
Machine

Mimesys

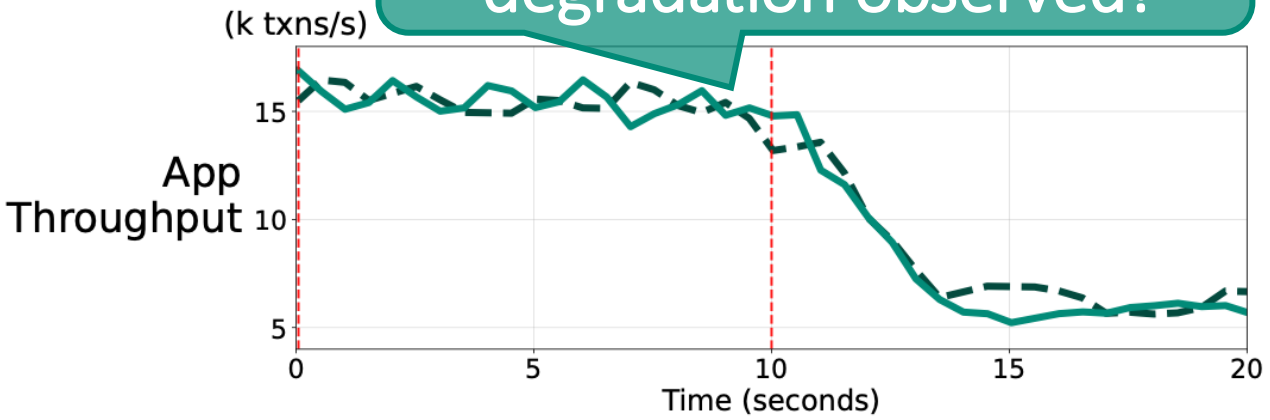
Executable Workload

VM3 (Database)

Developer



Similar performance degradation observed!



Machine

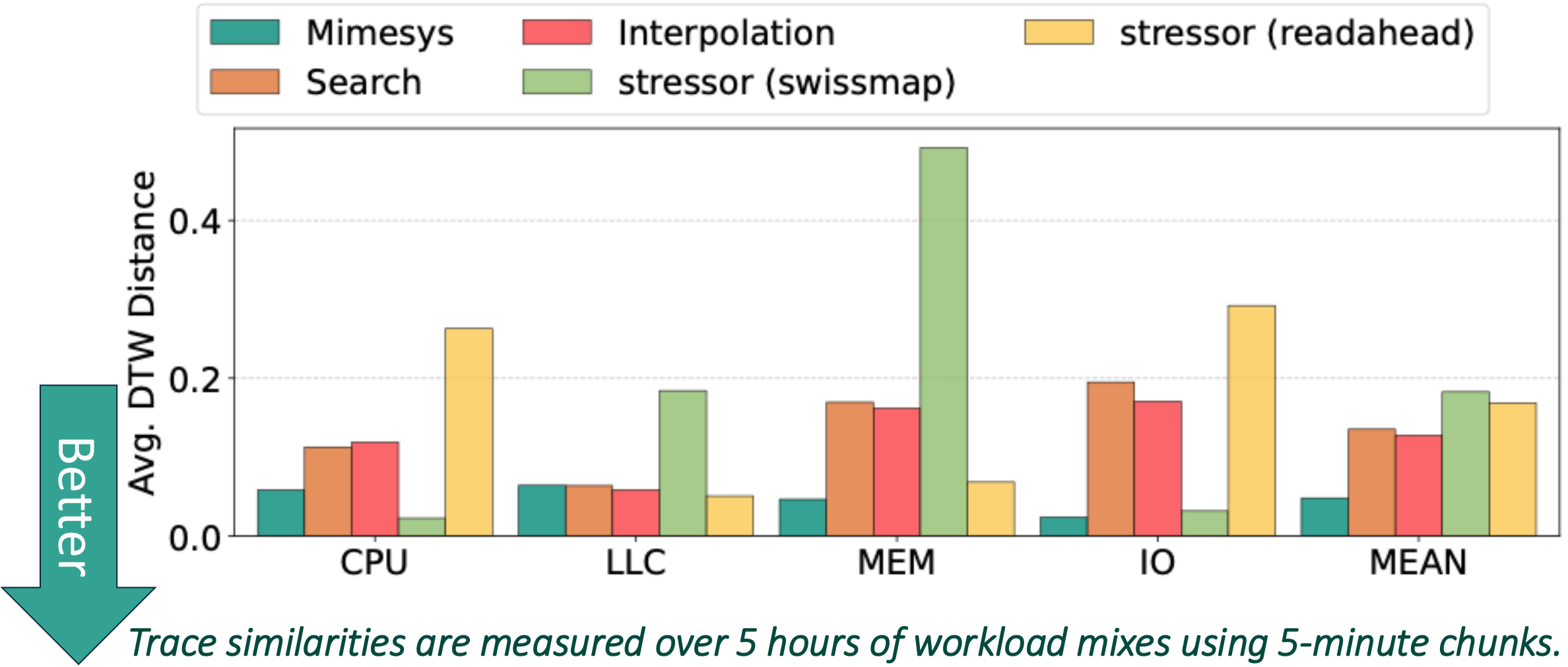
Mimesys

Executable Workload

VM3 (Database)

Developer

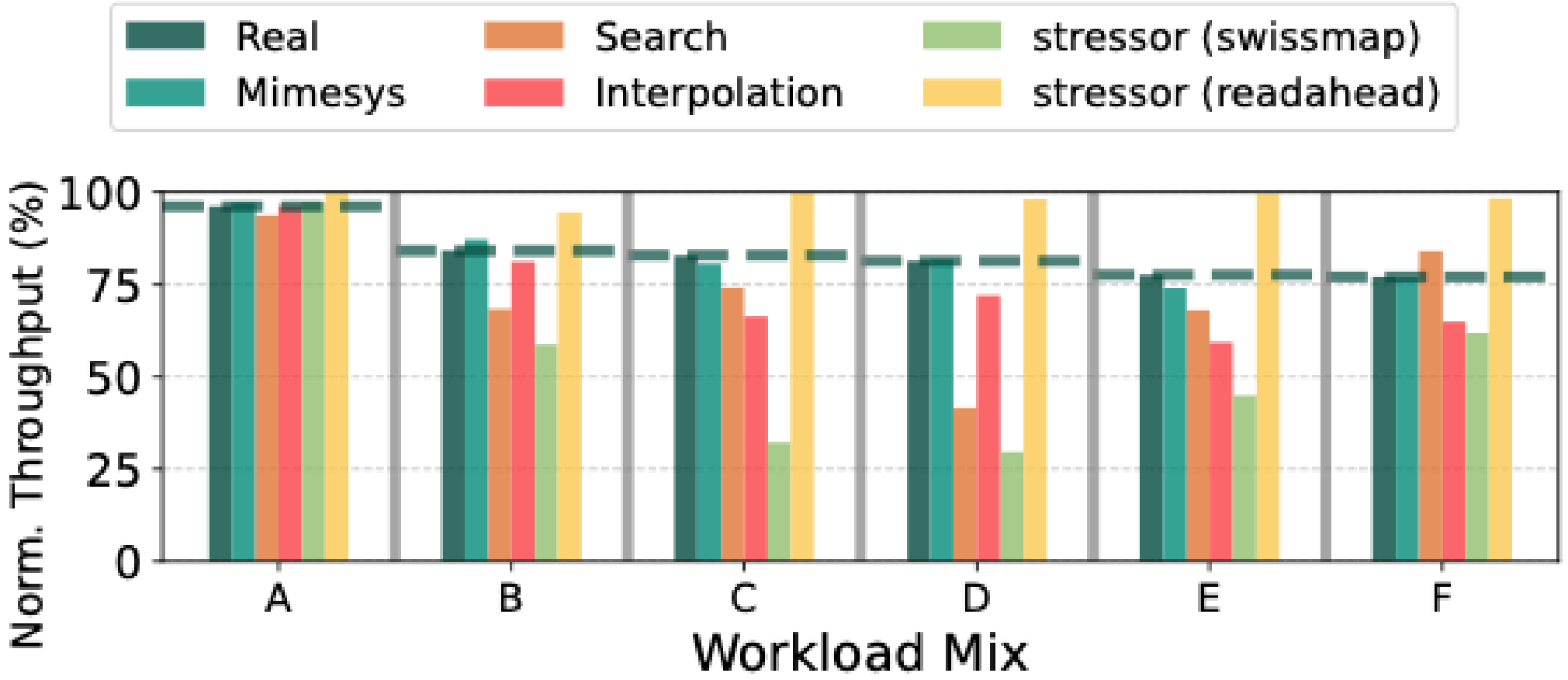
Trace Similarities



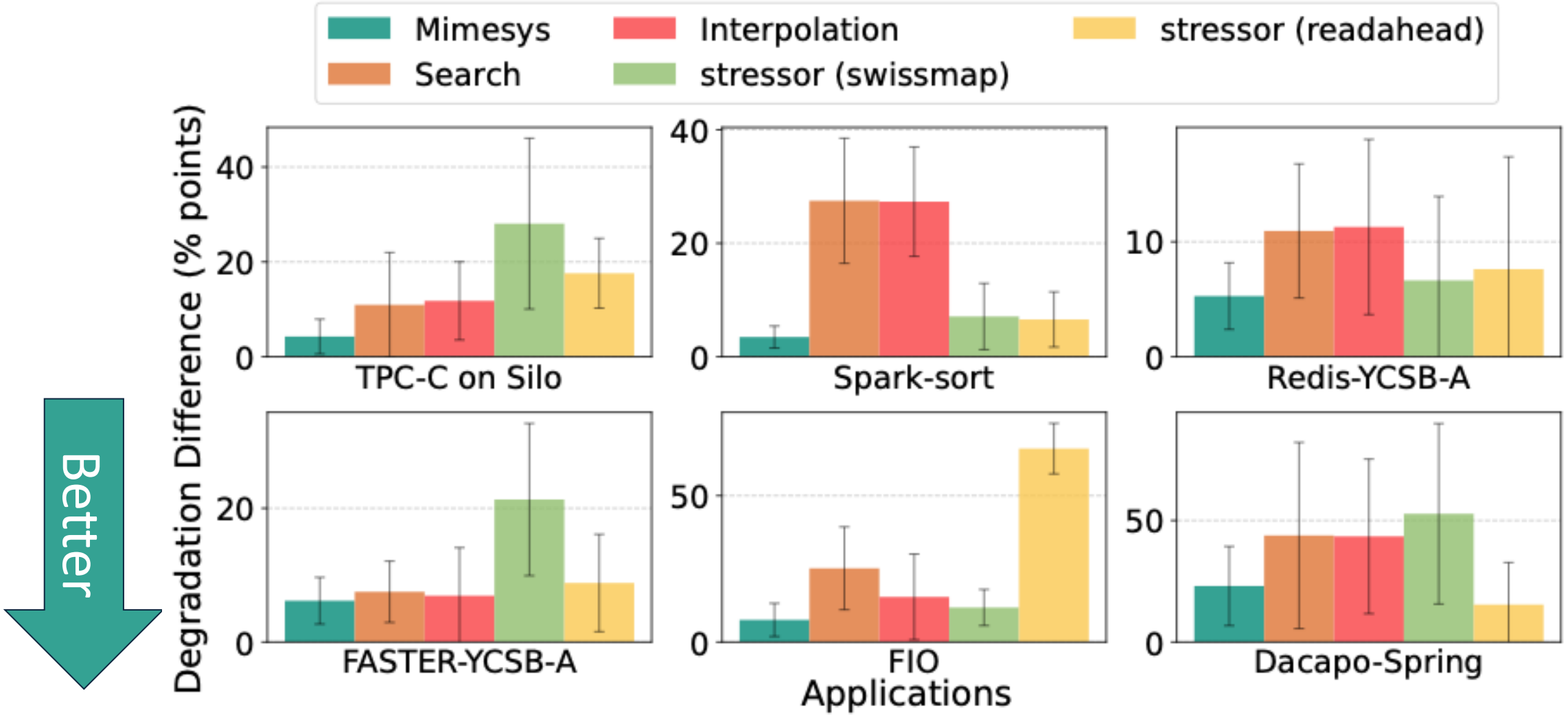
Trace similarities are measured over 5 hours of workload mixes using 5-minute chunks.

Reproducing Noisy Neighbor Effects

<TPC-C on Silo> performance varying colocated workloads



Reproducing Noisy Neighbor Effects



Summary

Artifact Available:
github.com/ldos-project/mimesys
Contact: donghyun@utexas.edu

Mimesys generates synthetic workloads from resource usage traces:

- **Diffusion Modeling:** train a diffusion model to generate stressor compositions from traces
- **Execution-driven Alignment:** fine-tune the model using trace similarity as a reward signal

Future Directions: Hardware-agnostic workload synthesis, reproducing richer and fine-grained system behaviors