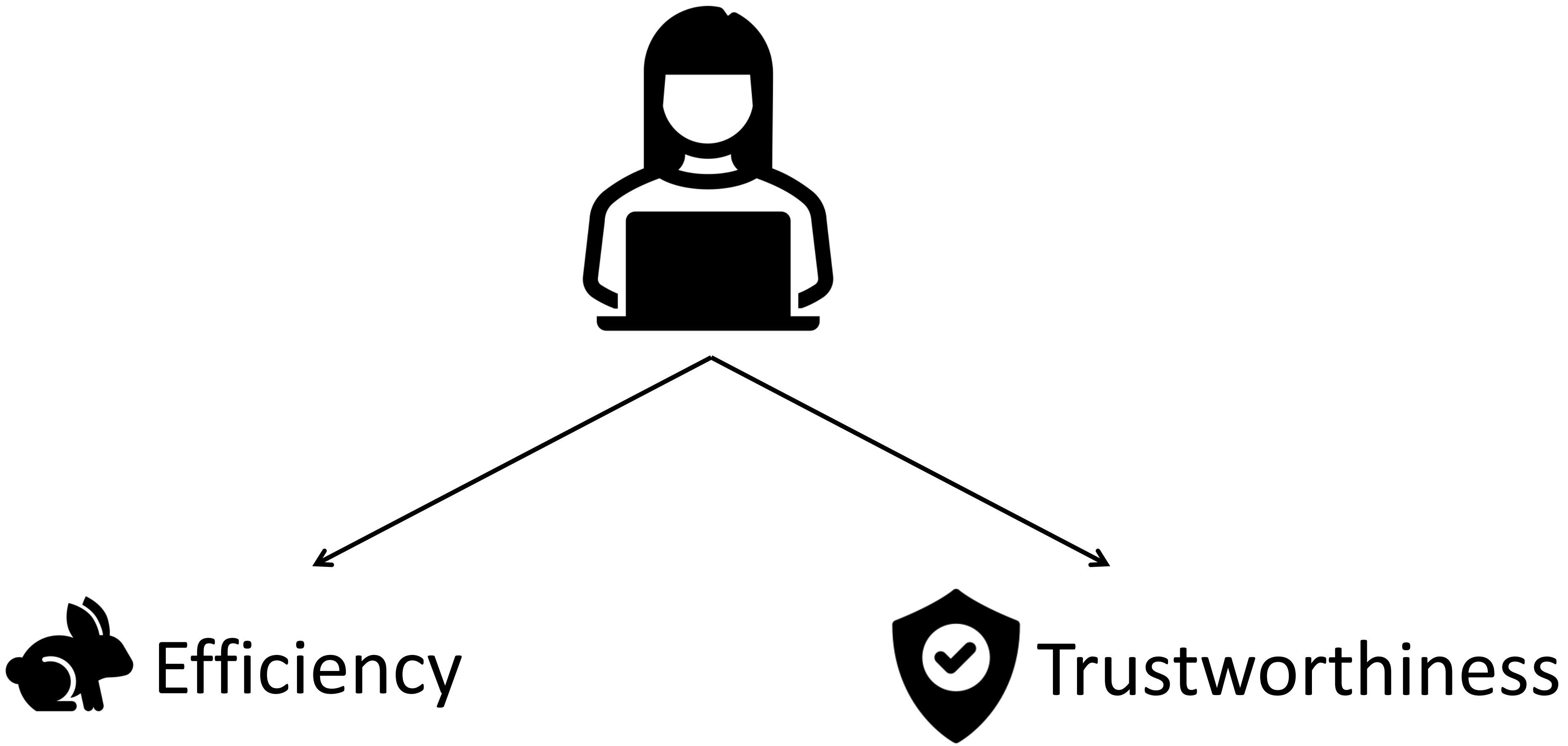
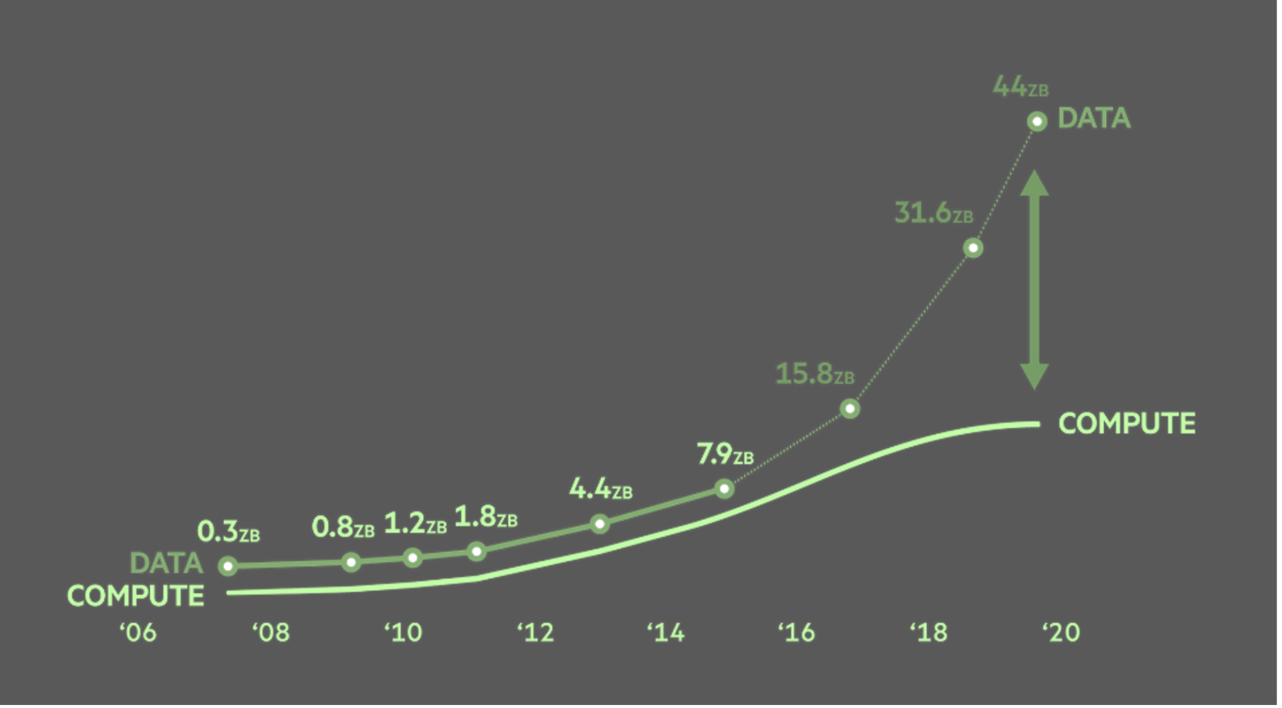
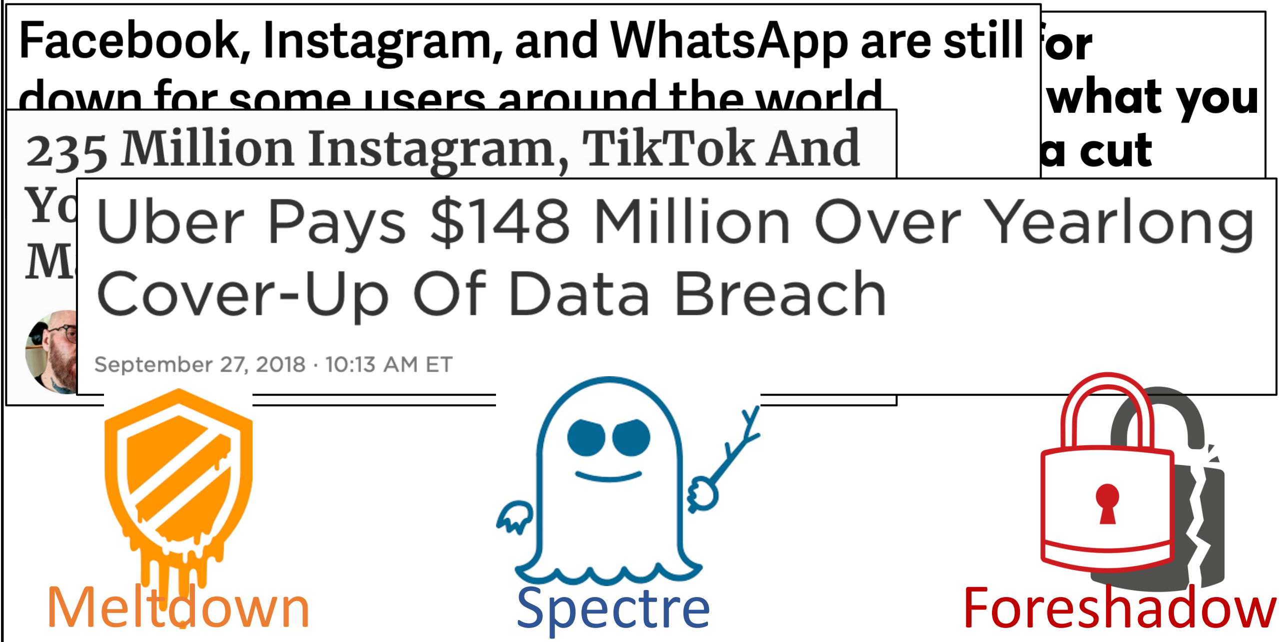
The Quest for Efficient and Trustworthy Systems

Baris Kasikci


Efficiency
Trustworthiness





Every 2 years, we create 2x more data than what we have created in all of human history[1]

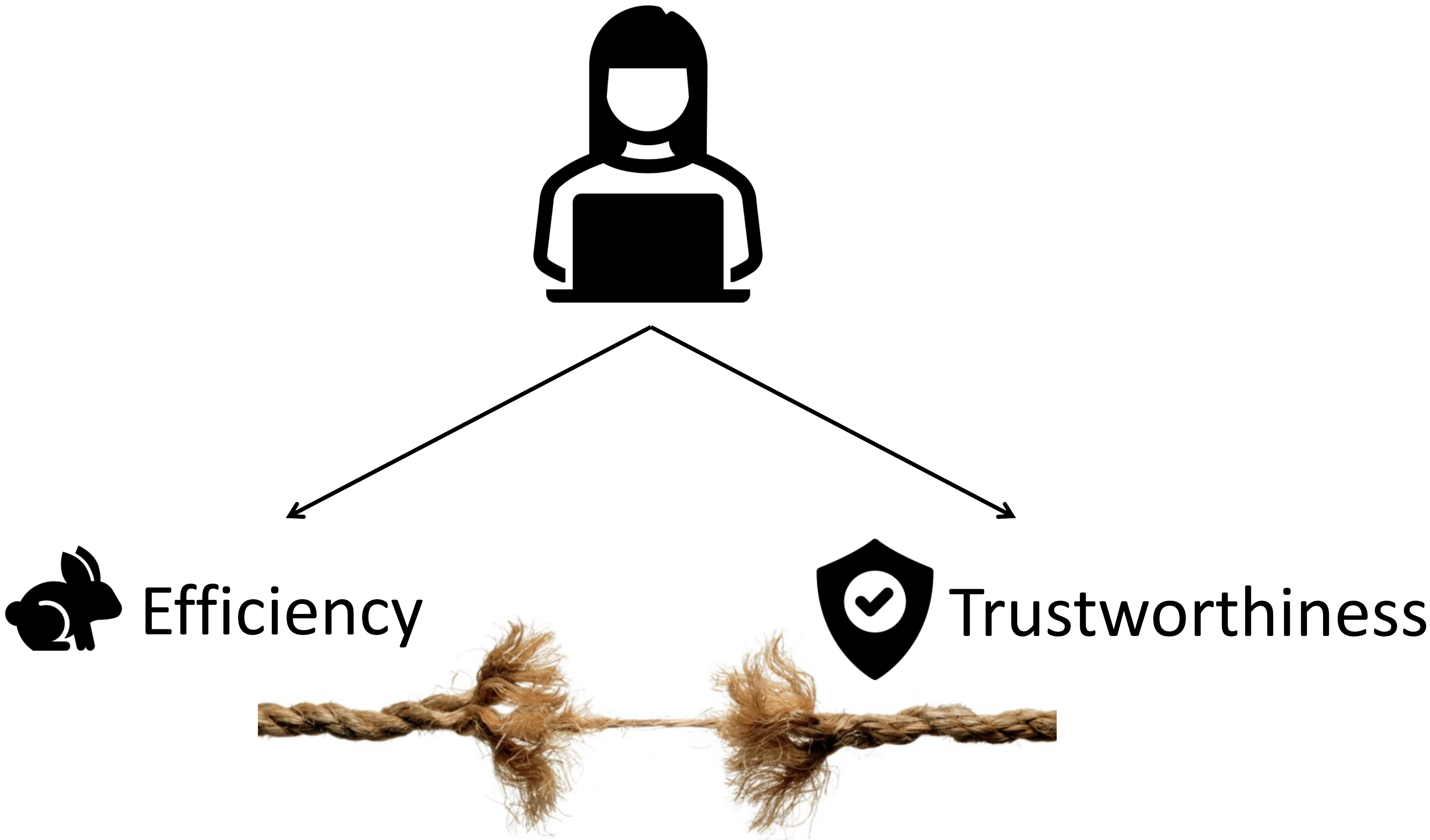
Efficiency of computer systems needs to catch up

Total cost of poor software quality > $2 Trillion in the US[2]

Trustworthiness (Reliability + Security) needs to improve

[1] Kirk Bresniker, World Economic Forum, 2018

[2] Consortium for Information & Software Quality, 2021 Report



My Approach

Designing efficient and trustworthy systems

based on a systematic understanding of program behavior

Efficiency

Datacenter Efficiency

Whisper [MICRO'22] Thermometer [ISCA'22]

Twig [MICRO'21] PDede [MICRO'21]

DMon [OSDI'21] I-SPY [MICRO'20]

Ripple [ISCA'21] Huron [PLDI'19] Cntr [ATC'18]

Heterogeneous Systems Support

Persistent Memory Indexing [FAST'21]

Optimus [ASPLOS'20]

Systems	Security
Architecture	PL

My work appears in major venues in all these areas

Trustworthiness

Failure Reproduction and Analysis

OmniTable [OSDI'22]

Debugging in the Brave New World [ASPLOS'22]

ER [PLDI'21] REPT [OSDI'18] Snorlax [SOSP'17]

Hippocrates [ASPLOS'21] Agamotto [OSDI'20]

Verified Distributed Systems

Sift [ATC'22] IGOR [RTAS'21] I4 [SOSP'19]

Hardware Security

MOESI-prime [ISCA'22]

Dolma [SEC'21] NDA [MICRO'19]

Foreshadow [SEC'18] Morpheus [ASPLOS'19]

: Award papers

	Offline	Online
Datacenter Efficiency		
Failure Reproduction and Analysis		
Hardware Security		

Navigating the efficiency-trustworthiness tension:
A careful balance between offline and online techniques

Outline

	Offline	Online
Datacenter Efficiency	Data-driven optimizations	Lightweight profiling
Failure Reproduction and Analysis		
Hardware Security		

Datacenters consume massive energy

- 3% of the global energy, large carbon footprint[1]
- $35 million/year savings from 1% less work[2]

Responsiveness impacts revenue

- 400ms delay decreases Google Search users by 0.4%[2]
- Two second delay on search responses reduces Microsoft Bing's revenue by 4.3%[2]

[1] Rano Danilak, Why Energy Is A Big And Rapidly Growing Problem For Data Centers?, 2017

[2] Kathryn McKinley , Tail Latency: Beyond Queuing Theory, 2017

amazon

Amazon CodeGuru

Google

AutoFDO: Automatic Feedback-Directed Optimization for Warehouse-Scale Applications

Dehao Chen
Google Inc.
dehao@google.com

David Xinliang Li
Google Inc.
davidxl@google.com

Tipp Moseley
Google Inc.
tipp@google.com

Meta

BOLT: A Practical Binary Optimizer for Data Centers and Beyond

Maksim Panchenko, Rafael Auler, Bill Nell, Guilherme Ottoni
Facebook, Inc.
Menlo Park, CA, USA
{maks,rafaelauler,bnell,ottoni}@fb.com

Microsoft

Vulcan

Binary transformation in a distributed environment

Amitabh Srivastava
Andrew Edwards
Hoi Vo

Profile-Guided Optimizations, PGO

e.g., use a profile of branch traces for reordering code to make it cache-friendly

Cooperative Prefetching: Compiler and Hardware Support for Effective Instruction Prefetching in Modern Processors

Chi-Keung Luk

Temporal Instruction Fetch Streaming

Limitations:

Significant hardware modifications

Impractical on-chip space overhead

Limited gains due to on-chip space limits

Lawrence Sp

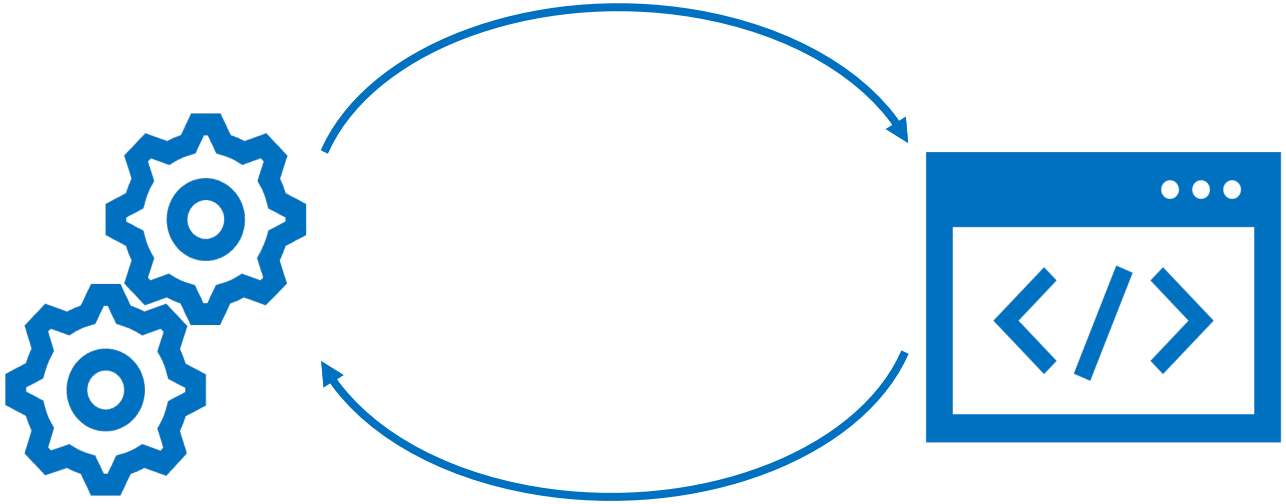
Blasting Through The Front-End Bottleneck With Shotgun

Rakesh Kumar*
Uppsala University

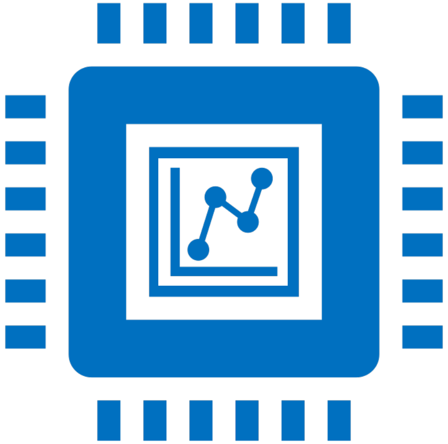
Boris Grot
University of Edinburgh

Vijay Nagarajan
University of Edinburgh

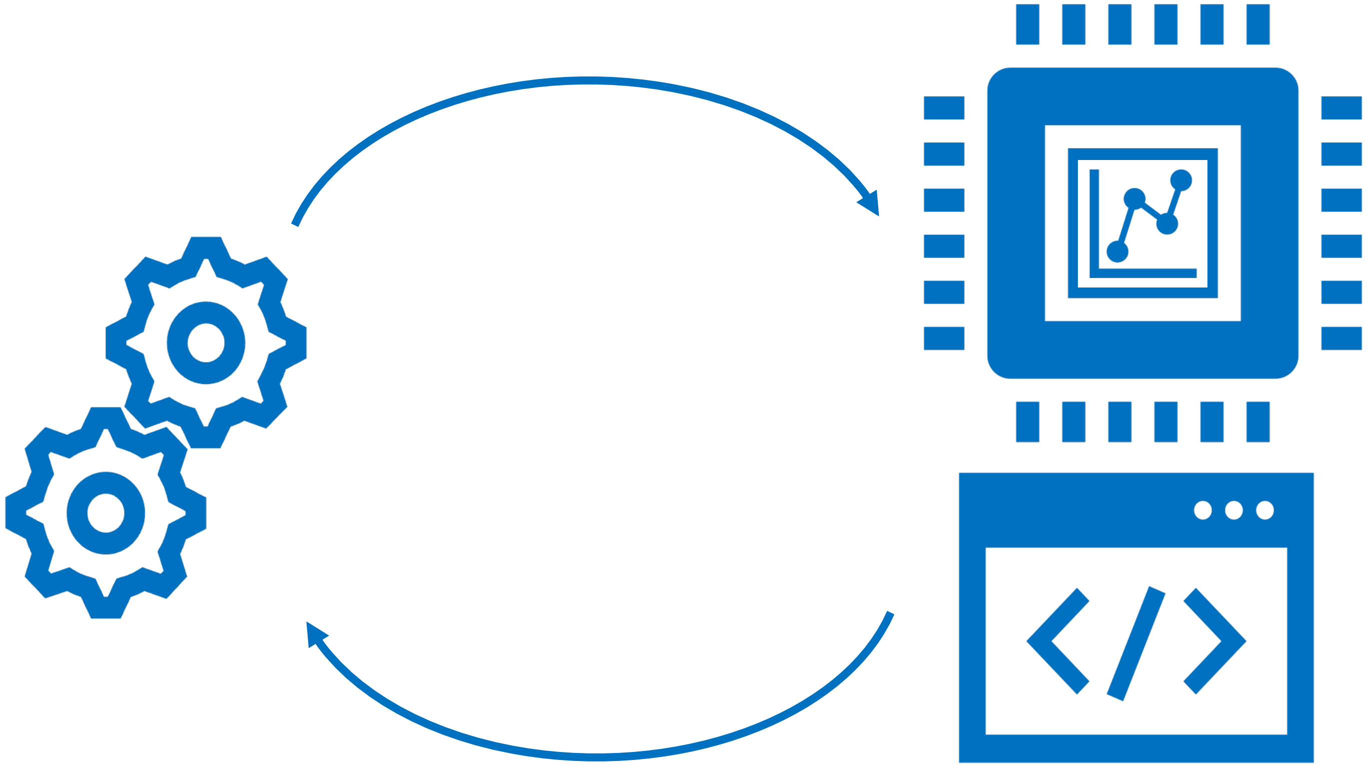
Profile-Guided Software Optimizations



On-Chip Analysis and Optimizations



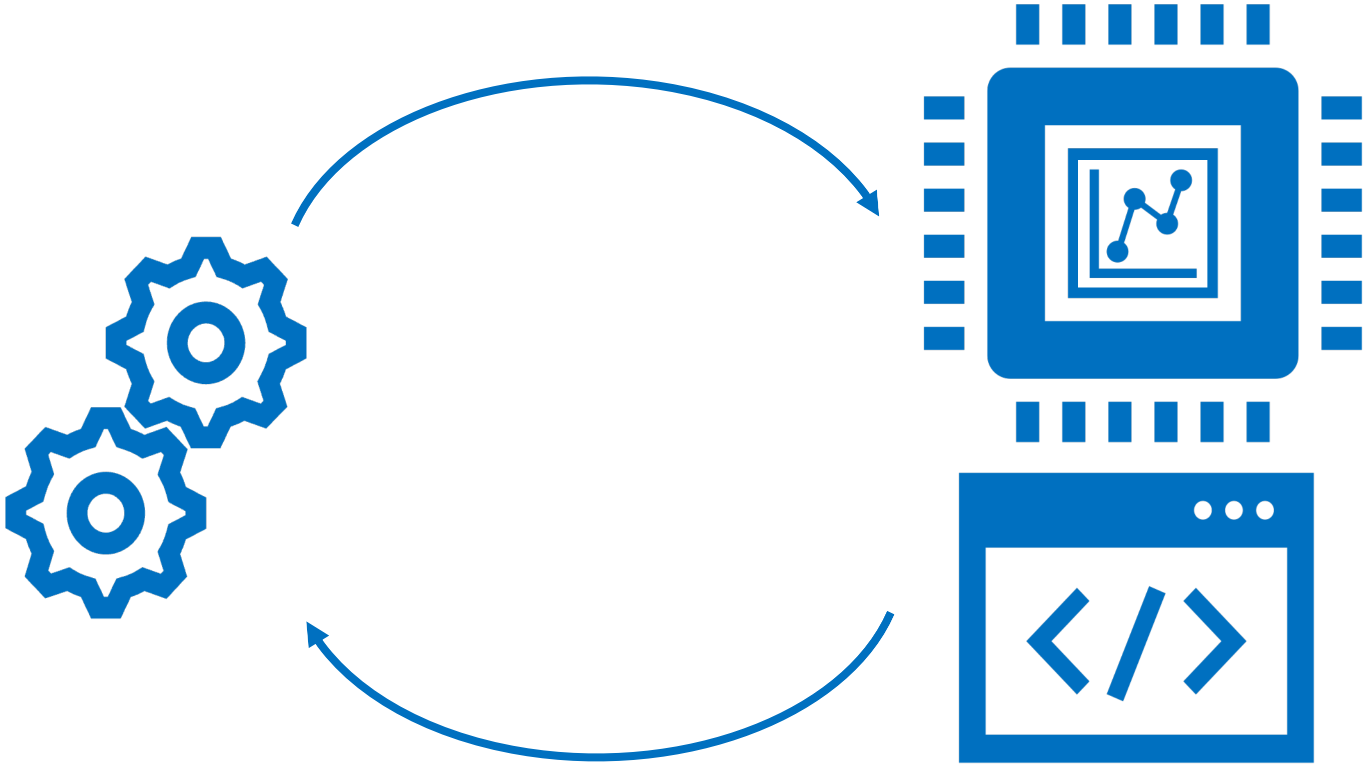
Profile-Guided Software and Hardware Optimizations



Online Lightweight Profiling

Profile-Guided Software and Hardware Optimizations

Offline Analysis of
Profiling Data



Hardware
Optimizations
Little/No Hardware
Modifications

Software
Optimizations

Online Lightweight Profiling

Performance improvement of up to 90% of the theoretical limit
Little-to-No hardware modifications (Intel & ARM technology transfer)

I-SPY: Context-Driven Conditional Instruction Prefetching with Coalescing

Tanvir Ahmed Khan* Akshitha Sriraman* Joseph Devietti† Gilles Pokam‡ Heiner Litz§ Baris Kasikci*
*University of Michigan †University of Pennsylvania ‡Intel Corporation §University of California, Santa Cruz
*{takh, akshitha, barisk}@umich.edu †devietti@cis.upenn.edu ‡gilles.a.pokam@intel.com §hlitz@ucsc.edu

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

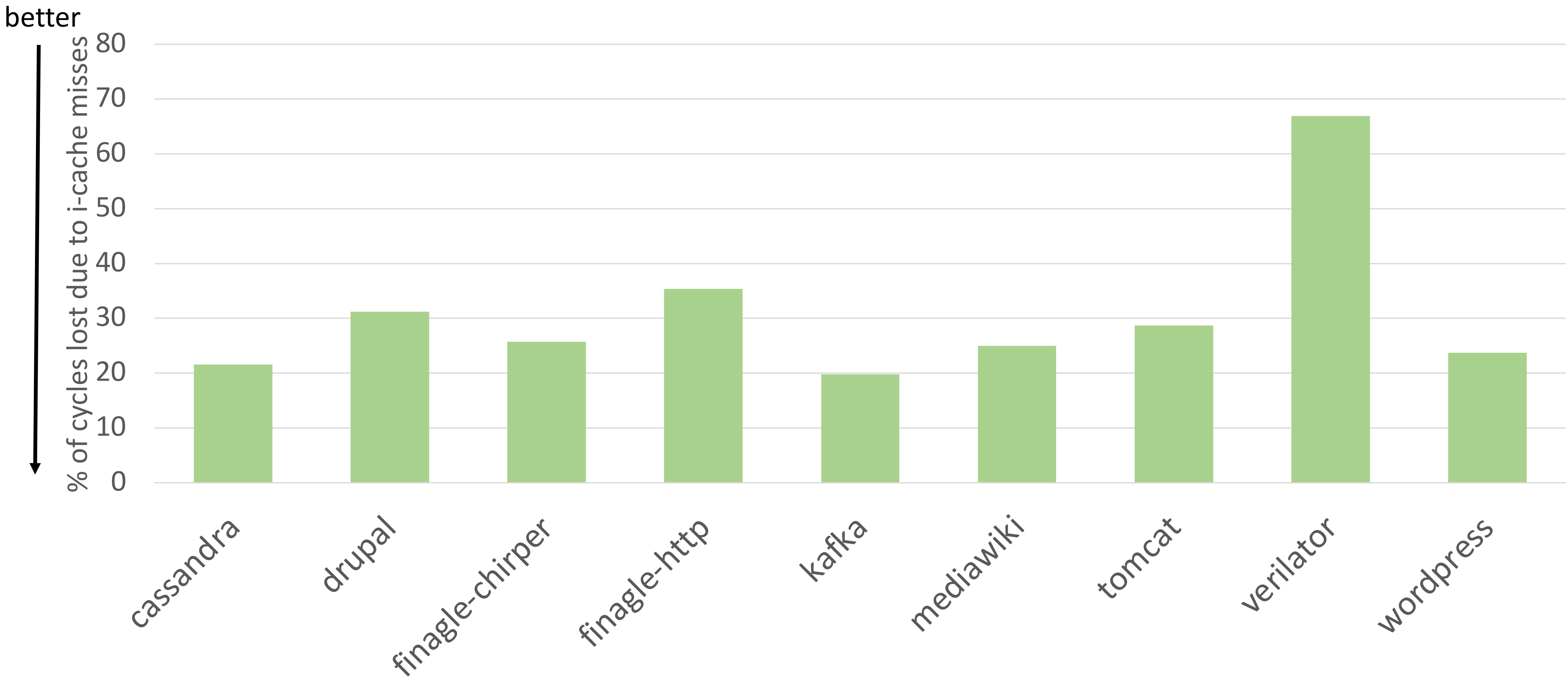
Penn
UNIVERSITY of PENNSYLVANIA

intel®

UC SANTA CRUZ

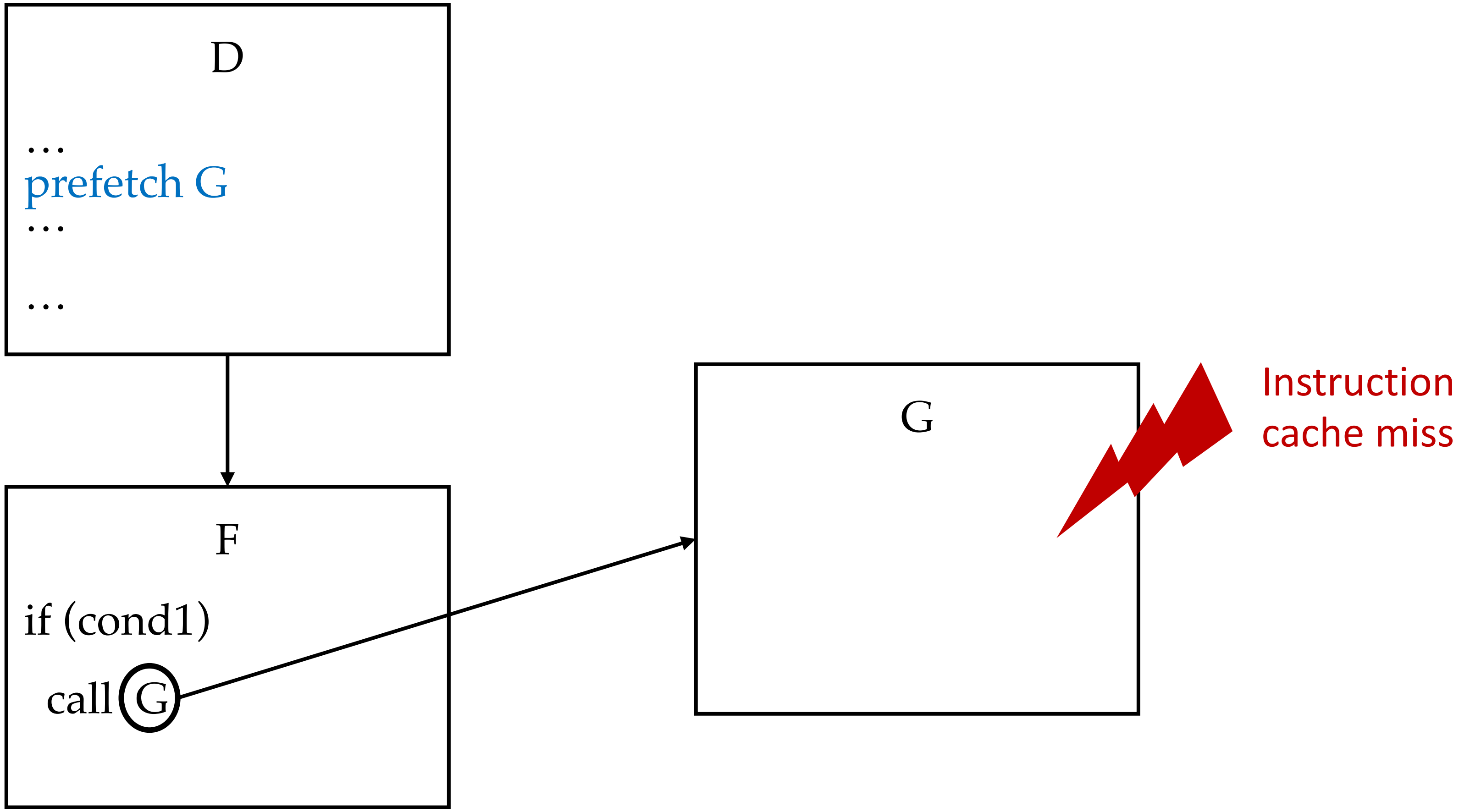
MICRO'20

Performance Impact of Instruction Cache Misses



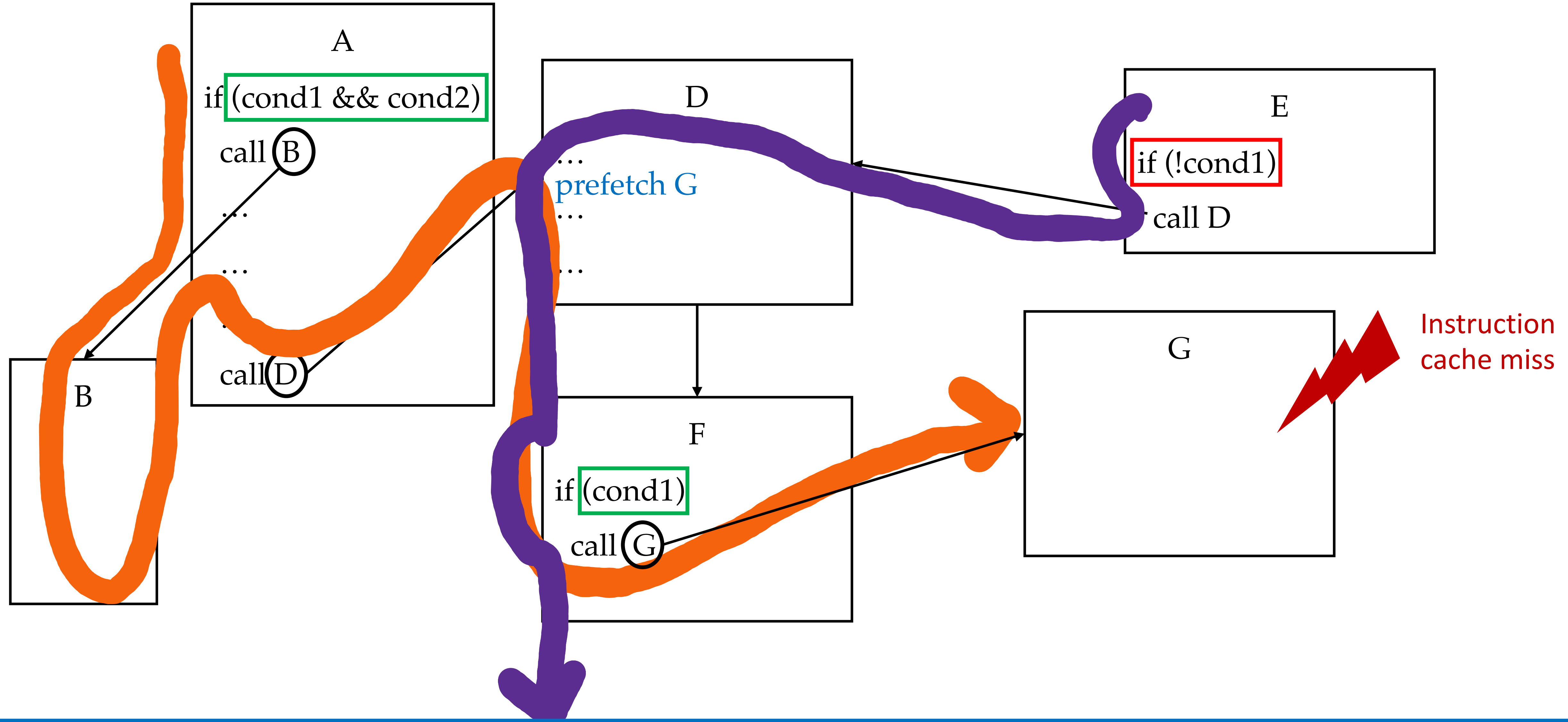
20-70% performance lost due to large instruction footprint (14-45x I-cache)

Why Does Prior Work Fall Short?



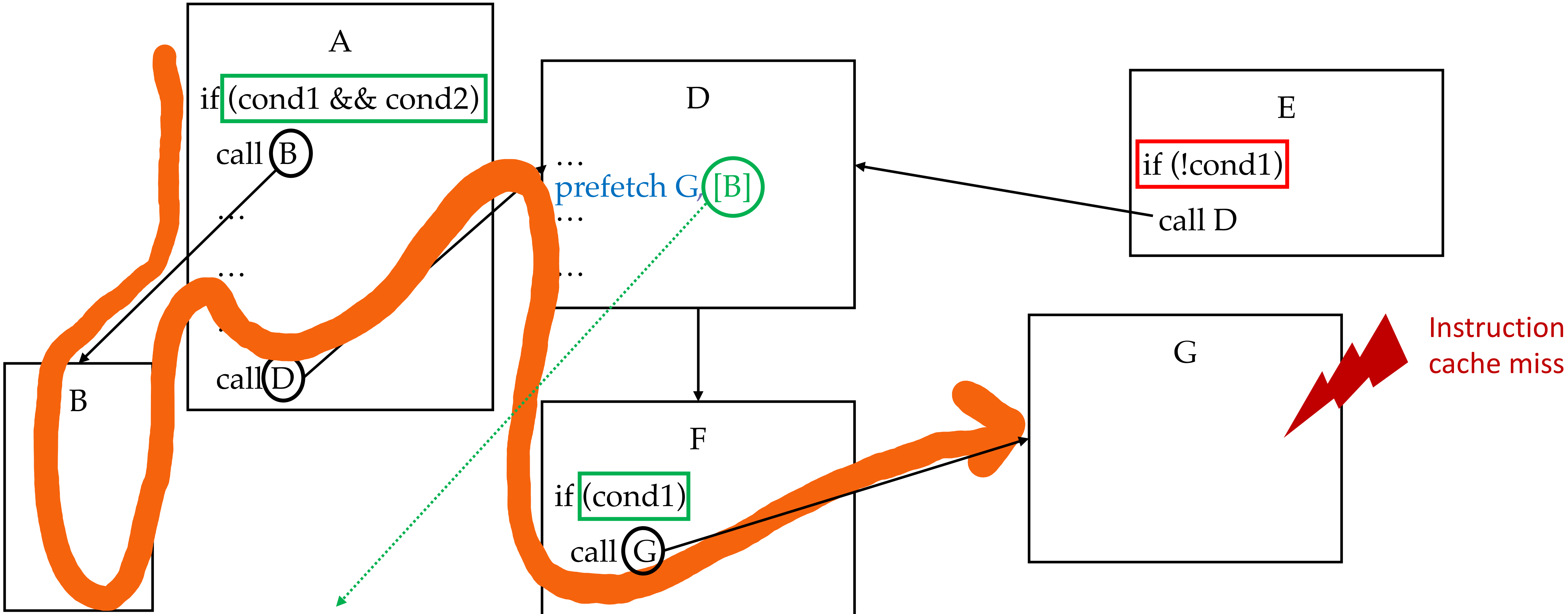
[1] Grant Ayers et al., AsmDB: understanding and mitigating front-end stalls in warehouse-scale computers, ISCA 2019

Why Does Prior Work Fall Short?



Overfitting prefetches based on limited execution information hurts speedup

Context-Driven* Conditional Prefetching



Prefetch only if B was executed recently

*Context = Control flow tracked using efficient online hardware tracing

4 branches-long context information allows 90% of the speed of an ideal cache

I-SPY

Context-Driven Conditional Prefetching

- A data-driven optimization powered by offline analysis of profiling information
- Avoids unnecessary prefetches
- Can be implemented with minor hardware support

Achieves 90% of the ideal cache performance → ~ $700 million in savings[1]

- Outperforms prior work by Google by 22.5%

[1] Rano Danilak, Why Energy Is A Big And Rapidly Growing Problem For Data Centers?, 2017

ISCA'22

Thermometer: Profile-Guided BTB Replacement for Data Center Applications.

Shixin Song, shixins@umich.edu, University of Michigan, USA
Tanvir Ahmed Khan, takh@umich.edu, University of Michigan, USA
Sara Mahdizadeh Shahri, smahdiz@umich.edu, University of Michigan, USA
Akshitha Sriraman, akshitha@cmu.edu, Carnegie Mellon University, USA
Niranjan Soundararajan, niranjan.k.soundararajan@intel.com, Intel Labs, India
Sreenivas Subramoney, sreenivas.subramoney@intel.com, Intel Labs, India
Heiner Litz, hlitz@ucsc.edu, University of California, Santa Cruz, USA
Baris Kasikci, barisk@umich.edu, University of Michigan, USA

MICRO'22

Whisper: Profile-Guided Branch Misprediction Elimination for Data Center Applications

Tanvir Ahmed Khan* Muhammed Ugur* Krishnendra Nathella† Dam Sunwoo† Heiner Litz‡ Daniel A. Jiménez§ Baris Kasikci*
*University of Michigan †ARM ‡University of California, Santa Cruz §Texas A&M University
*{takh, meugur, barisk}@umich.edu †{Krishnendra.Nathella, Dam.Sunwoo}@arm.com ‡hlitz@ucsc.edu
§djimenez@acm.org

MICRO'22

OCOLOS: Online COde Layout OptimizationS

Yuxuan Zhang* Tanvir Ahmed Khan† Gilles Pokam‡ Baris Kasikci† Heiner Litz§ Joseph Devietti*
*University of Pennsylvania †University of Michigan ‡Intel Corporation §University of California, Santa Cruz
*{zyuxuan, devietti}@seas.upenn.edu †{takh, barisk}@umich.edu
‡gilles.a.pokam@intel.com §hlitz@ucsc.edu

EuroSys'22

APT-GET: Profile-Guided Timely Software Prefetching

Saba Jamilan* Tanvir Ahmed Khan‡ Grant Ayers† Baris Kasikci‡ Heiner Litz*
*University of California, Santa Cruz †Google ‡University of Michigan

OSDI'21

DMon: Efficient Detection and Correction of Data Locality Problems Using Selective Profiling

Tanvir Ahmed Khan, *University of Michigan*
Ian Neal, *University of Michigan*
Gilles Pokam, *Intel Corporation*
Barzan Mozafari, *University of Michigan*
Baris Kasikci, *University of Michigan*

ISCA'21

Ripple: Profile-Guided Instruction Cache Replacement for Data Center Applications

Tanvir Ahmed Khan* Dexin Zhang† Akshitha Sriraman* Joseph Devietti‡ Gilles Pokam§ Heiner Litz¶ Baris Kasikci*
*University of Michigan †University of Science and Technology of China ‡University of Pennsylvania
§Intel Corporation ¶University of California, Santa Cruz
*{takh, akshitha, barisk}@umich.edu †zhangdexin@mail.ustc.edu.cn
‡devietti@cis.upenn.edu §gilles.a.pokam@intel.com ¶hlitz@ucsc.edu

MICRO'21

Twig: Profile-Guided BTB Prefetching for Data Center Applications

Tanvir Ahmed Khan, takh@umich.edu, University of Michigan, USA
Nathan Brown, nlbrow@umich.edu, University of Michigan, USA
Akshitha Sriraman, akshitha@umich.edu, University of Michigan, USA
Niranjan Soundararajan, niranjan.k.soundararajan@intel.com, Intel Labs, India
Rakesh Kumar, rakesh.kumar@ntnu.no, Norwegian University of Science and Technology, Norway
Joseph Devietti, devietti@cis.upenn.edu, University of Pennsylvania, USA
Sreenivas Subramoney, sreenivas.subramoney@intel.com, Intel Labs, India
Gilles Pokam, gilles.a.pokam@intel.com, Intel Labs, USA
Heiner Litz, hlitz@ucsc.edu, University of California, Santa Cruz, USA
Baris Kasikci, barisk@umich.edu, University of Michigan, USA

MICRO'21

PDede: Partitioned, Deduplicated, Delta Branch Target Buffer

Niranjan Soundararajan, niranjan.k.soundararajan@intel.com, Processor Architecture Research Lab, Intel Labs, India
Peter Braun, pvbraun@ucsc.edu, University of California, Santa Cruz USA
Tanvir Ahmed Khan, takh@umich.edu, University of Michigan USA
Baris Kasikci, barisk@umich.edu, University of Michigan USA
Heiner Litz, hlitz@ucsc.edu, University of California, Santa Cruz USA
Sreenivas Subramoney, sreenivas.subramoney@intel.com, Processor Architecture Research Lab, Intel Labs, India

Datacenter Efficiency

Thermometer [ISCA'22]

Twig [MICRO'21] PDede [MICRO'21]

DMon [OSDI'21] I-SPY [MICRO'20]

Ripple [ISCA'21] Huron [PLDI'19] Cntr [ATC'18]

Awards

VMware Early Career Grant

Intel Rising Star Award

Intel Faculty Awards

- 2017, 2018

Rackham Ph.D. Fellowship

- Tanvir Ahmed Khan

MICRO'22 Best Paper Award

Grants

- NSF, Intel, SRC

Intel[1] and ARM[2] technology transfer

Collaborations

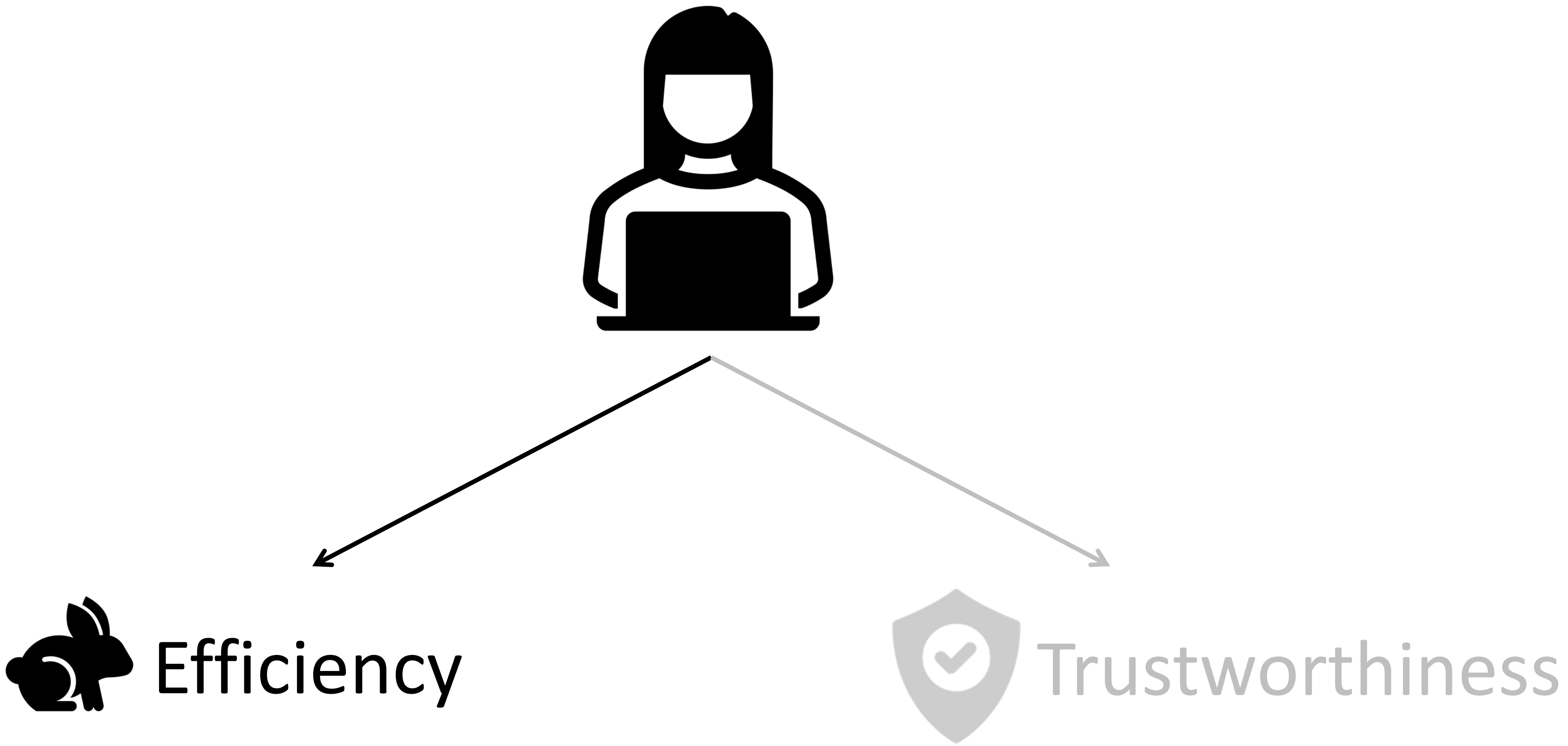
- ARM
- University of Pennsylvania
- UC Santa Cruz

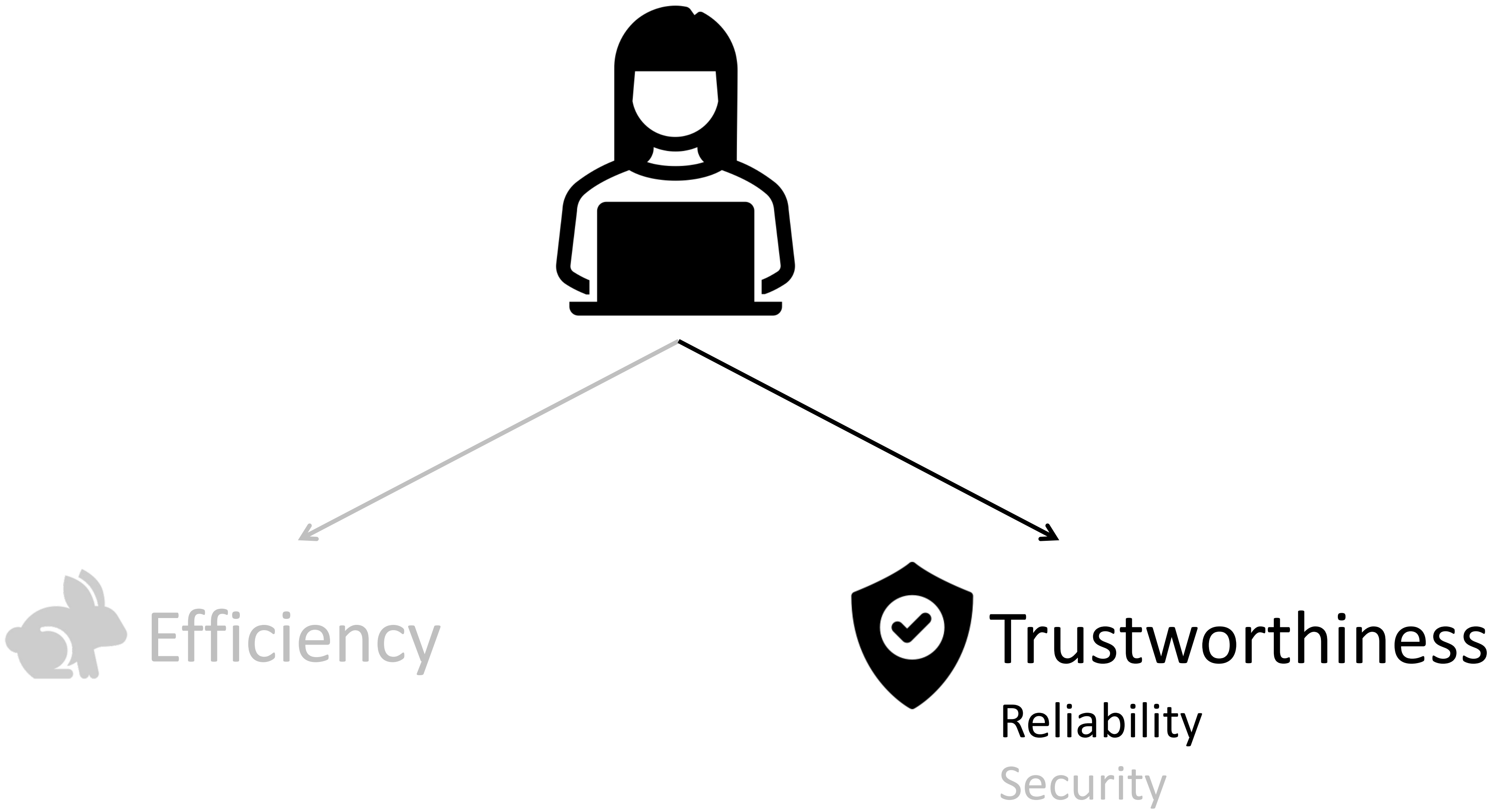
[1] https://patents.google.com/patent/US20210342134A1/en

[2] https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/arm-neoverse-n1-performance-analysis-methodology

Outline

	Offline	Online
Datacenter Efficiency	Data-driven optimizations	Lightweight profiling
Failure Reproduction and Analysis		
Hardware Security		


Efficiency
Trustworthiness


Efficiency
Trustworthiness
Reliability
Security

Outline

	Offline	Online
Datacenter Efficiency	Data-driven optimizations	Lightweight profiling
Failure Reproduction and Analysis	Static & symbolic program analysis	Selective information monitoring
Hardware Security		

Characterizing and Predicting Which Bugs Get Fixed: An Empirical Study of Microsoft Windows

Philip J. Guo* Thomas Zimmermann+ Nachiappan Nagappan+ Brendan Murphy+

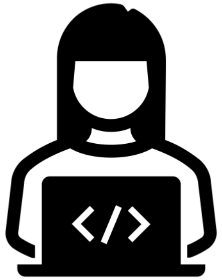
* Stanford University
+ Microsoft Research

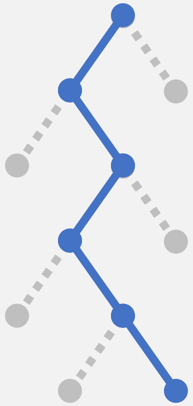
"Developers can fix a bug if they can reproduce the associated failure"

Reproducing failures is difficult, especially for production use cases

Memory dump

Manual
Failure Reproduction



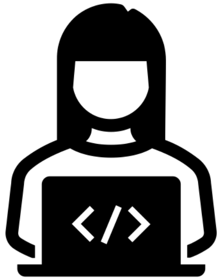


Last state
of data in memory

Branches
monitored for profiling
(profile-guided optimizations)

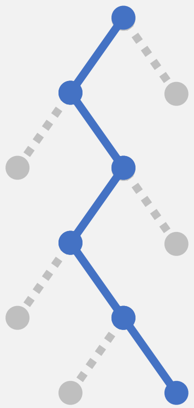
Control flow trace

Manual
Failure Reproduction



Memory dump

Branches
monitored for profiling
(profile-guided optimizations)



Control flow trace

\+ Last state
of data in memory

→ **REPT &**
Execution Reconstruction

Automatic Failure
Reproduction

Navigating the efficiency/trustworthiness tension:

Use branch traces for both optimizations and reproducing failures

REPT and Execution Reconstruction

OSDI 2018
Best Paper Award

REPT: Reverse Debugging of Failures in Deployed Software

- Most-widely deployed failure reproduction and analysis system in the world
- Used in **~ 1 billion** Microsoft Windows systems

Execution Reconstruction (ER)

- Offline symbolic program analysis
- Online selective hardware monitoring (control and data)
- Reproduces arbitrarily longer executions than what REPT can

Execution Reconstruction: Harnessing Failure Reoccurrences for Failure Reproduction

Gefei Zuo
gefeizuo@umich.edu
University of Michigan, USA

Jiacheng Ma
jcma@umich.edu
University of Michigan, USA

Andrew Quinn
arquinn@umich.edu
University of Michigan, USA

Pramod Bhatotia
pramod.bhatotia@in.tum.de
TU Munich, Germany

Pedro Fonseca
pfonseca@purdue.edu
Purdue University, USA

Baris Kasikci
barisk@umich.edu
University of Michigan, USA

COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MICHIGAN

PURDUE UNIVERSITY

PLDI'21

Prior Work vs. Execution Reconstruction (ER)

ER (our work)

ESD (Execution Synthesis)[2]

REPT[4]

BugRedux[3]

Mozilla RR (Record Replay)[1]

Record nothing
(e.g., symbolic execution)

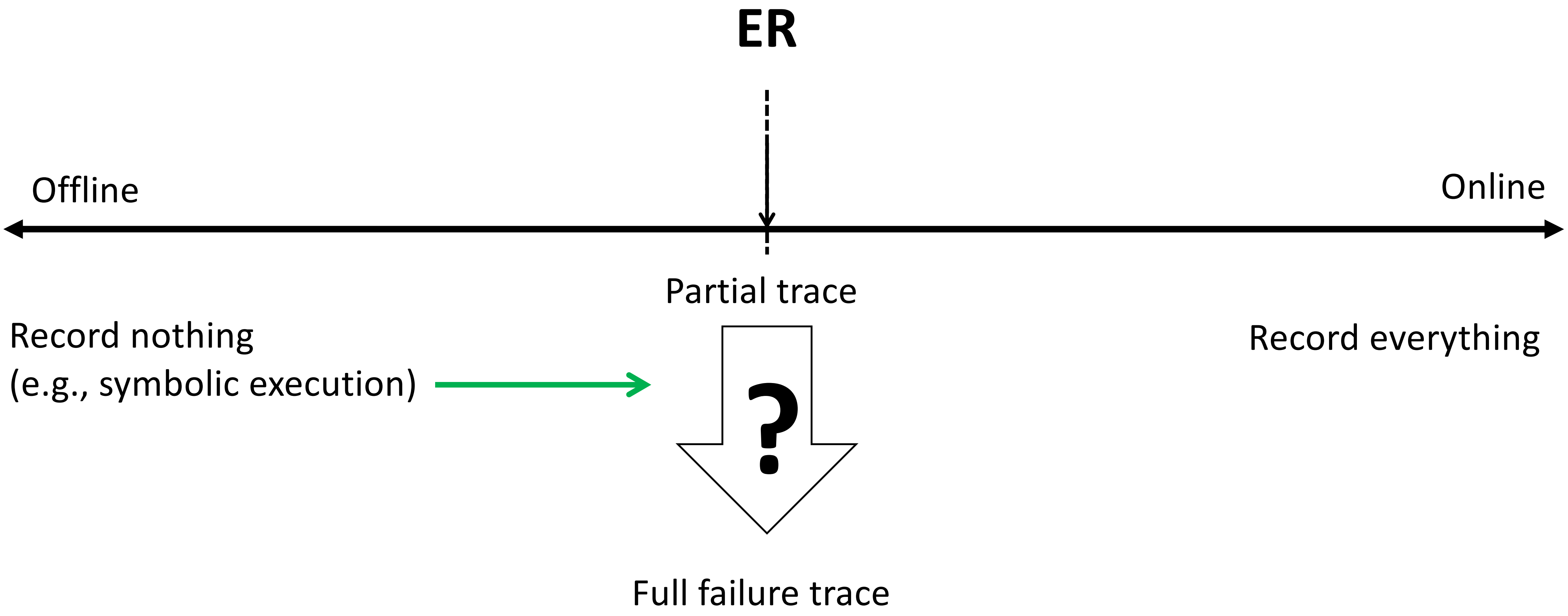
- no runtime overhead ✓
- poor reproducibility ✗

Record everything

- high runtime overhead ✗
- great reproducibility ✓

Existing trade-offs are insufficient to reproduce complex production failures

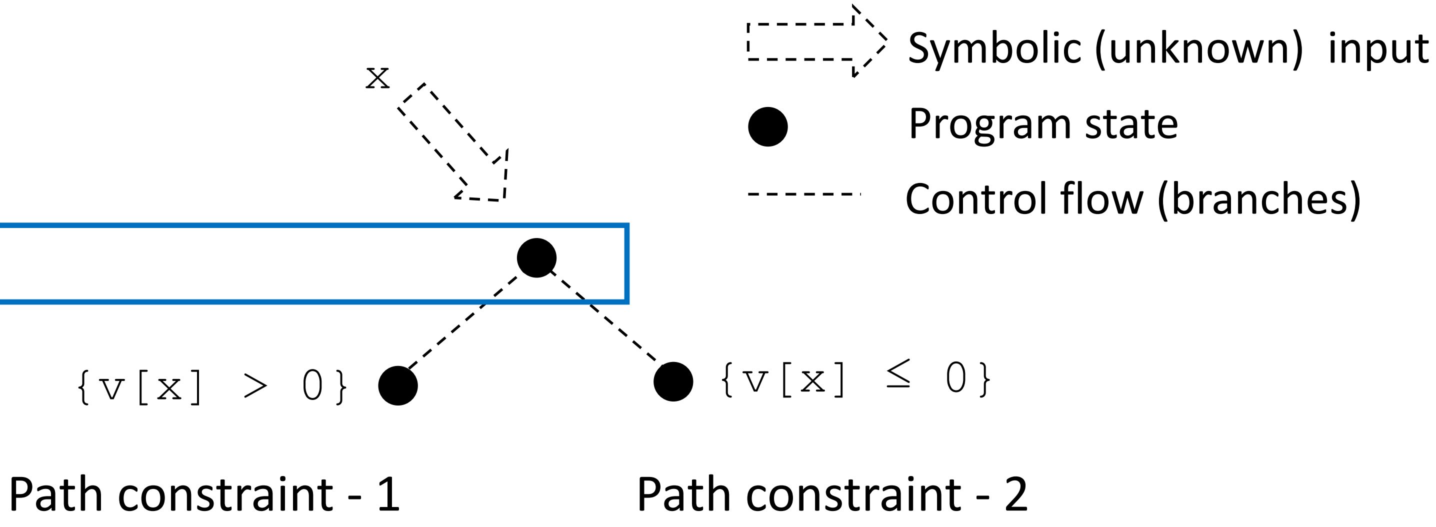
Challenges



ER
Offline
Online
Partial trace
Record nothing
(e.g., symbolic execution)
Record everything
?
Full failure trace

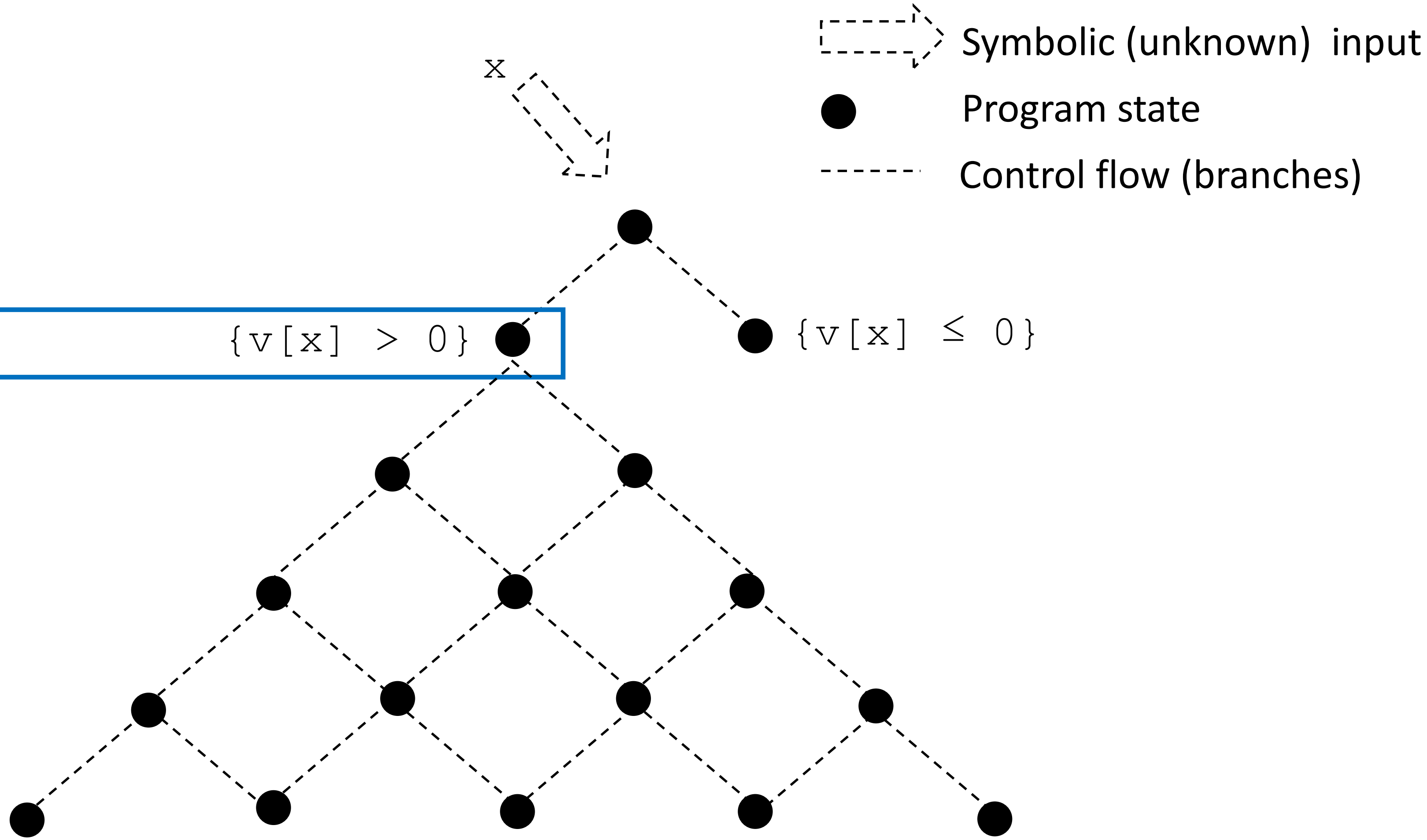
Background: Symbolic Execution

```
void foo(int x) {
  int v[16] = {0};
  if (v[x] > 0) {
```



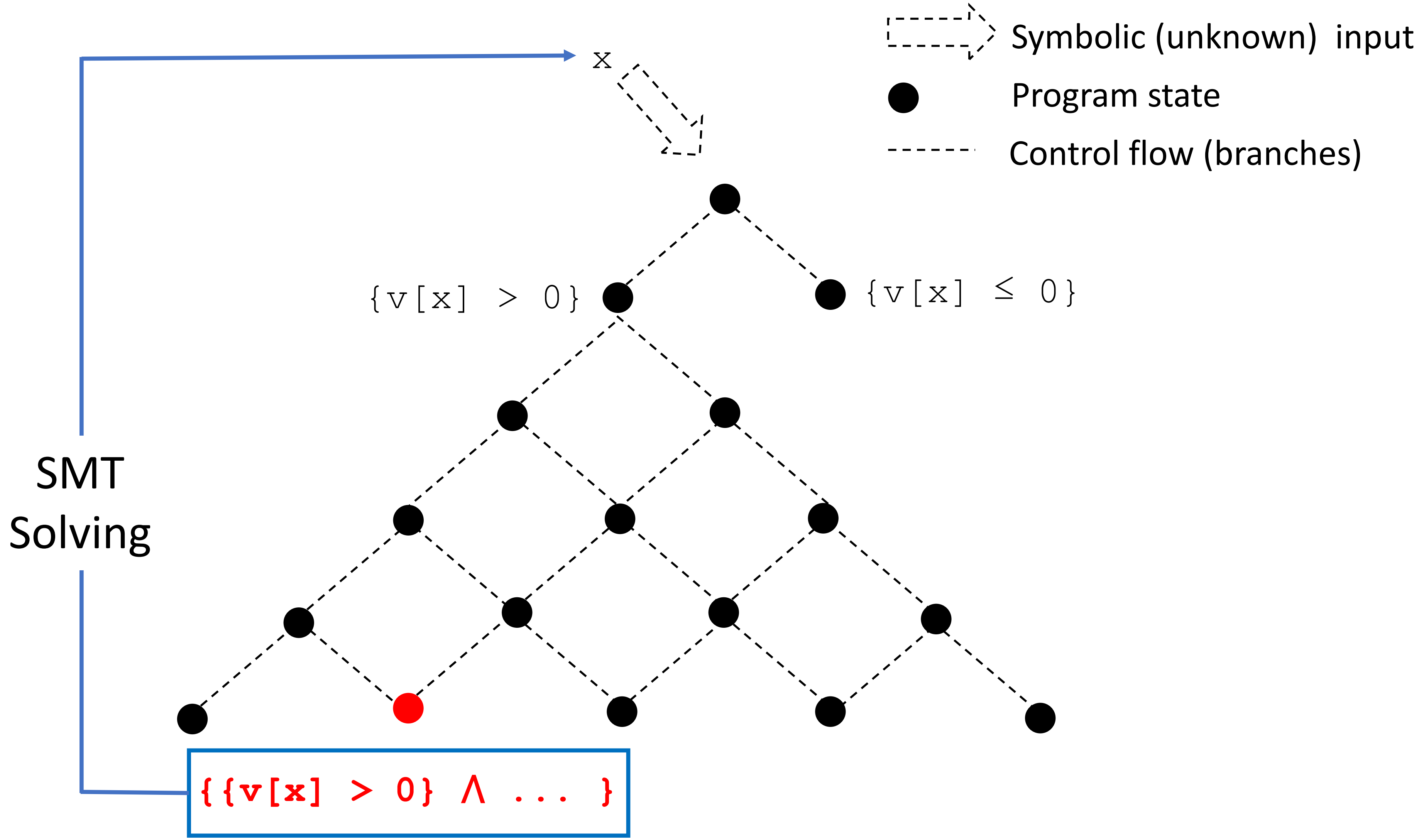
Background: Symbolic Execution

```
void foo(int x) {
  int v[16] = {0};
  if (v[x] > 0) {
    ...
    if{
    ...
    ...
    ...
    ...
    ...
    ...
    ...
```

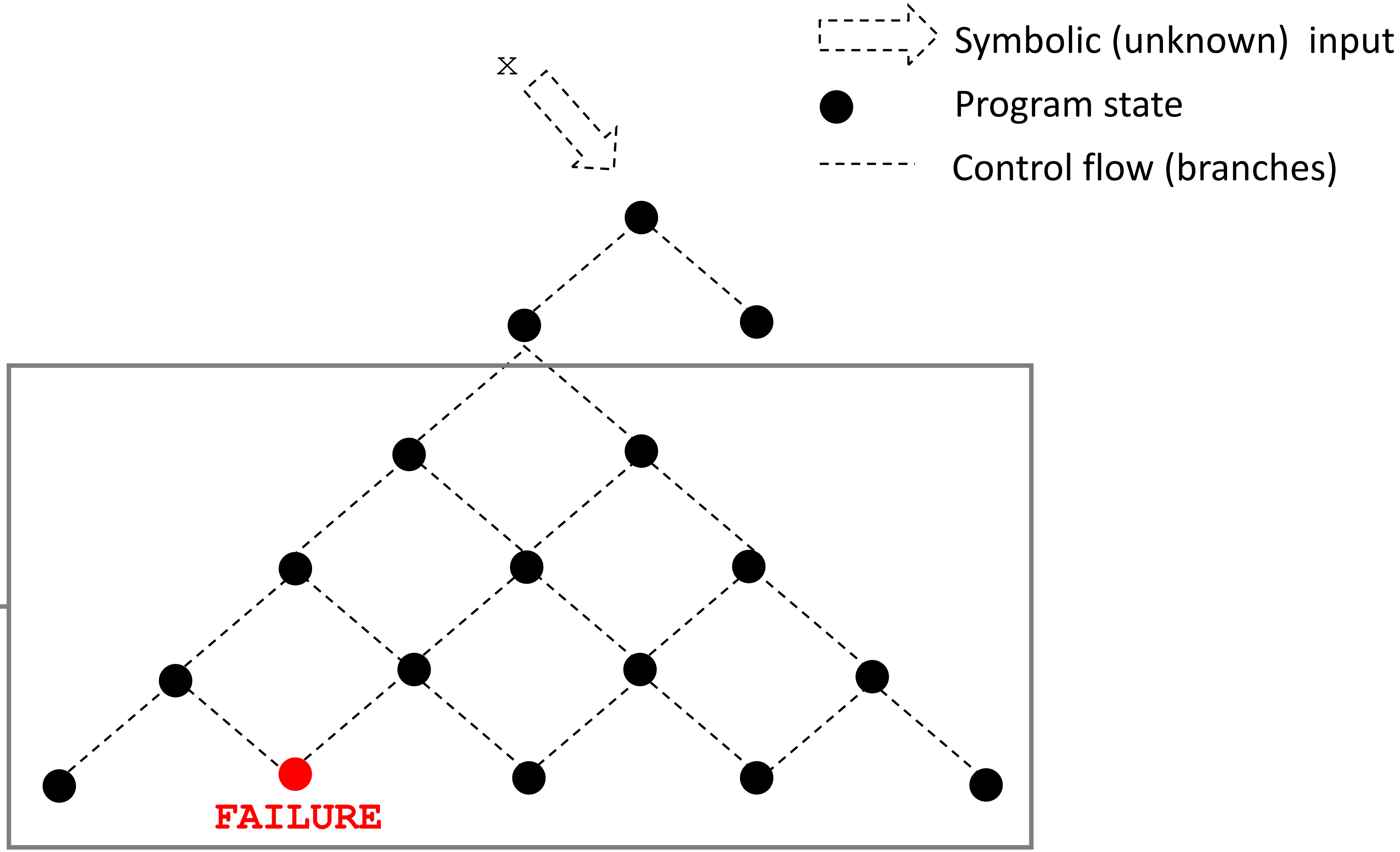


Background: Symbolic Execution

```
void foo(int x) {
  int v[16] = {0};
  if (v[x] > 0) {
    ...
    if{
    ...
    ...
    ...
    ...
    ...
    ...
    // FAILURE
```



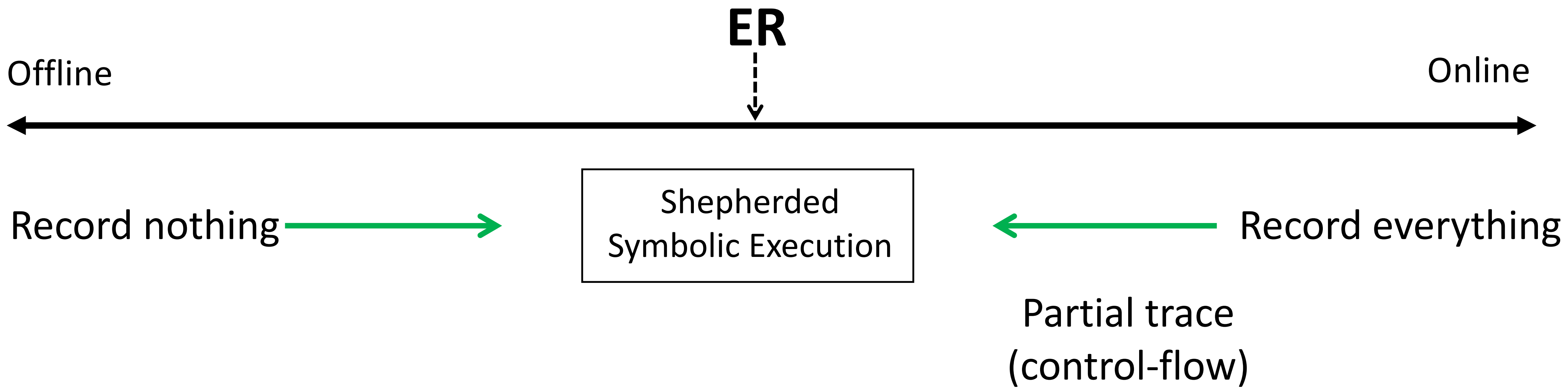
Background: Symbolic Execution



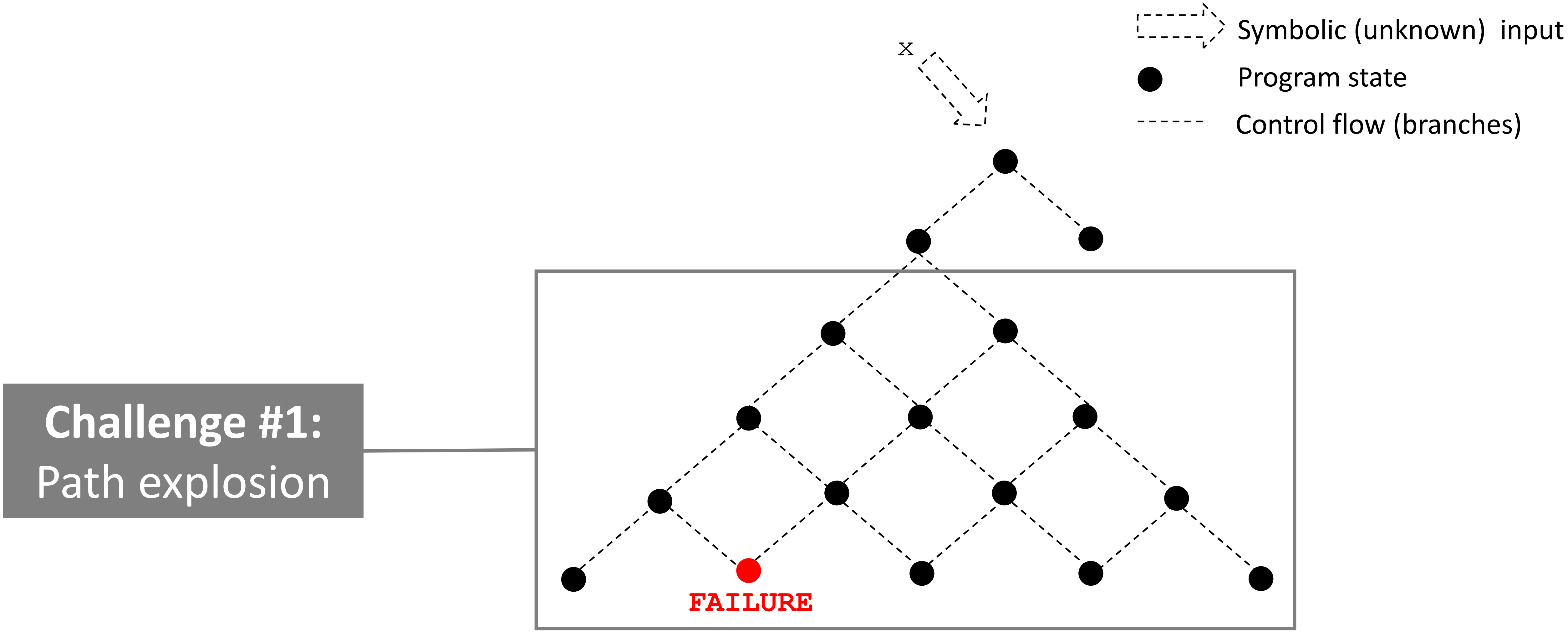
Challenge #1:
Path explosion

Shepherded Symbolic Execution

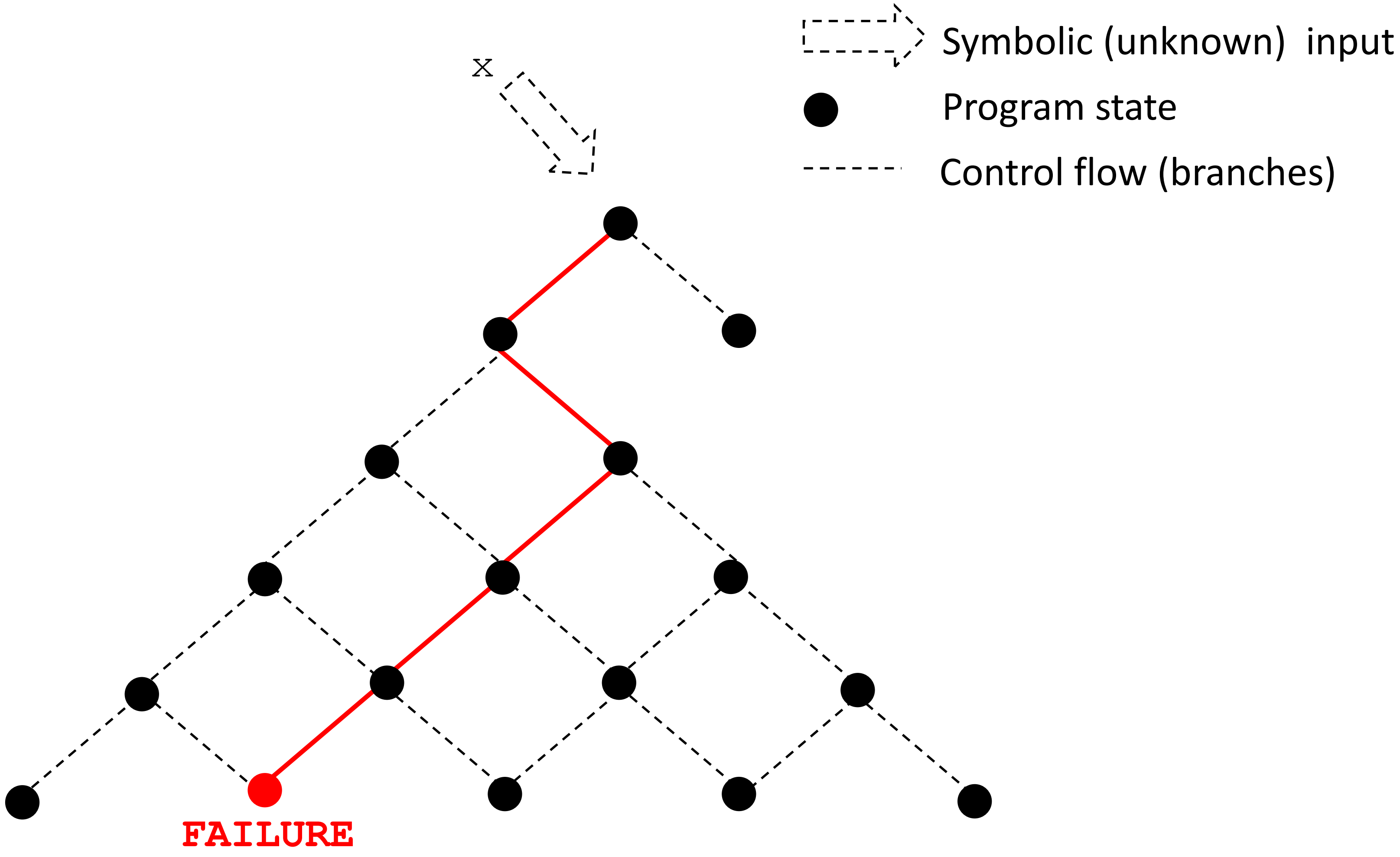
- Avoids path-explosion by following a control flow trace recorded in production



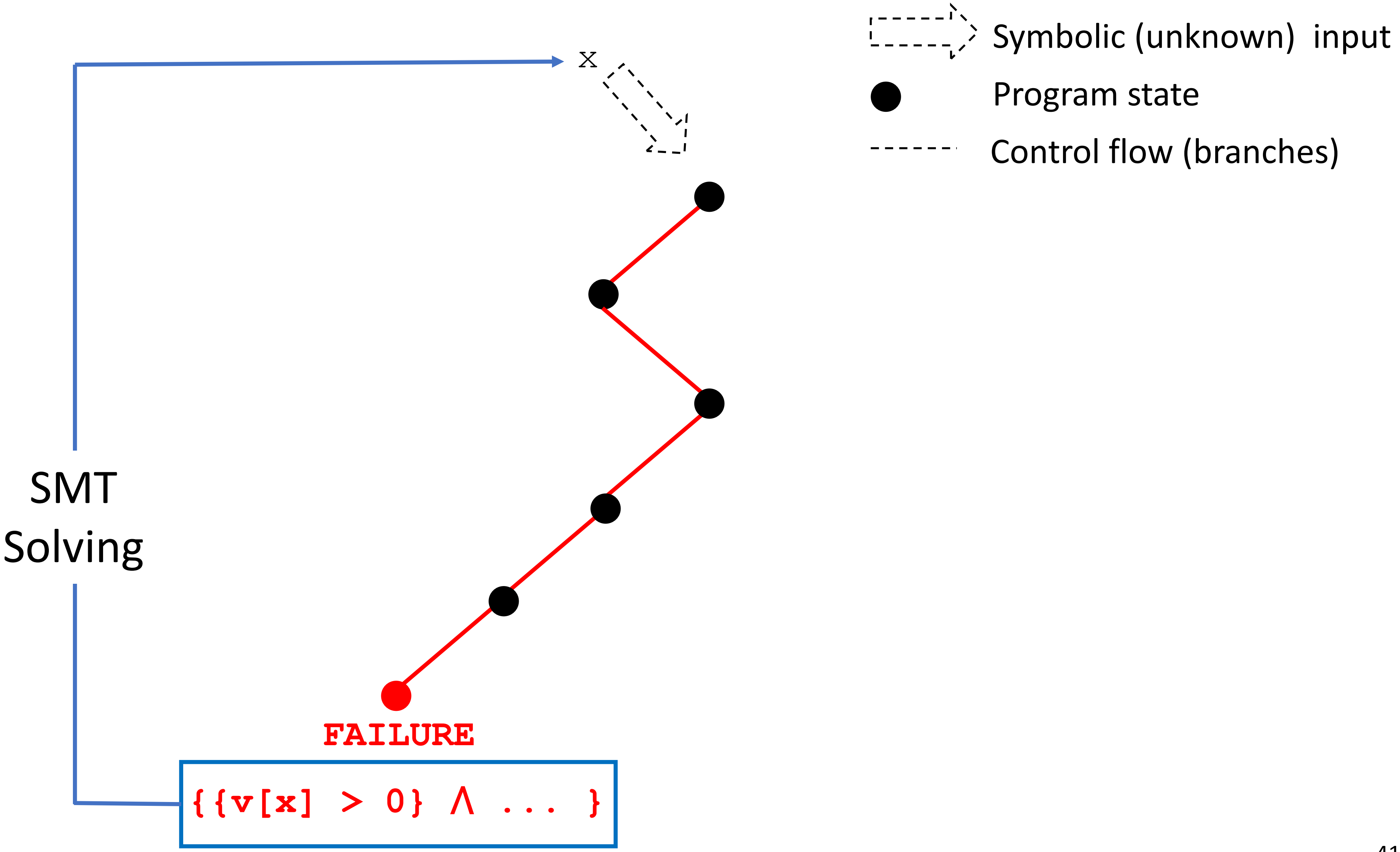
Shepherded Symbolic Execution



Shepherded Symbolic Execution



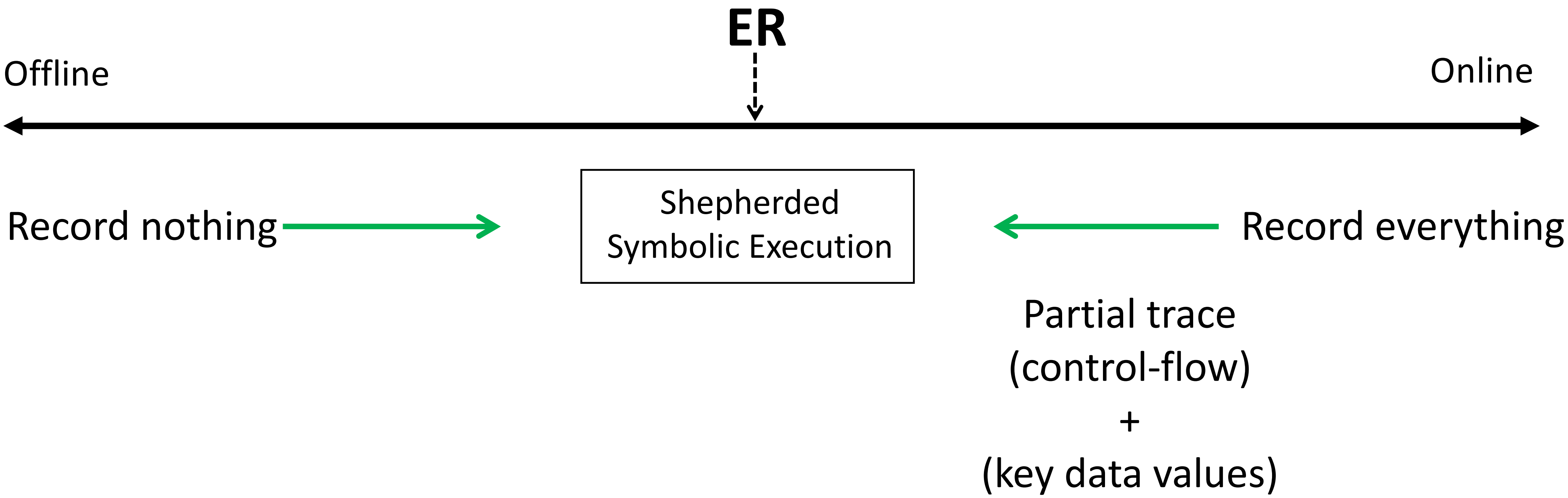
SMT Solving is Difficult



**Challenge #2:
Constraint solving
is difficult (NP complete)**

Shepherded Symbolic Execution

- Avoids path-explosion by following a control flow trace recorded in production
- Reduces (simplifies) constraints using key data values recorded in production

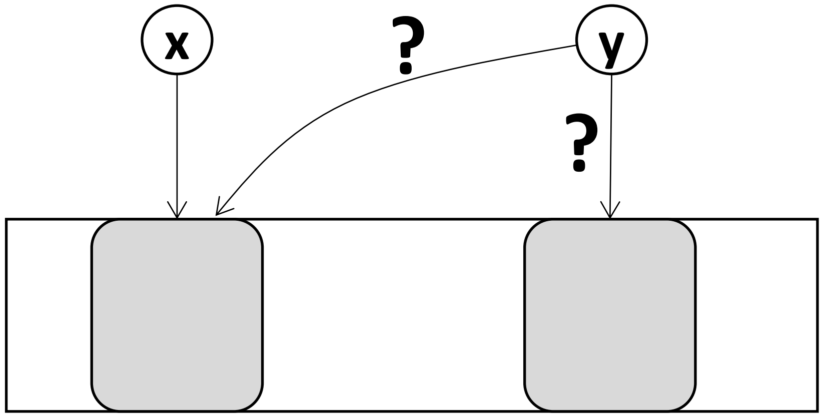


Key question: What data values best simplify constraint solving?

Constraint Simplification: Intuition

- SMT solving is a hard problem (NP-Complete)
- **Observation**: reasoning about memory aliasing takes the most time

Memory Addresses: **x** may-alias **y**

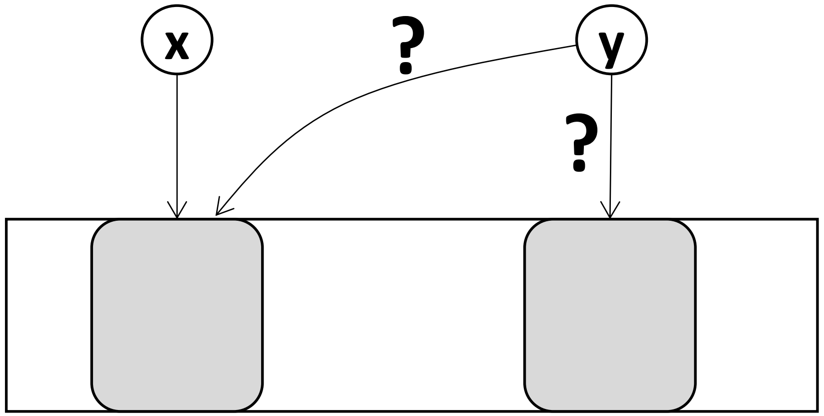


Symbolic Memory

Constraint Simplification: Intuition

- SMT solving is a hard problem (NP-Complete)
- **Observation**: reasoning about memory aliasing takes the most time

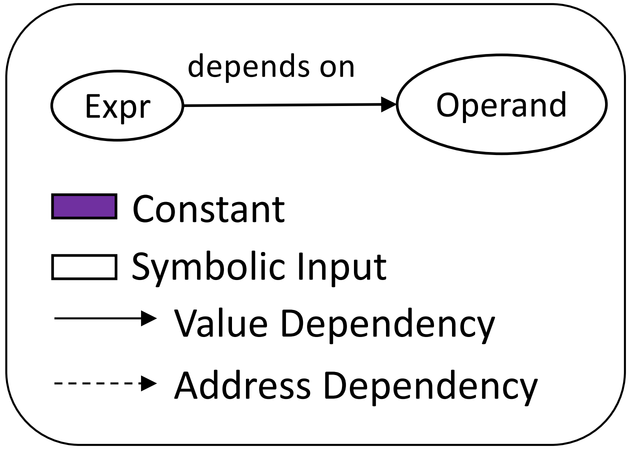
Memory Addresses: **x** may-alias **y**

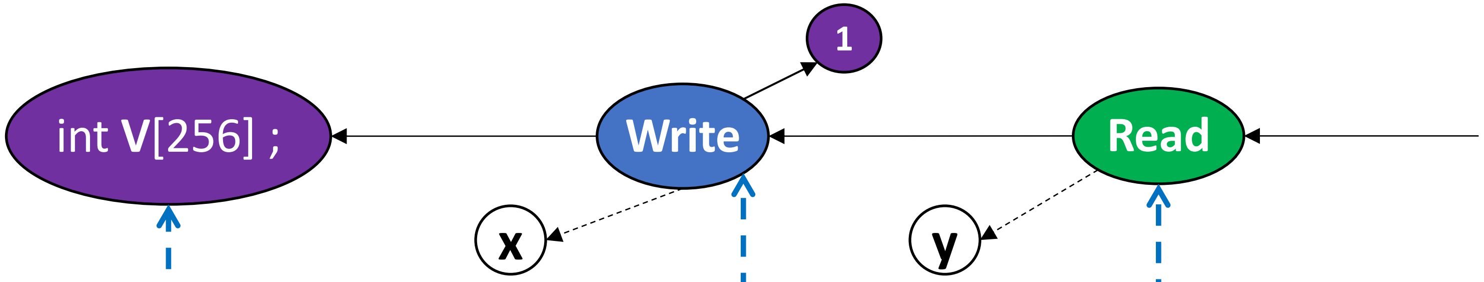


Symbolic Memory

Hypothesis: Recording addresses can simplify constraint solving

Constraint Simplification: Example

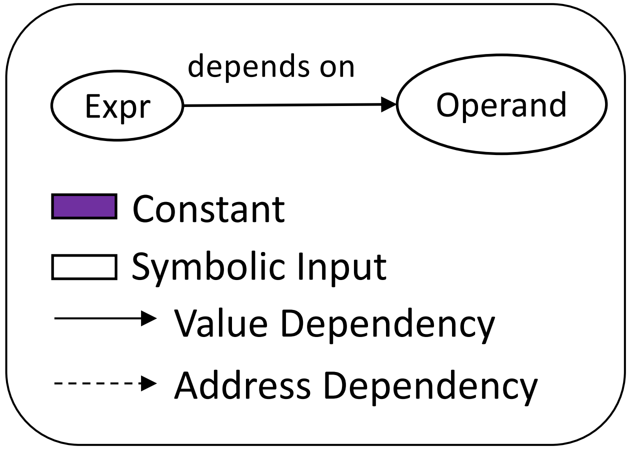


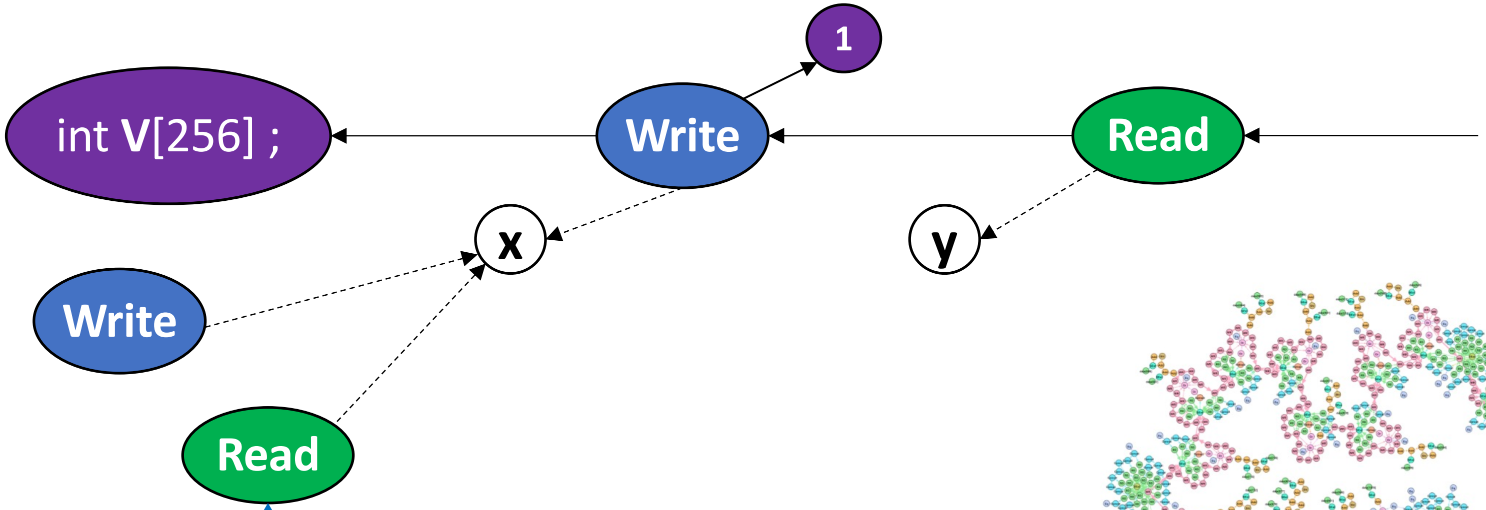


Symbolic inputs

```
void foo(int x, int y) {
 int V[256] = {0};
 V[x]= 1;
  ...V[y]...
  ...
}
```

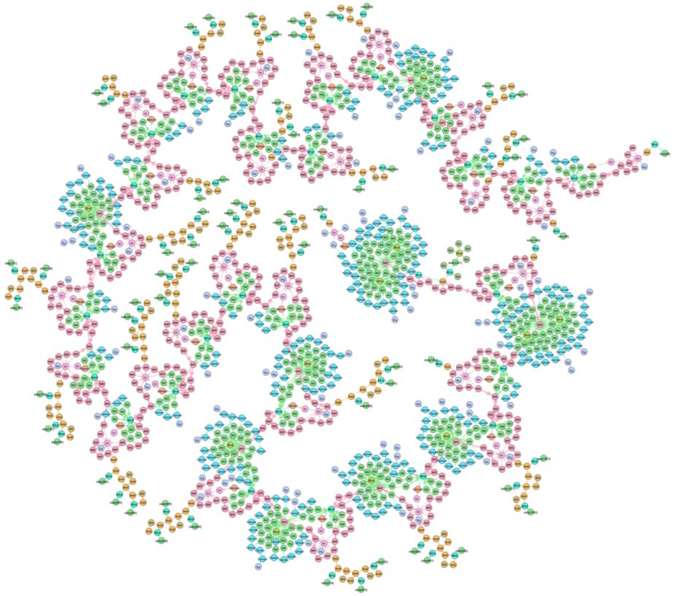
Constraint Simplification: Example





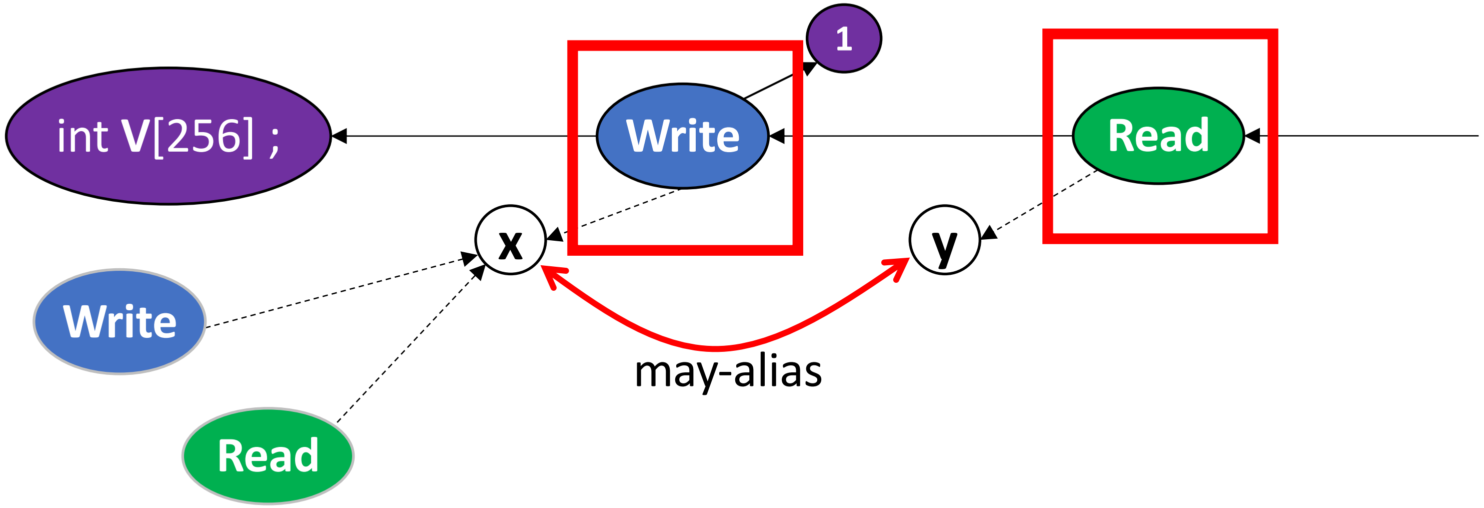
Symbolic inputs

```
void foo(int x, int y) {
 int V[256] = {0};
 V[x]= 1;
  ...V[y]...
  ...
}
```



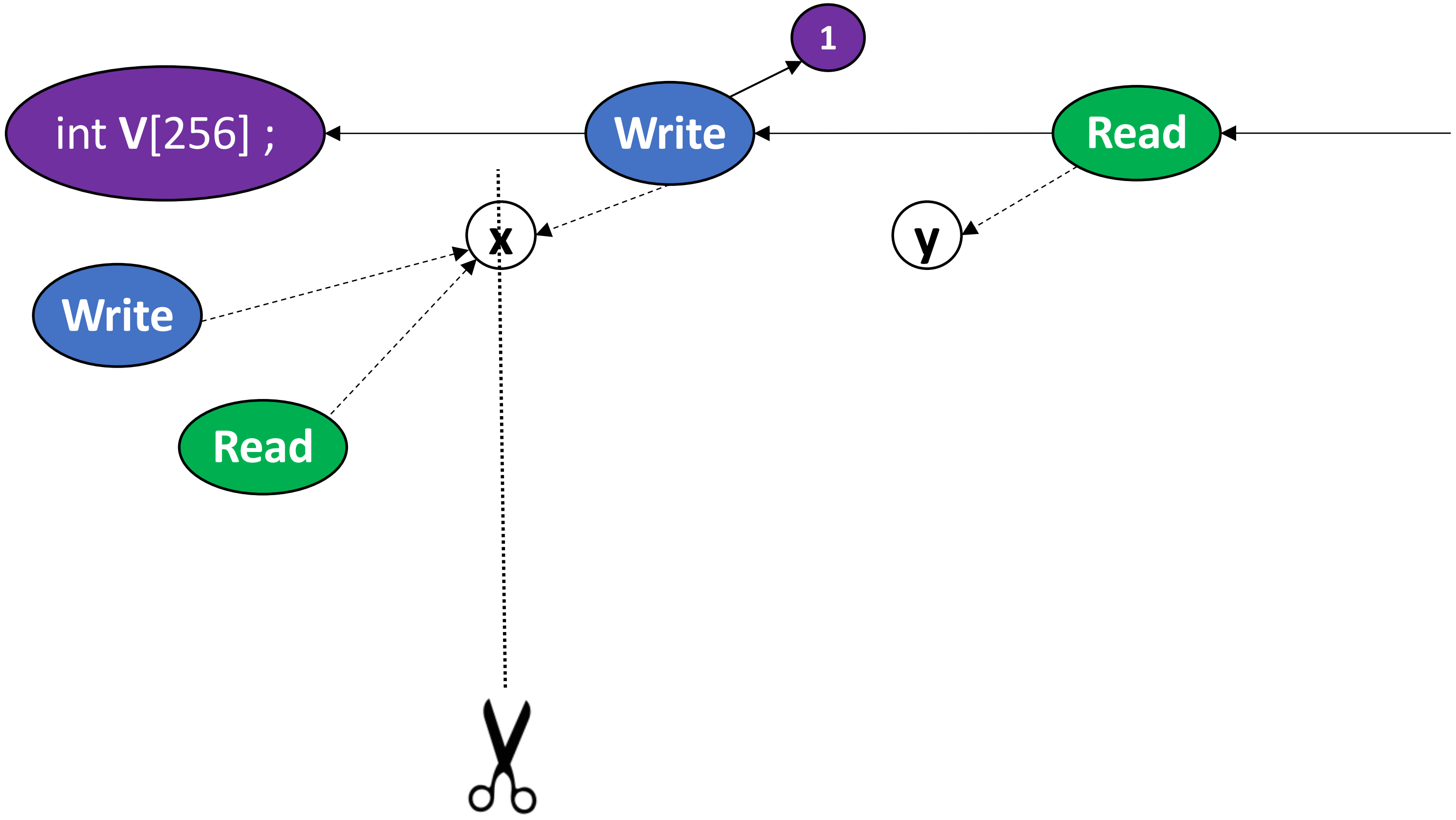
PHP-2012-2386

Constraint Simplification: Example



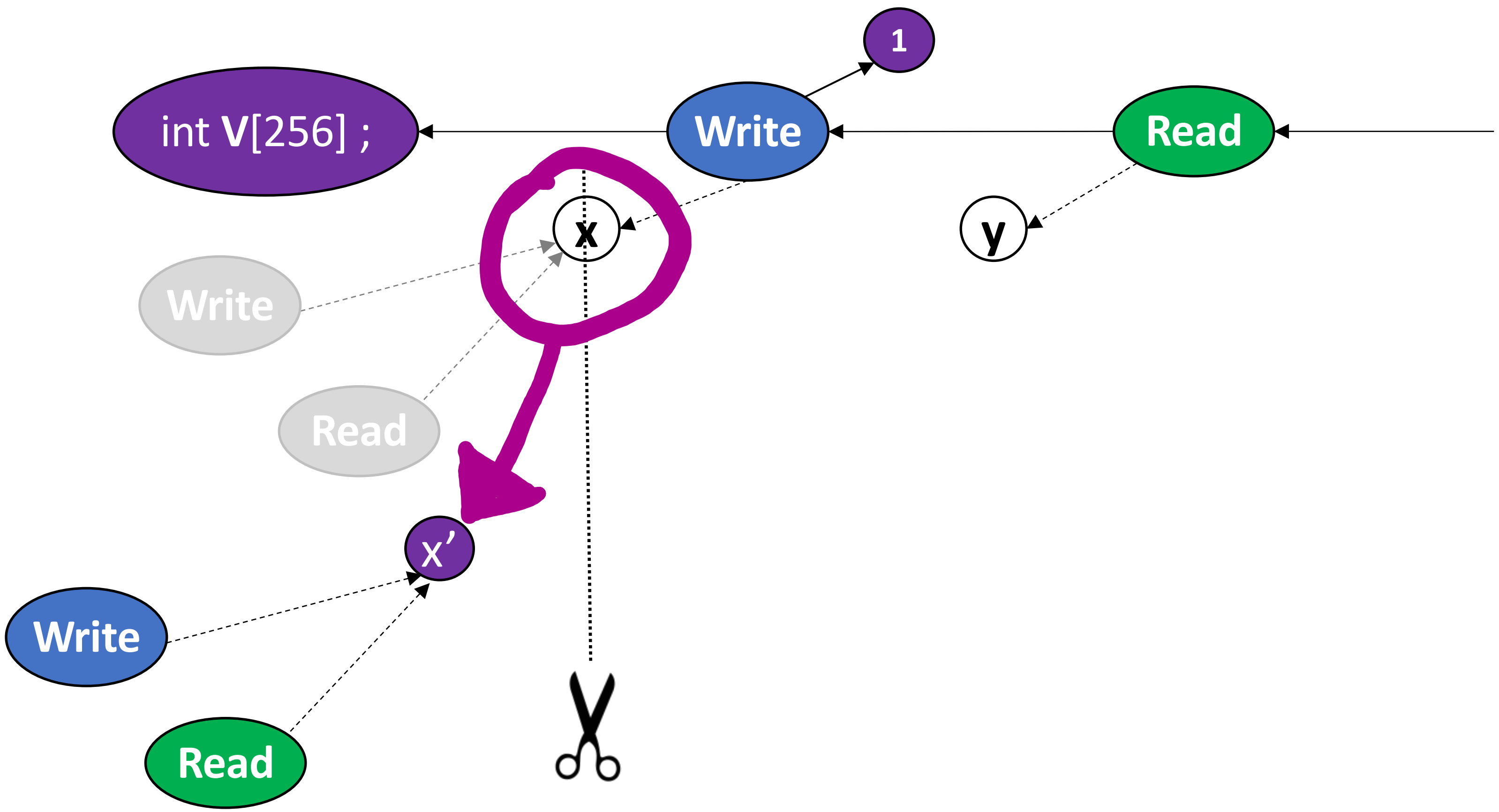
Constraint Simplification: Example

Record the runtime value of (x) as (x')



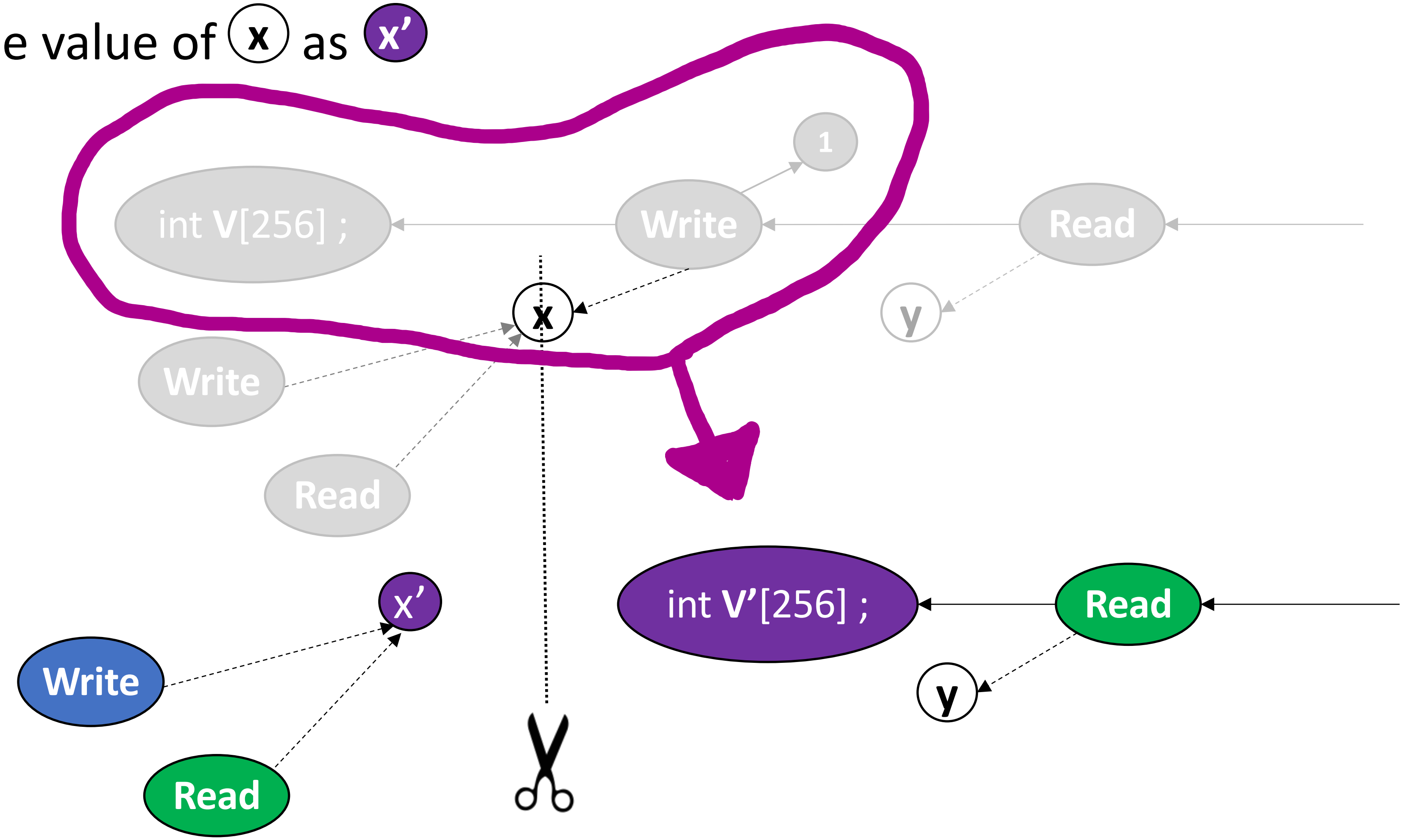
Constraint Simplification: Example

Record the runtime value of (x) as (x')



Constraint Simplification: Example

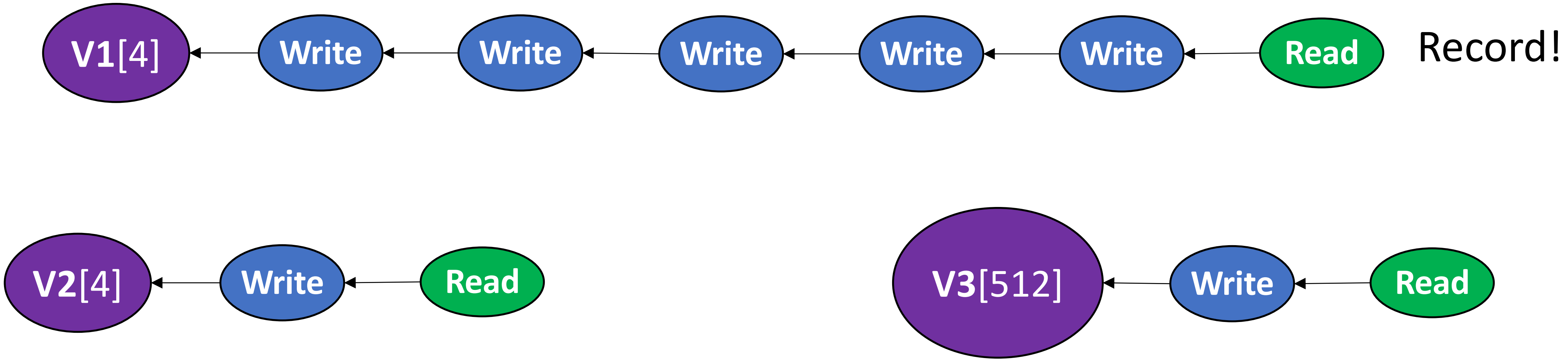
Record the runtime value of x as x’



Constraint Simplification: Heuristics

Record “key” symbolic memory addresses, which are used in:

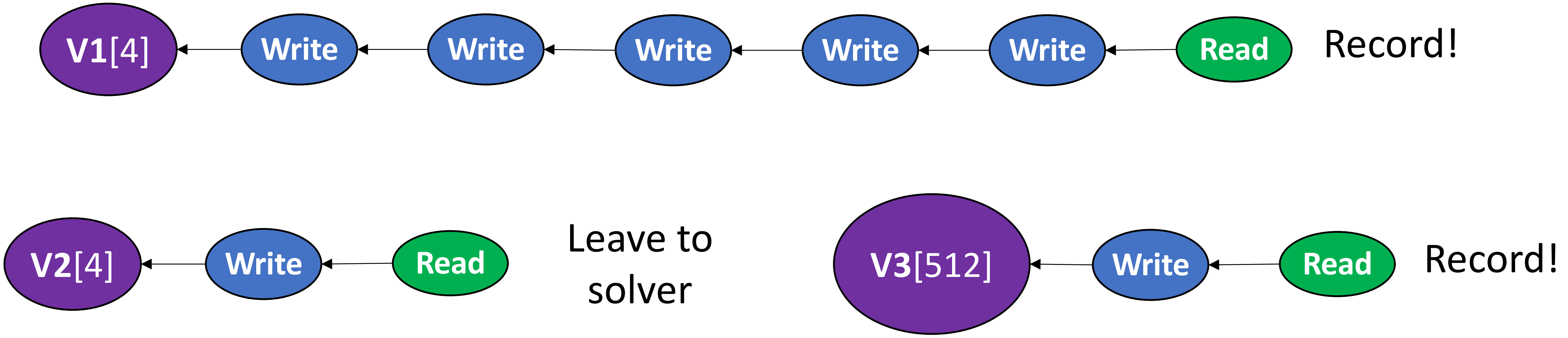
- The longest symbolic write chain
- The write chain that accesses the largest symbolic memory object



Constraint Simplification: Heuristics

Record “key” symbolic memory addresses, which are used in:

- The longest symbolic write chain
- The write chain that accesses the largest symbolic memory object



Execution Reconstruction Summary

In-production Tracing Engine (Online)

Analysis Engine (Offline)

Application + Hardware Tracing

FAILURE

❶ *Send control-flow trace*

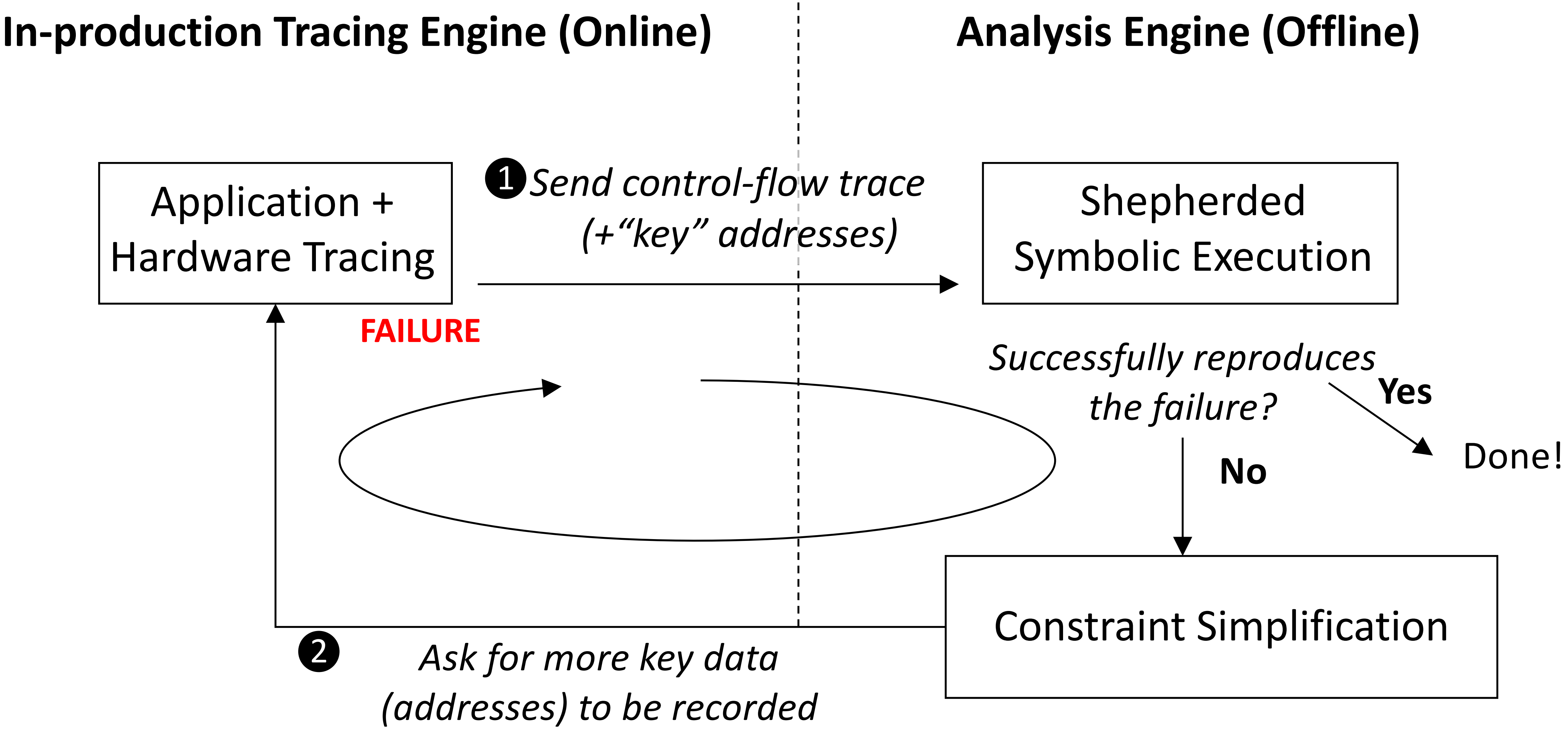
Shepherded Symbolic Execution

Successfully reproduces the failure?

Yes

Done!

Execution Reconstruction Summary



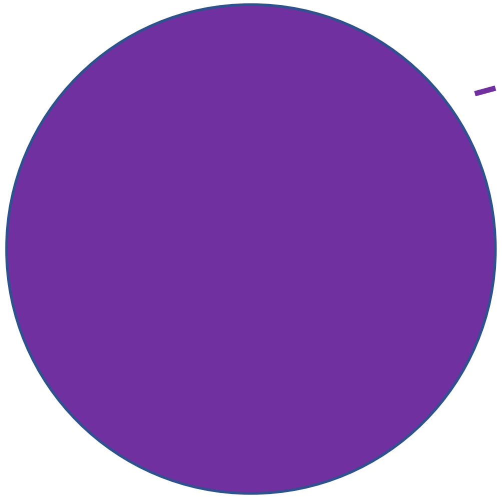
REPT vs ER

Records

Control Flow

+

Key Data Values (Addresses)



Reproduces O(10^4) instructions

Reproduces O(10^7) instructions

Execution Reconstruction – Results

Eliminates path explosion & simplifies constraint solving

- By recording control flow and key data values

Can reproduce failures in complex, long-running executions

- 1000x longer than REPT (deployed in Windows), without recording a longer trace
- Requires only 3.5 reoccurrences on average per failure

0.3% runtime performance overhead

Failure Reproduction and Analysis

OmniTable [OSDI'22]

Debugging in the Brave New World [ASPLOS'22]

ER [PLDI'21] REPT [OSDI'18] Snorlax [SOSP'17]

Hippocrates [ASPLOS'21] Agamotto [OSDI'20]

Awards

- NSF CAREER Award
- Microsoft Research Faculty Fellowship
- Google Faculty Award
  - 2019, 2021
- Microsoft Research PhD Fellowship
  - Andrew Quinn
- NSF Graduate Research Fellowship
  - Andrew Loveless, Andrew Quinn
- Towner Prize
  - Ian Neal
- OSDI Best Paper Award
- IEEE MICRO Top Pick Honorable Mention

Grants

NSF, SRC, Google

Collaborations

UT Austin, KAIST, Intel

Real-world deployment in Windows

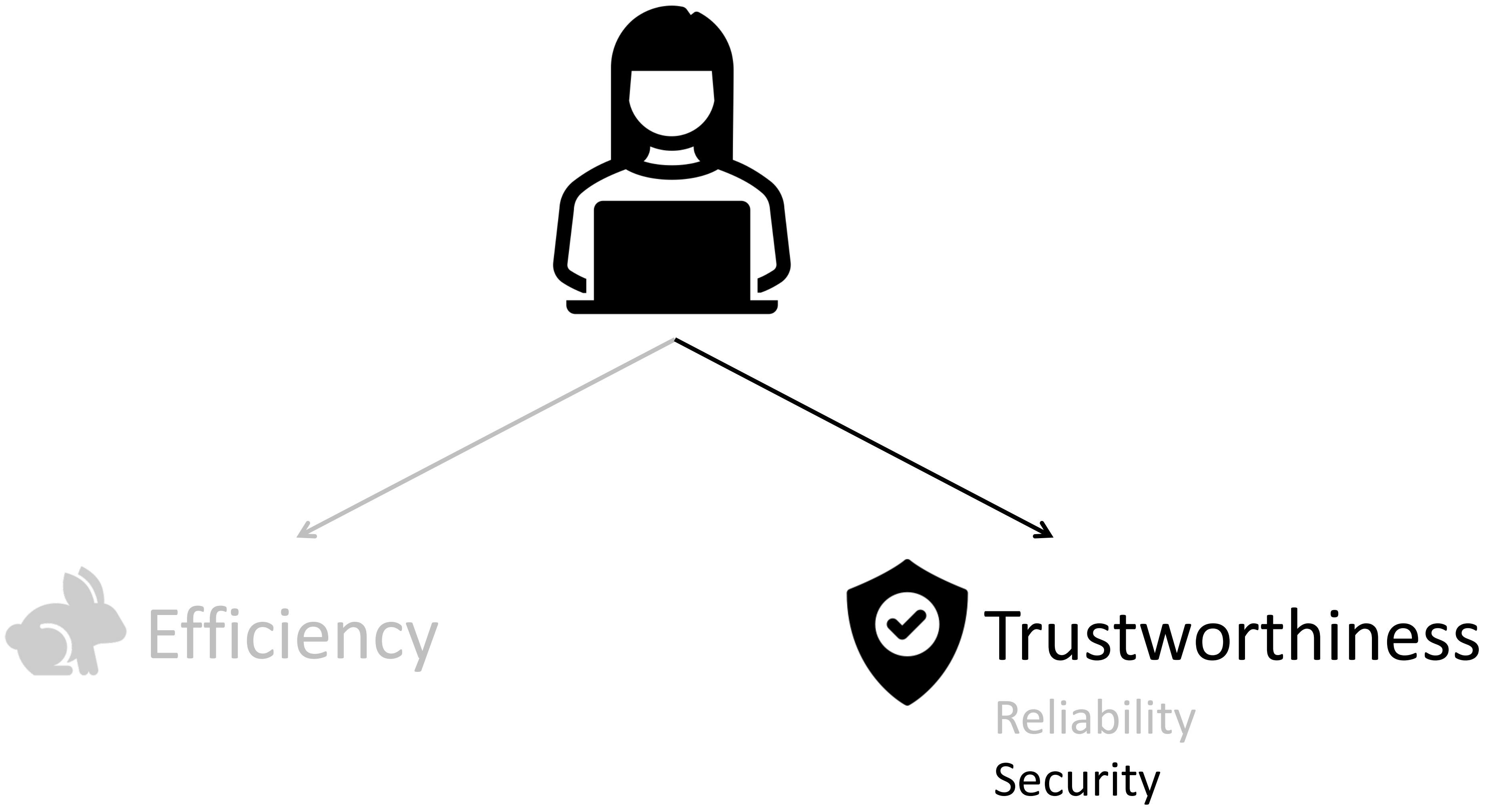
~1 billion systems

Adoption at Meta[1], Intel

[1] https://engineering.fb.com/2021/04/27/developer-tools/reverse-debugging/

Outline

	Offline	Online
Datacenter Efficiency	Data-driven optimizations	Lightweight profiling
Failure Reproduction and Analysis	Static & symbolic program analysis	Selective information monitoring
Hardware Security		


Efficiency
Trustworthiness
Reliability
Security

Outline

	Offline	Online
Datacenter Efficiency	Data-driven optimizations	Lightweight profiling
Failure Reproduction and Analysis	Static & symbolic program analysis	Selective information monitoring
Hardware Security	Classification of attacks	Threat model-specific defenses

WIRED

LILY HAY NEWMAN SECURITY 08.14.18 01:00 PM

SPECTRE-LIKE FLAW UNDERMINES INTEL PROCESSORS' MOST SECUR

FORESHADOW

Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution

Read the paper | Cite | Watch a demo

USENIX Security'18

The Register®

Biting the hand that feeds IT

Security

Foreshadow and Intel SGX software attestation: 'The whole trust model collapses'

ZDNet

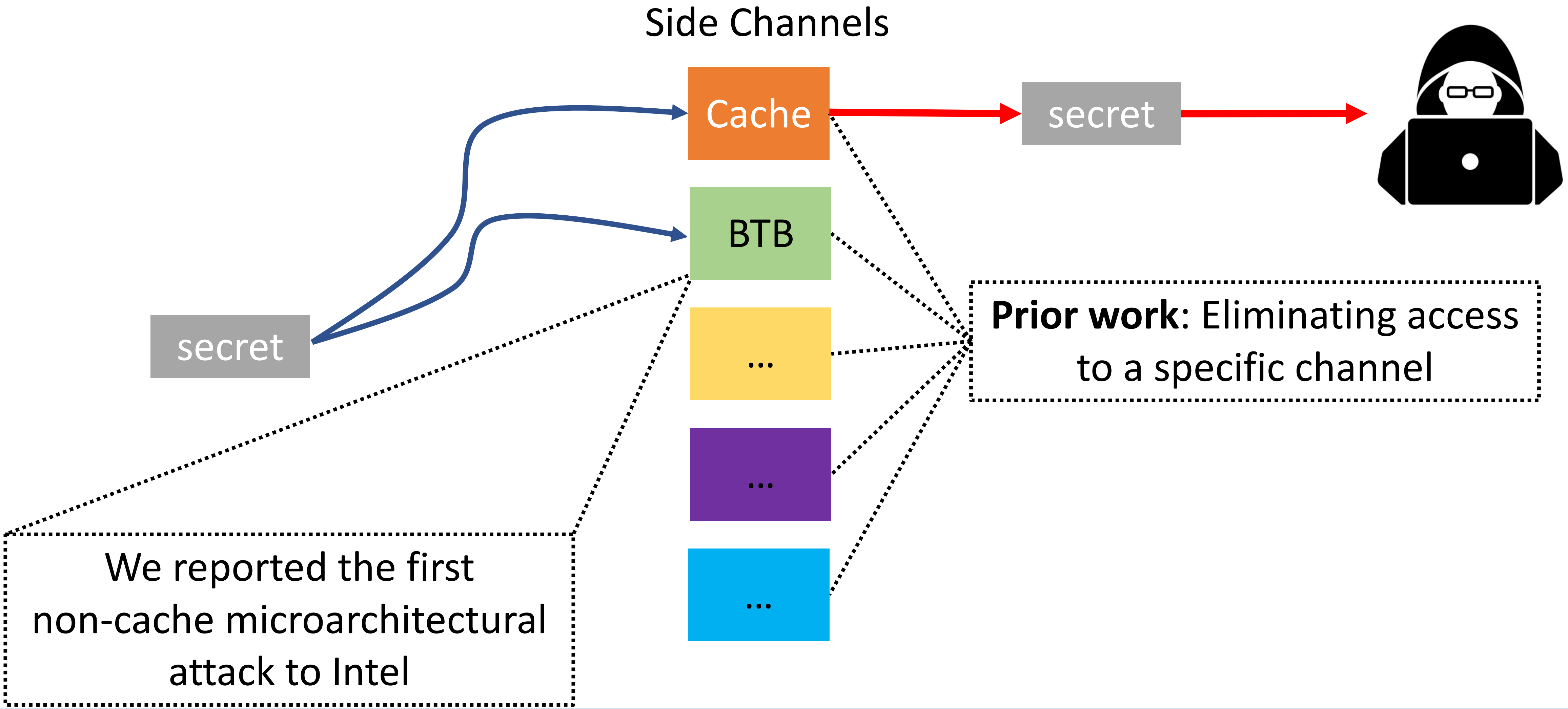
TECHNICA

WON'T BE THE LAST SUCH PROBLEM

s SGX blown wide open by, you sed it, a speculative execution attack

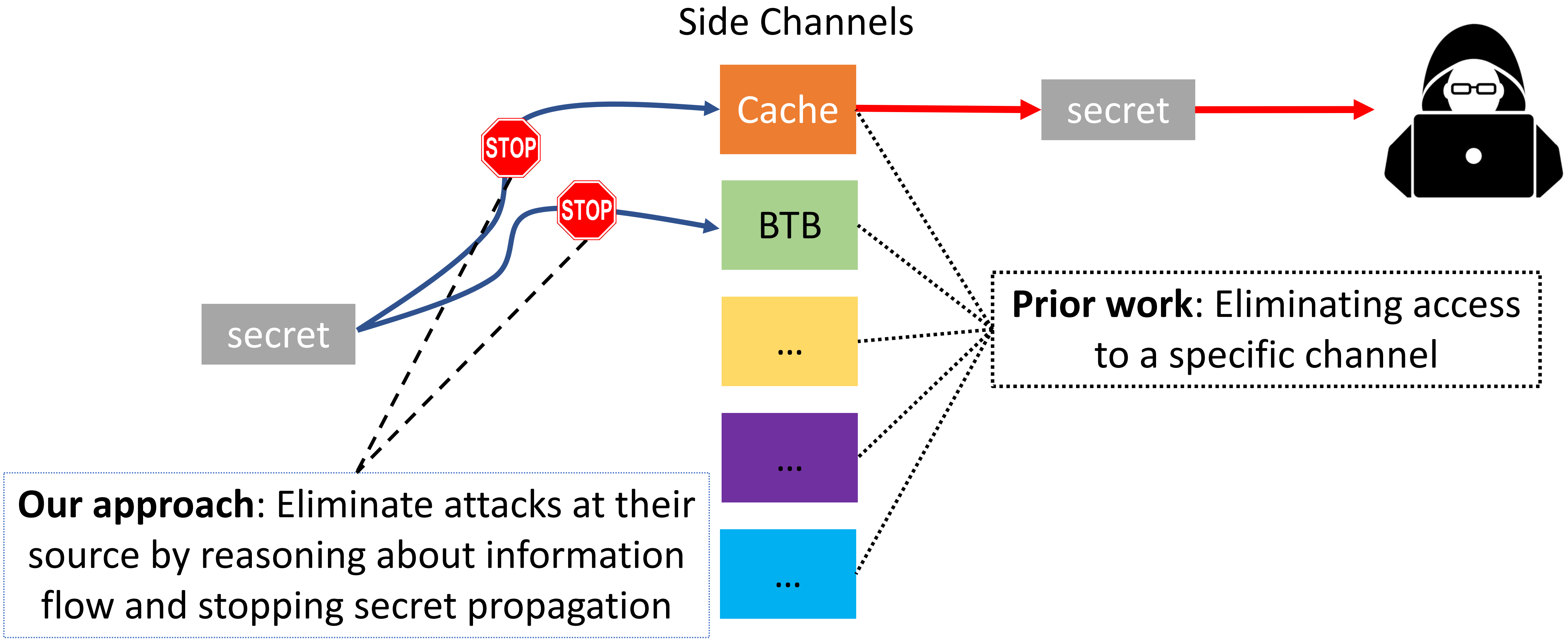
IEEE MICRO Top Pick

Foreshadow bypasses the virtual memory abstraction:
A VM in the Cloud can leak secrets from someone else's VM



Prior defenses were channel-specific

Principled and Comprehensive Defenses



Navigating the efficiency/trustworthiness tension: defenses tailored to threats

NDA: Preventing Speculative Execution Attacks at Their Source

Ofir Weisse
University of Michigan

Ian Neal
University of Michigan

Kevin Loughlin
University of Michigan

Thomas F. Wenisch
University of Michigan

Baris Kasikci
University of Michigan

COMPUTER SCIENCE
& ENGINEERING
UNIVERSITY OF MICHIGAN

MICRO'19

IEEE Micro Top-Pick
Honorable Mention

NDA's Key Insight

Speculative execution attacks require a chain of **dependent instructions** to access and transmit secrets.

By controlling data propagation, NDA can **break these dependency chains**, thwarting the code sequences required to mount attacks.

Analysis of Attacks: A Chain of Dependent Instructions

I) Access Phase

↓

II) Transmit Phase
e.g., via the cache

↓

III) Recover Phase
using timing measurements

I) and II): During speculation

III): After speculation

Analysis of Attacks: A Chain of Dependent Instructions

I) Access Phase

II) Transmit Phase
e.g., via the cache

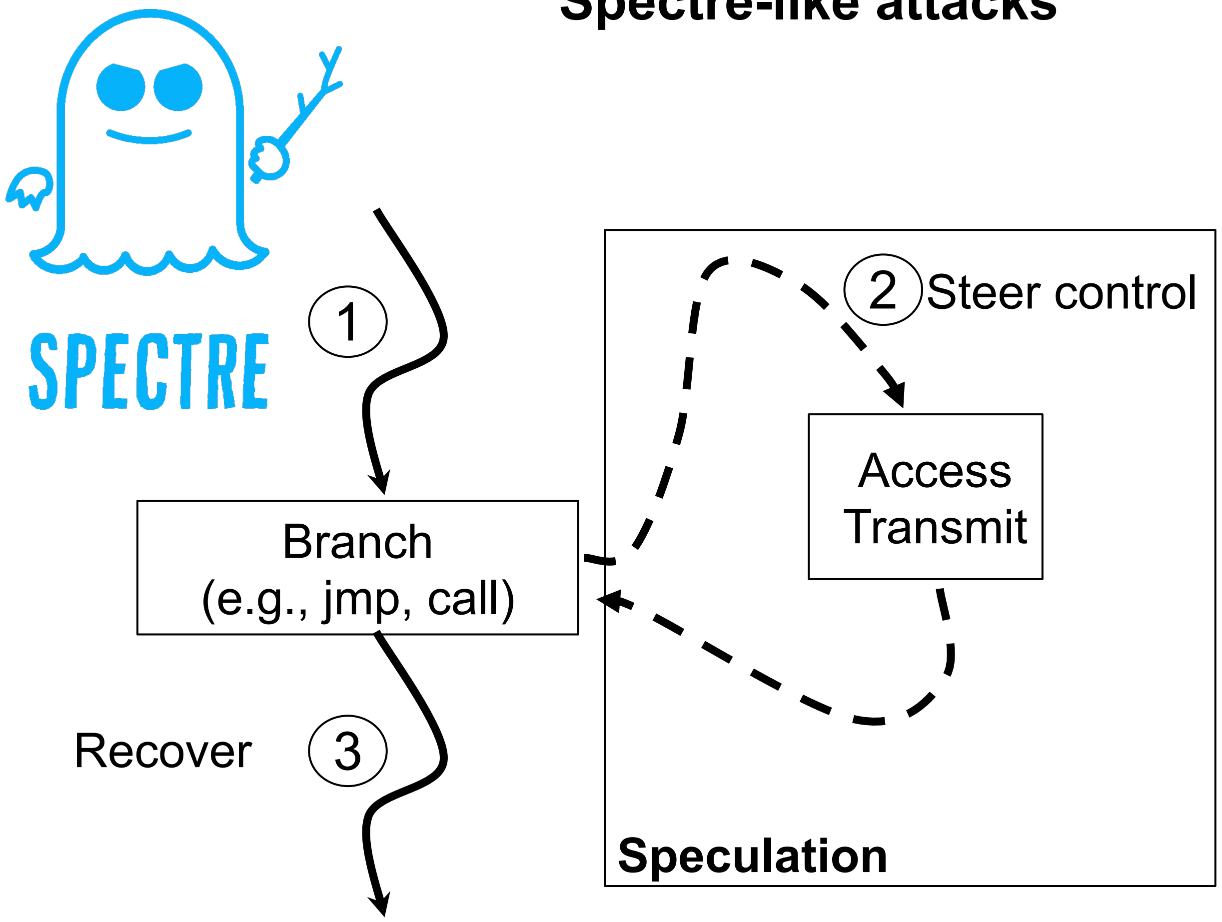
III) Recover Phase
using timing
measurements

NDA can **break** the chain of
dependent instructions

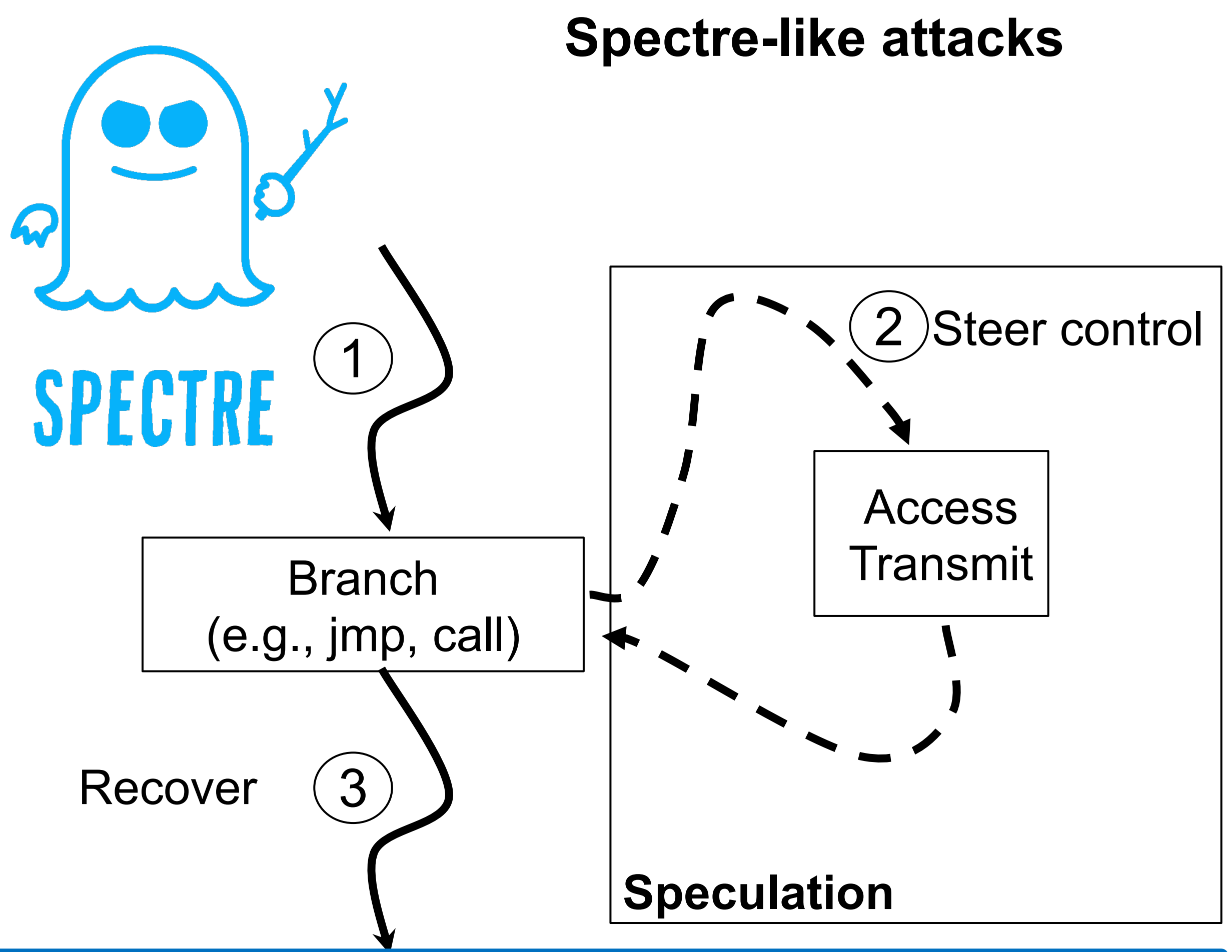
Only **"unsafe"** instructions are not allowed to speculatively transmit secrets

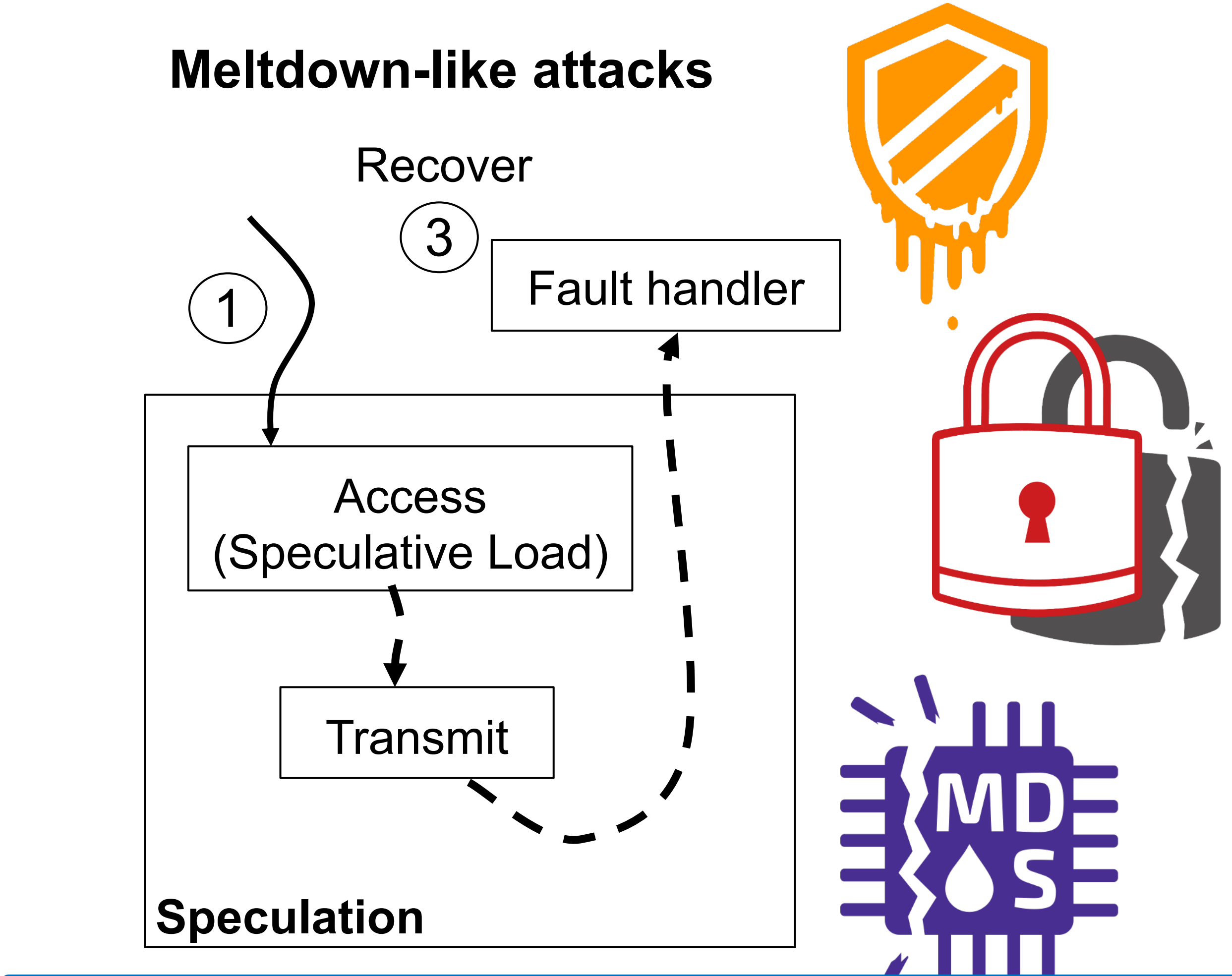
Unsafe Instructions: A Threat-Model-Centric View

Spectre-like attacks



Unsafe Instructions: A Threat-Model-Centric View





In Spectre-like attacks, instructions after a branch are potentially unsafe

In Meltdown-like attacks, all loads are potentially unsafe

NDA - Summary

During speculation, NDA:

- Allows the execution of unsafe instructions (access)
- Disallows broadcasting the effects of unsafe instructions (transmission)

Much lower overhead than in-order execution (4.8x)

- 10.7% overhead against Spectre-like attacks
- 36.1% overhead against Meltdown-like attacks

More comprehensive security than channel-specific defenses

- Protection against existing and future side-channels

Hardware Security

MOESI-prime [ISCA'22]

Dolma [SEC'21]

NDA [MICRO'19]

Foreshadow [SEC'18]

Morpheus [ASPLOS'19]

Awards

Facebook Fellowship
- Marina Minkin

NSF Graduate Research Fellowship
- Kevin Loughlin

Google Fellowship
- Kevin Loughlin

IEEE MICRO Top Pick

IEEE MICRO Top Pick Honorable Mention

Grants

DARPA, ONR

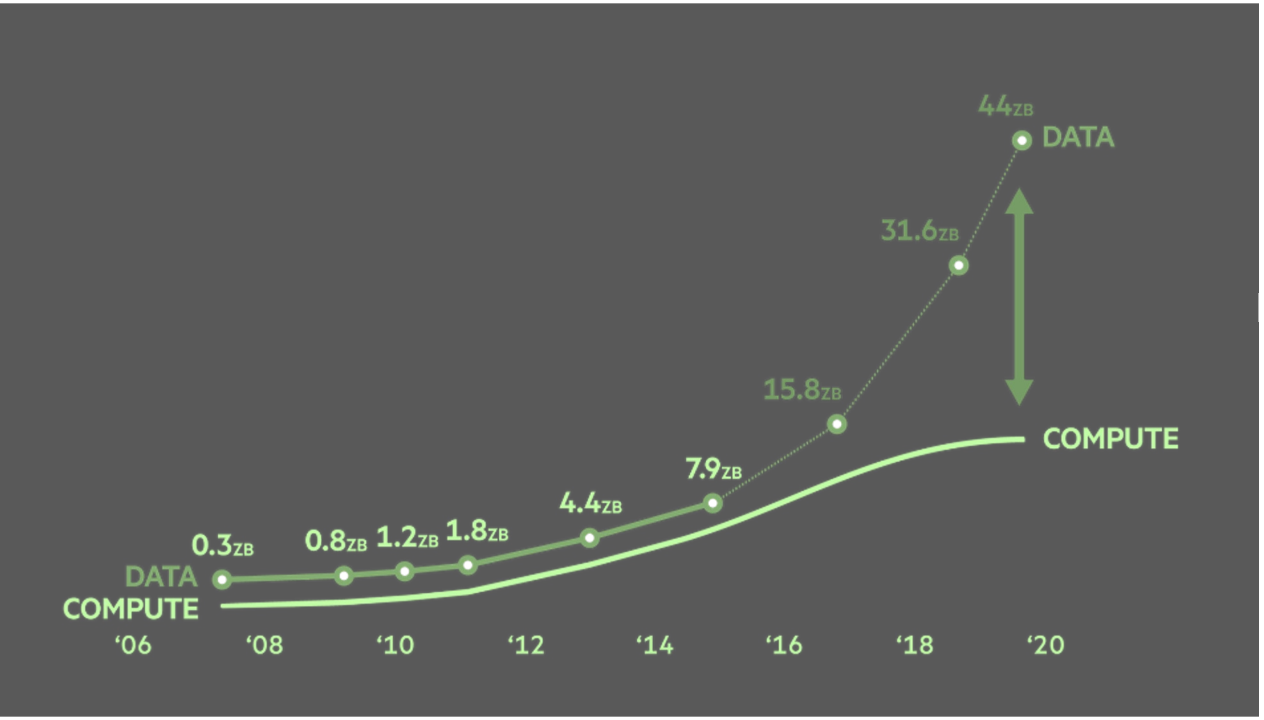
Collaborations

Microsoft, KU Leuven, Technion, University of Adelaide

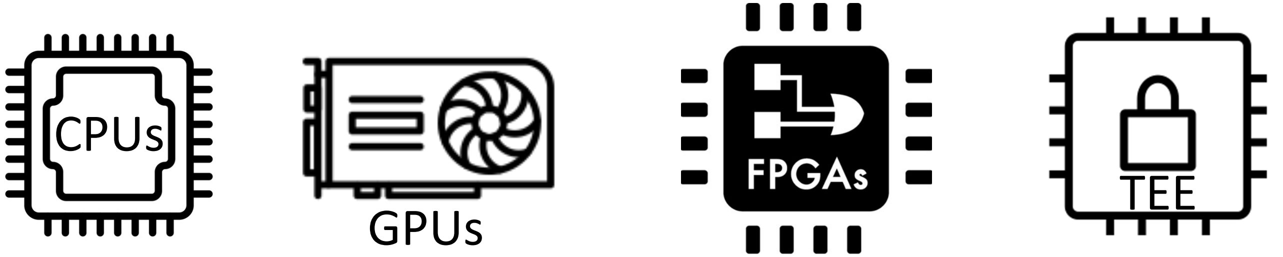
Improved cloud security

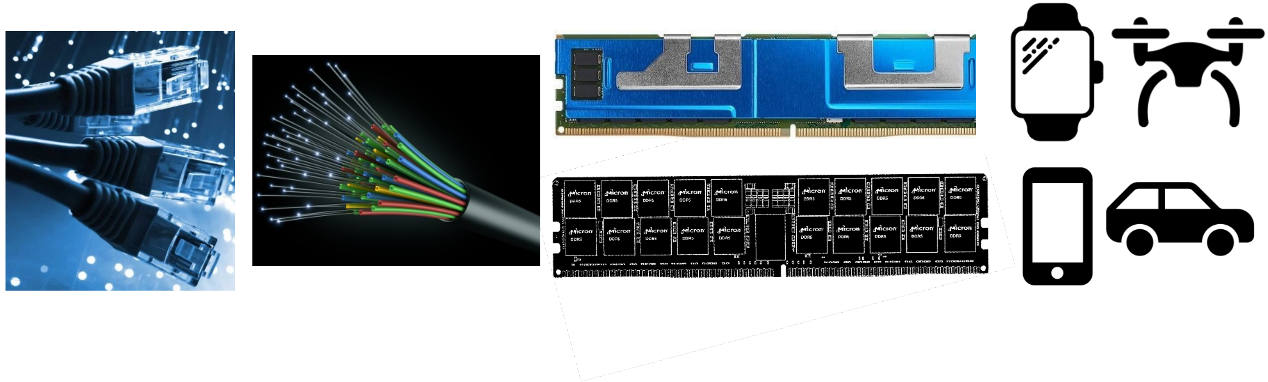
Processor upgrades and patches

Future Work



Data trends will continue driving software complexity up





Increased heterogeneity, new interconnects, and more edge devices will bring entirely new efficiency and trustworthiness challenges

Exploring emerging computer systems problems
through the lens of computer architecture, programming languages, and security

Data Center Efficiency

Trends

Increasing memory, compute, and interconnect heterogeneity

- Multi-tier memory, specialized hardware, diverse interconnects

Future Work

#1: Rethinking profile-guided optimizations (via HW/SW co-design) for heterogeneous systems

Data Center Efficiency

Trends

Increasing memory, compute, and interconnect heterogeneity

- Multi-tier memory, specialized hardware, diverse interconnects

Future Work

#1: Rethinking profile-guided optimizations (via HW/SW co-design) for heterogeneous systems

#2: Designing new systems abstractions for resource management in a disaggregated environment

Reliability

Trends

Increased heterogeneity and hardware specialization

Always-on profiling/monitoring on the edge and datacenter

- Primarily used for performance optimizations

Future Work

#1: Using production data to rethink our approach to trustworthiness

Reliability

Trends

Increased heterogeneity and hardware specialization

Always-on profiling/monitoring on the edge and datacenter

- Primarily used for performance optimizations

Future Work

#1: Using production data to rethink our approach to trustworthiness

#2: Techniques for building more reliable heterogeneous systems[1]

[1] Debugging in the Brave New World of Reconfigurable Hardware. Jiacheng Ma, Gefei Zuo, Kevin Loughlin, Andrew Quinn, Baris Kasikci. ASPLOS 2022

Hardware Security

Trends

Microarchitectural isolation is a recurring problem

Increased intermittent and silent hardware errors[1,2]

Future Work

#1: Microarchitectural isolation as a foundational security primitive

[1] Cores That Don't Count, Peter H. Hochschild Paul Jack Turner Jeffrey C. Mogul Rama Krishna Govindaraju Parthasarathy Ranganathan David E Culler Amin Vahdat Proc. HotOS 2021

[2] Silent Data Corruptions at Scale, Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, Sriram Sankar, Arxiv, 2021

Hardware Security

Trends

Microarchitectural isolation is a recurring problem

Increased intermittent and silent hardware errors[1,2]

Future Work

#1: Microarchitectural isolation as a foundational security primitive

#2: Techniques for eliminating/reducing hardware errors[3]

[1] Cores That Don't Count, Peter H. Hochschild Paul Jack Turner Jeffrey C. Mogul Rama Krishna Govindaraju Parthasarathy Ranganathan David E Culler Amin Vahdat Proc. HotOS 2021

[2] Silent Data Corruptions at Scale, Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, Sriram Sankar, Arxiv, 2021

[3] Preventing Coherence-Induced Hammering in Commodity Workloads. Kevin Loughlin, Stefan Saroiu, Alec Wollman, Yatin Manerkar, **Baris Kasikci**, ISCA'22


VANS

Efficiency

Trustworthiness

Datacenter Efficiency

Whisper [MICRO'22] Thermometer [ISCA'22]

Twig [MICRO'21] PDede [MICRO'21]

DMon [OSDI'21] I-SPY [MICRO'20]

Ripple [ISCA'21] Huron [PLDI'19] Cntr [ATC'18]

Heterogeneous Systems Support

Persistent Memory Indexing [FAST'21]

Optimus [ASPLOS'20]

Systems	Security
Architecture	PL

Failure Reproduction and Analysis

OmniTable [OSDI'22]

Debugging in the Brave New World [ASPLOS'22]

ER [PLDI'21] REPT [OSDI'18] Snorlax [SOSP'17]

Hippocrates [ASPLOS'21] Agamotto [OSDI'20]

Verified Distributed Systems

Sift [ATC'22] IGOR [RTAS'21] I4 [SOSP'19]

Hardware Security

MOESI-prime [ISCA'22]

Dolma [SEC'21] NDA [MICRO'19]

Foreshadow [SEC'18] Morpheus [ASPLOS'19]