Hardware Mechanisms for Distributed Dynamic Software Analysis

Joseph L. Greathouse

Advisor: Prof. Todd Austin

May 10, 2012

Software Errors Abound

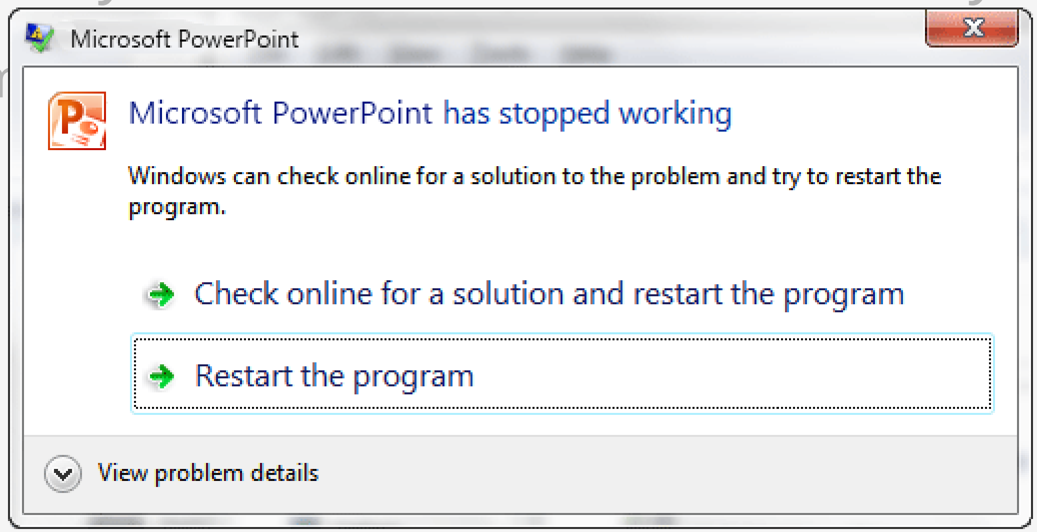
- NIST: Software errors cost U.S. ~$60 billion/year

Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >⅓ from viruses, network intrusion, etc.

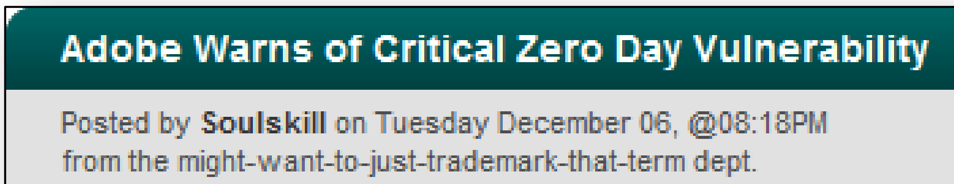
Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >$\frac{1}{3}$ from



Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >⅓ from viruses, network intrusion, etc.


Adobe Warns of Critical Zero Day Vulnerability

Posted by **Soulskill** on Tuesday December 06, @08:18PM
from the might-want-to-just-trademark-that-term dept.

Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >⅓ from viruses, network intrusion, etc.

Adobe Warns of Critical Zero Day Vulnerability

Posted by **Soulskill** on Tuesday December 06, @08:18PM
from the might-want-to-just-trademark-that-term dept.

Global Spam Drops by a Third After Rustock Botnet Gets Crushed, Symantec Says

By SecurityWeek News on March 29, 2011

Software Errors Abound

- NIST: Software errors cost U.S. ~$60 billion/year
- FBI: Security Issues cost U.S. $67 billion/year
  - >⅓ from viruses, network intrusion, etc.

Adobe Warns of Critical Zero Day Vulnerability

Posted by **Soulskill** on Tuesday December 06, @08:18PM
from the might-want-to-just-trademark-that-term dept.

Global Spam Drops by a Third After Rustock Botnet Gets Crushed, Symantec Says

By SecurityWeek News on March 29, 2011

Stuxnet attackers used 4 Windows zero-day exploits

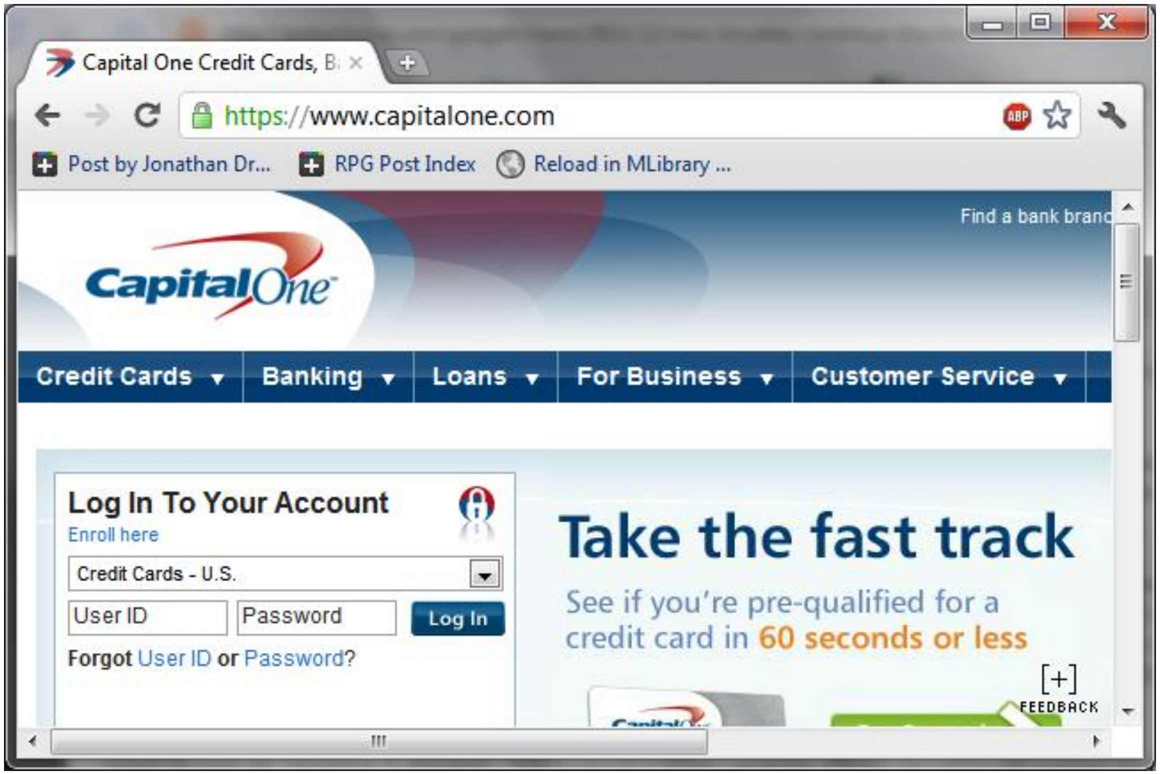
By Ryan Naraine | September 14, 2010, 11:18am PDT

Example of a Modern Bug

Nov. 2010 OpenSSL Security Flaw

Example of a Modern Bug

Nov. 2010 OpenSSL Security Flaw

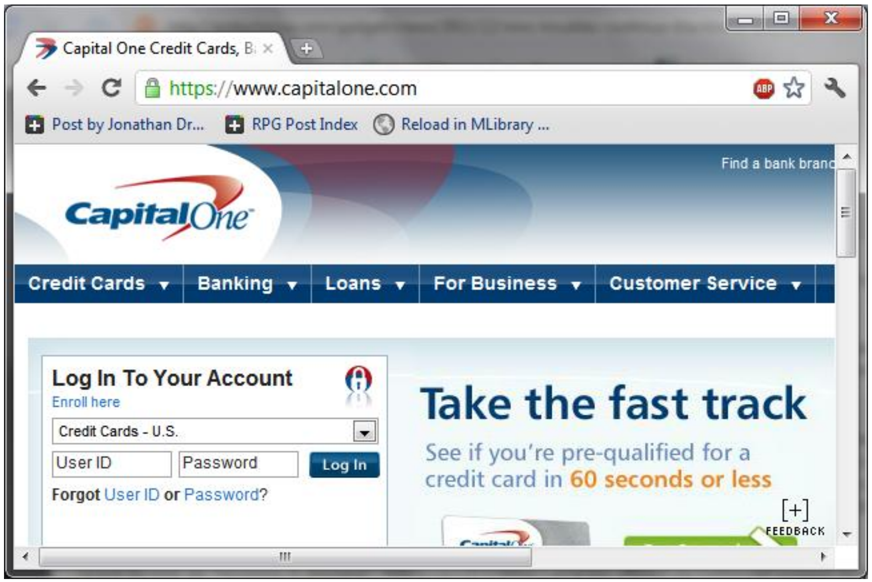


Example of a Modern Bug

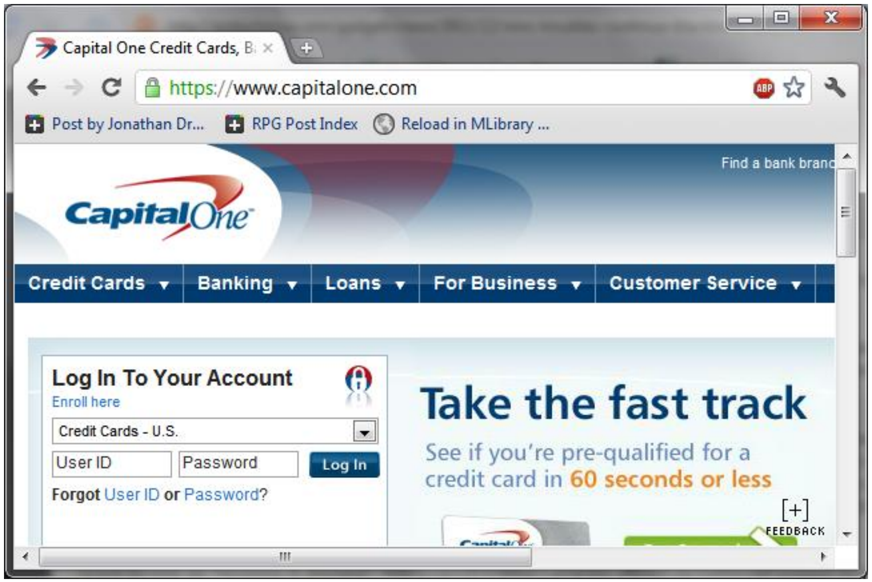
```
if(ptr == NULL) {
   len=thread_local->mylen;
   ptr=malloc(len);
   memcpy(ptr, data, len);
}
```

Example of a Modern Bug

Thread 1
mylen=small



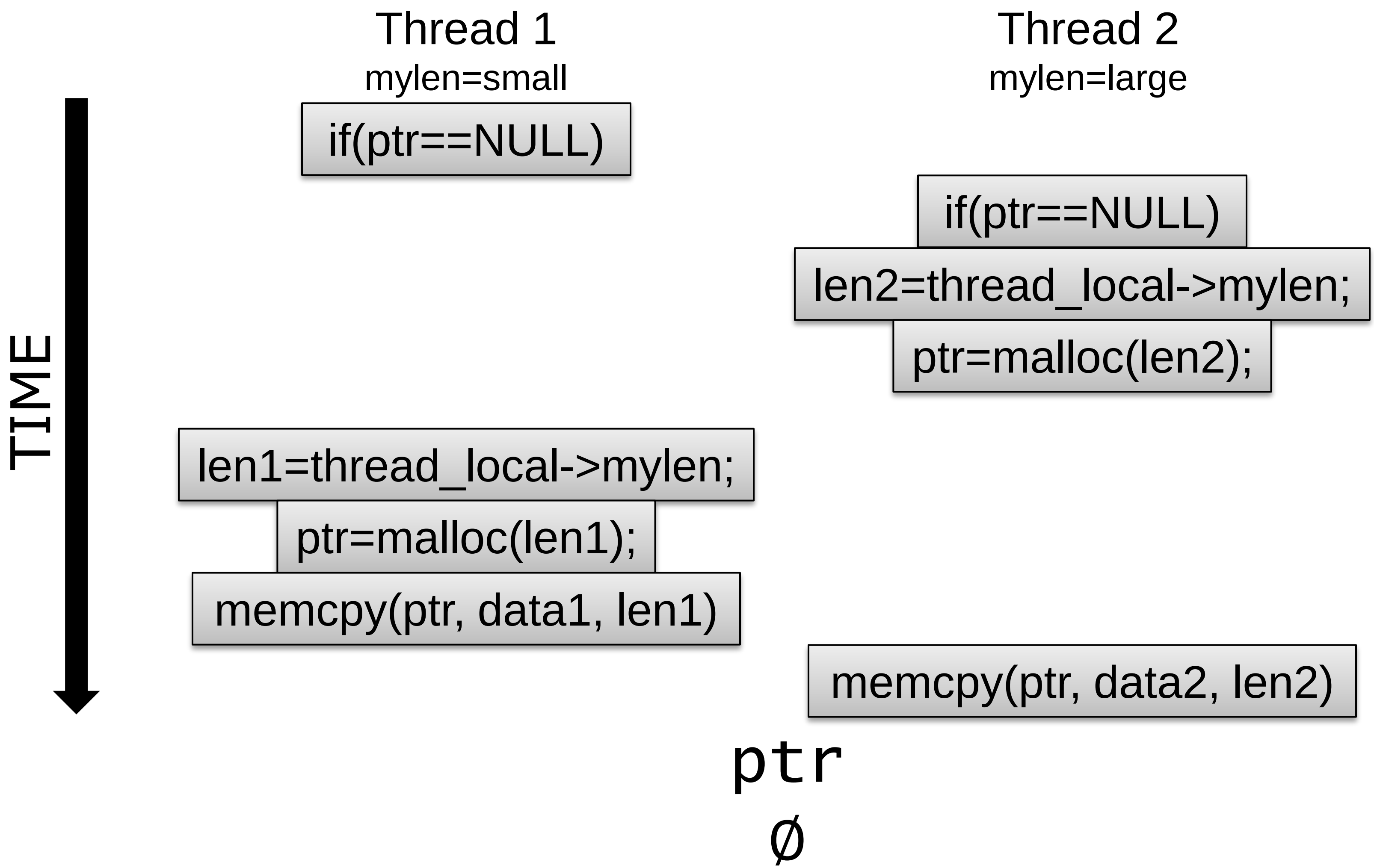
Thread 2
mylen=large



`ptr`

∅

Example of a Modern Bug



Example of a Modern Bug

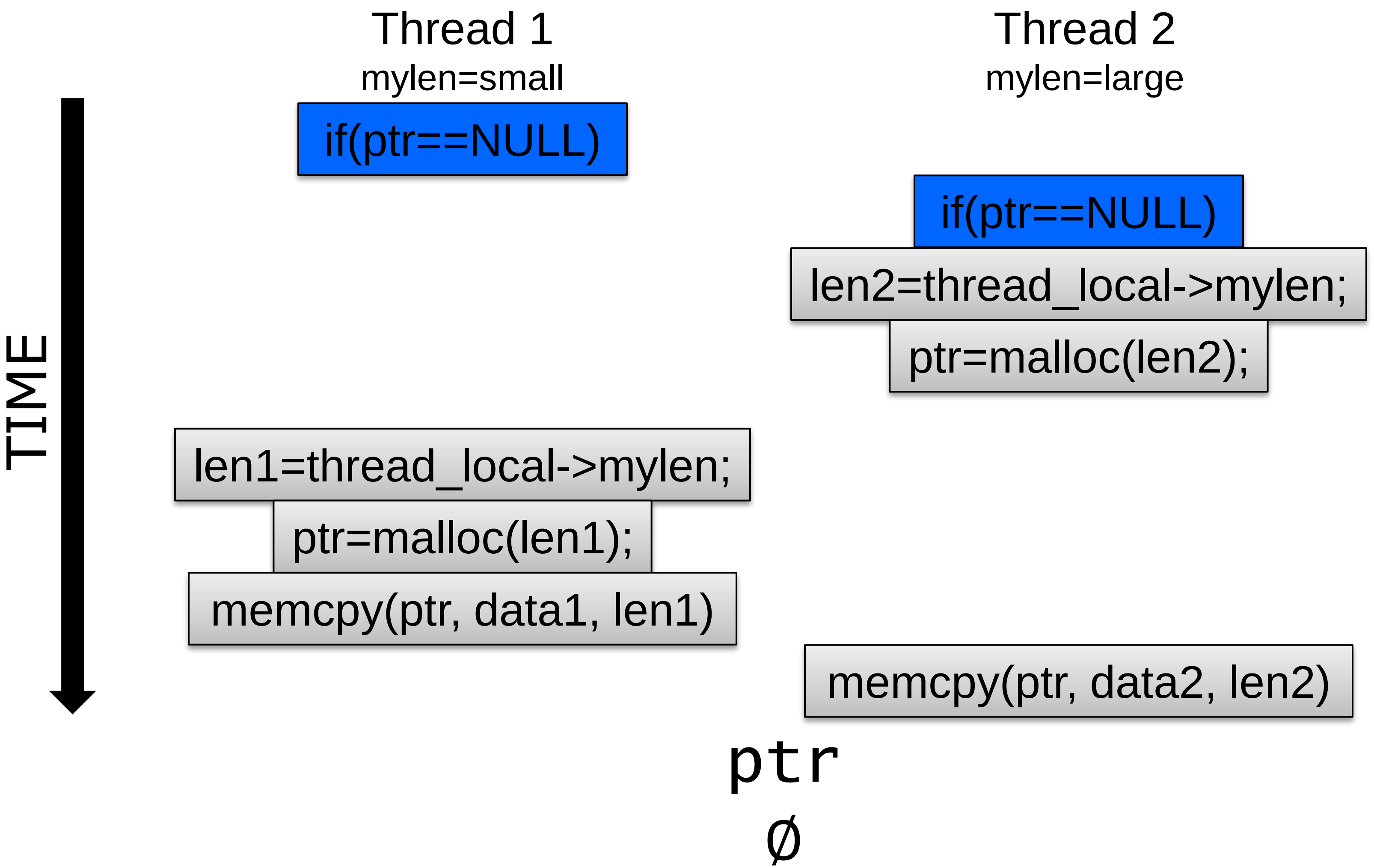
TIME

Thread 1 (mylen=small)	Thread 2 (mylen=large)
if(ptr==NULL)	
	if(ptr==NULL)
	len2=thread_local->mylen;
	ptr=malloc(len2);
len1=thread_local->mylen;	
ptr=malloc(len1);	
memcpy(ptr, data1, len1)	
	memcpy(ptr, data2, len2)

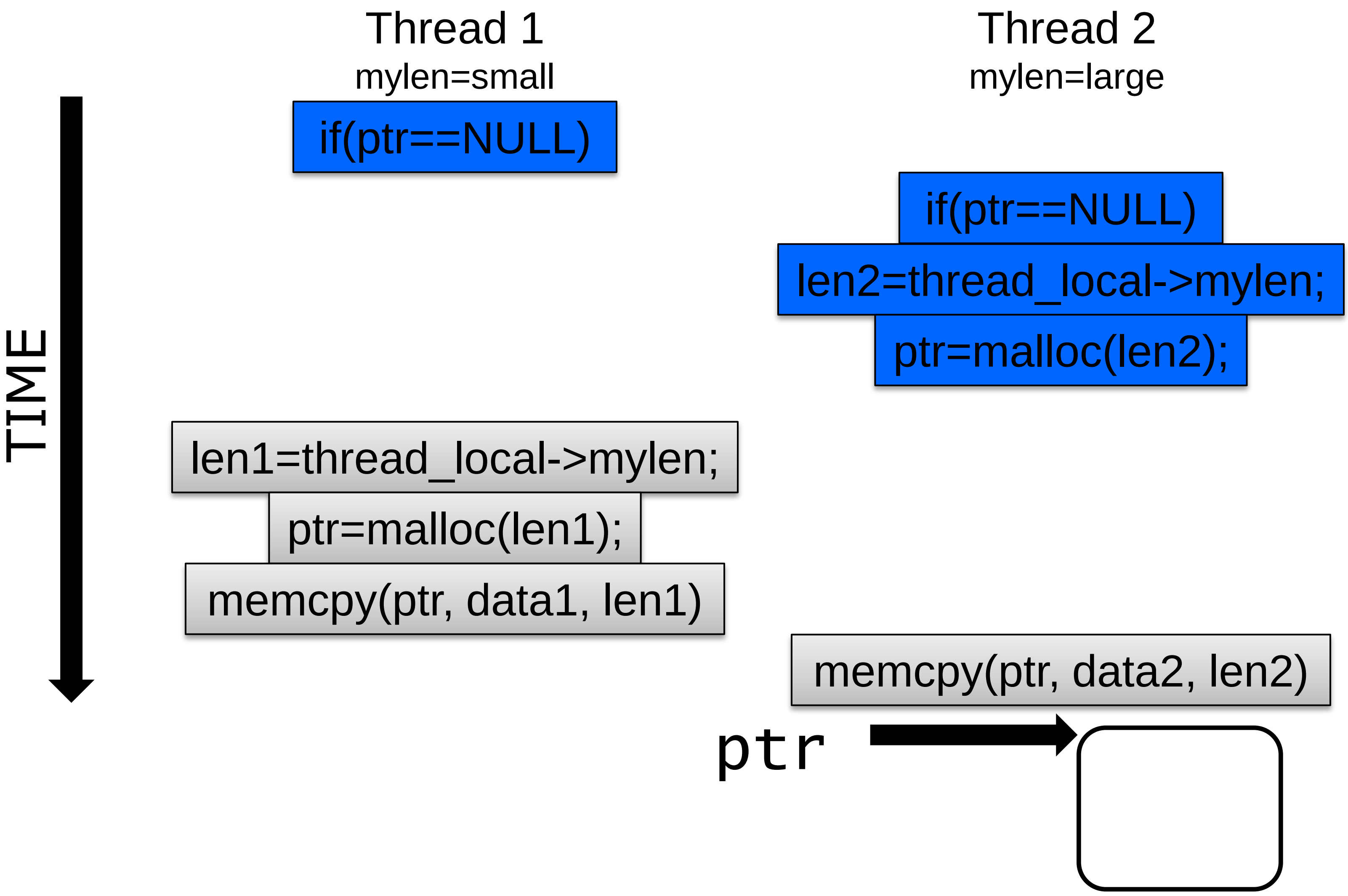
ptr

∅

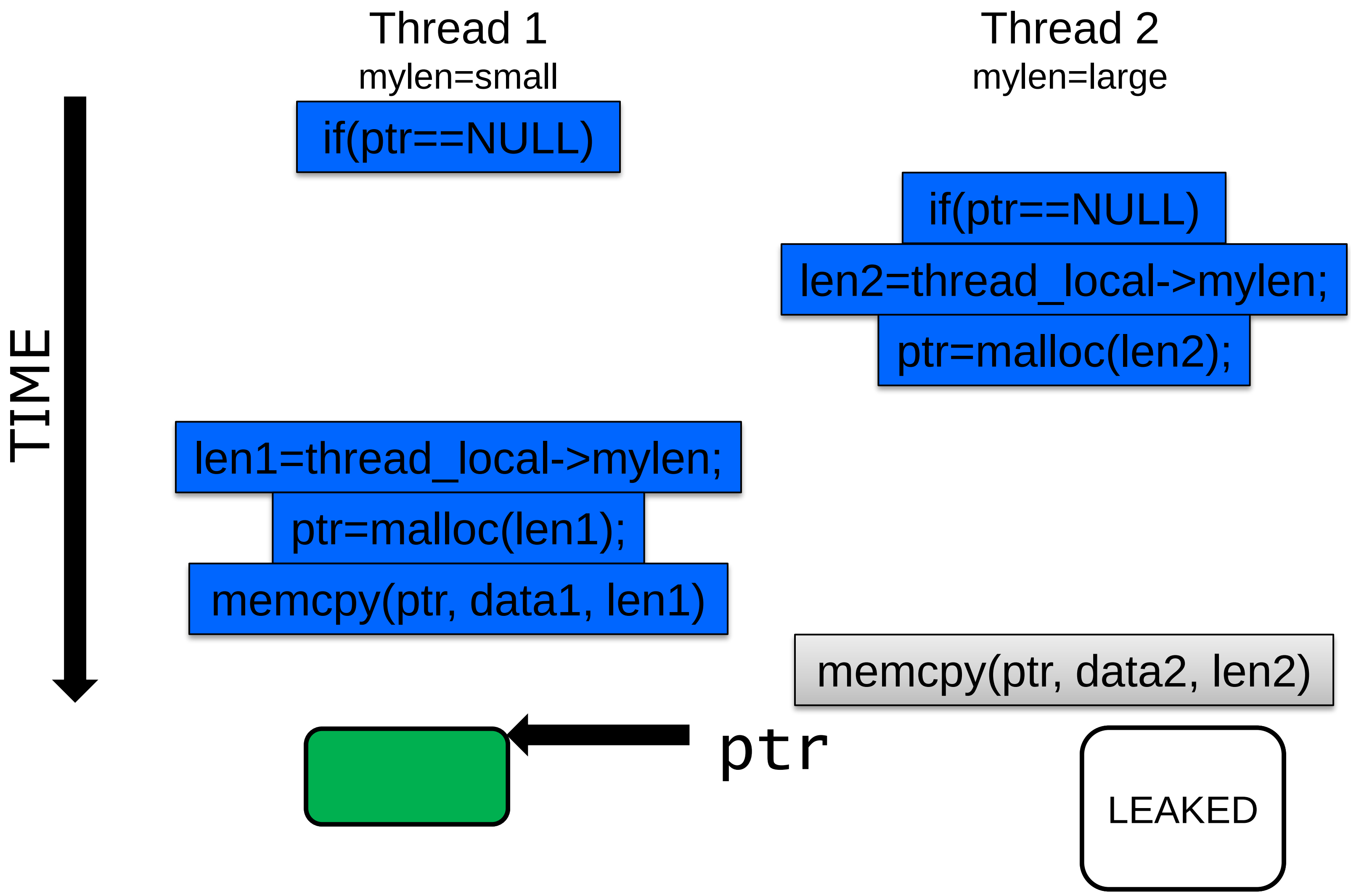
Example of a Modern Bug



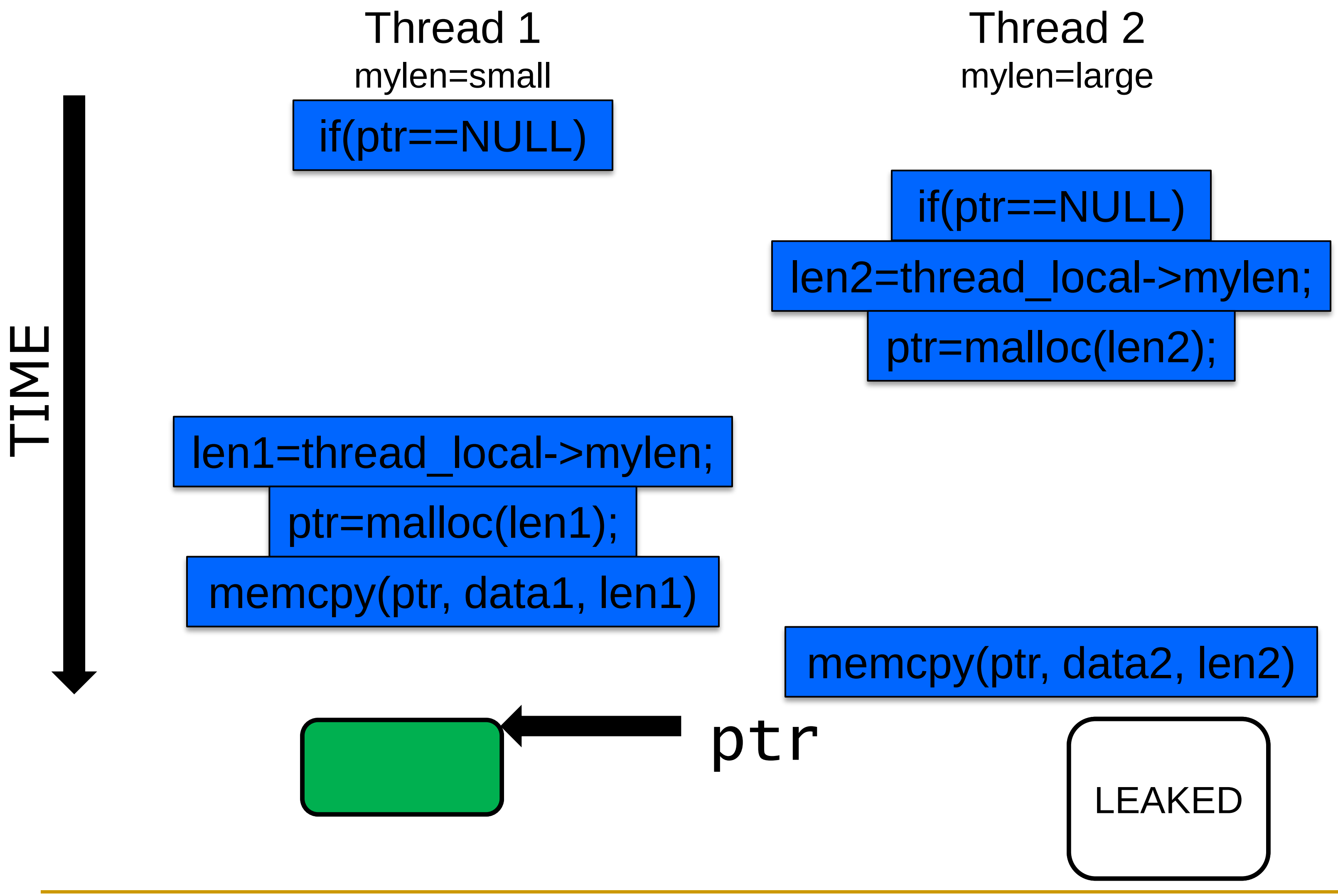
Example of a Modern Bug



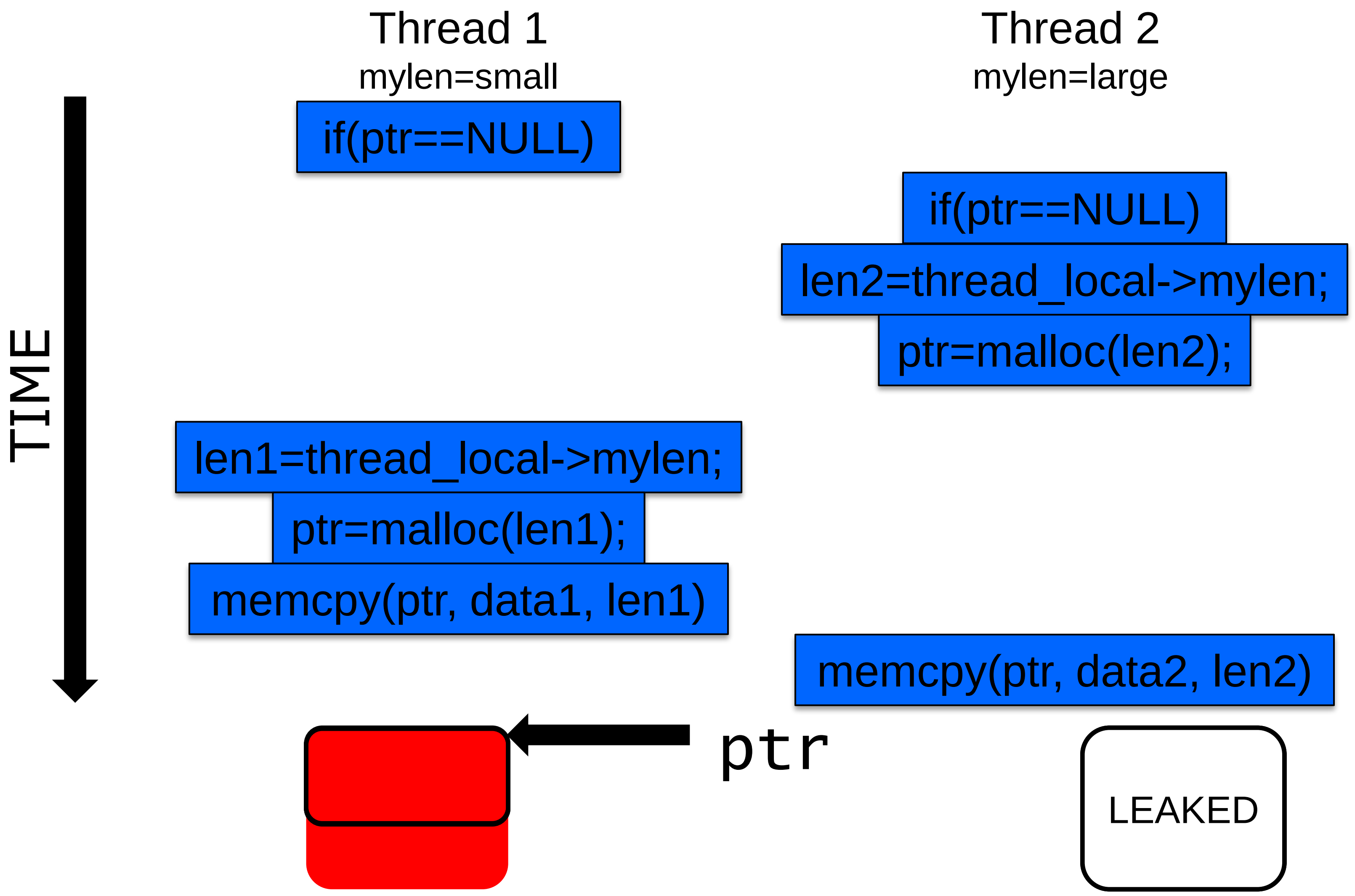
Example of a Modern Bug



Example of a Modern Bug

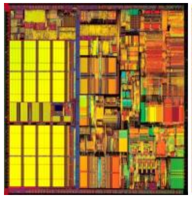


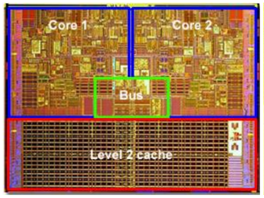
Example of a Modern Bug

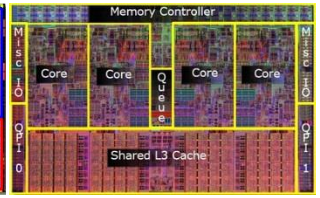


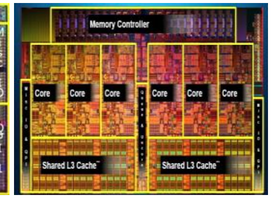
Hardware Plays a Role in this Problem

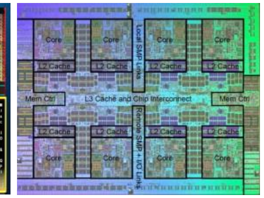
Hardware Plays a Role in this Problem

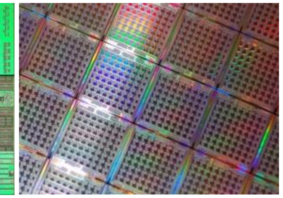


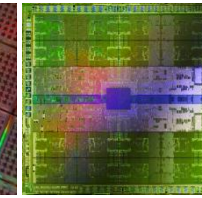




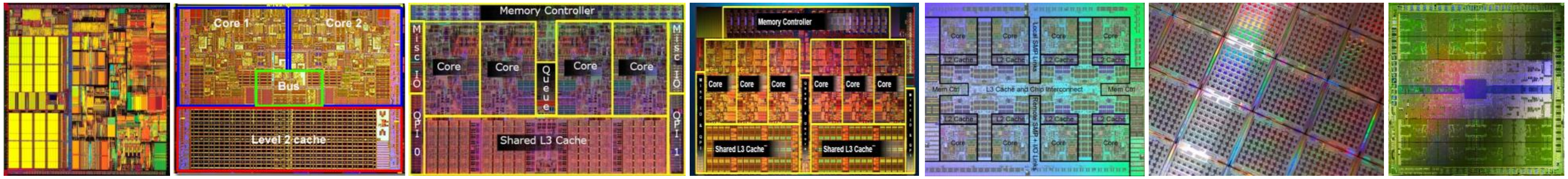






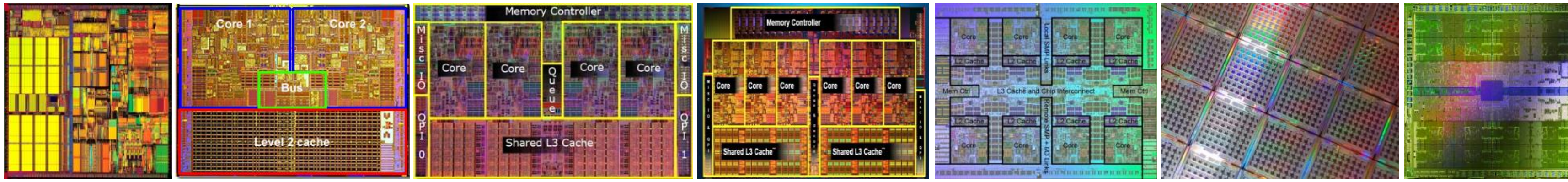


Hardware Plays a Role in this Problem



In spite of proposed hardware solutions

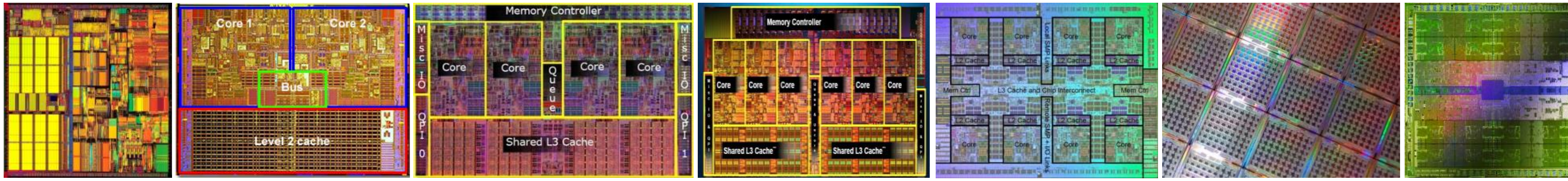
Hardware Plays a Role in this Problem



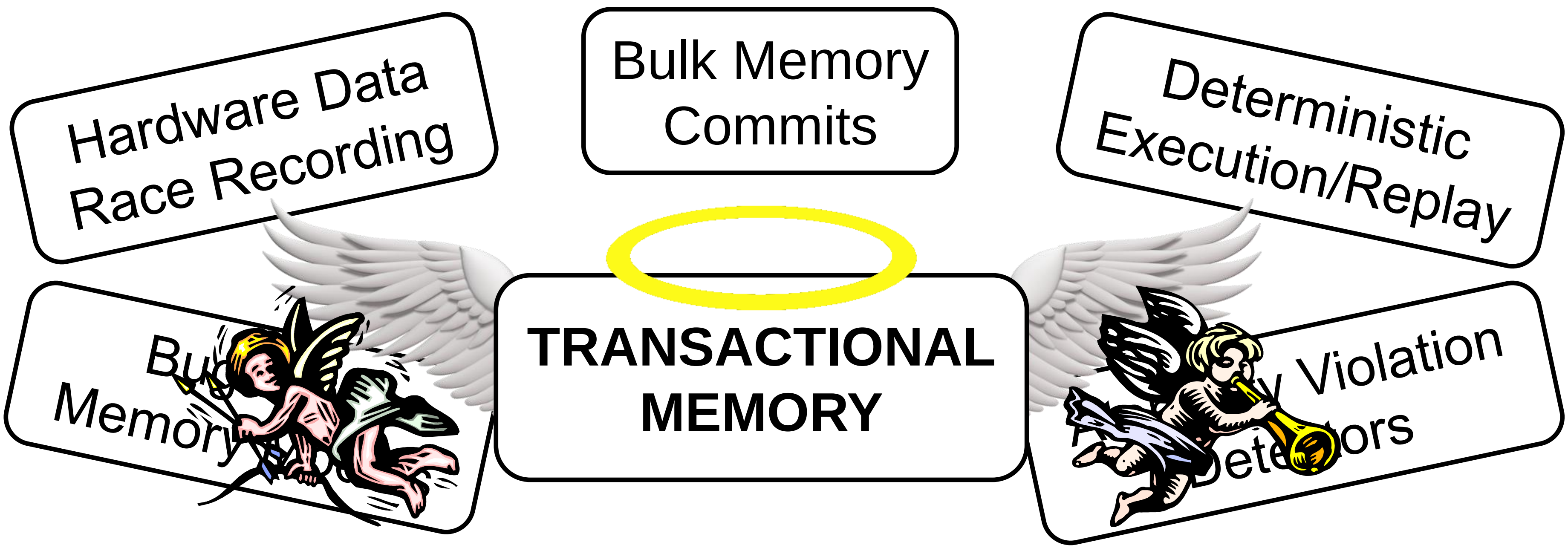
In spite of proposed hardware solutions

- Hardware Data Race Recording
- Bulk Memory Commits
- Deterministic Execution/Replay
- Bug-Free Memory Models
- Atomicity Violation Detectors

Hardware Plays a Role in this Problem



In spite of proposed hardware solutions



Dynamic Software Analyses

- Analyze the program as it runs
  - + Find errors on any executed path

Dynamic Software Analyses

- Analyze the program as it runs
  - + Find errors on any executed path

- Data Race Detection (e.g. Inspector XE)
- Taint Analysis
- Memory Checking (e.g. MemCheck)
- Dynamic Bounds Checking

Dynamic Software Analyses

- Analyze the program as it runs
  - + Find errors on any executed path
  - – LARGE overheads, only test one path at a time
- Data Race Detection (e.g. Inspector XE)
- Taint Analysis
- Memory Checking (e.g. MemCheck)
- Dynamic Bounds Checking

Dynamic Software Analyses

- Analyze the program as it runs
  - \+ Find errors on any executed path
  - – LARGE overheads, only test one path at a time

- Data Race Detection (e.g. Inspector XE)

  2-300x

- Taint Analysis

  2-200x

- Memory Checking (e.g. MemCheck)

  5-50x

- Dynamic Bounds Checking

  2-80x

Goals of this Thesis

- Allow high quality dynamic software analyses
  - Find **difficult bugs** that weaker analyses miss
- **Distribute the tests** to large populations
  - Must be low overhead or users will get angry
- **Sampling + Hardware** to accomplished this
  - Each user only tests a small part of the program
  - Each test should be helped by hardware

Meeting These Goals - Thesis Overview

Meeting These Goals - Thesis Overview

Allow high quality dynamic software analyses

Meeting These Goals - Thesis Overview

Dataflow
Analysis

Allow high quality dynamic
software analyses

Data Race
Detection

Meeting These Goals - Thesis Overview

Dataflow Analysis

Data Race Detection

Allow high quality dynamic software analyses

Meeting These Goals - Thesis Overview

	Software Support	Hardware Support
Dataflow Analysis		
Data Race Detection		

Meeting These Goals - Thesis Overview

	Software Support	Hardware Support
Dataflow Analysis		
Data Race Detection		

Meeting These Goals - Thesis Overview

	Software Support	Hardware Support
Dataflow Analysis		
Data Race Detection		

Distribute the tests +
Sample the analyses

Meeting These Goals - Thesis Overview

Software Support

Hardware Support

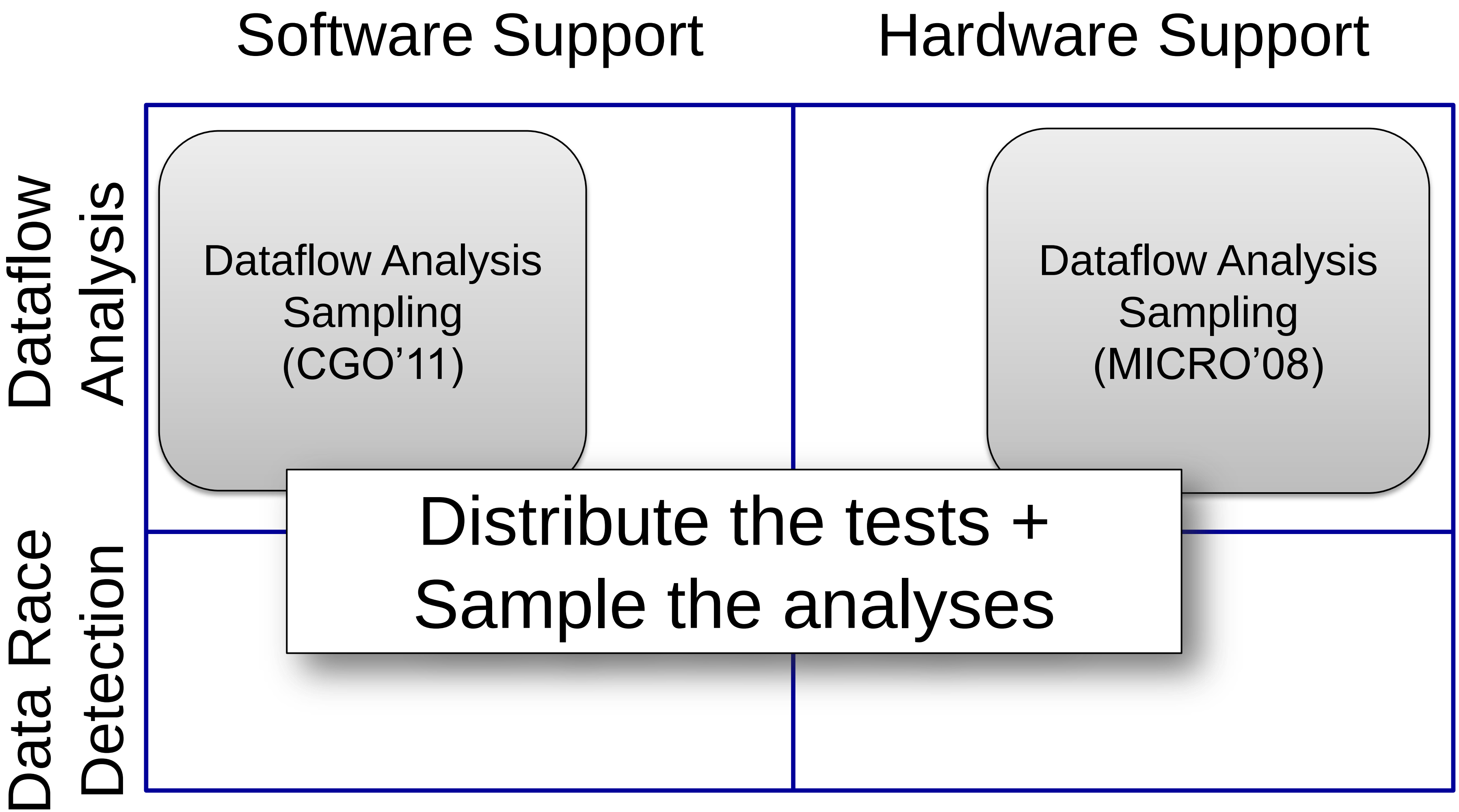
Dataflow Analysis

Dataflow Analysis Sampling (CGO'11)

Data Race Detection

Distribute the tests +
Sample the analyses

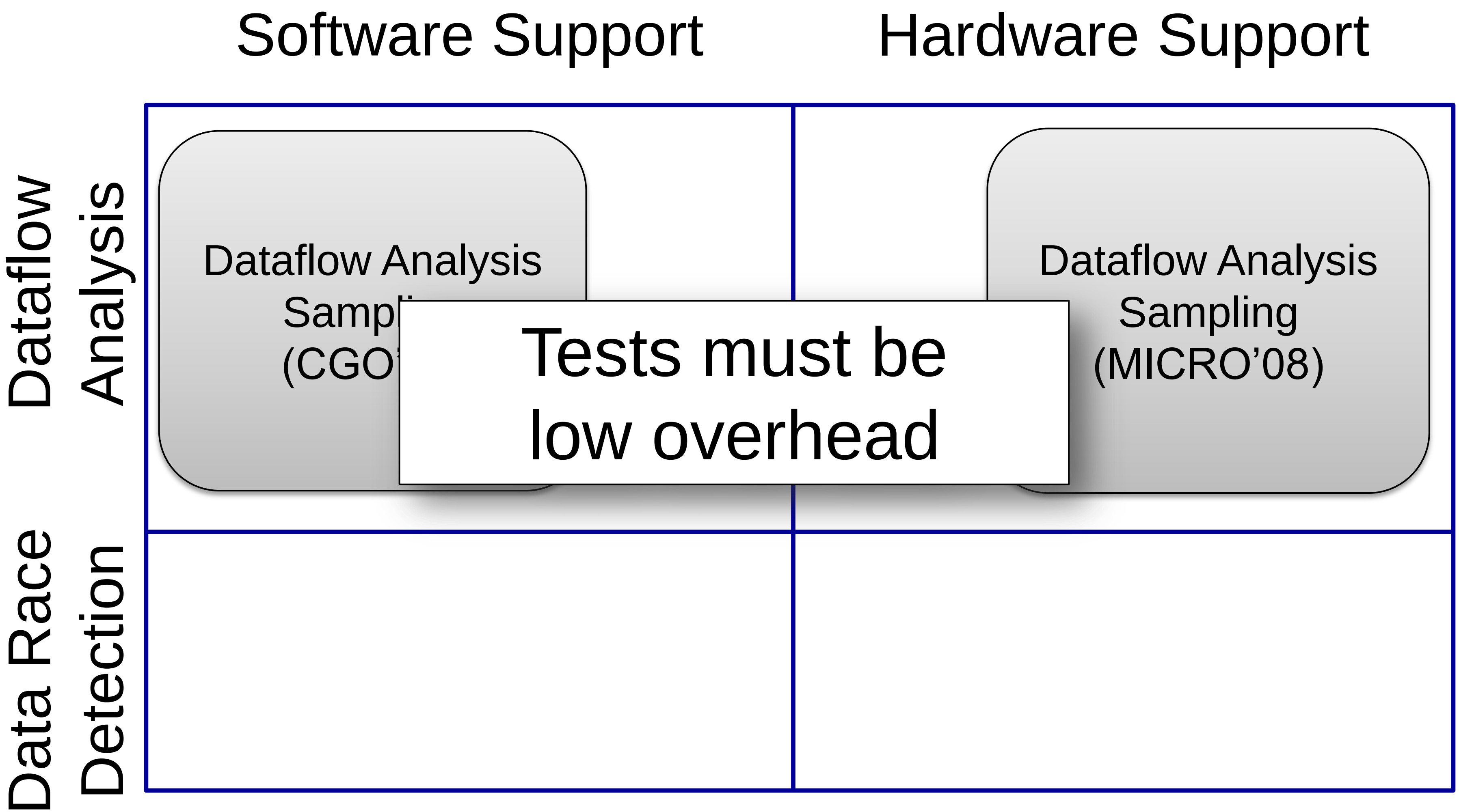
Meeting These Goals - Thesis Overview



Meeting These Goals - Thesis Overview

	Software Support	Hardware Support
Dataflow Analysis	Dataflow Analysis Sampling (CGO’11)	Dataflow Analysis Sampling (MICRO’08)
Data Race Detection		

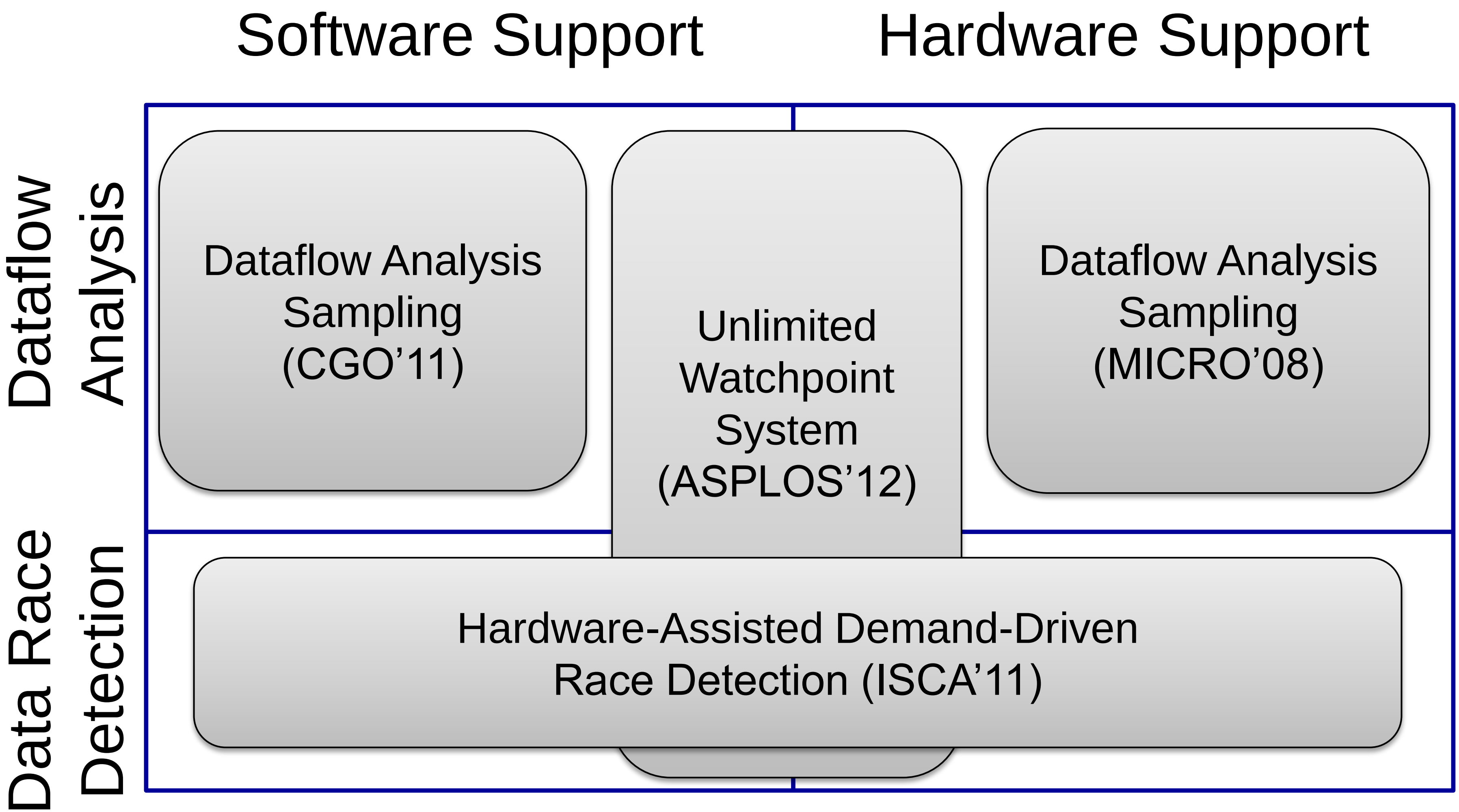
Meeting These Goals - Thesis Overview



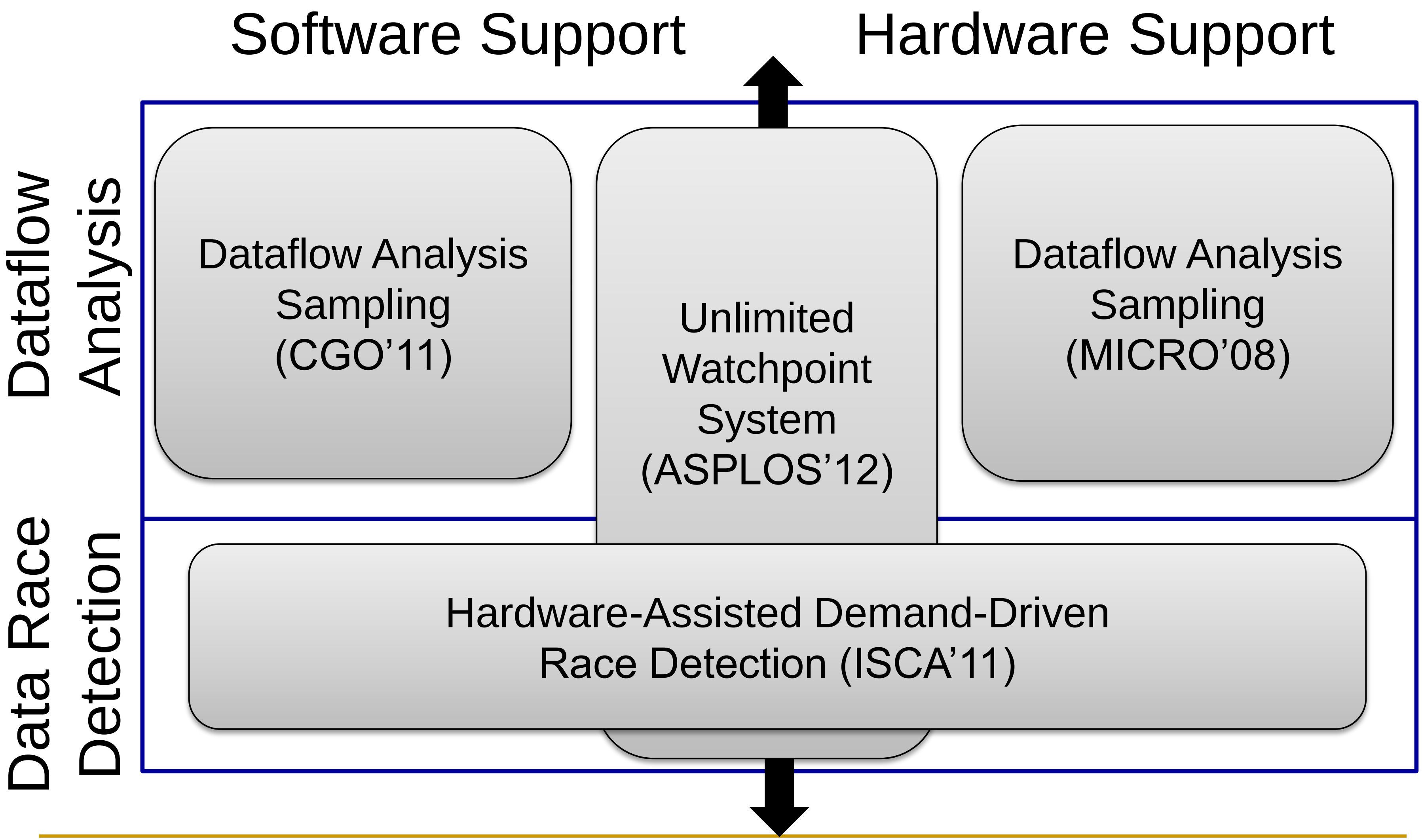
Meeting These Goals - Thesis Overview

	Software Support	Hardware Support
Dataflow Analysis	Dataflow Analysis Sampling (CGO'11)	Dataflow Analysis Sampling (MICRO'08)
Data Race Detection	Hardware-Assisted Demand-Driven Race Detection (ISCA'11)	

Meeting These Goals - Thesis Overview



Meeting These Goals - Thesis Overview



Outline

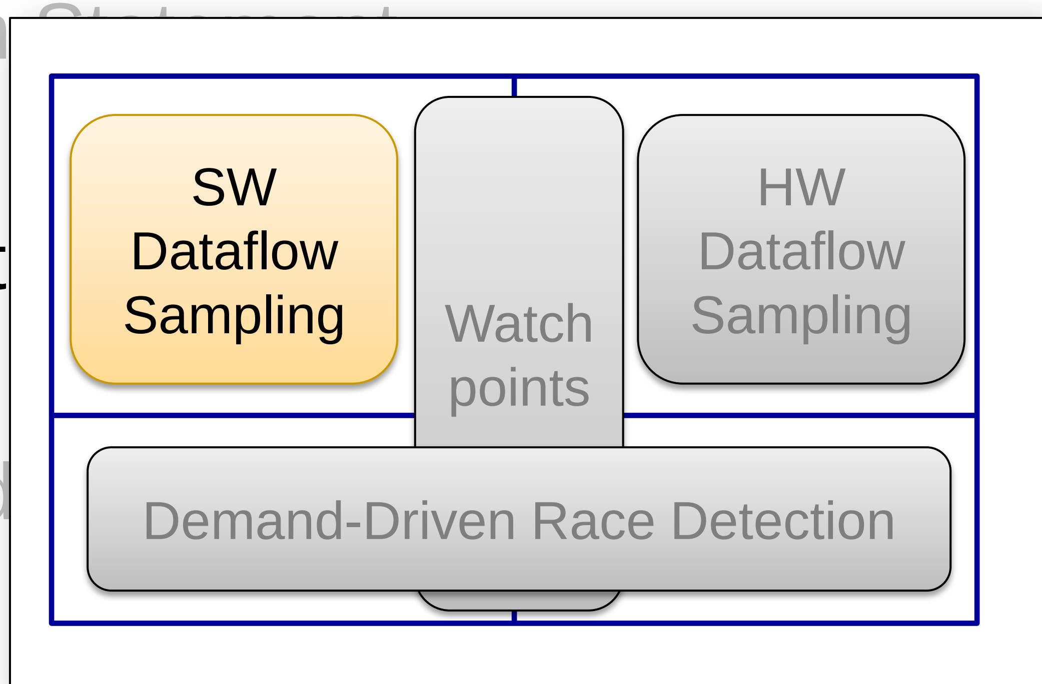
- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints

Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints

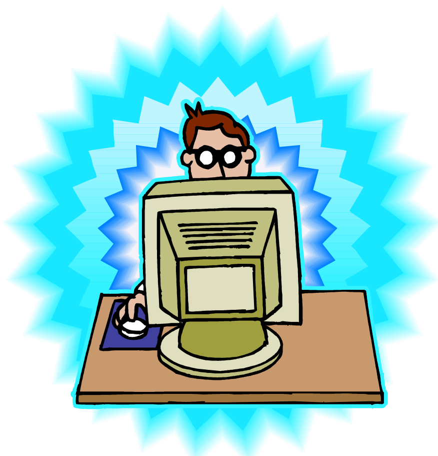
Outline

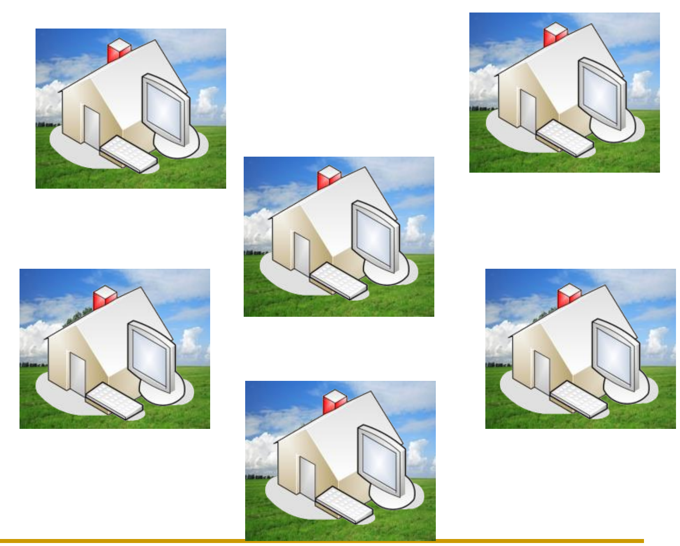
- Problem
- Distribut s
- Demand
- Unlimited Watchpoints



Distributed Dynamic Dataflow Analysis

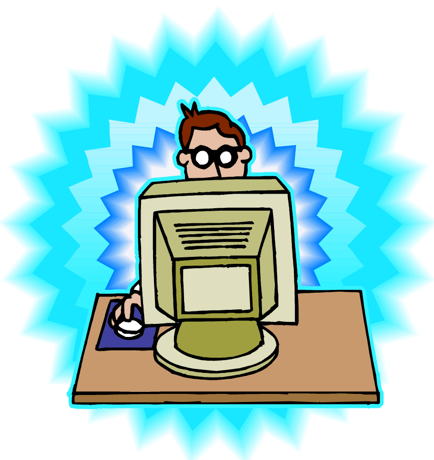
- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test

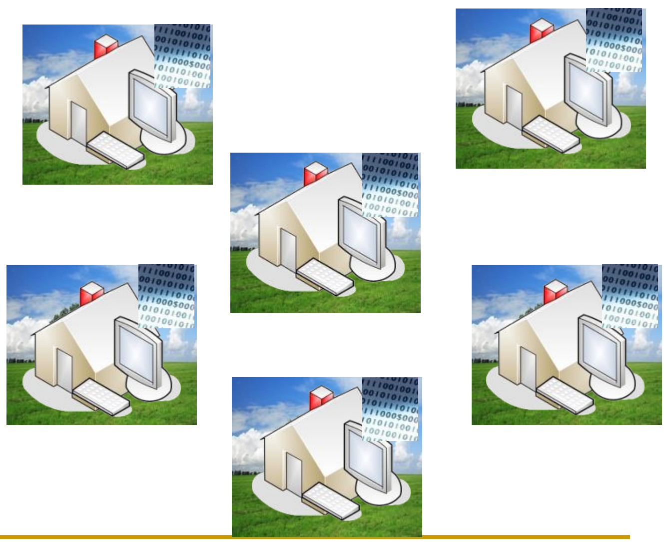




Distributed Dynamic Dataflow Analysis

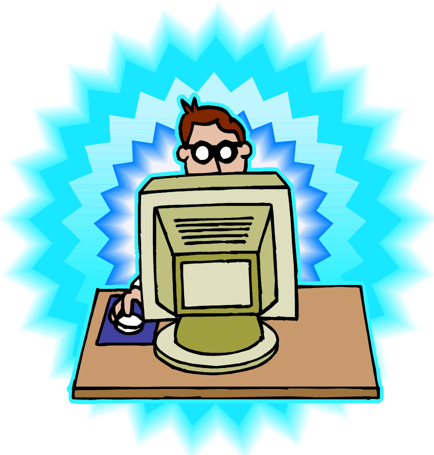
- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test

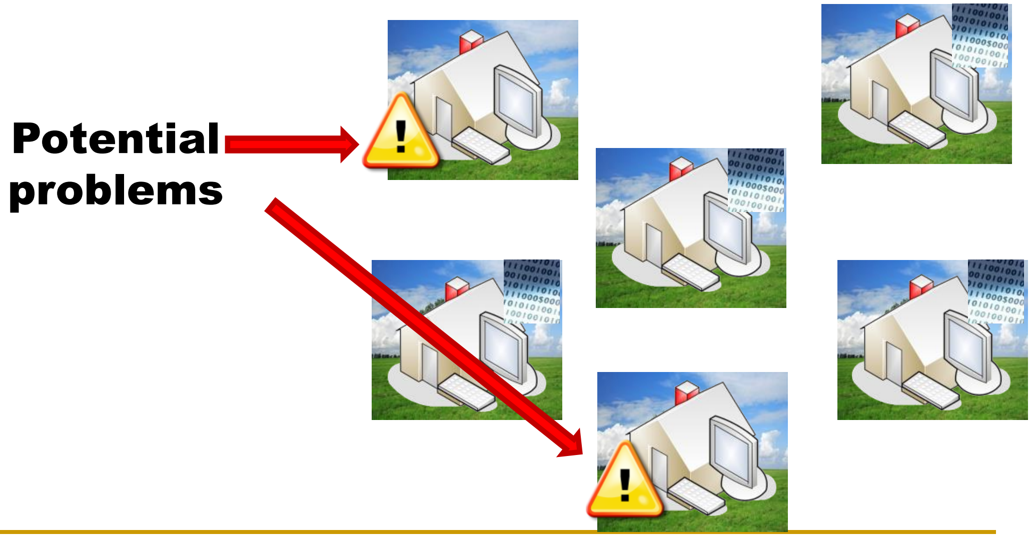




Distributed Dynamic Dataflow Analysis

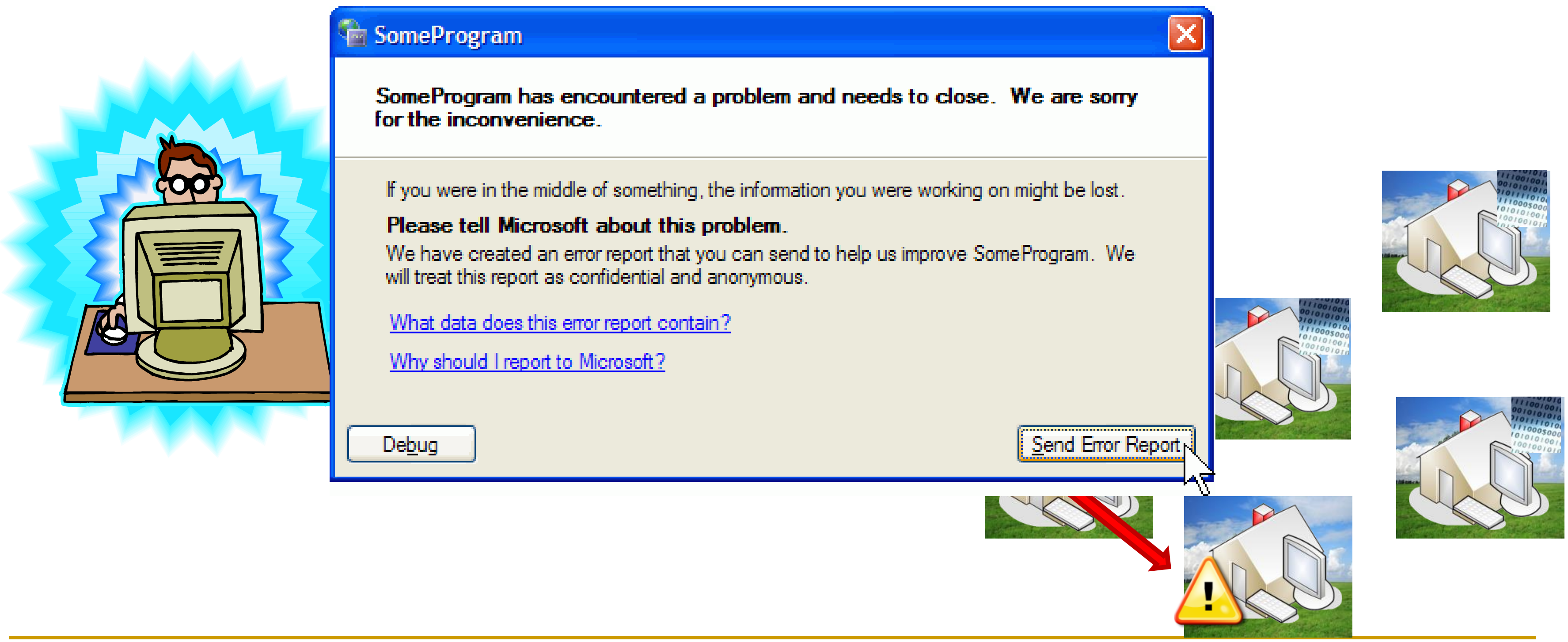
- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test





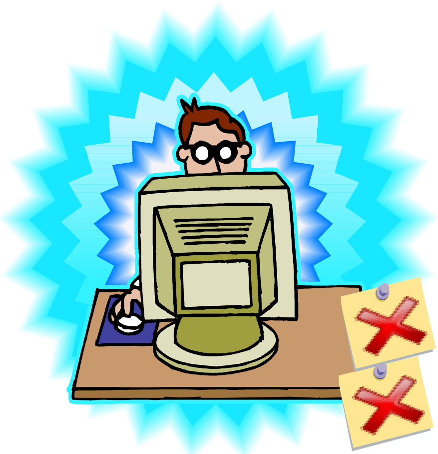
Distributed Dynamic Dataflow Analysis

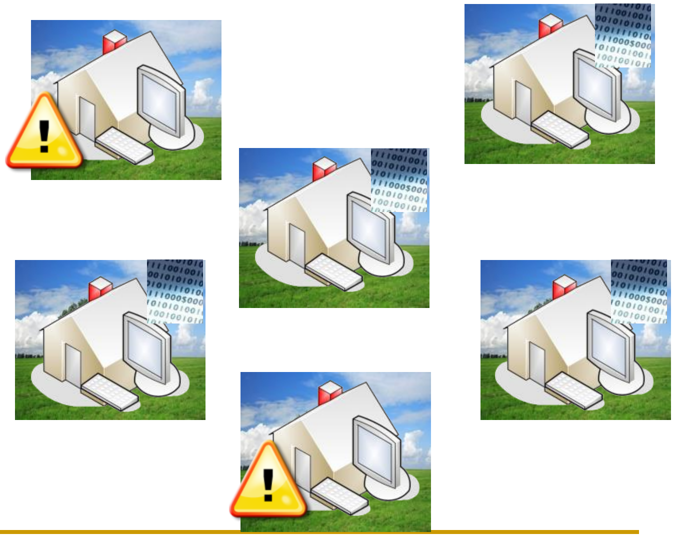
- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test



Distributed Dynamic Dataflow Analysis

- Split analysis across large populations
  - Observe more runtime states
  - Report problems developer never thought to test





The Problem: OVERHEADS

- Analyze the program as it runs
  - \+ System state, find errors on any executed path
  - – LARGE runtime overheads, only test one path
- Data Race Detection (e.g. Thread Analyzer)

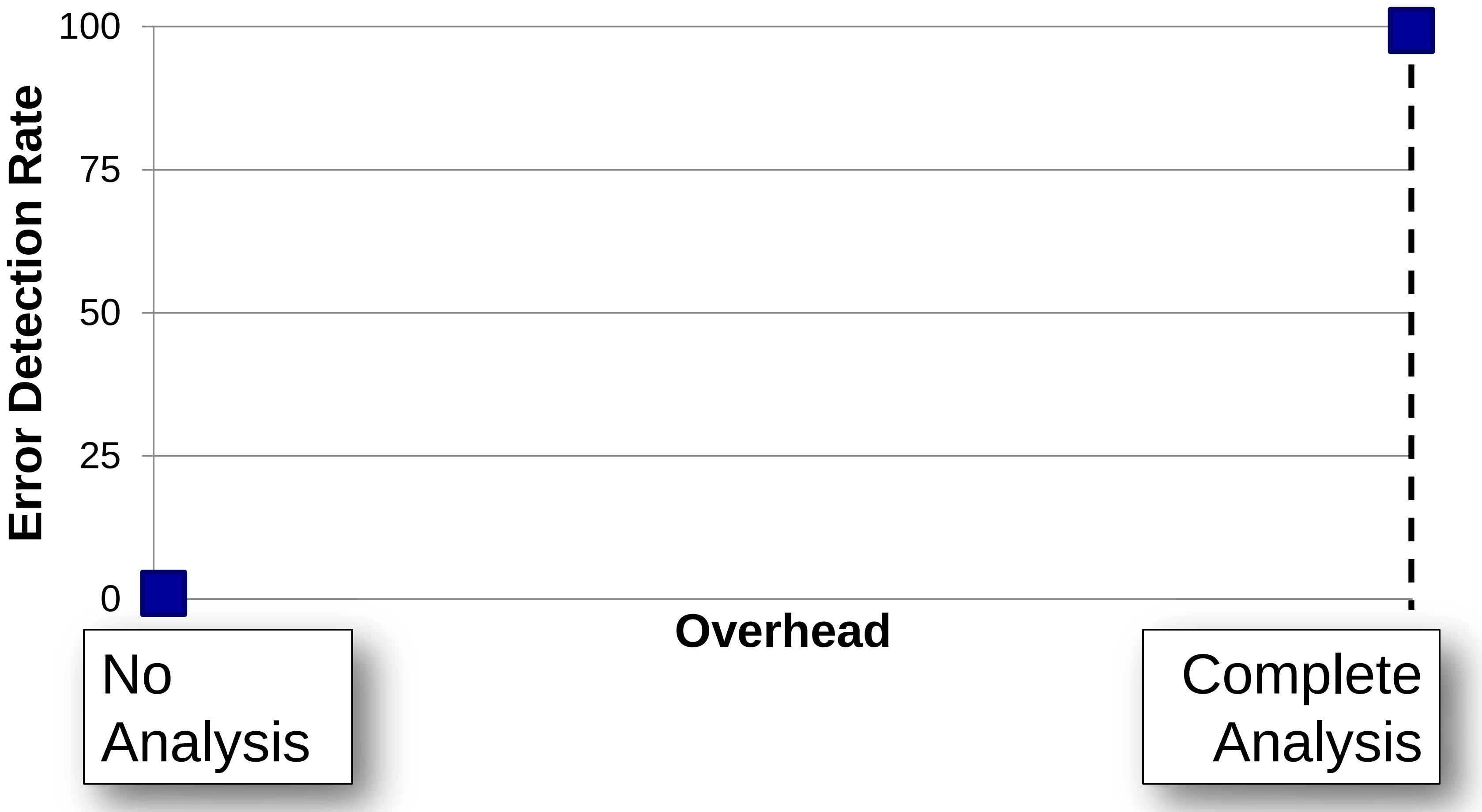
  2-300x
- Taint Analysis (e.g.TaintCheck)

  2-200x
- Memory Checking (e.g. MemCheck)

  5-50x
- Dynamic Bounds Checking

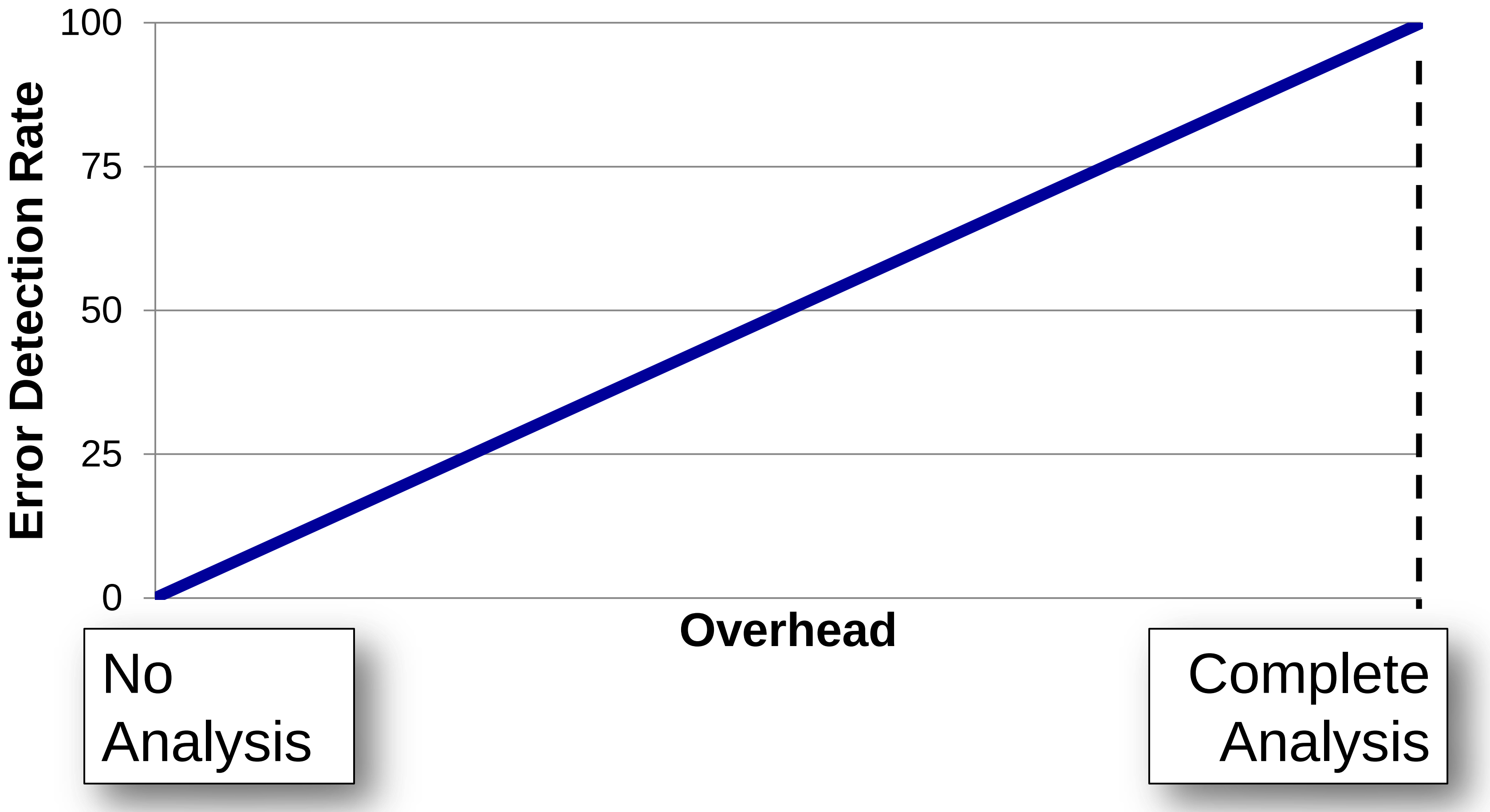
  2-80x

Current Options Limited



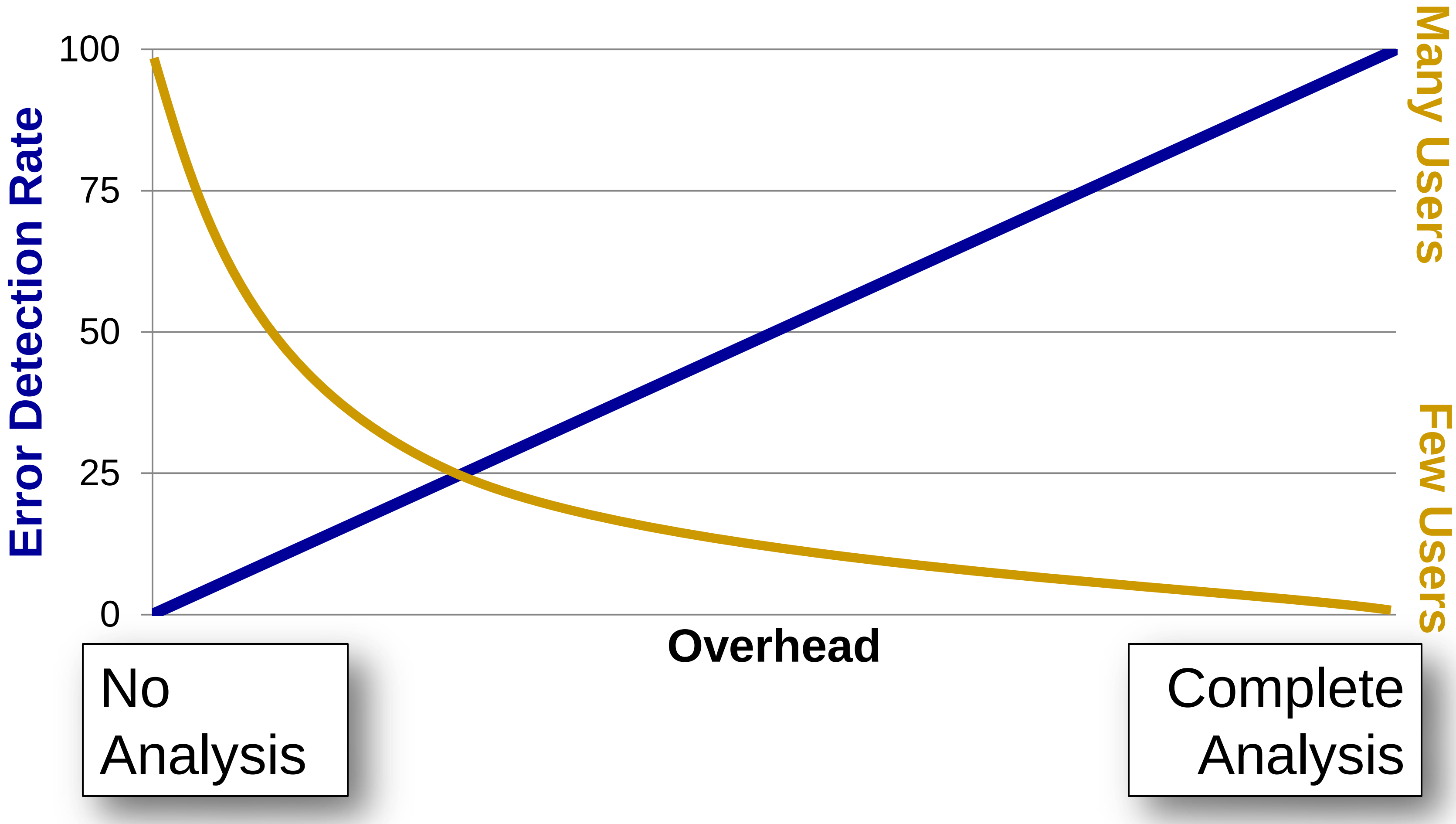
Solution: Sampling

- Lower overheads by skipping some analyses



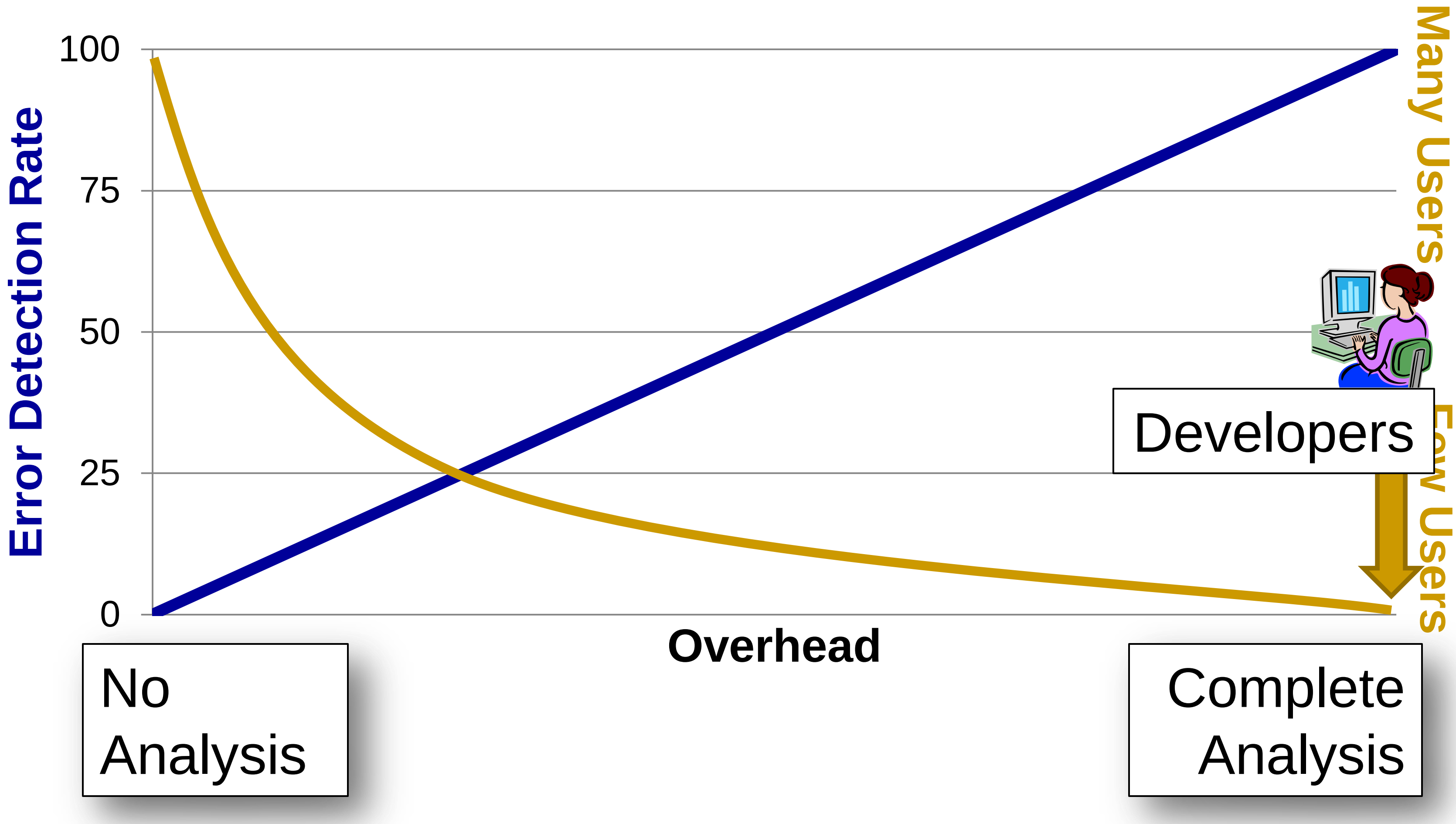
Sampling Allows Distribution

- Lower overheads mean more users



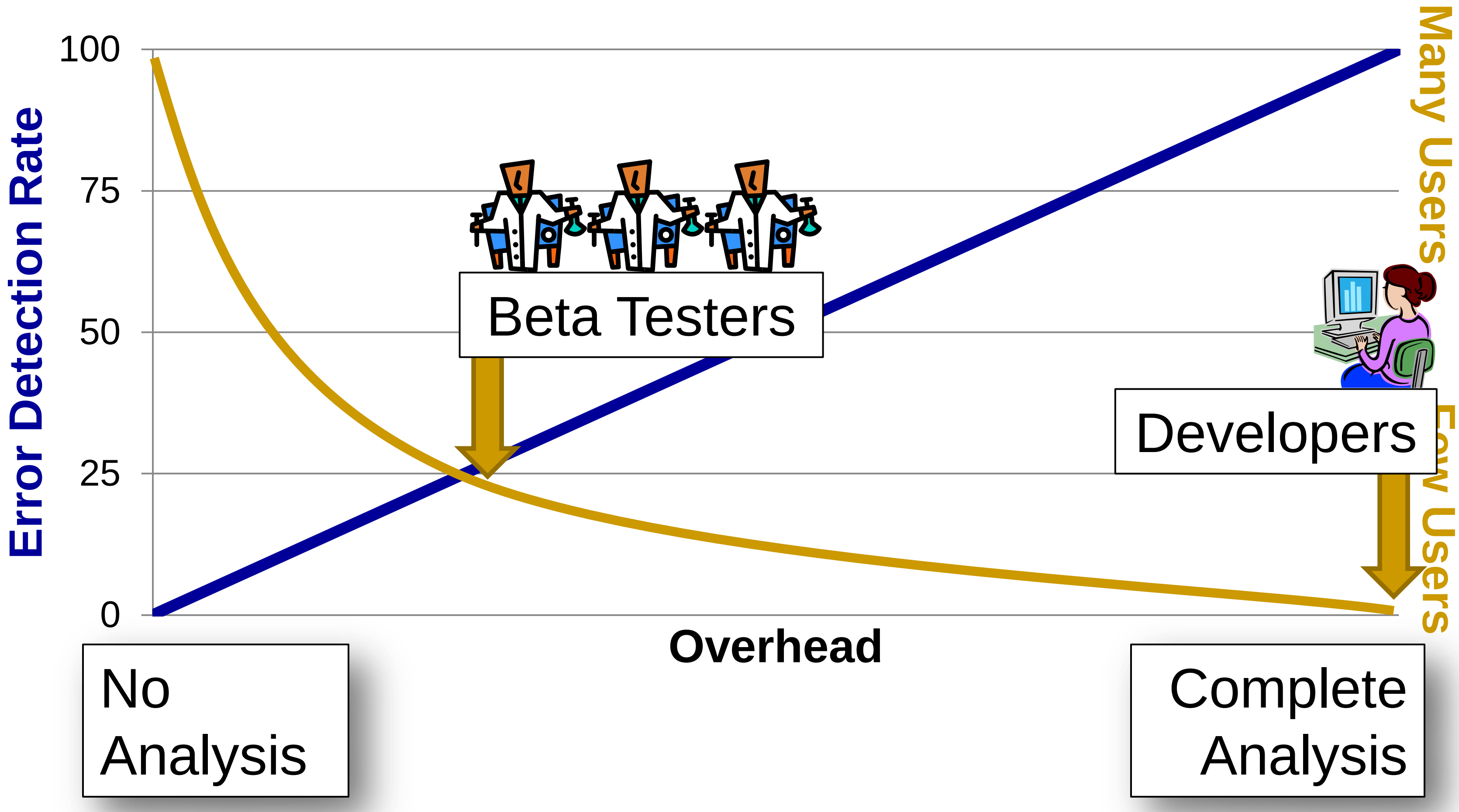
Sampling Allows Distribution

- Lower overheads mean more users



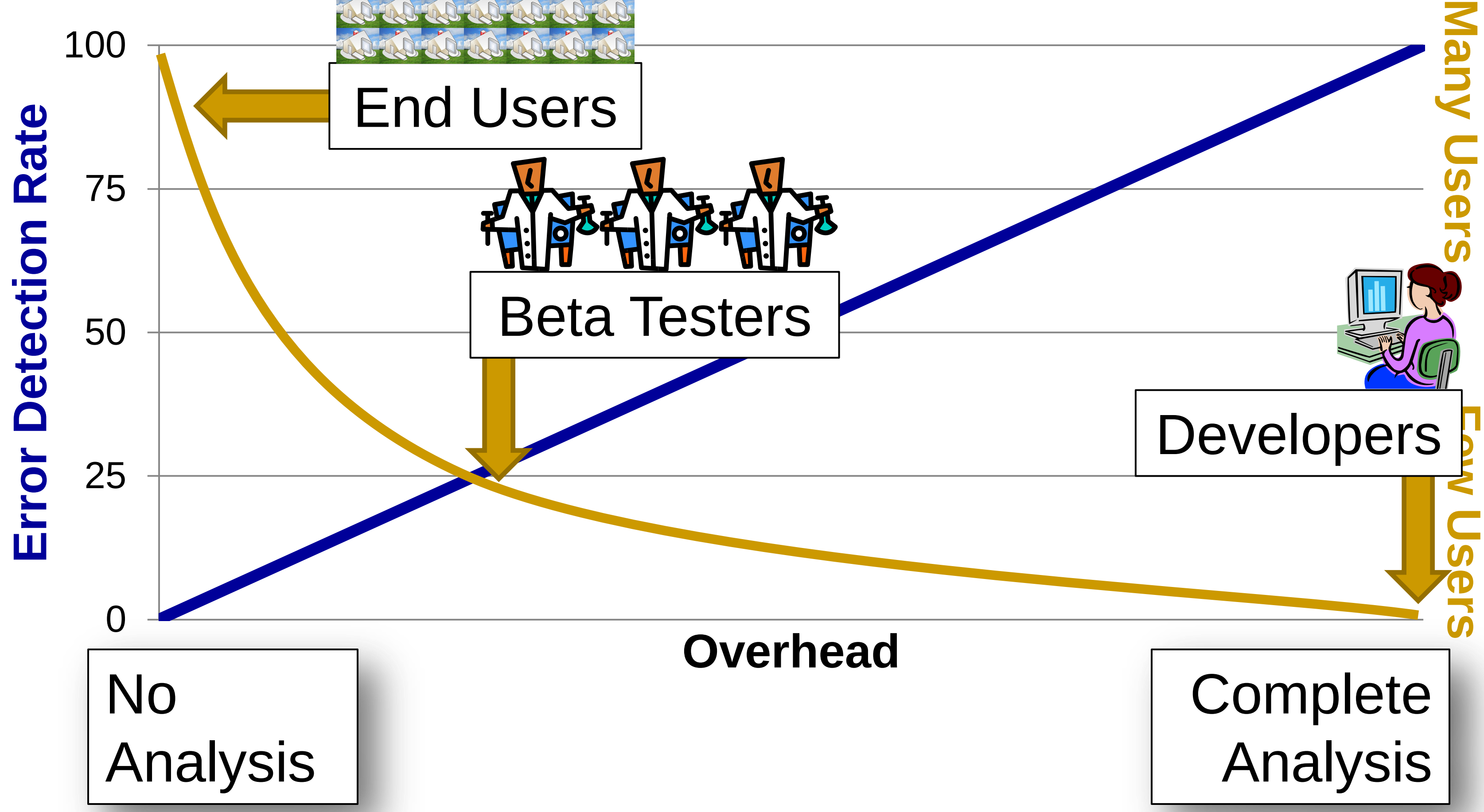
Sampling Allows Distribution

- Lower overheads mean more users



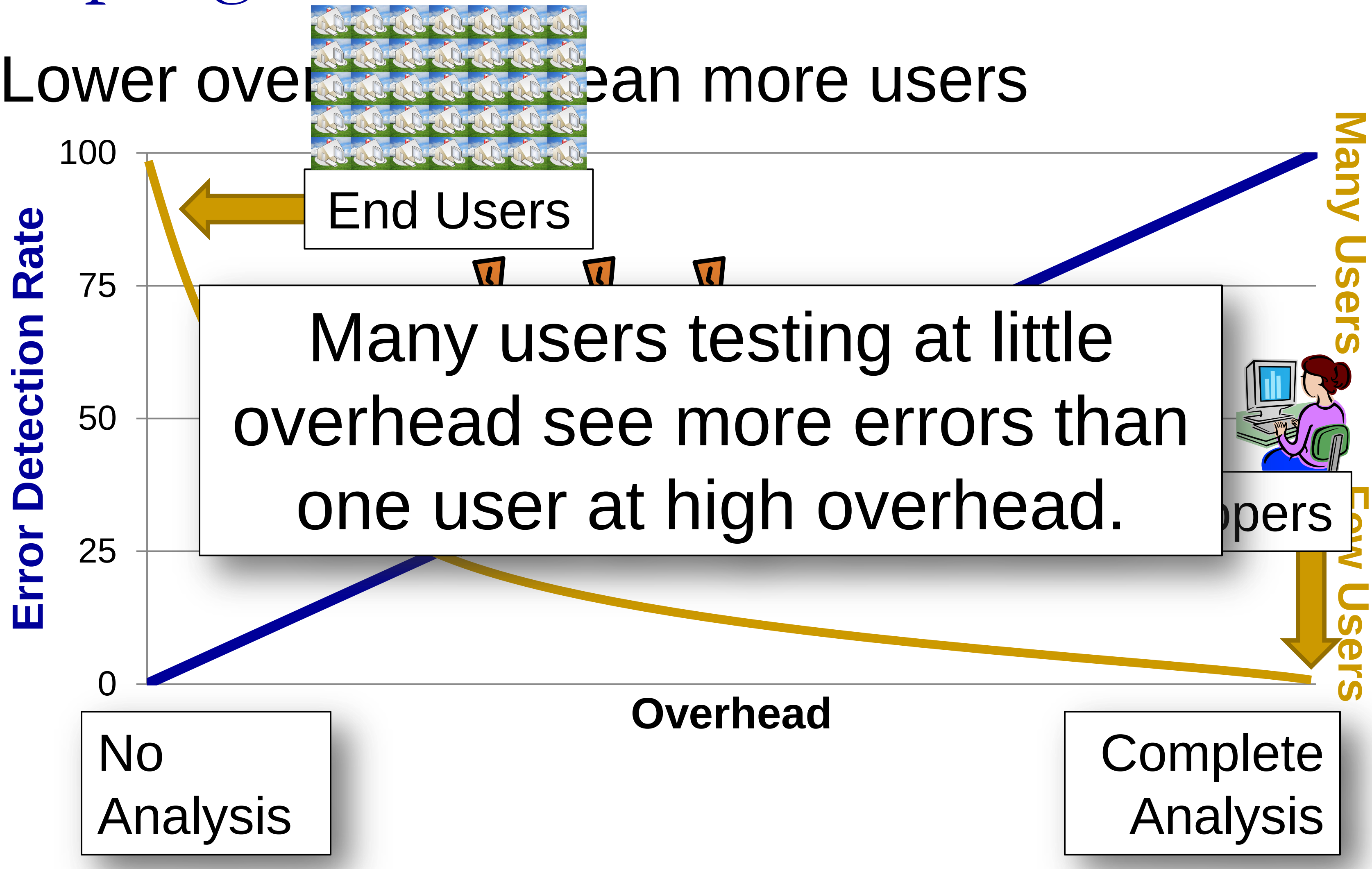
Sampling Allows Distribution

- Lower ove… ean more users

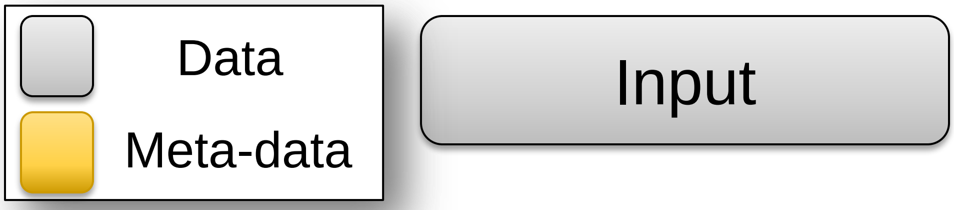


Sampling Allows Distribution

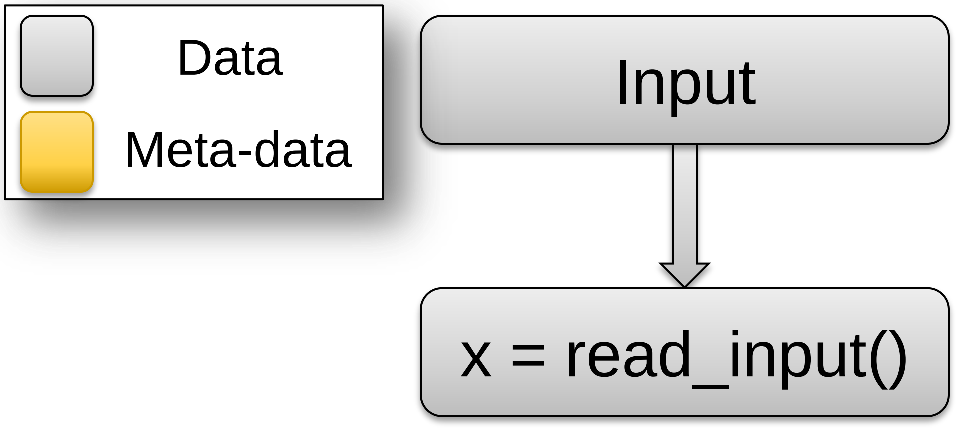
- Lower ove[illegible]ean more users



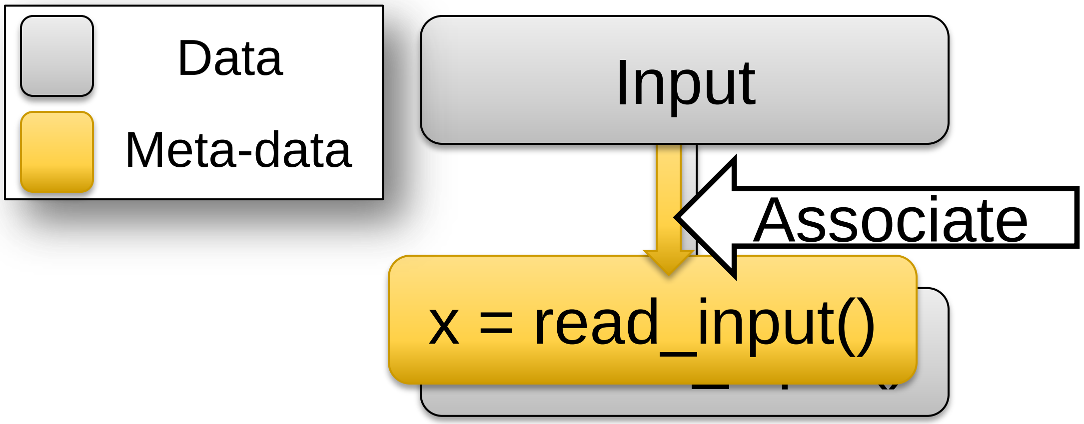
Example Dynamic Dataflow Analysis



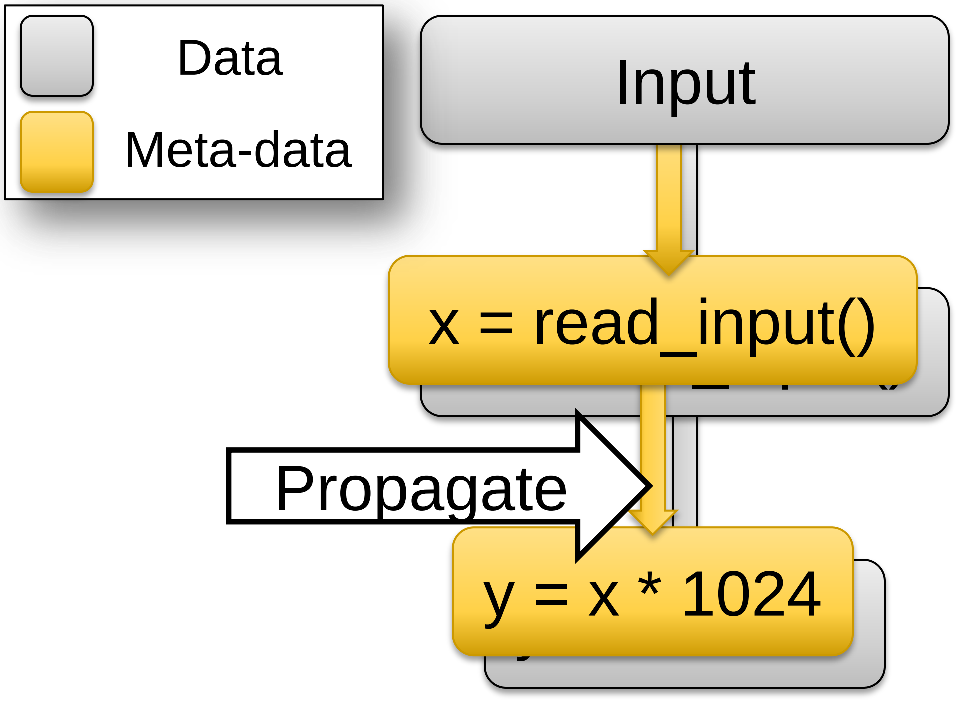
Example Dynamic Dataflow Analysis



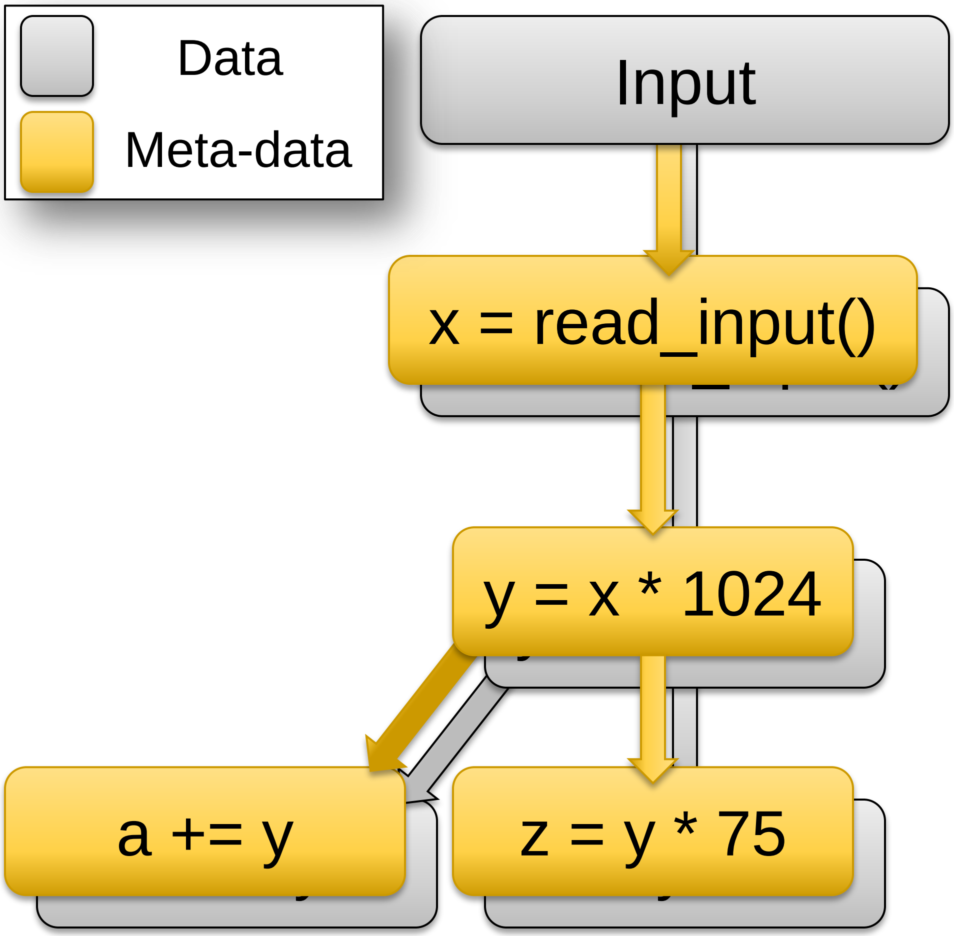
Example Dynamic Dataflow Analysis



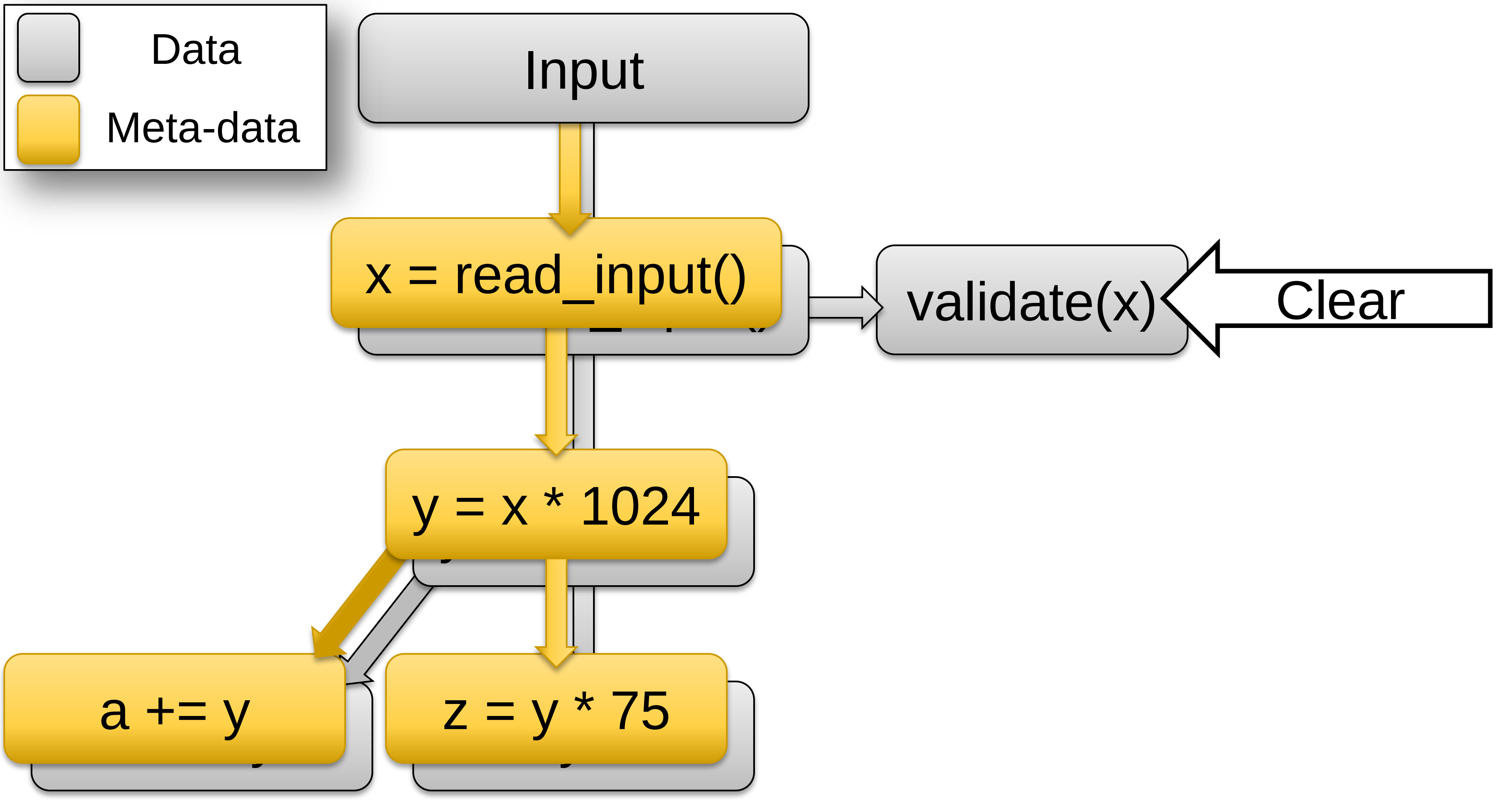
Example Dynamic Dataflow Analysis



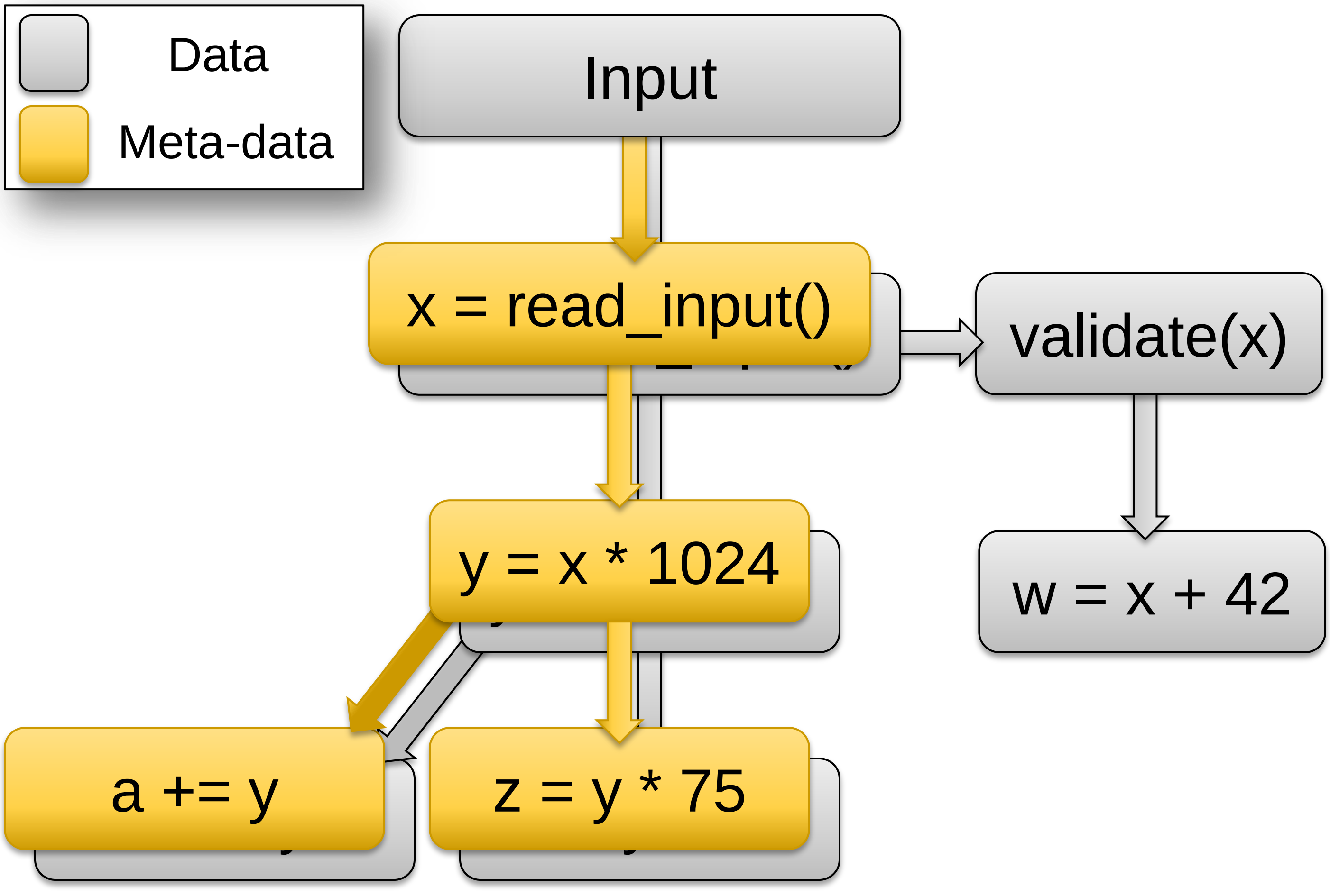
Example Dynamic Dataflow Analysis



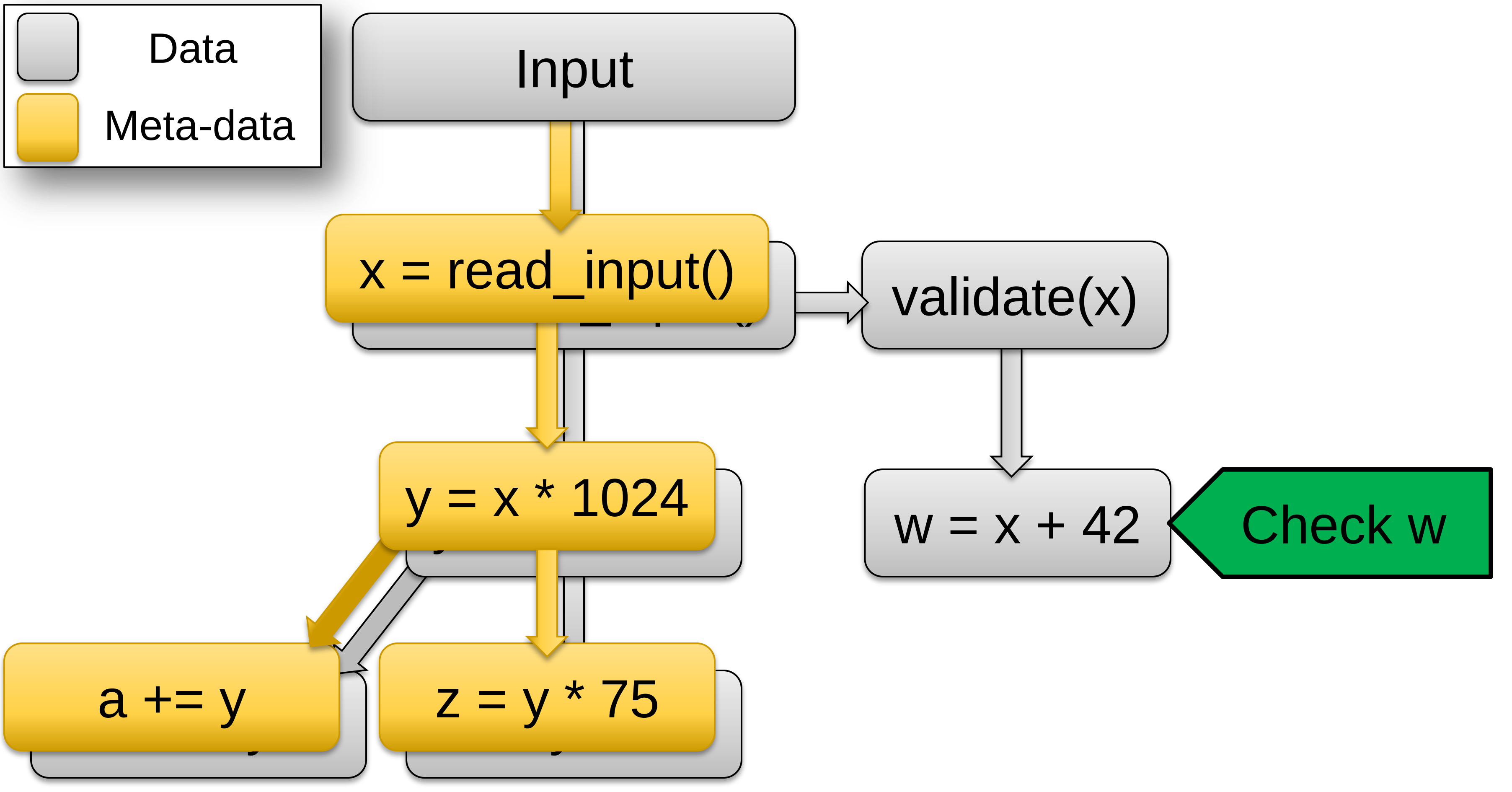
Example Dynamic Dataflow Analysis



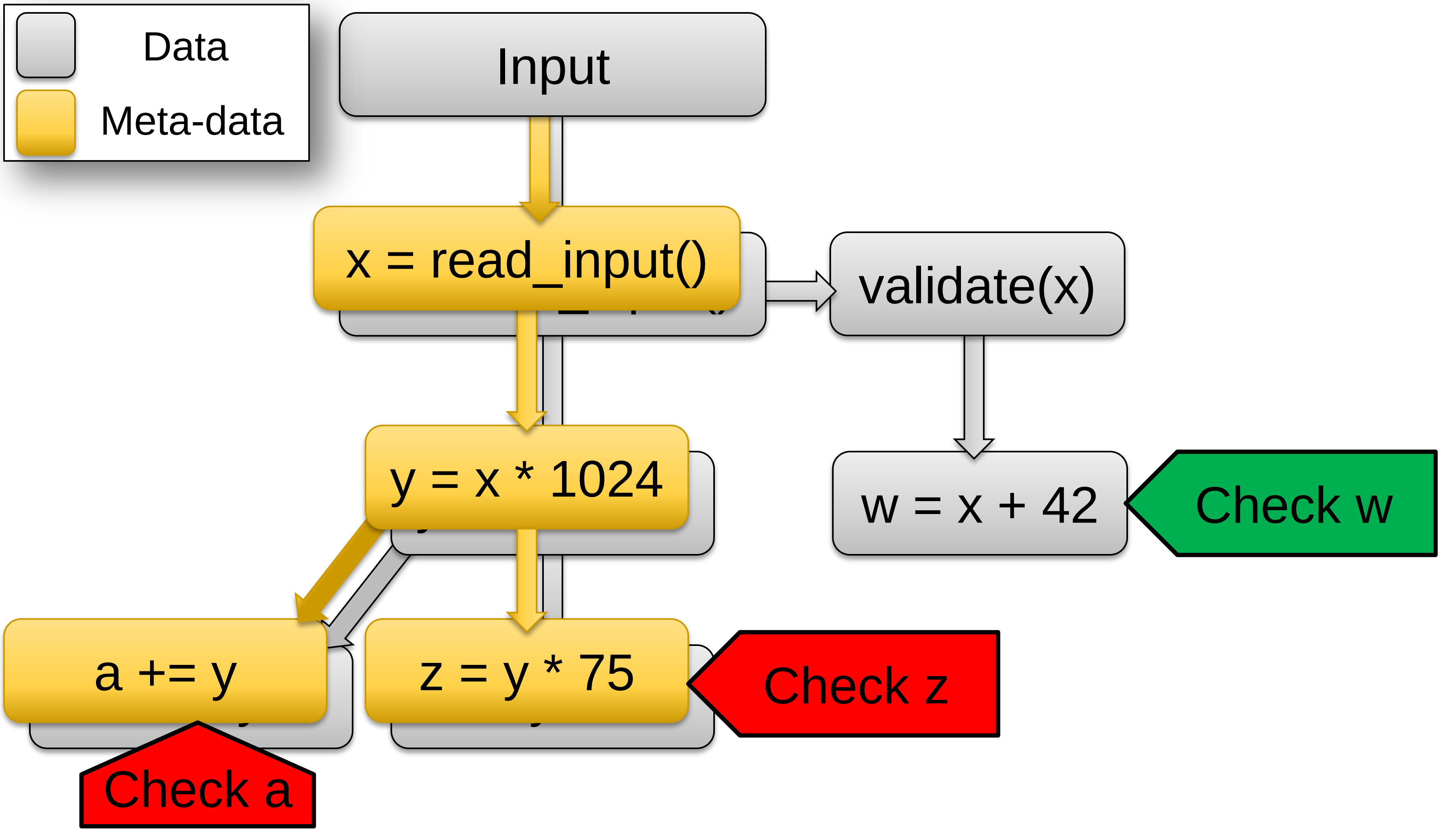
Example Dynamic Dataflow Analysis



Example Dynamic Dataflow Analysis

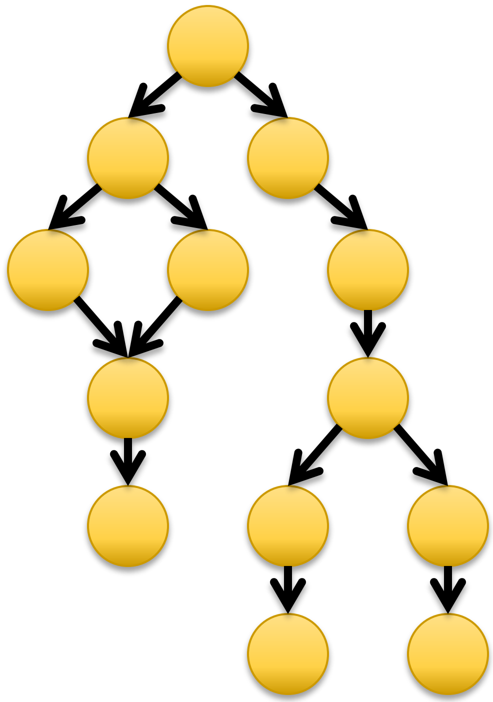


Example Dynamic Dataflow Analysis



Sampling Dataflows

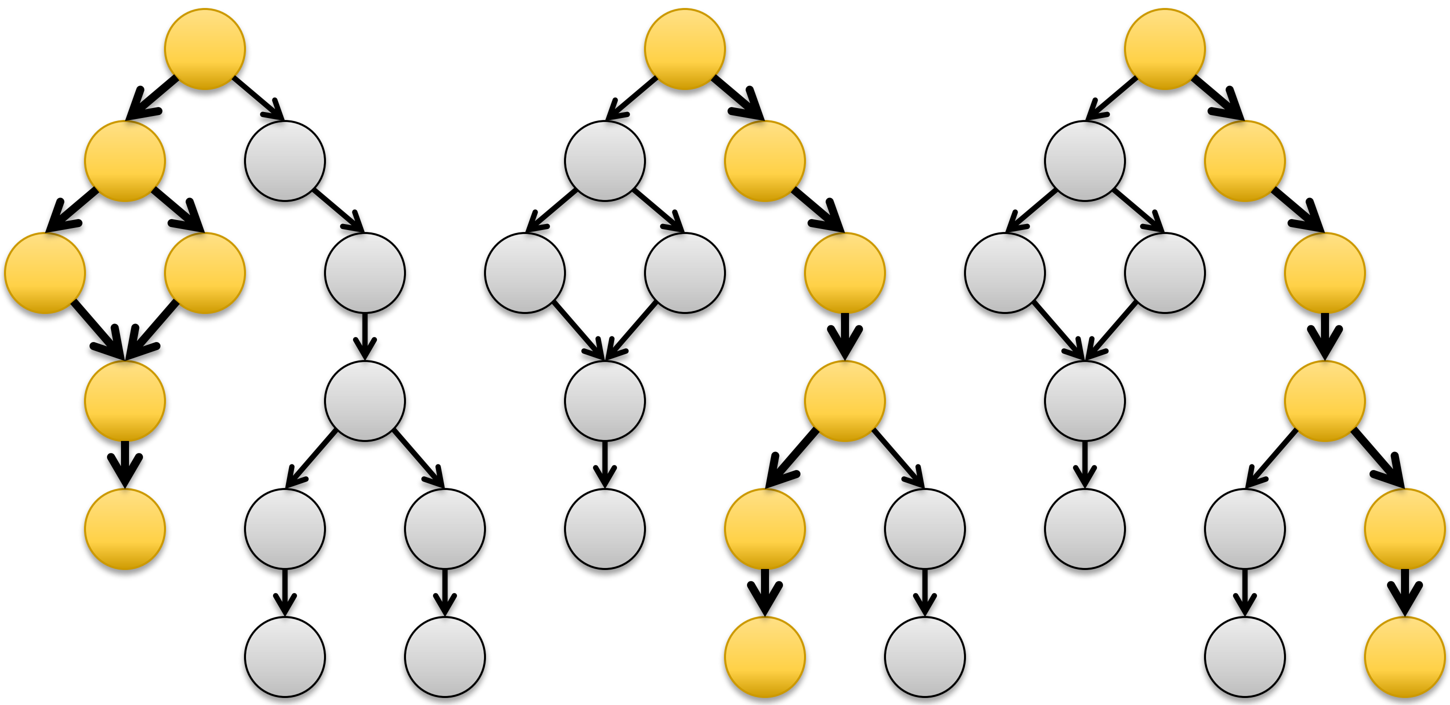
- Sampling must be aware of meta-data



- Remove meta-data from skipped dataflows

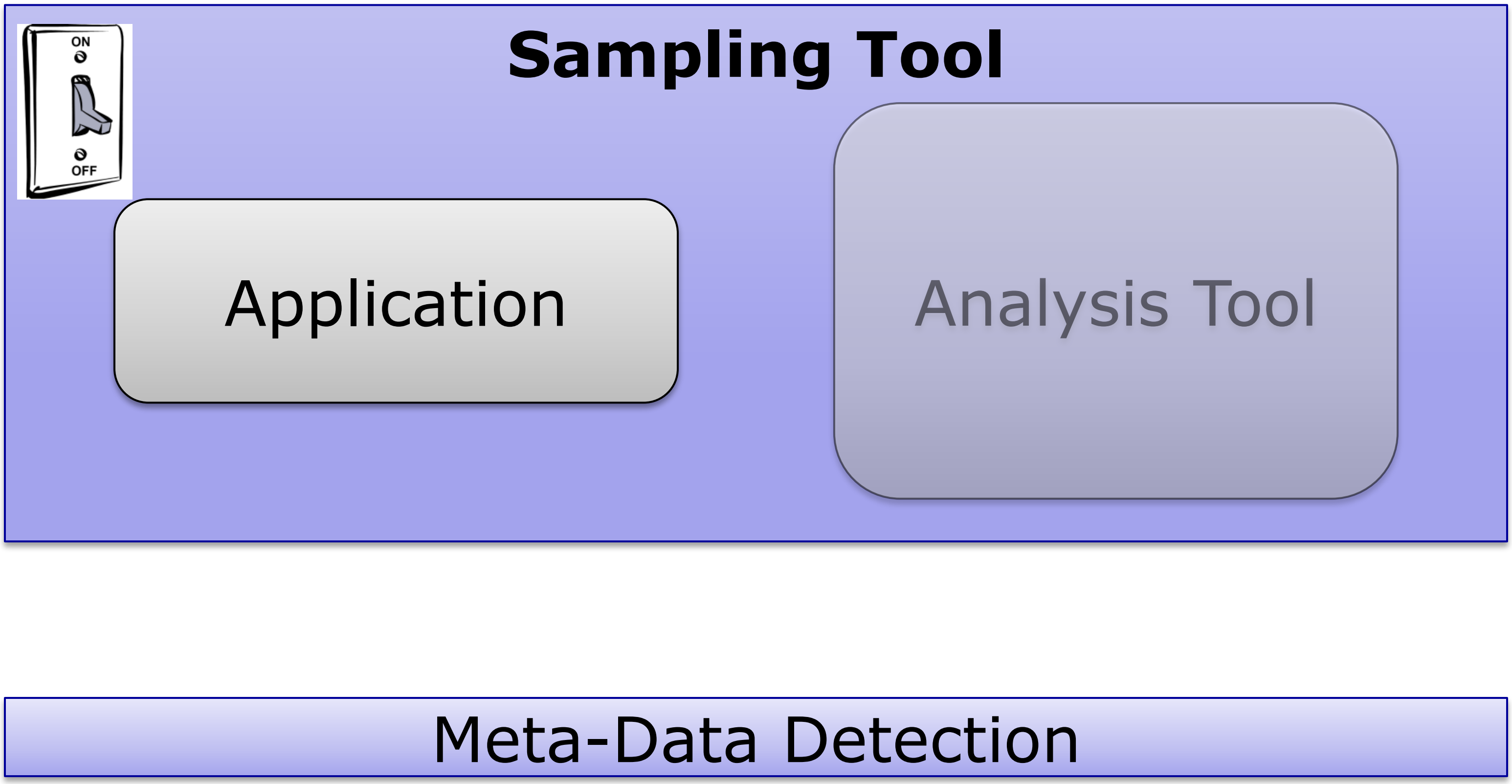
Sampling Dataflows

- Sampling must be aware of meta-data



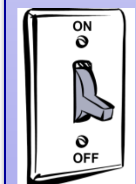
- Remove meta-data from skipped dataflows

Dataflow Sampling



Dataflow Sampling

Sampling Tool

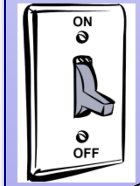


Application

Analysis Tool

Meta-Data Detection

Dataflow Sampling



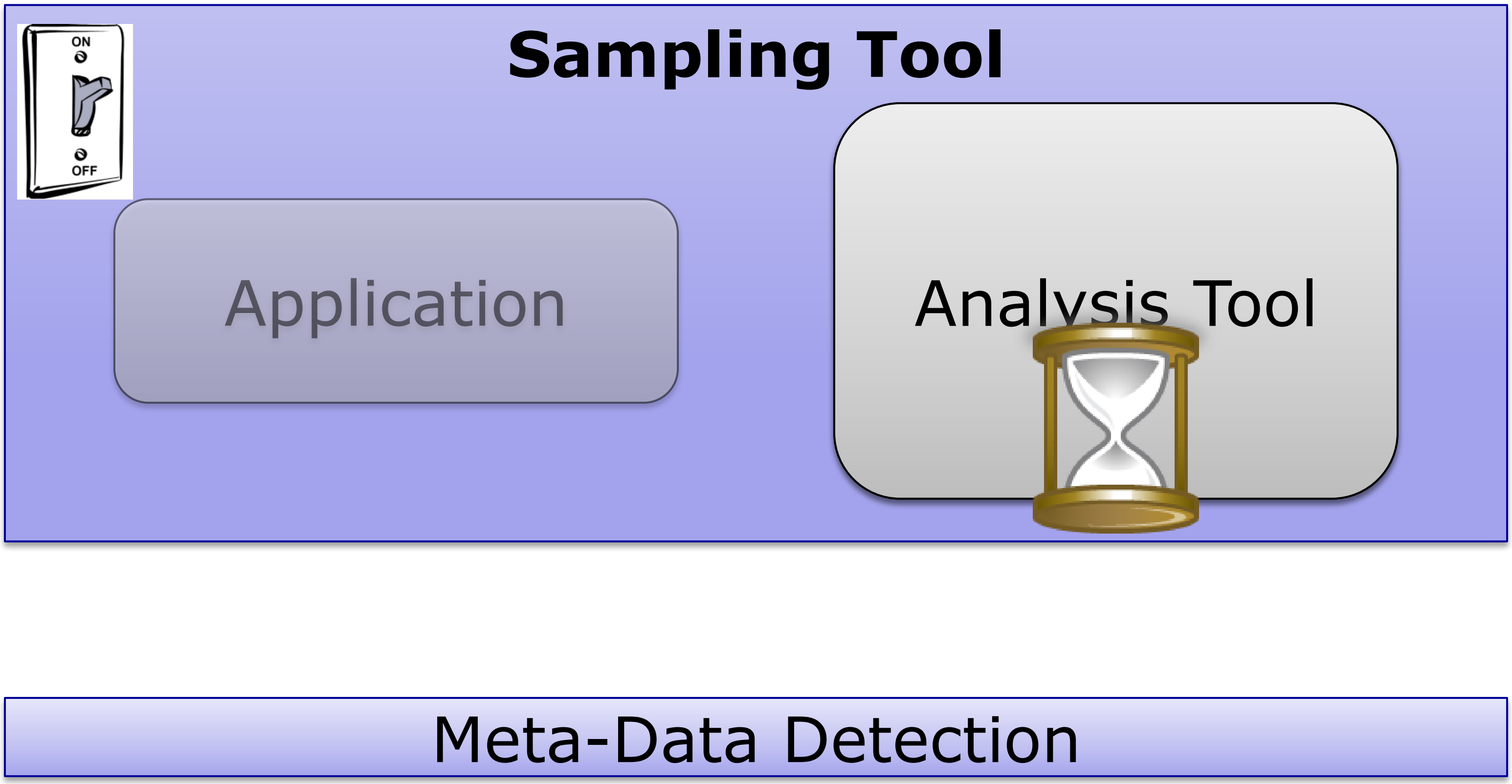
Sampling Tool

Application

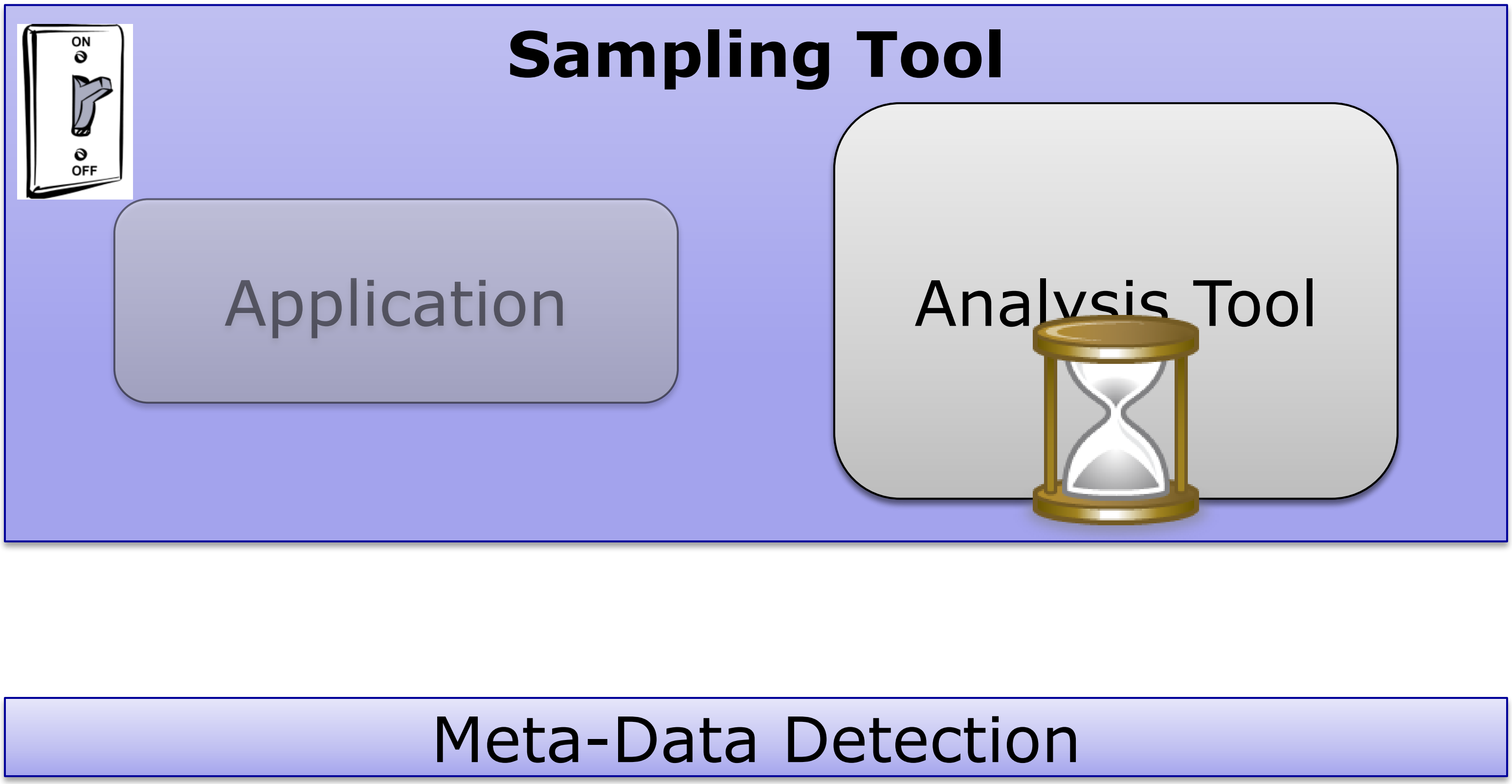
Analysis Tool

Meta-Data Detection

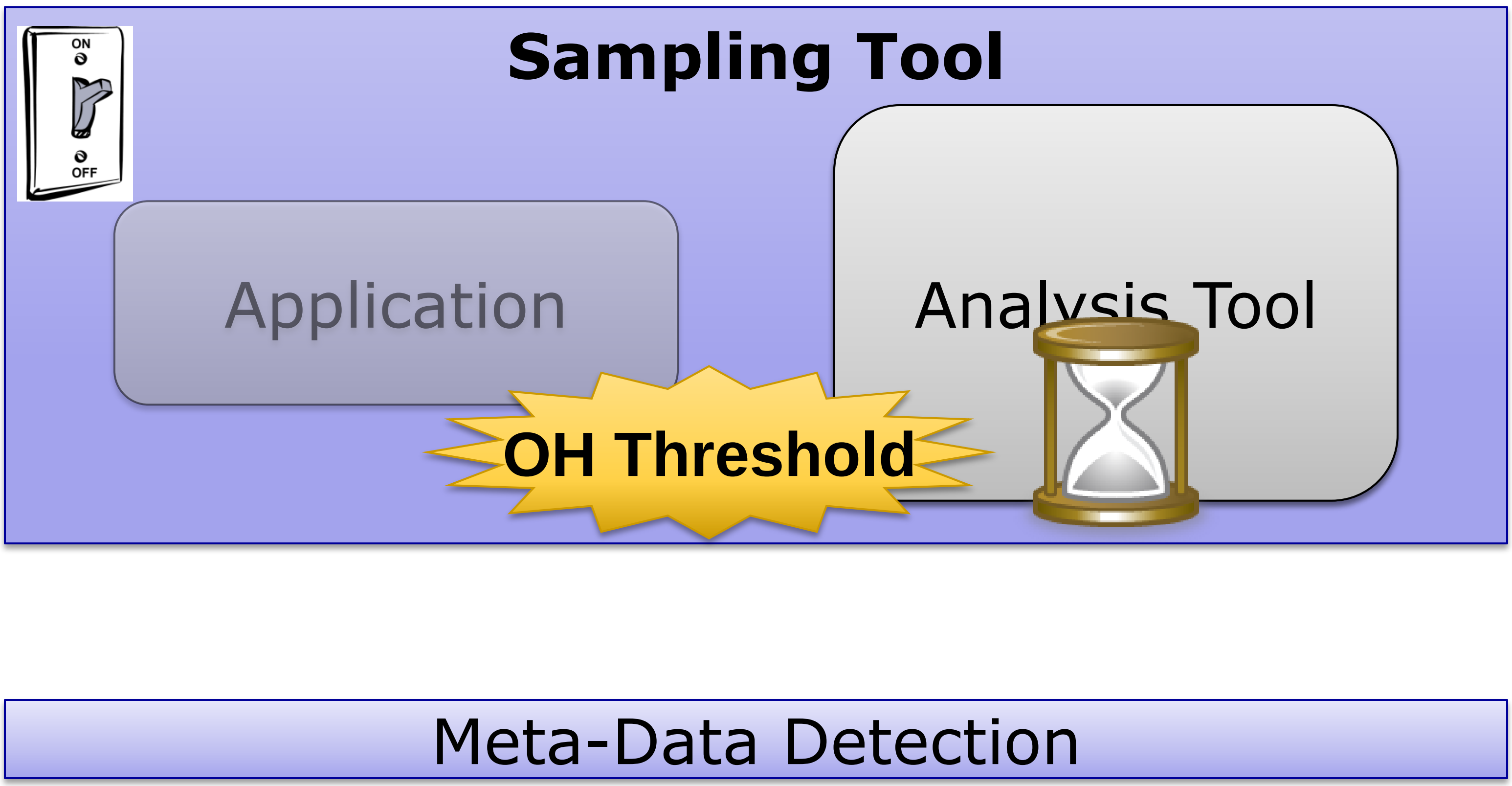
Dataflow Sampling



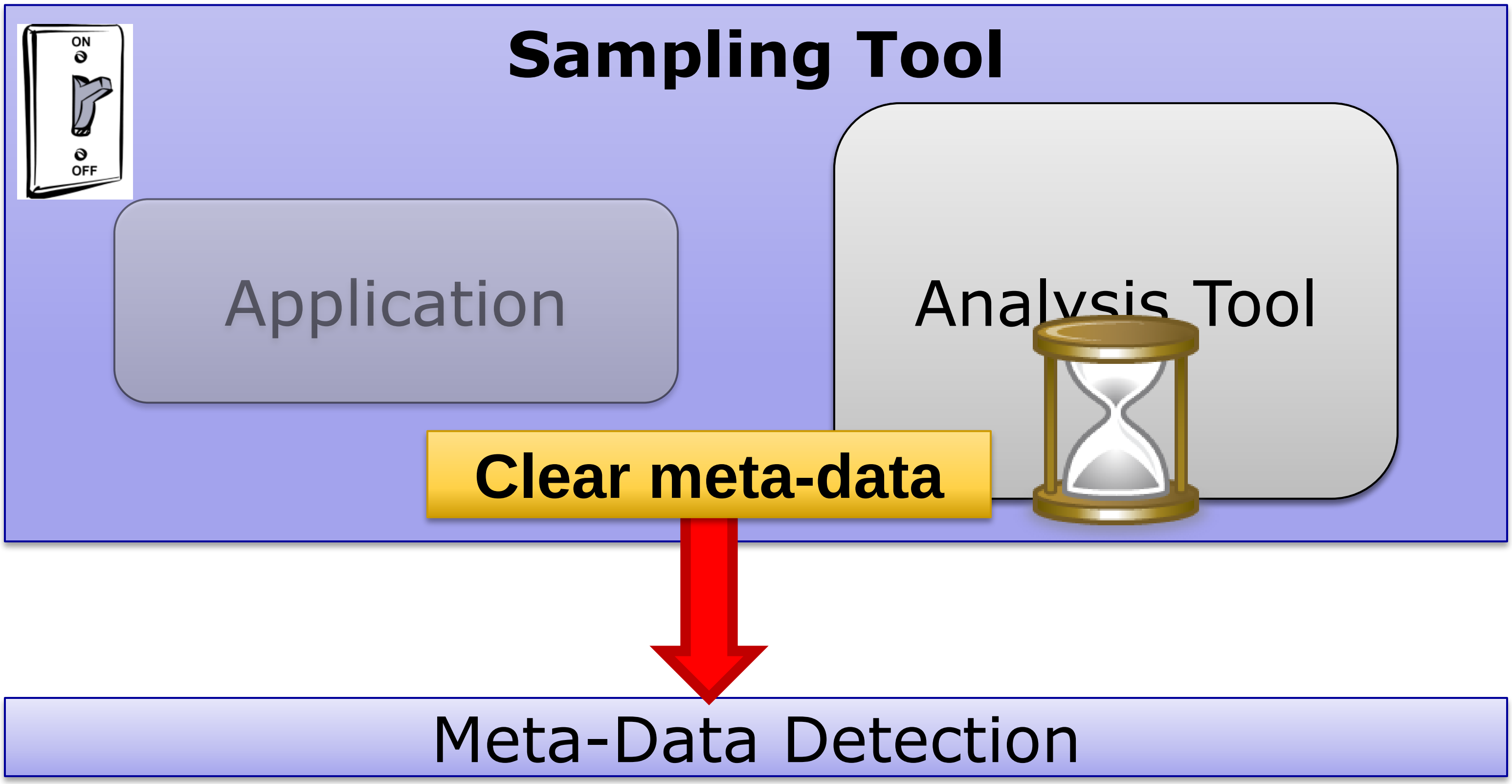
Dataflow Sampling



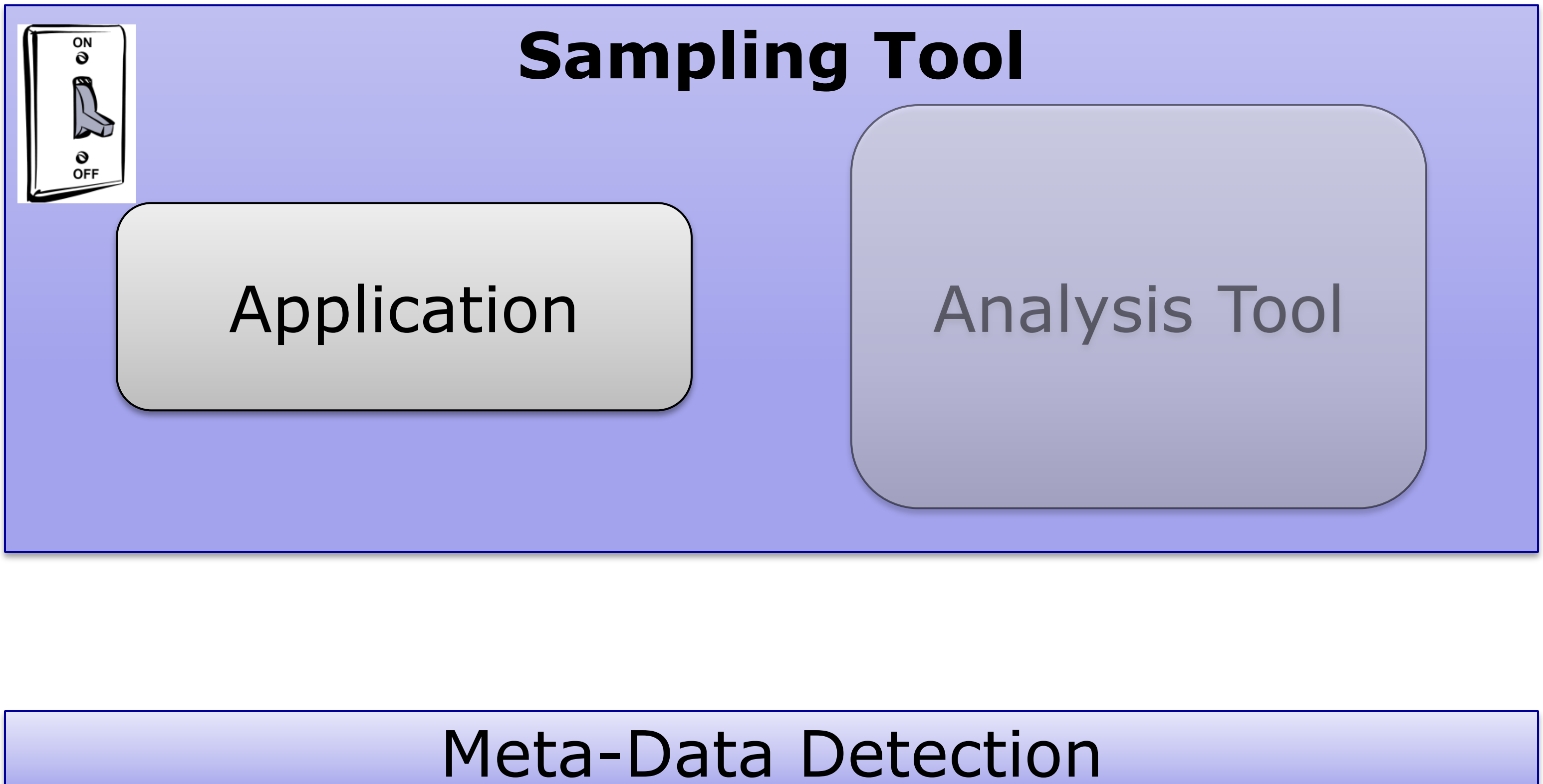
Dataflow Sampling



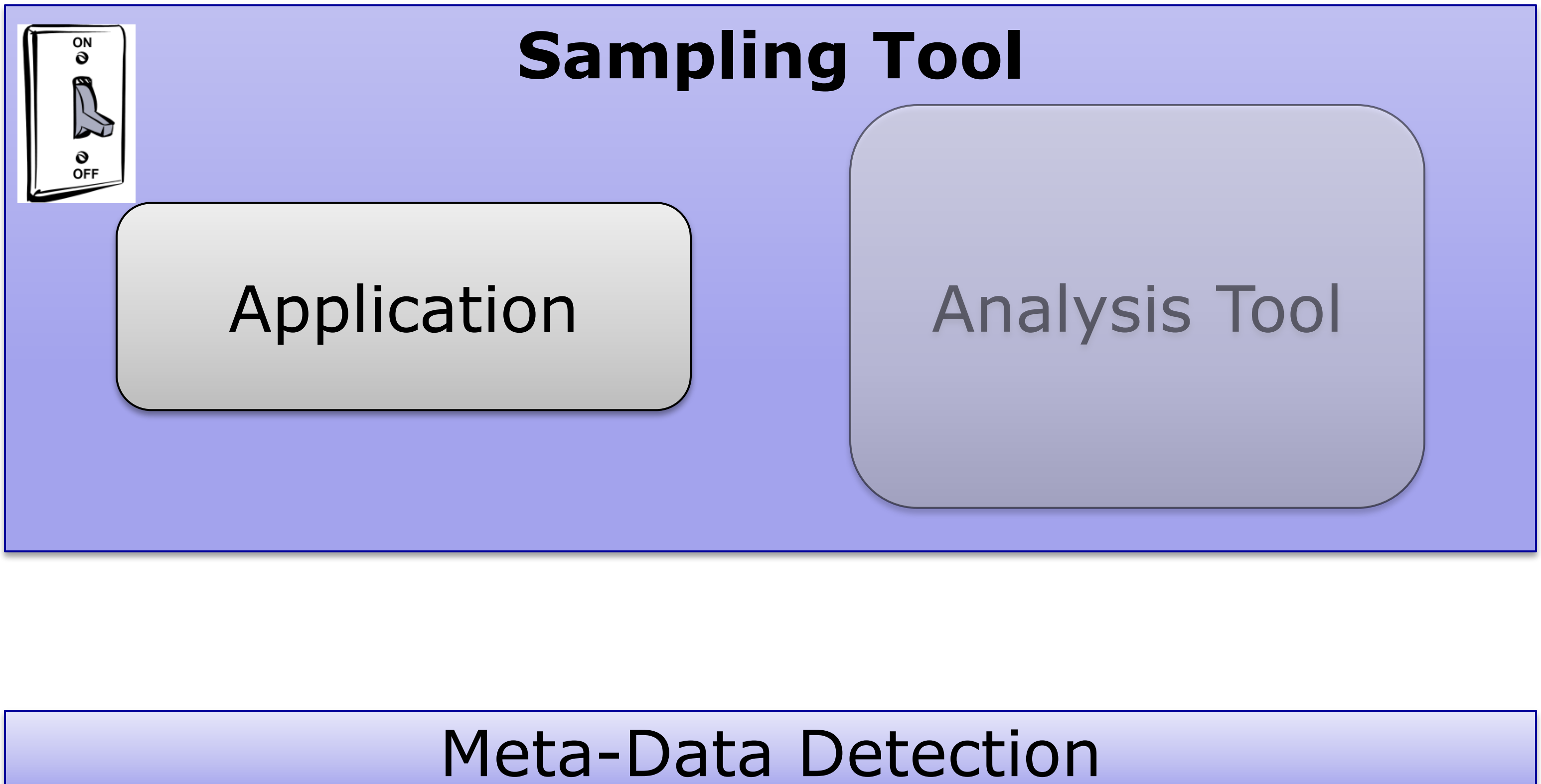
Dataflow Sampling



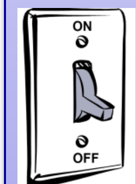
Dataflow Sampling



Dataflow Sampling



Dataflow Sampling



Sampling Tool

Application

Analysis Tool

Meta-Data Detection

Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

Finding Meta-Data

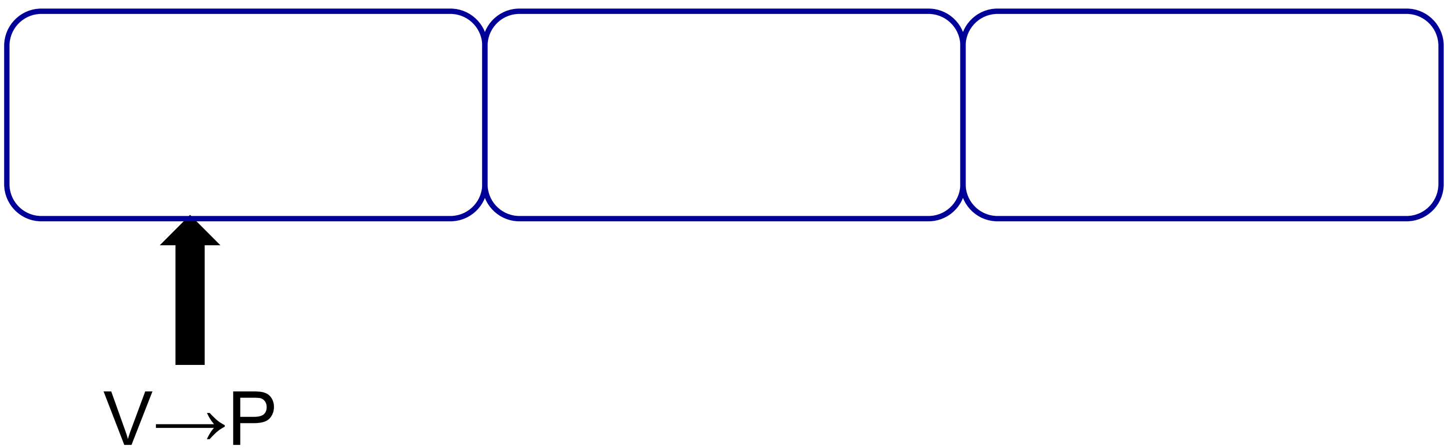
- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

❿Solution: Virtual Memory Watchpoints

Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

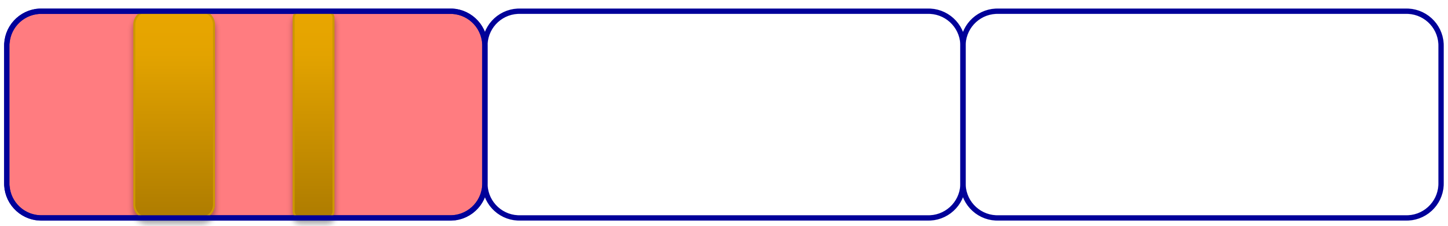
❿ Solution: Virtual Memory Watchpoints



Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

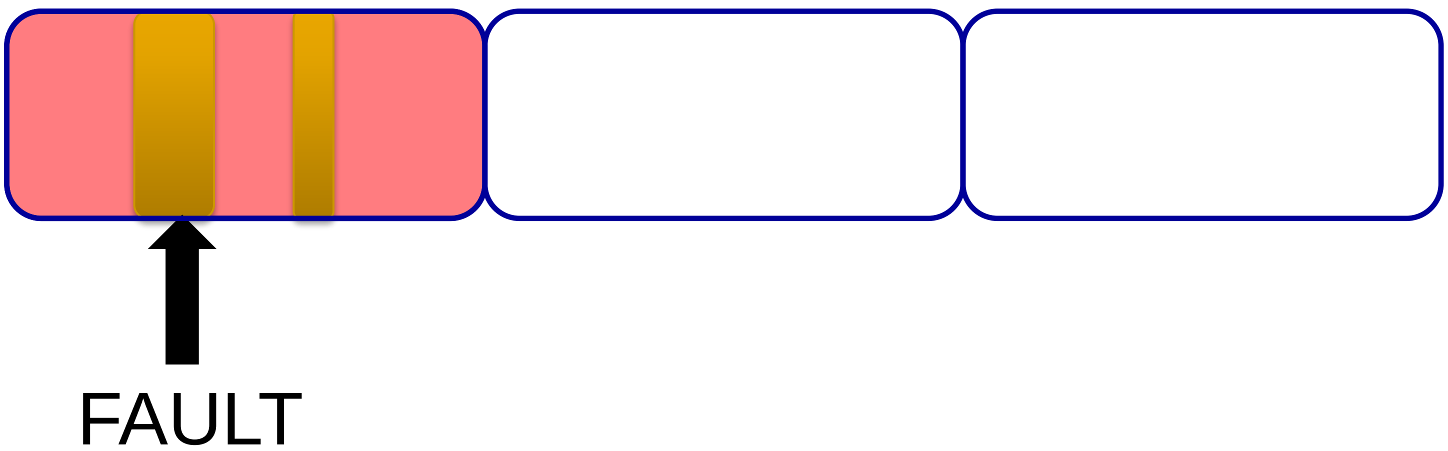
➓Solution: Virtual Memory Watchpoints



Finding Meta-Data

- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

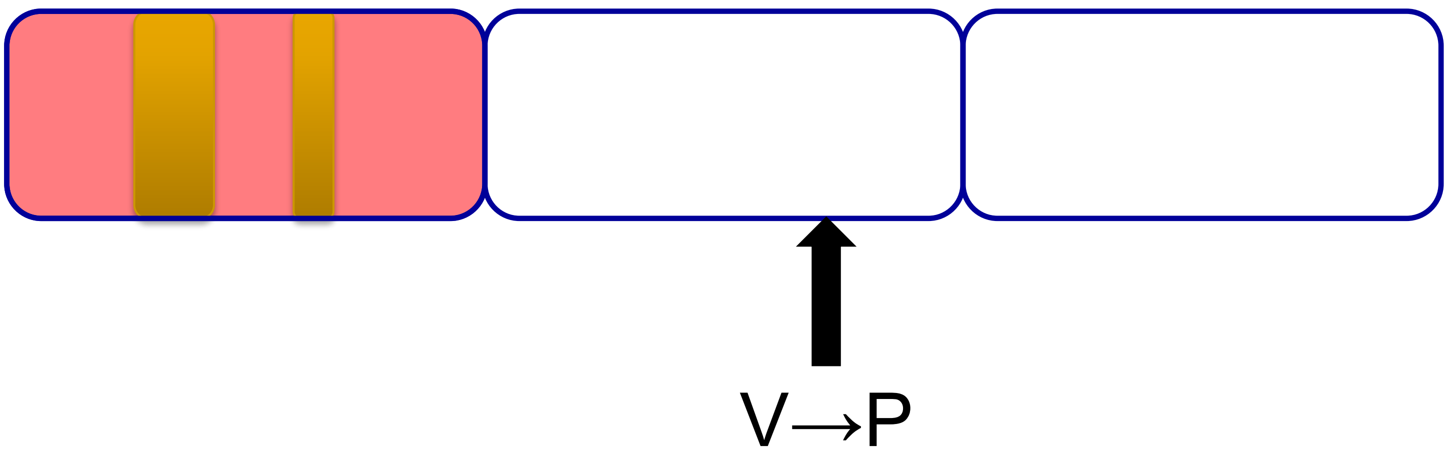
❿Solution: Virtual Memory Watchpoints



Finding Meta-Data

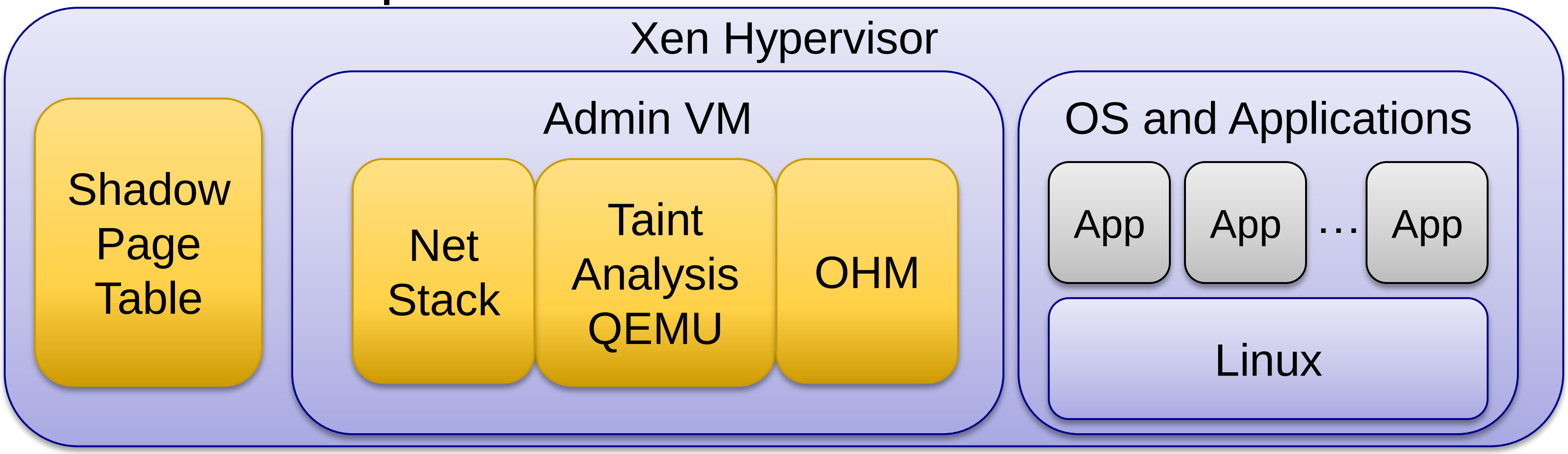
- No additional overhead when no meta-data
  - Needs hardware support
- Take a fault when touching shadowed data

❿Solution: Virtual Memory Watchpoints



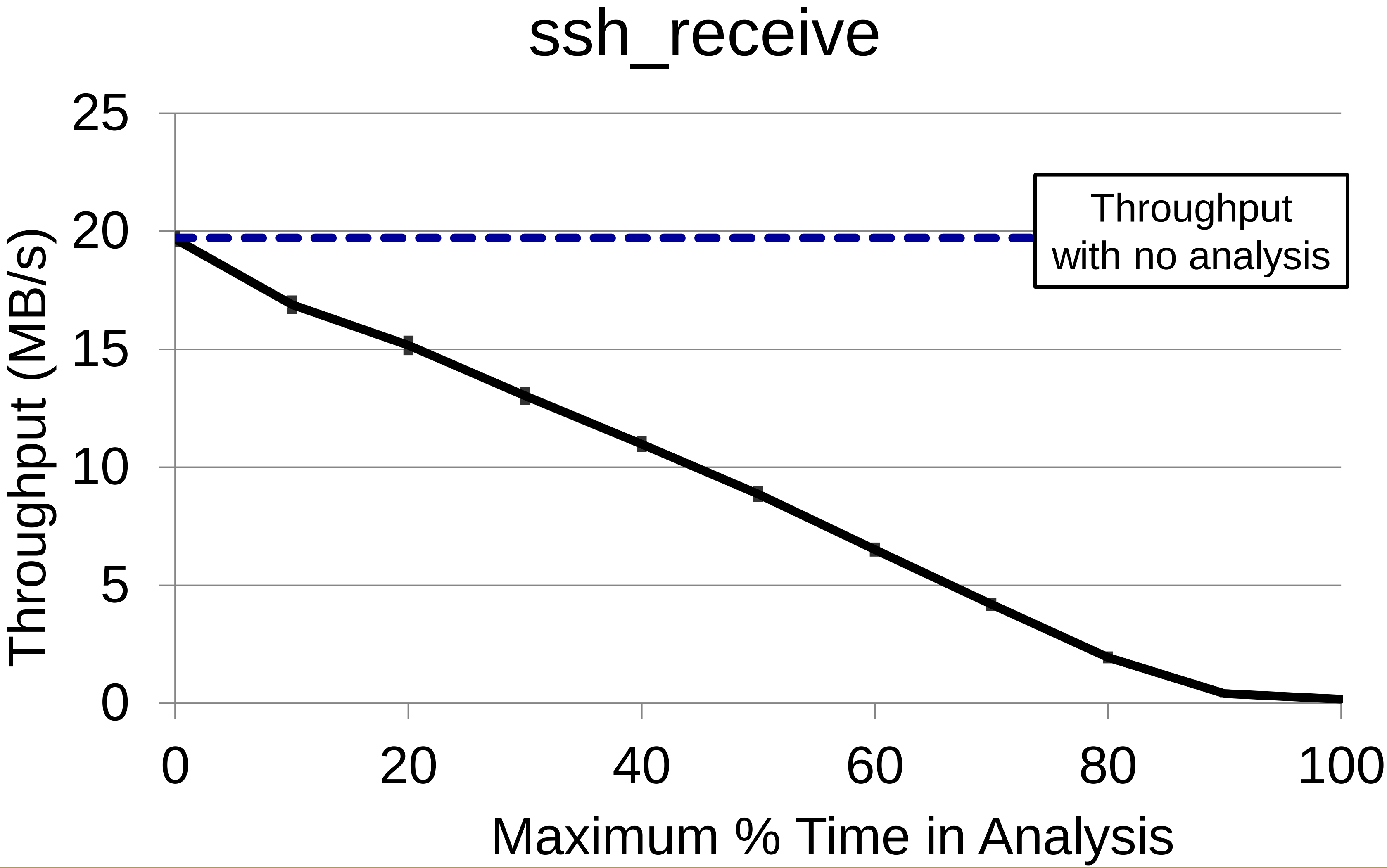
Prototype Setup

- Xen+QEMU Taint analysis sampling system
  - Network packets untrusted



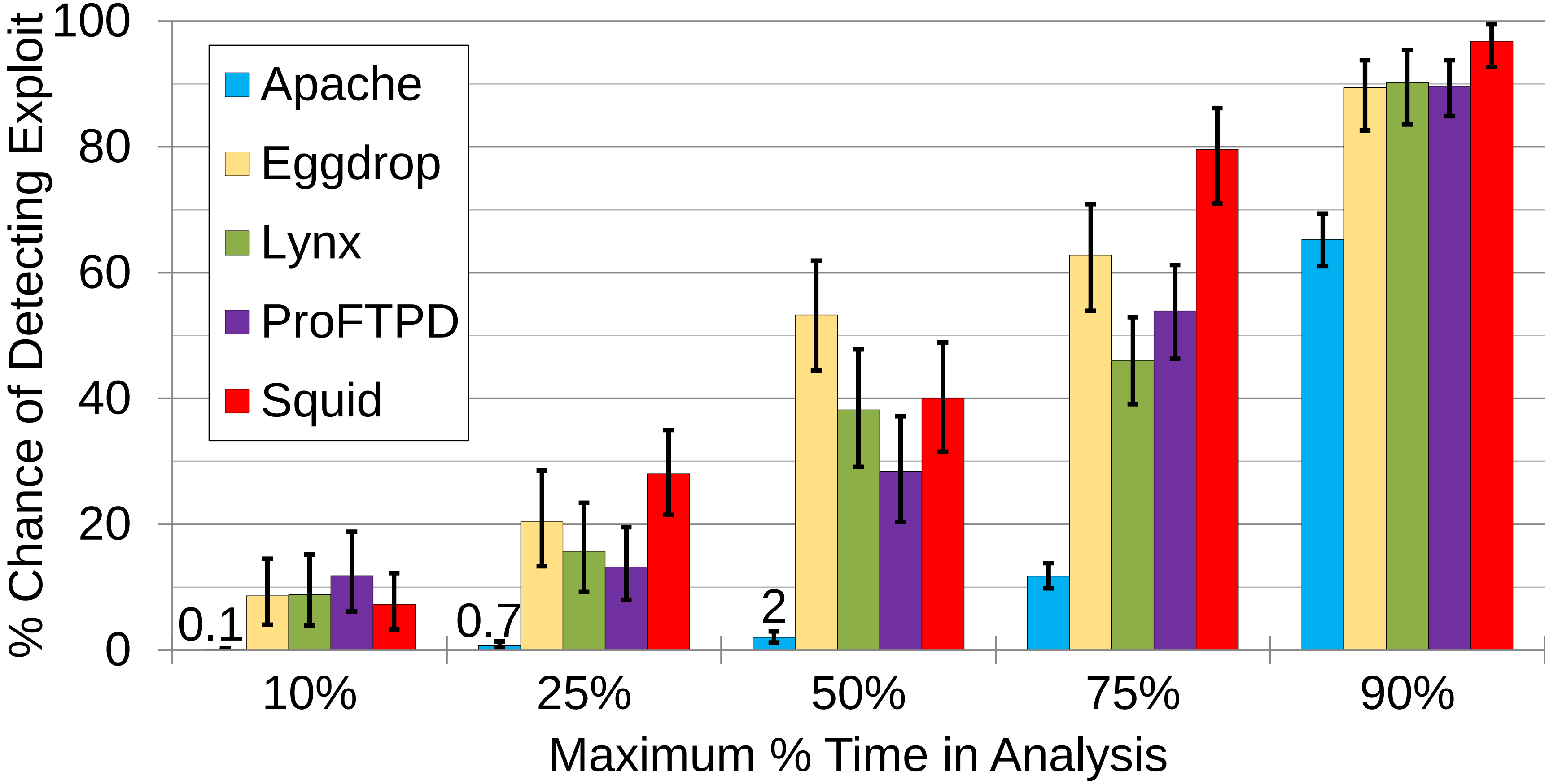
- Performance Tests – Network Throughput
  - *Example: **ssh_receive***
- Sampling Accuracy Tests
  - Real-world Security Exploits

Performance of Dataflow Sampling



Accuracy with Background Tasks

ssh_receive running in background

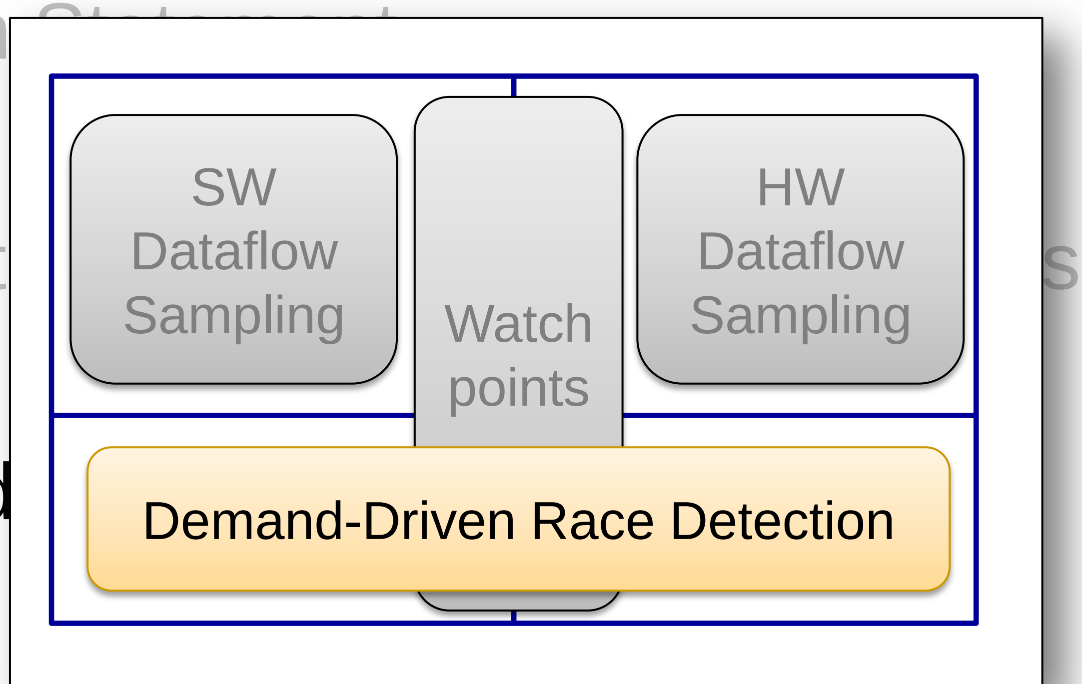


Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints

Outline

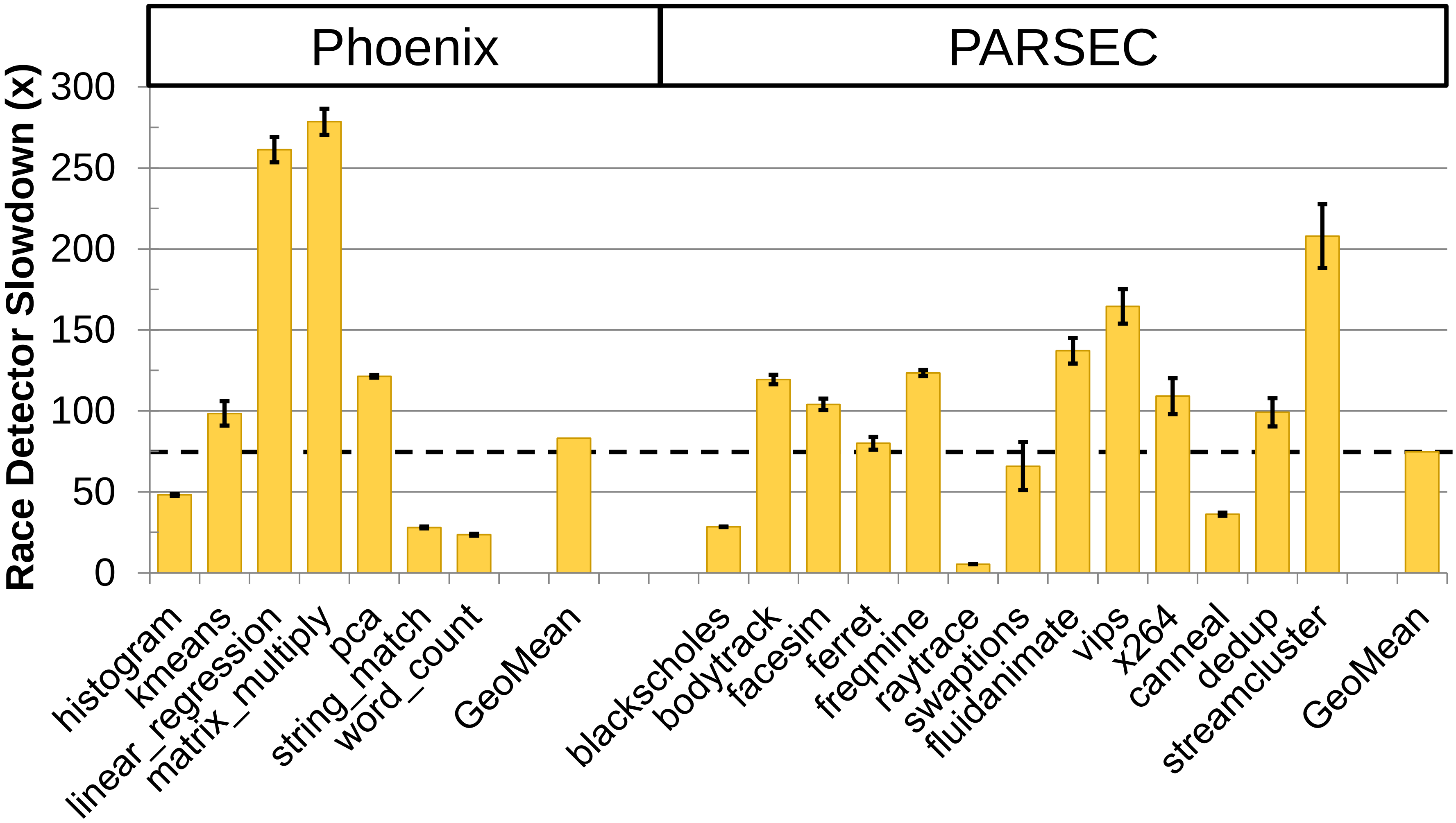
- Problem
- Distribut s
- Demand
- Unlimited Watchpoints



Dynamic Data Race Detection

- Add checks around every memory access
- Find inter-thread sharing
- Synchronization between write-shared accesses?
  - No? Data race.

SW Race Detection is Slow



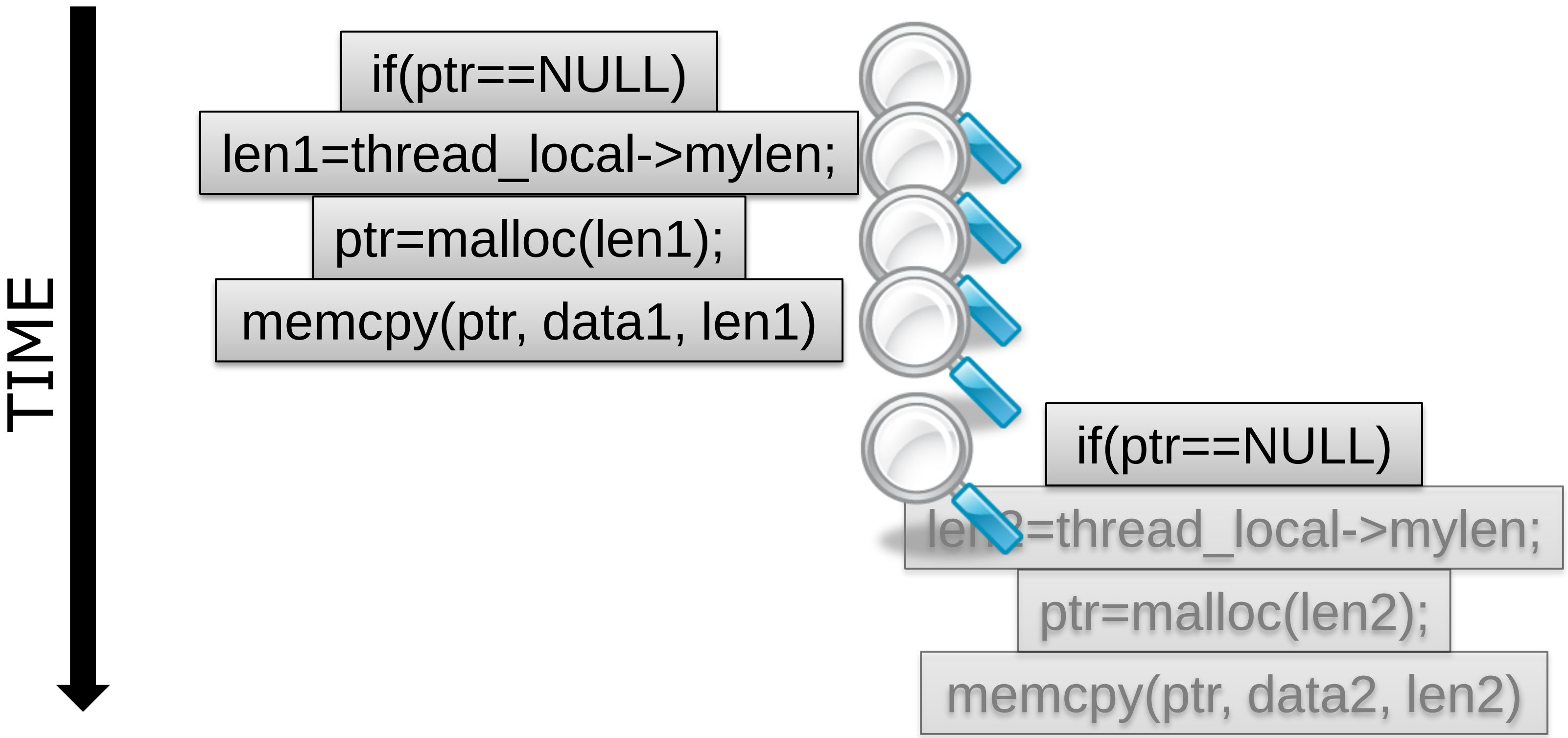
Inter-thread Sharing is What's Important

TIME

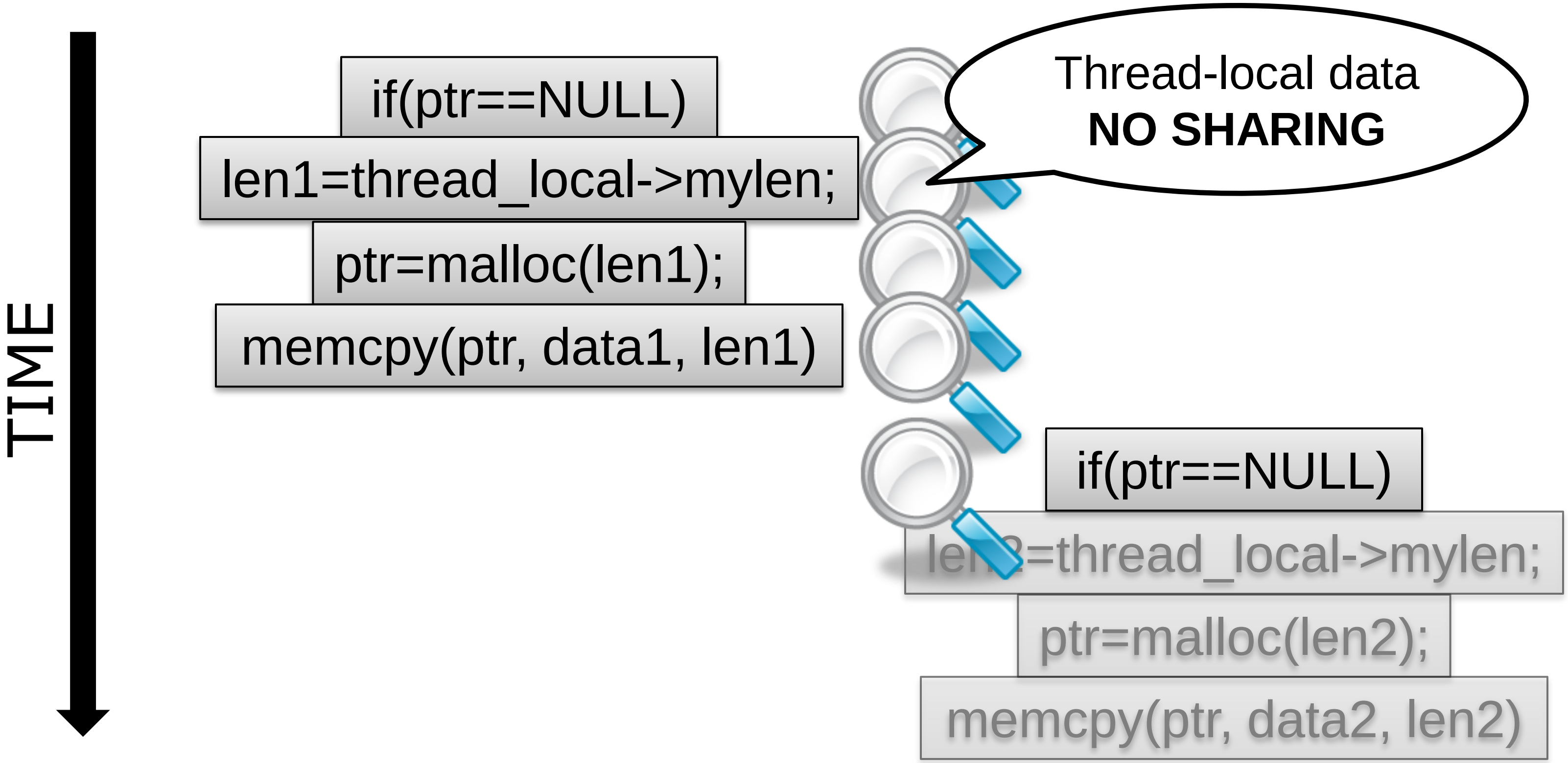
```
if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)
```

```
if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
```

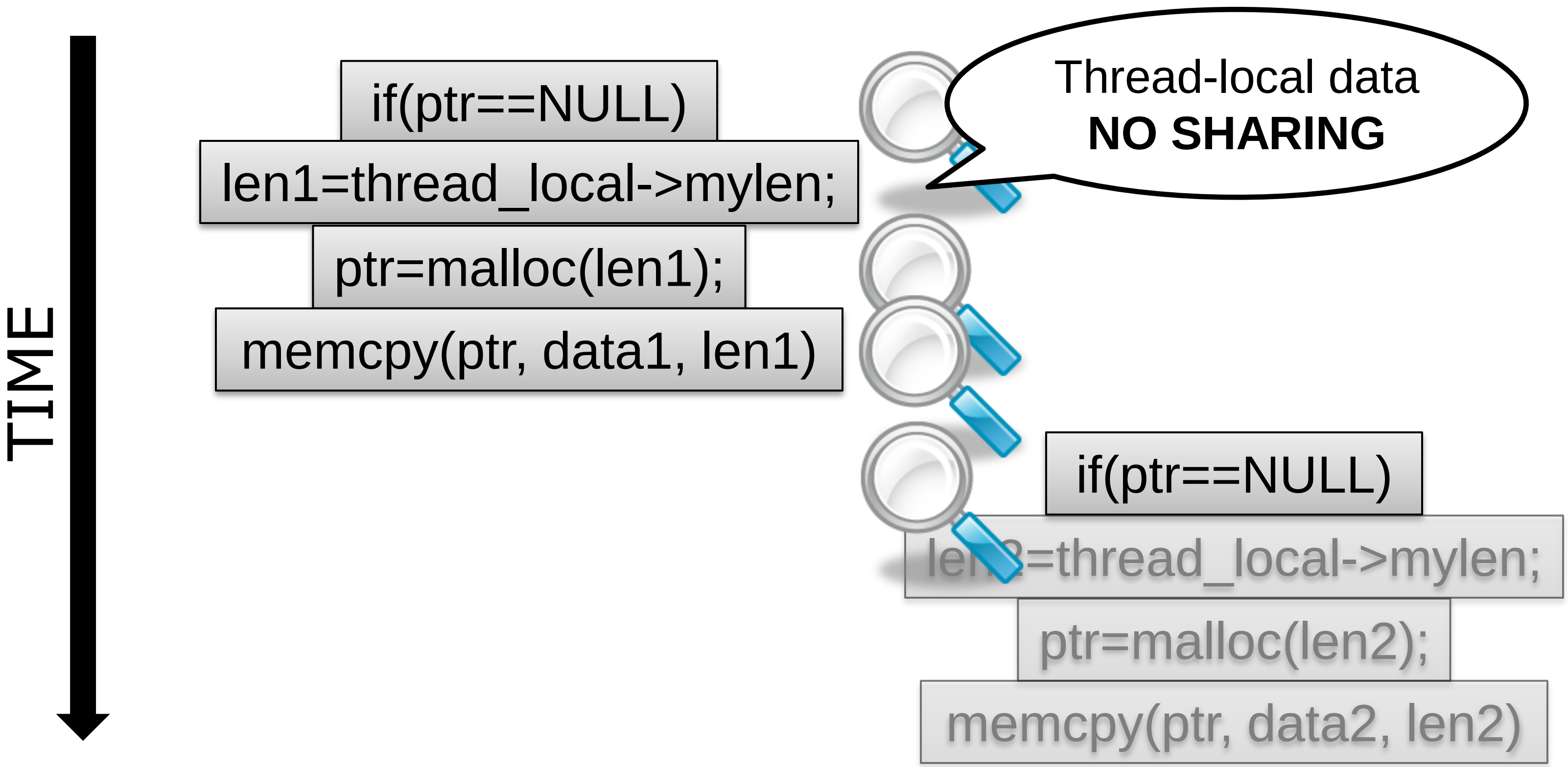
Inter-thread Sharing is What's Important



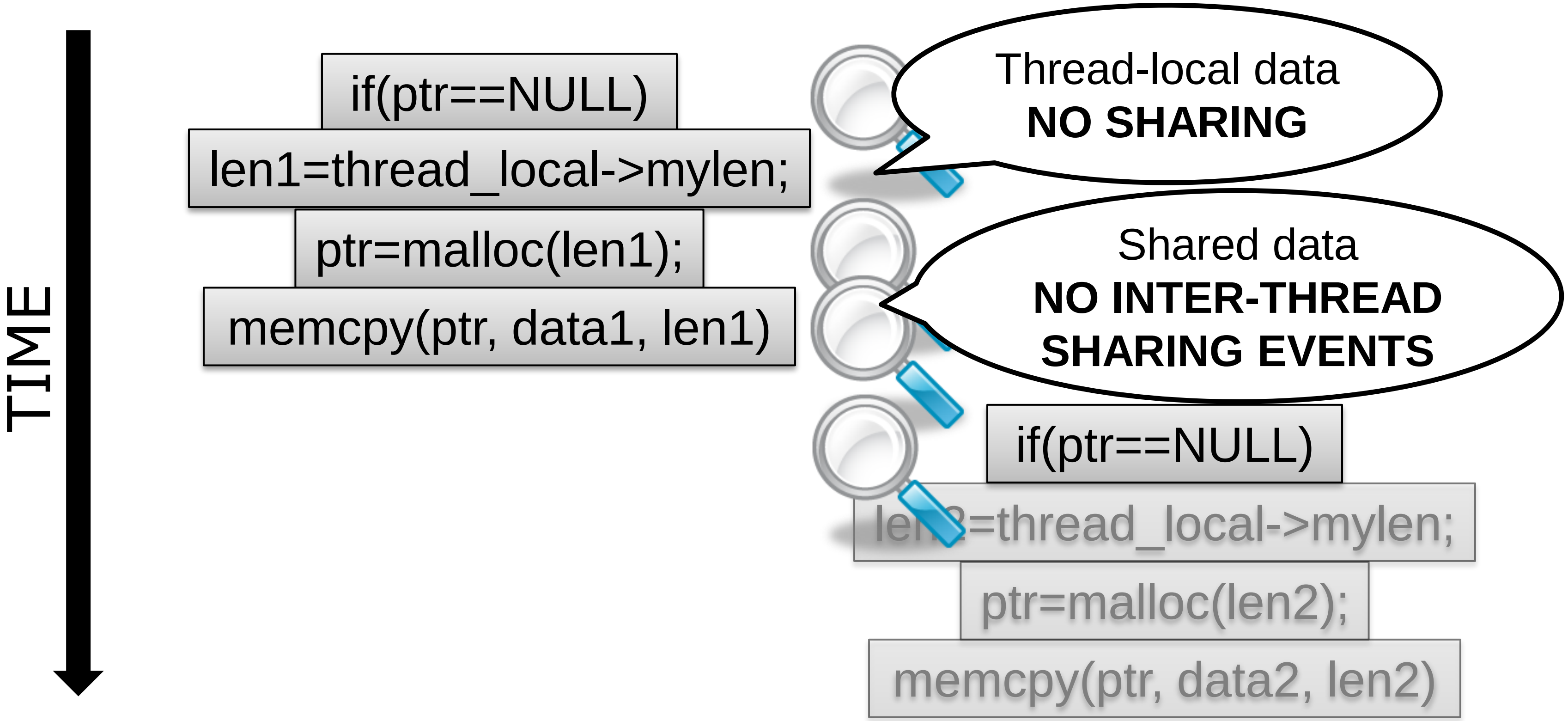
Inter-thread Sharing is What's Important



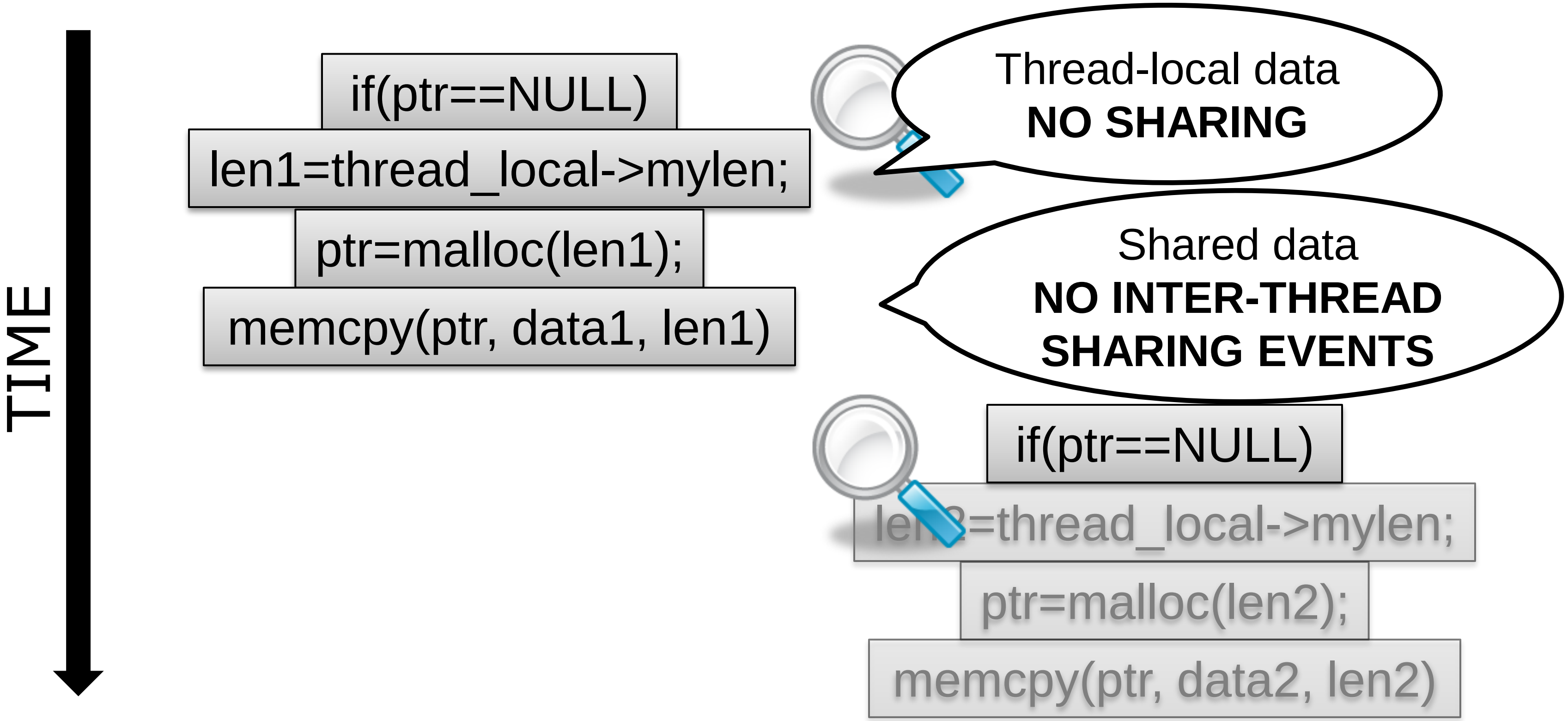
Inter-thread Sharing is What's Important



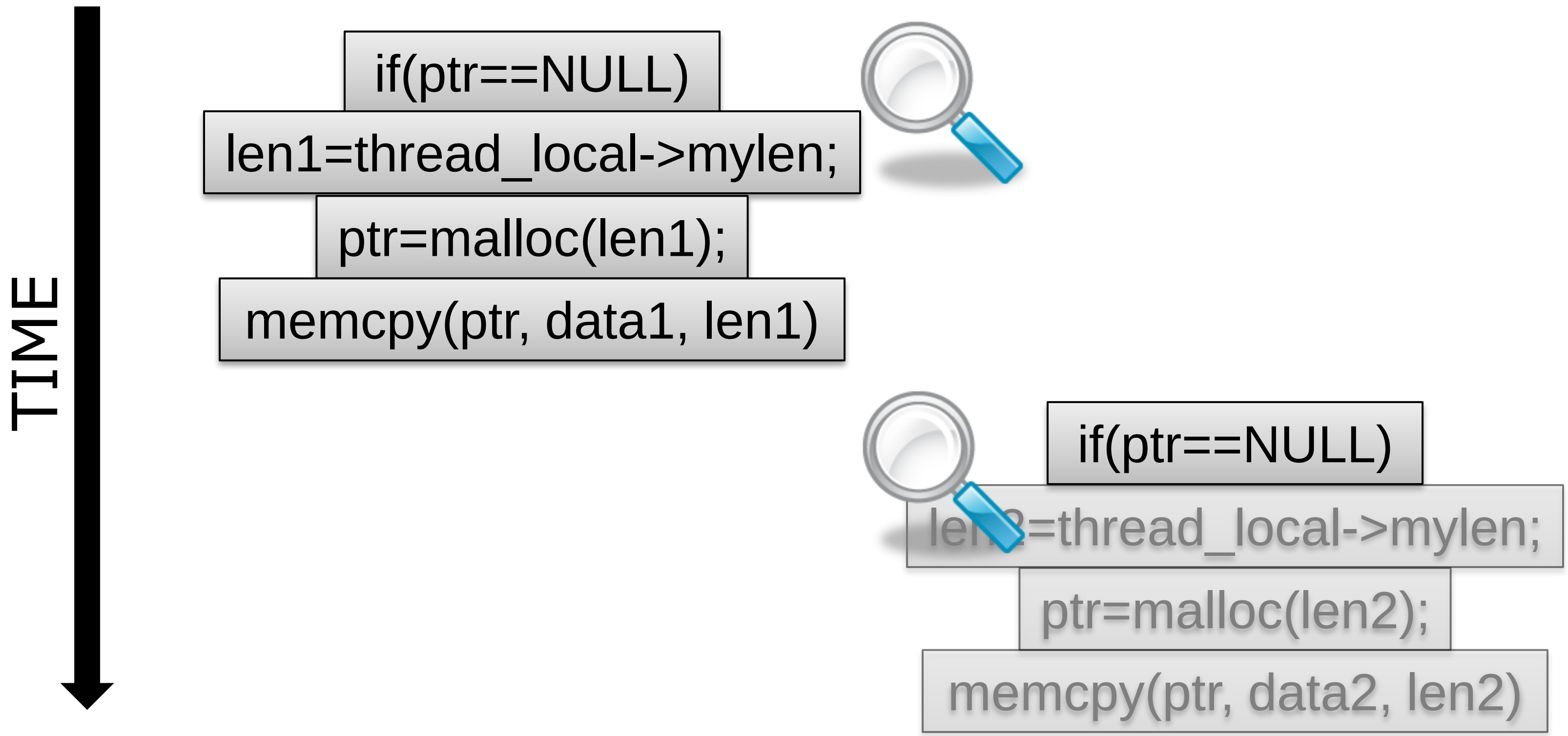
Inter-thread Sharing is What's Important



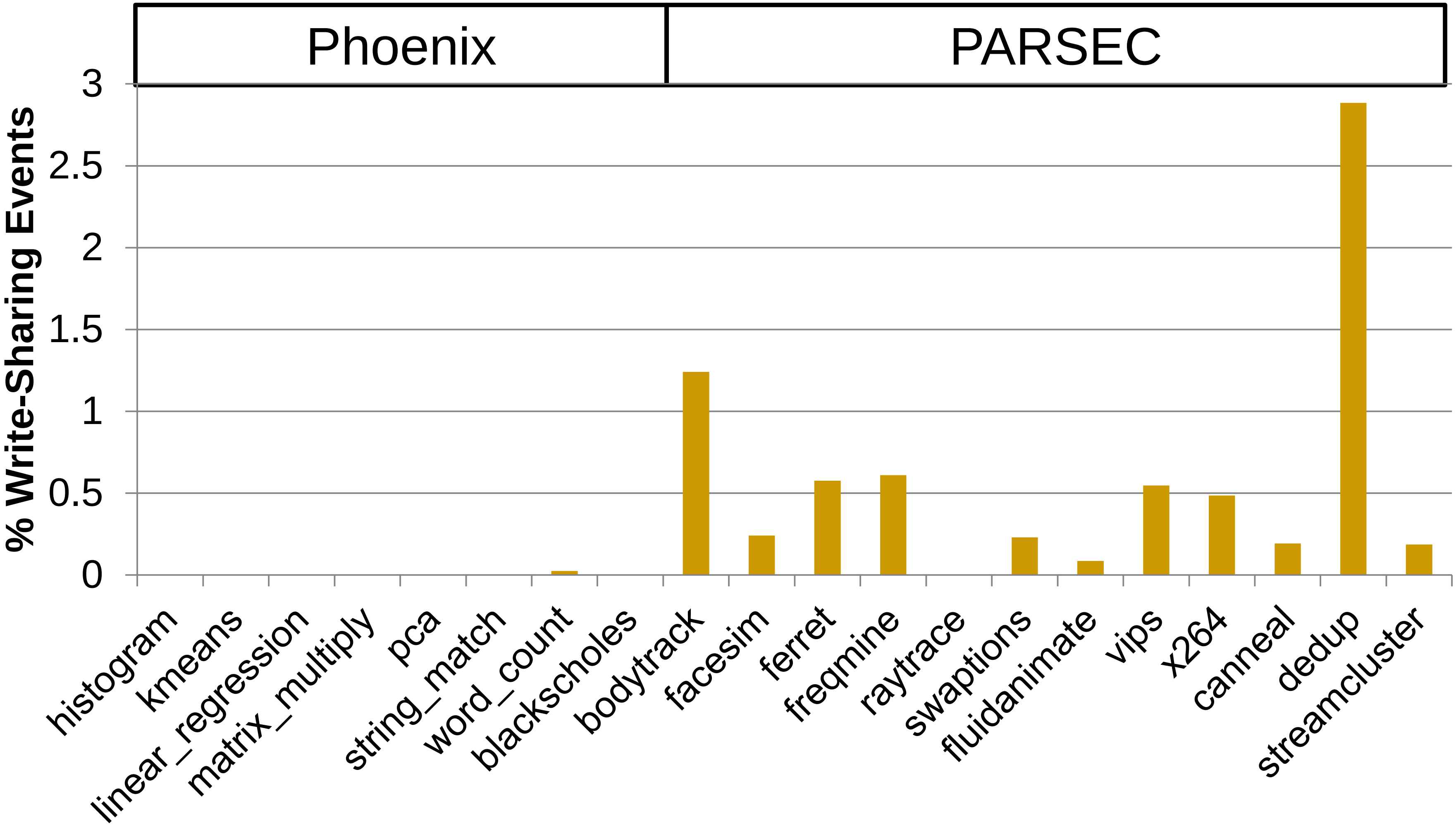
Inter-thread Sharing is What's Important



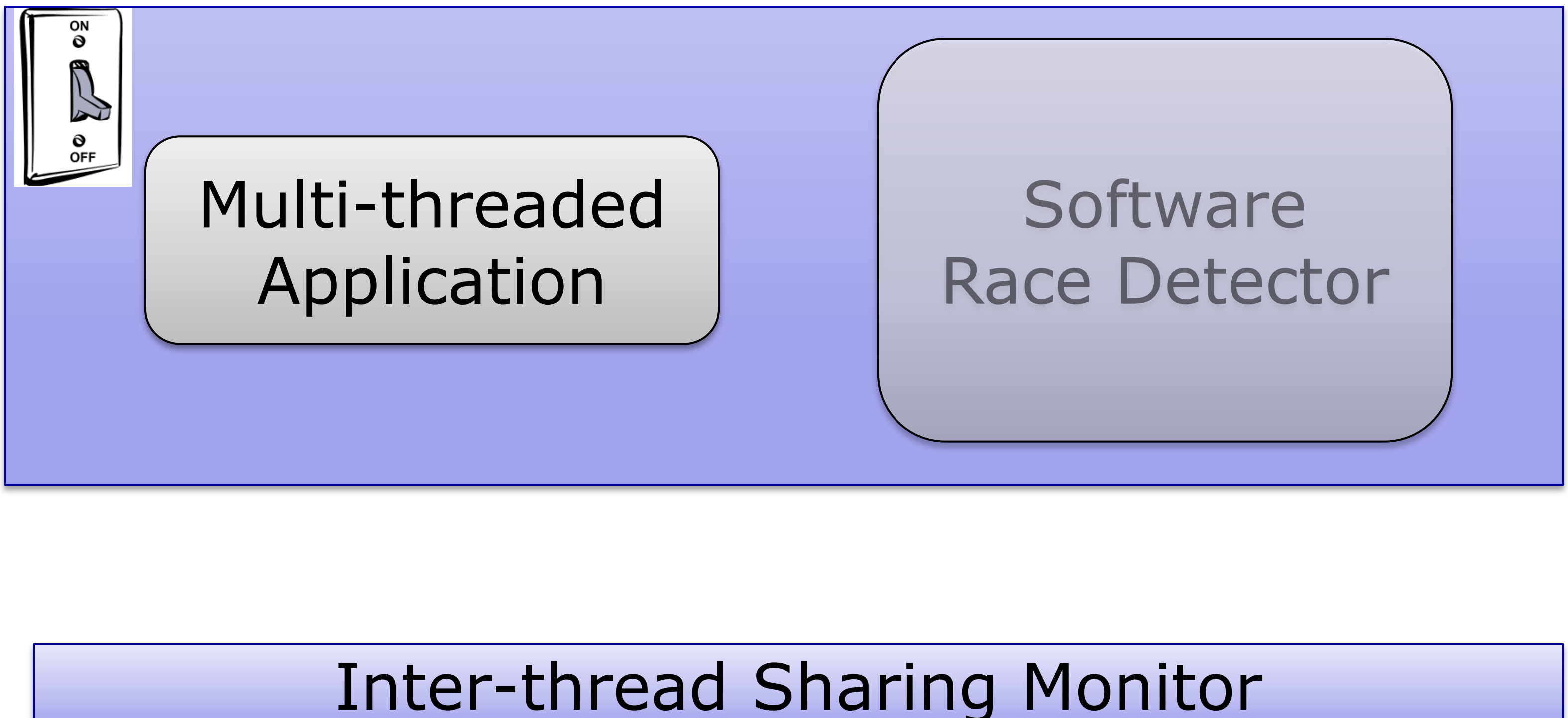
Inter-thread Sharing is What's Important



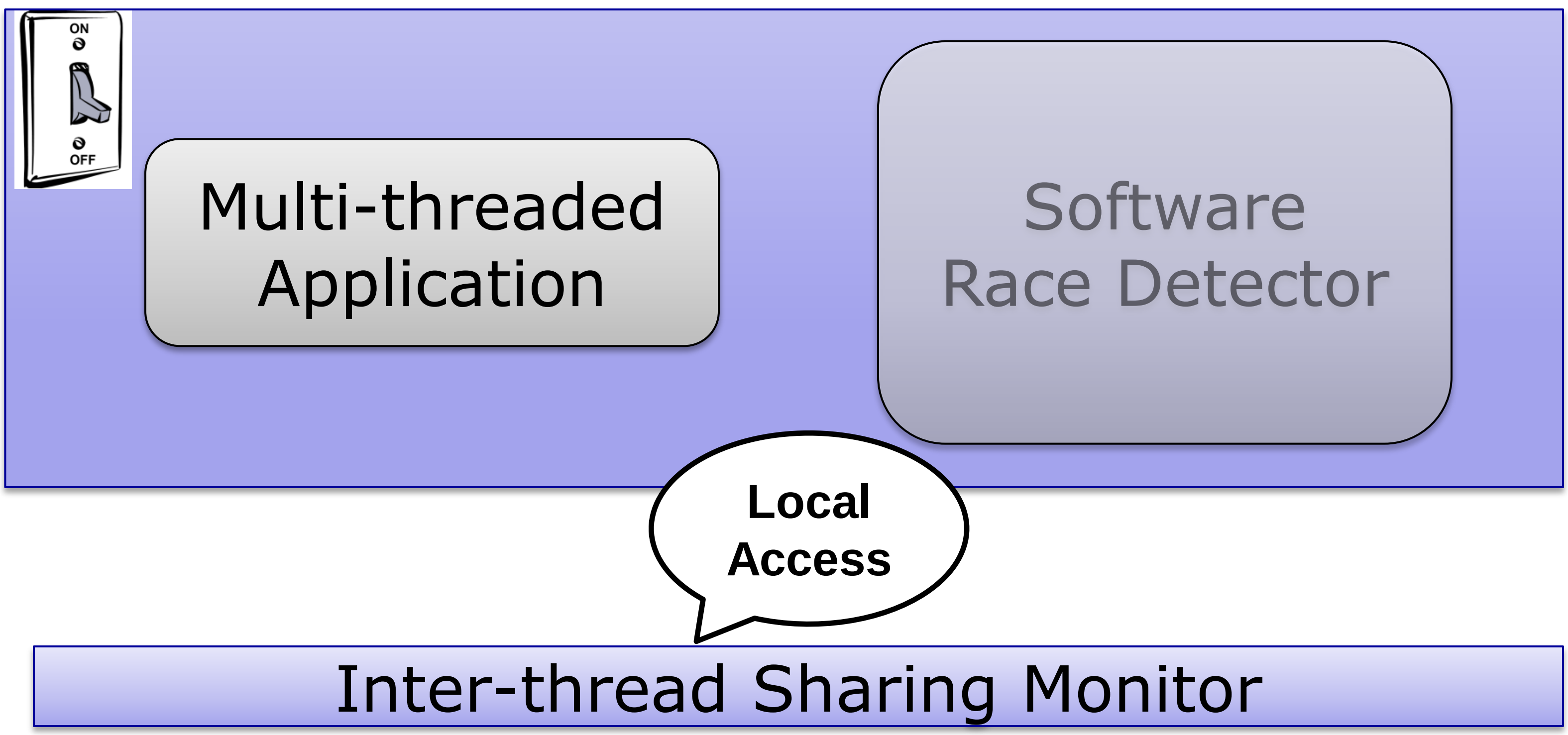
Very Little Dynamic Sharing



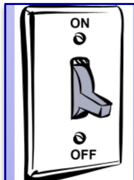
Run the Analysis On Demand



Run the Analysis On Demand



Run the Analysis On Demand



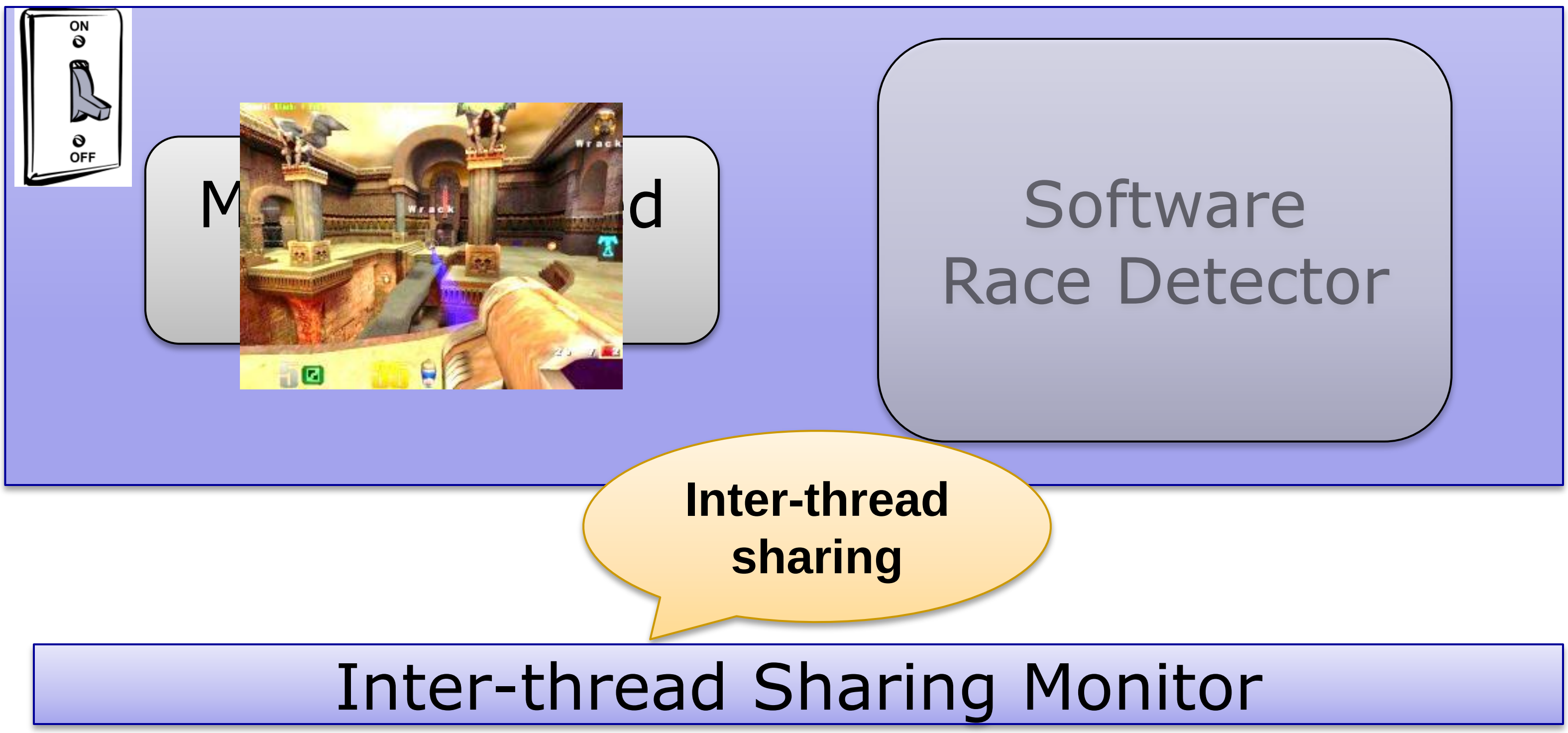
M

d

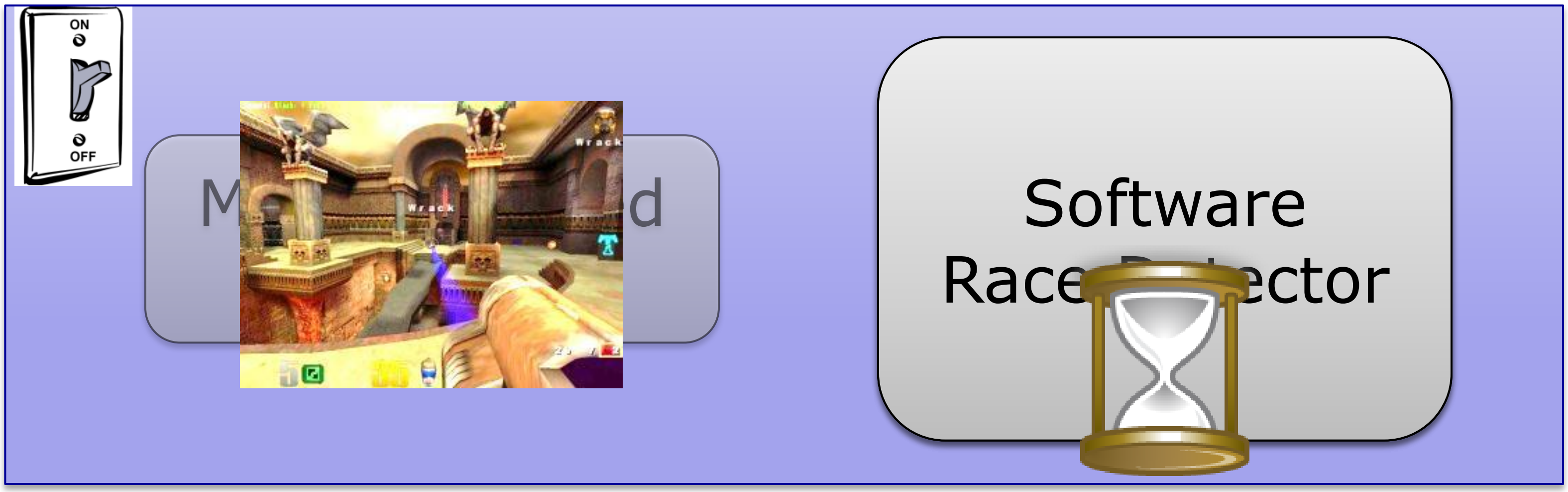
Software Race Detector

Inter-thread Sharing Monitor

Run the Analysis On Demand

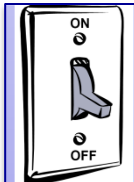


Run the Analysis On Demand



Inter-thread Sharing Monitor

Run the Analysis On Demand



M

d

Software
Race Detector

Inter-thread Sharing Monitor

Finding Inter-thread Sharing

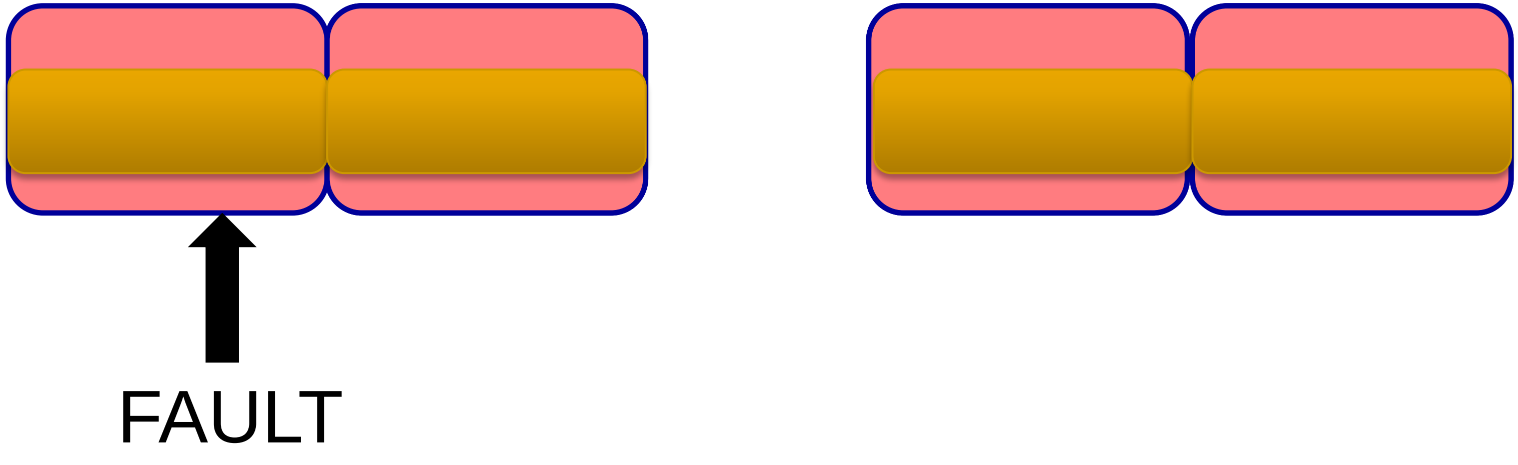
- Virtual Memory Watchpoints?

Finding Inter-thread Sharing

- Virtual Memory Watchpoints?

Finding Inter-thread Sharing

- Virtual Memory Watchpoints?



Finding Inter-thread Sharing

- Virtual Memory Watchpoints?

Finding Inter-thread Sharing

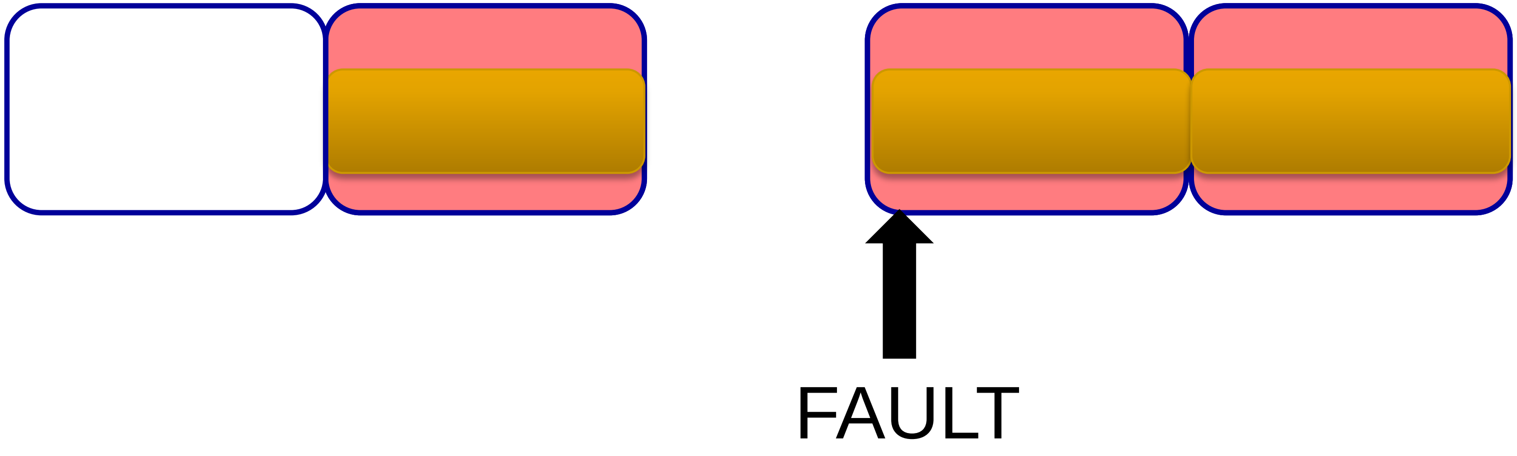
- Virtual Memory Watchpoints?

Finding Inter-thread Sharing

- Virtual Memory Watchpoints?

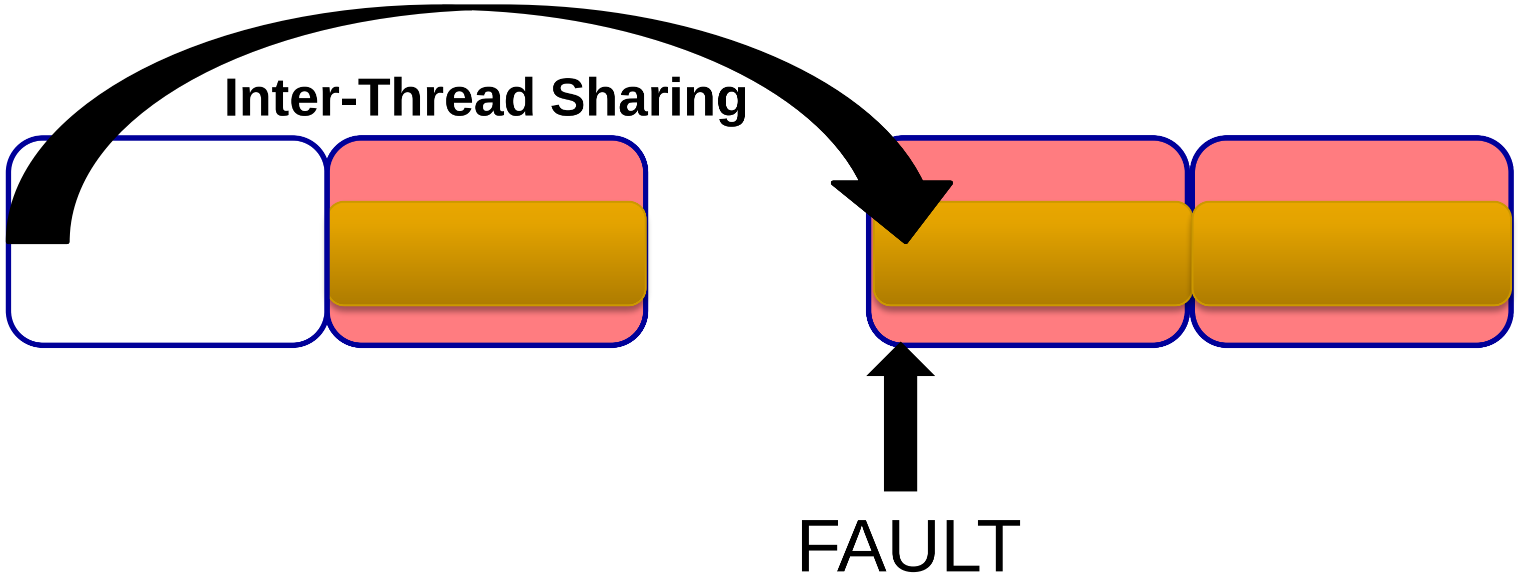
Finding Inter-thread Sharing

- Virtual Memory Watchpoints?



Finding Inter-thread Sharing

- Virtual Memory Watchpoints?



Finding Inter-thread Sharing

- Virtual Memory Watchpoints?

  – ~100% of accesses cause page faults

Finding Inter-thread Sharing

- Virtual Memory Watchpoints?

  – ~100% of accesses cause page faults

- Granularity Gap

Finding Inter-thread Sharing

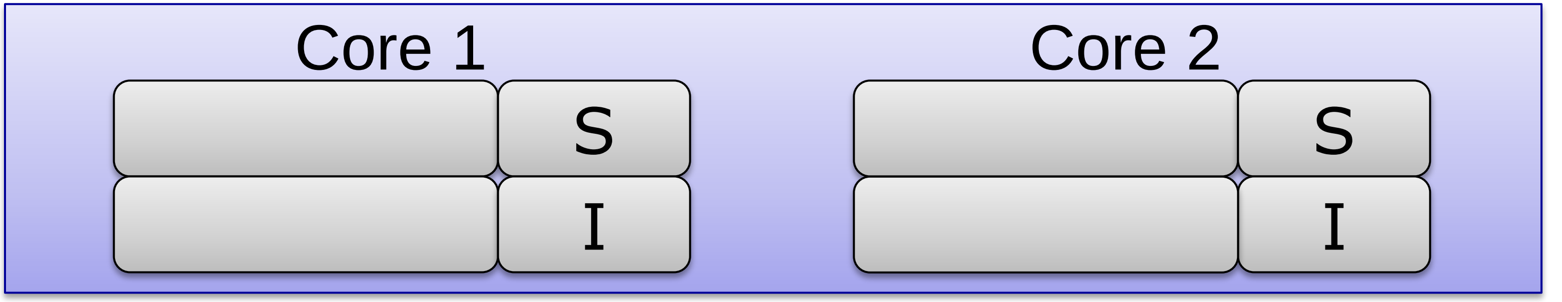
- Virtual Memory Watchpoints?

  - ~100% of accesses cause page faults

- Granularity Gap
- Per-process not per-thread

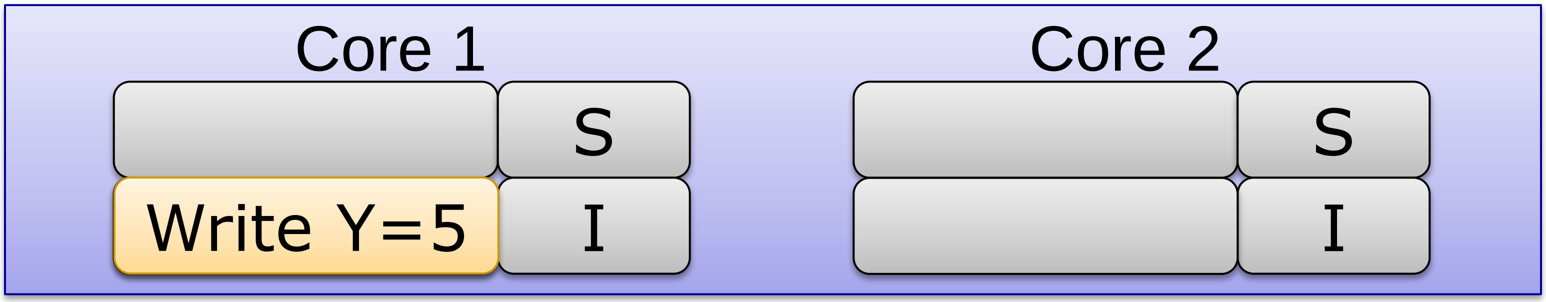
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing



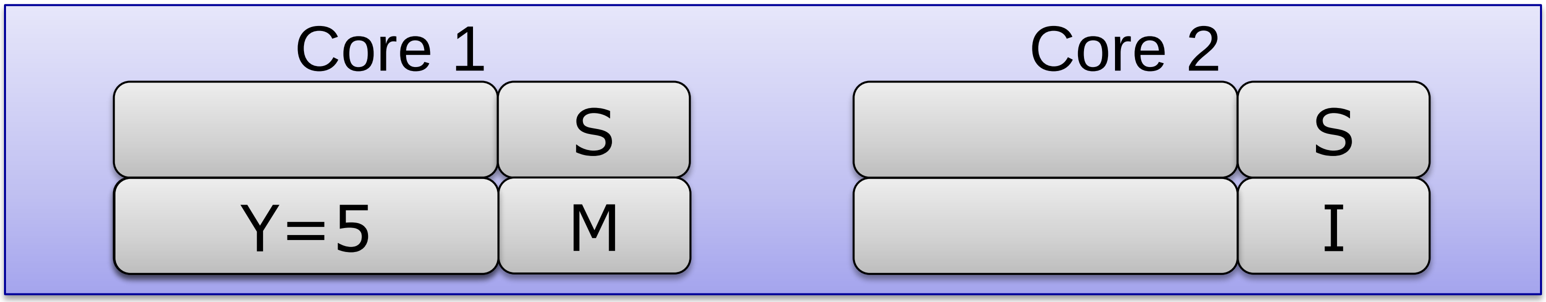
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing



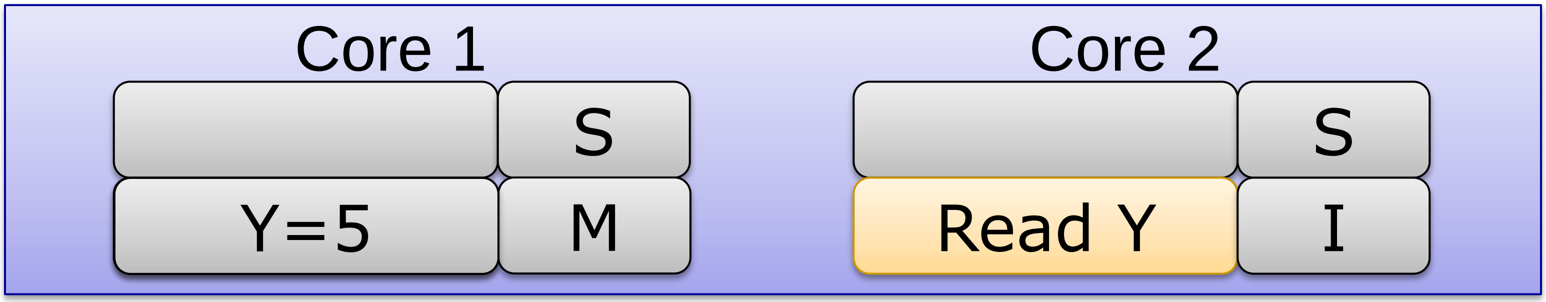
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing



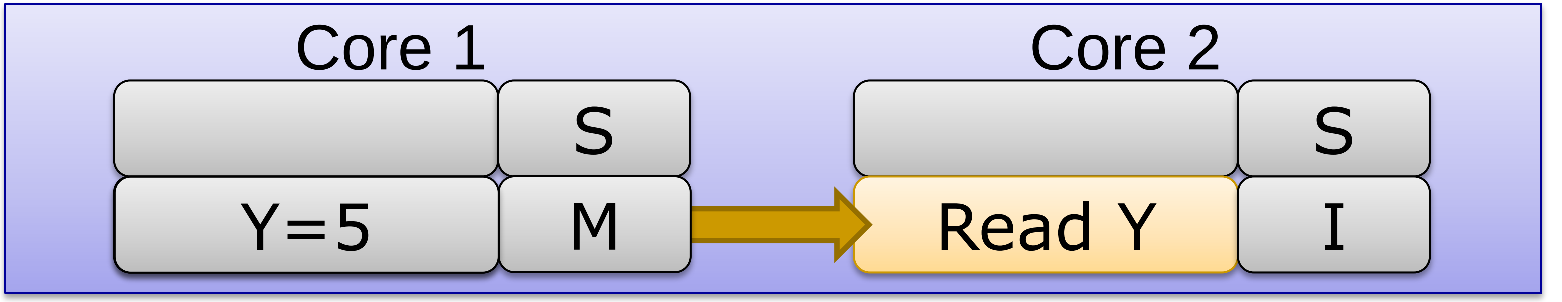
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing



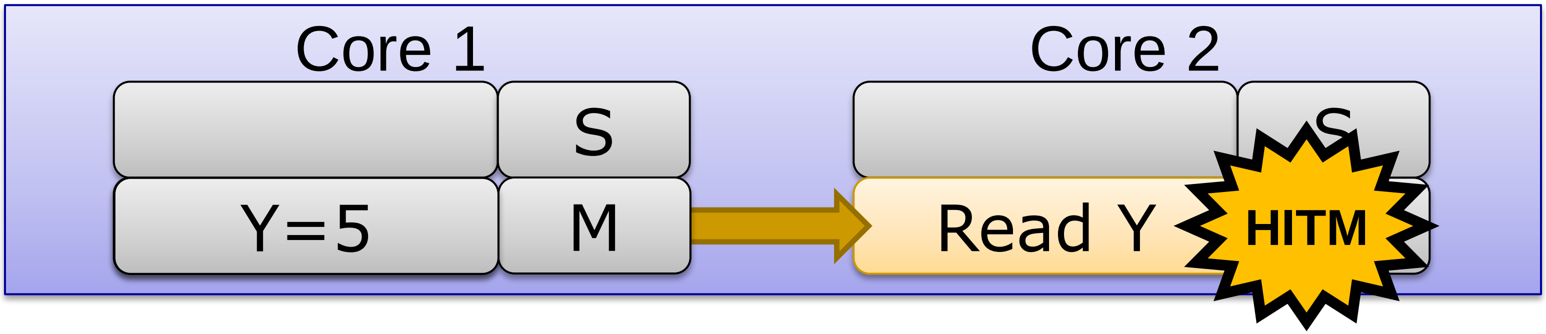
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing



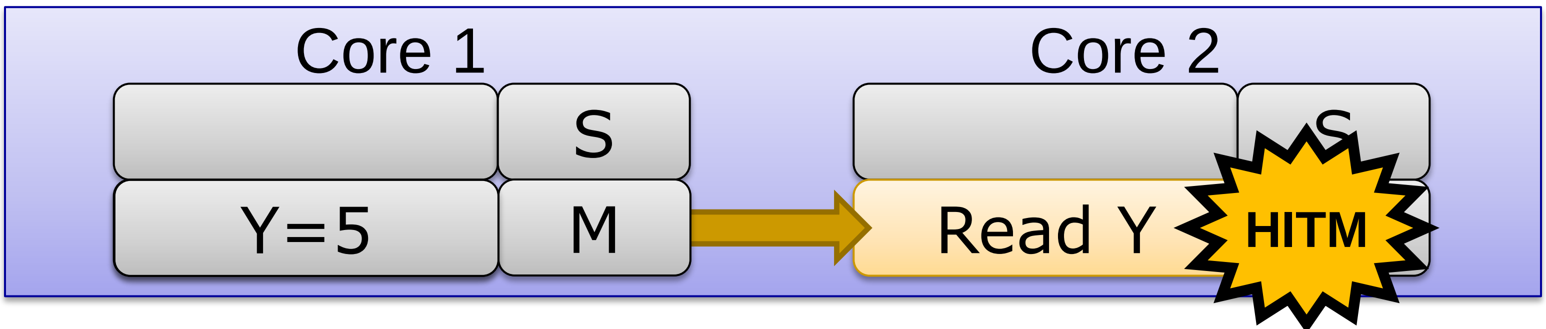
Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

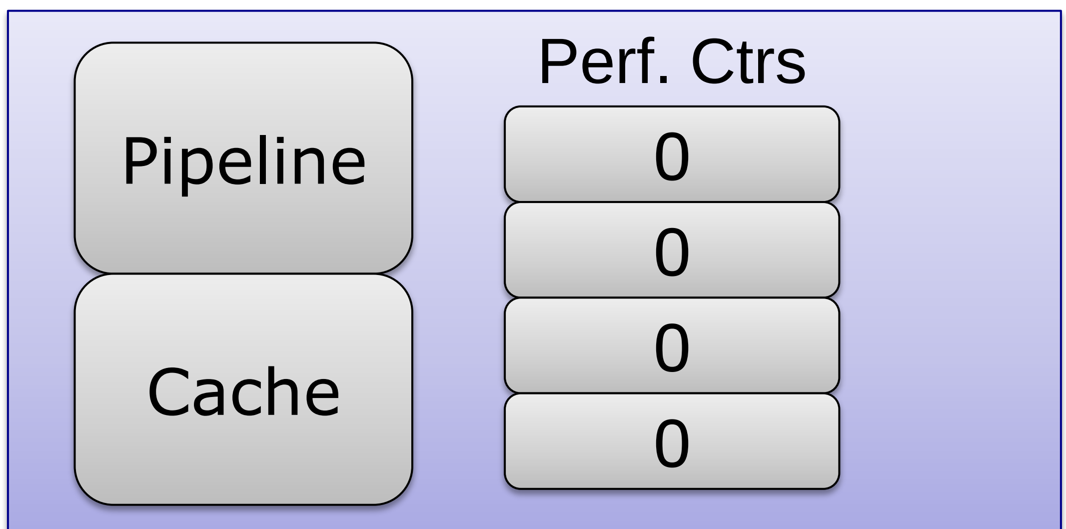


Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

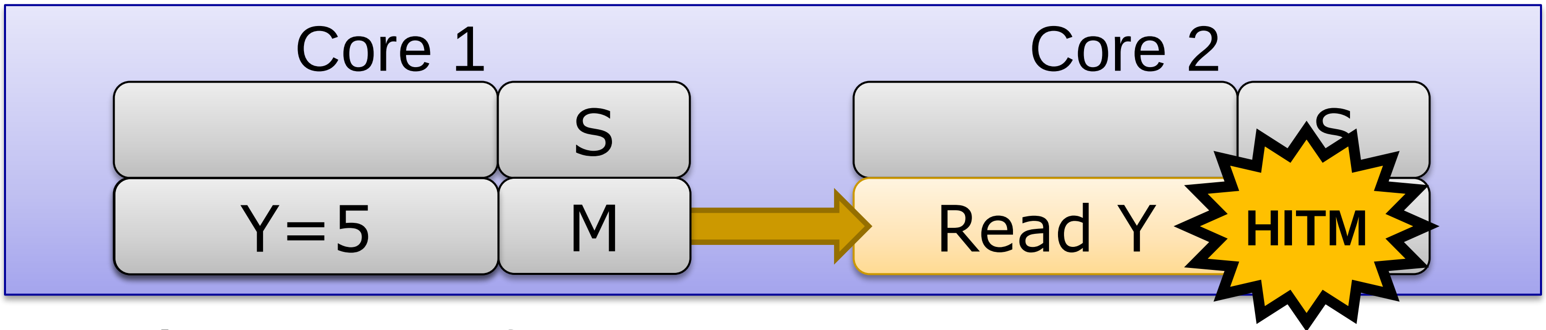


- Hardware Performance Counters

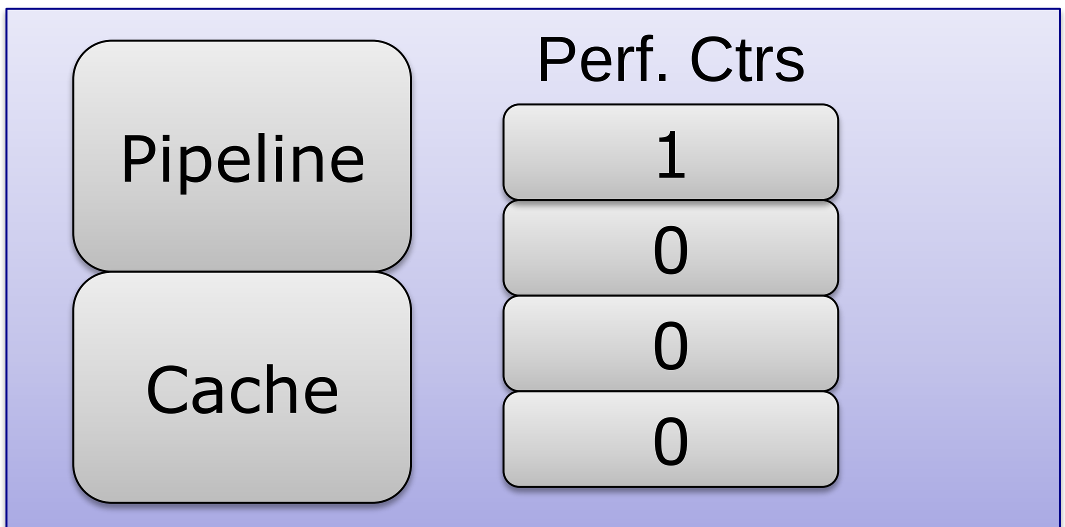


Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

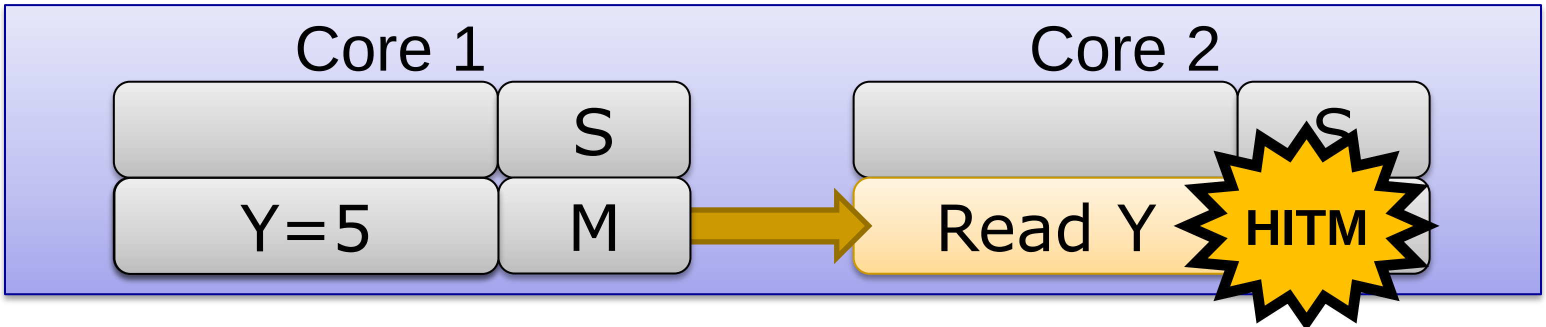


- Hardware Performance Counters

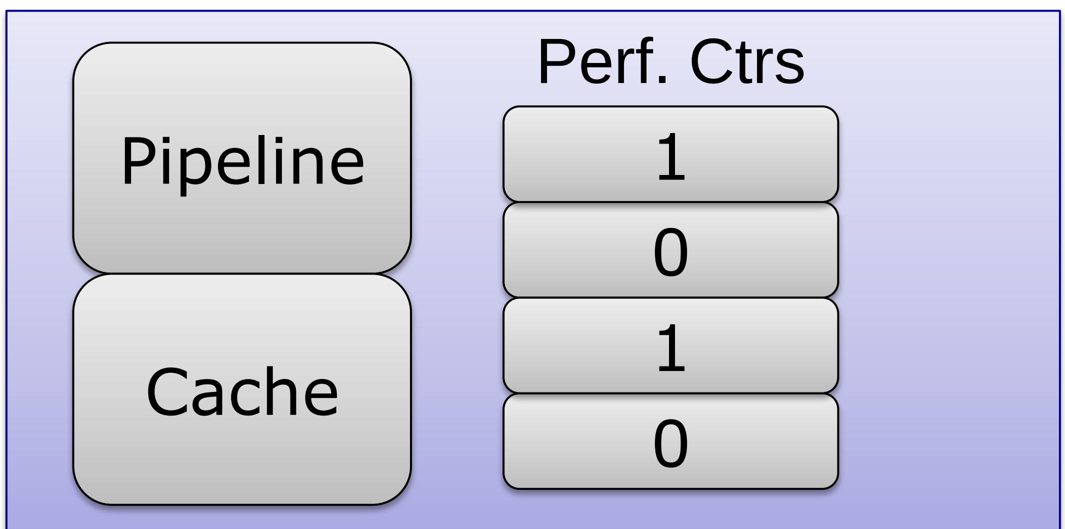


Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

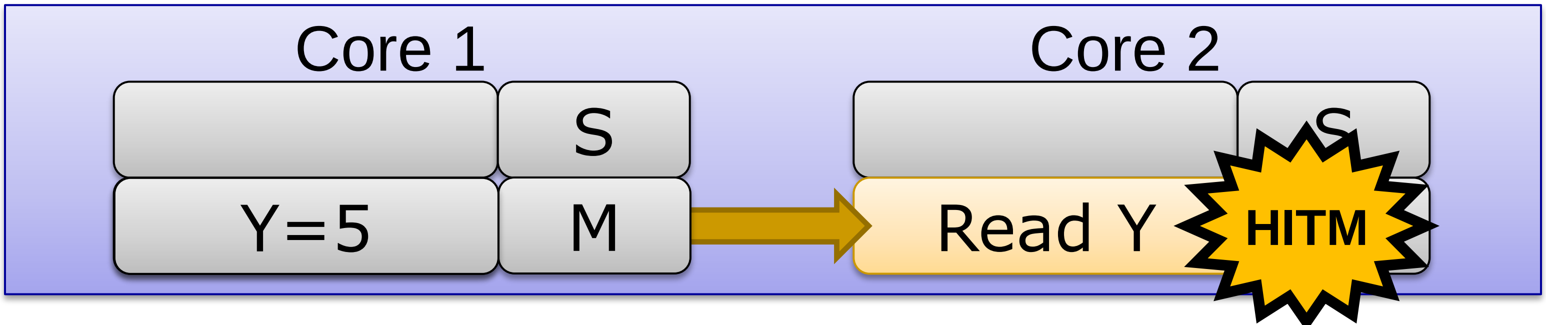


- Hardware Performance Counters

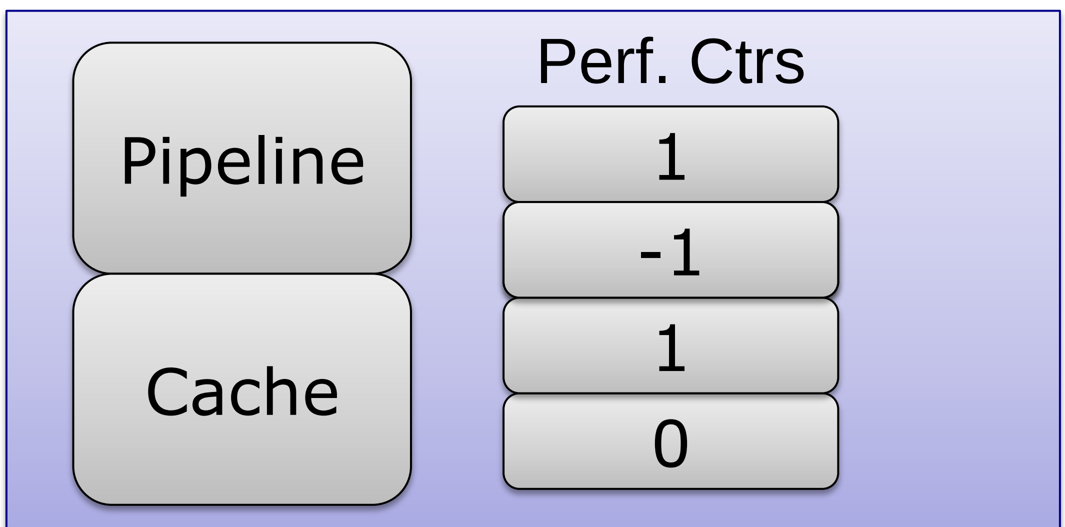


Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

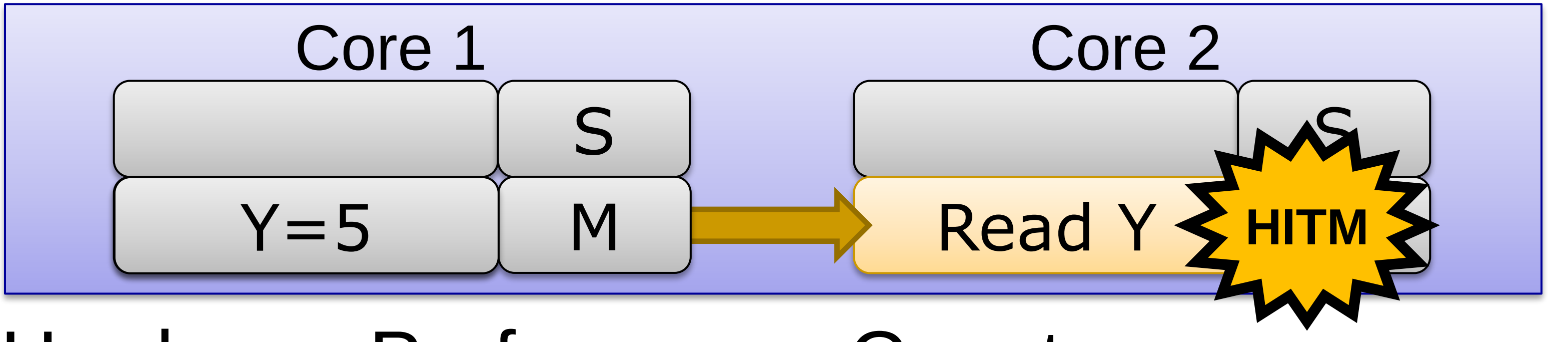


- Hardware Performance Counters

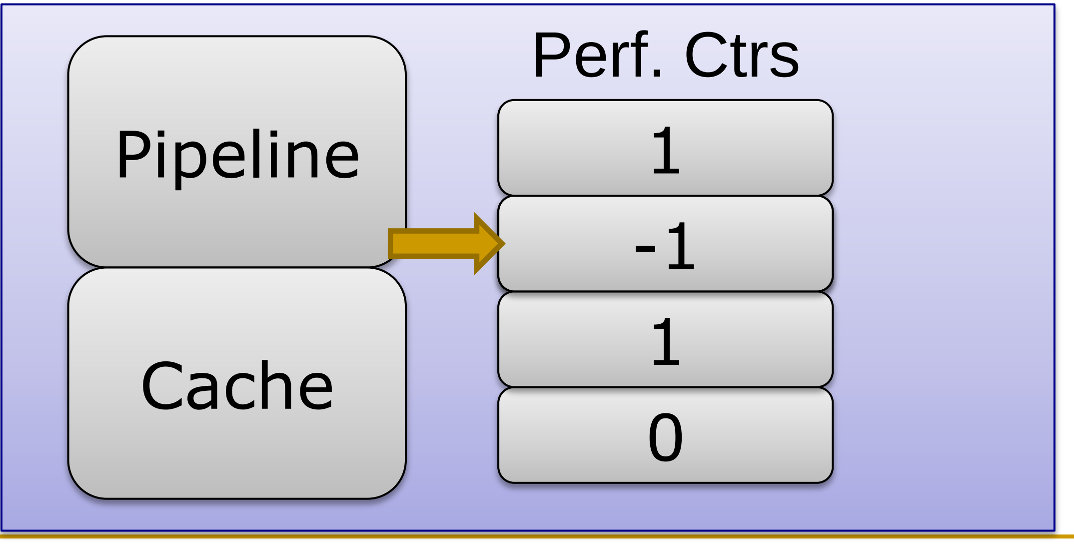


Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing

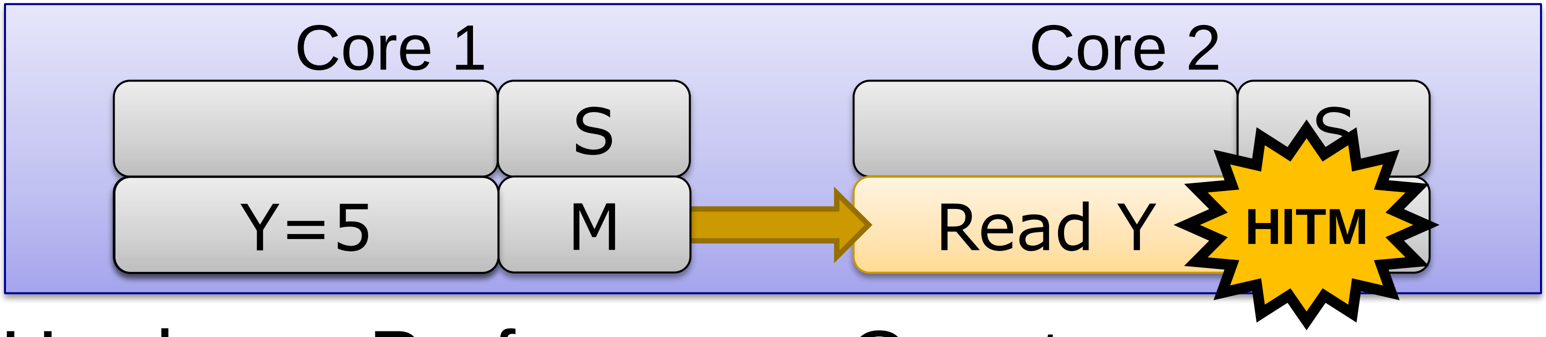


- Hardware Performance Counters

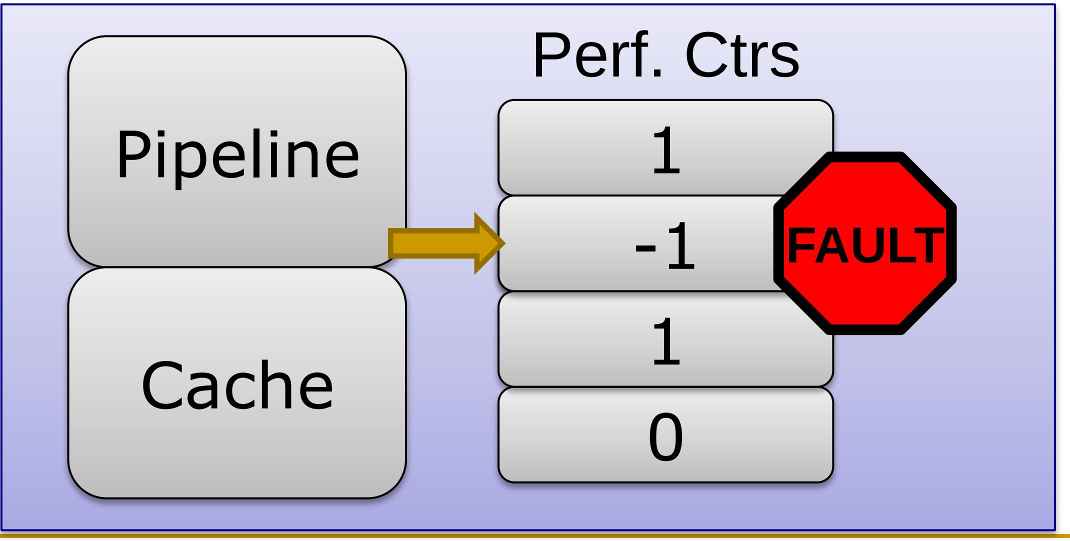


Hardware Sharing Detector

- HITM in Cache Memory: W→R Data Sharing



- Hardware Performance Counters



Potential Accuracy & Perf. Problems

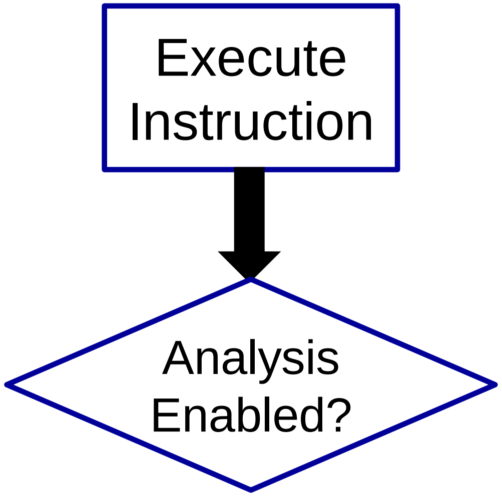
- Limitations of Performance Counters
  - Intel HITM only finds W→R Data Sharing
- Limitations of Cache Events
  - SMT sharing can't be counted
  - Cache eviction causes missed events
- Events go through the kernel

On-Demand Analysis on Real HW

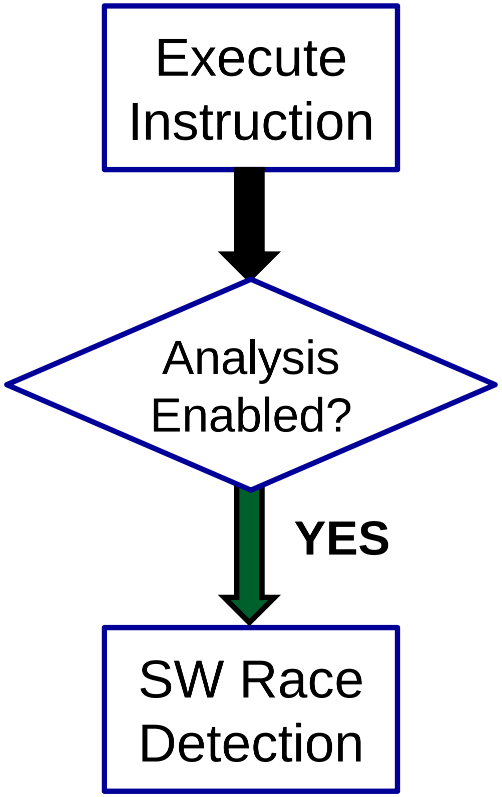
On-Demand Analysis on Real HW

Execute Instruction

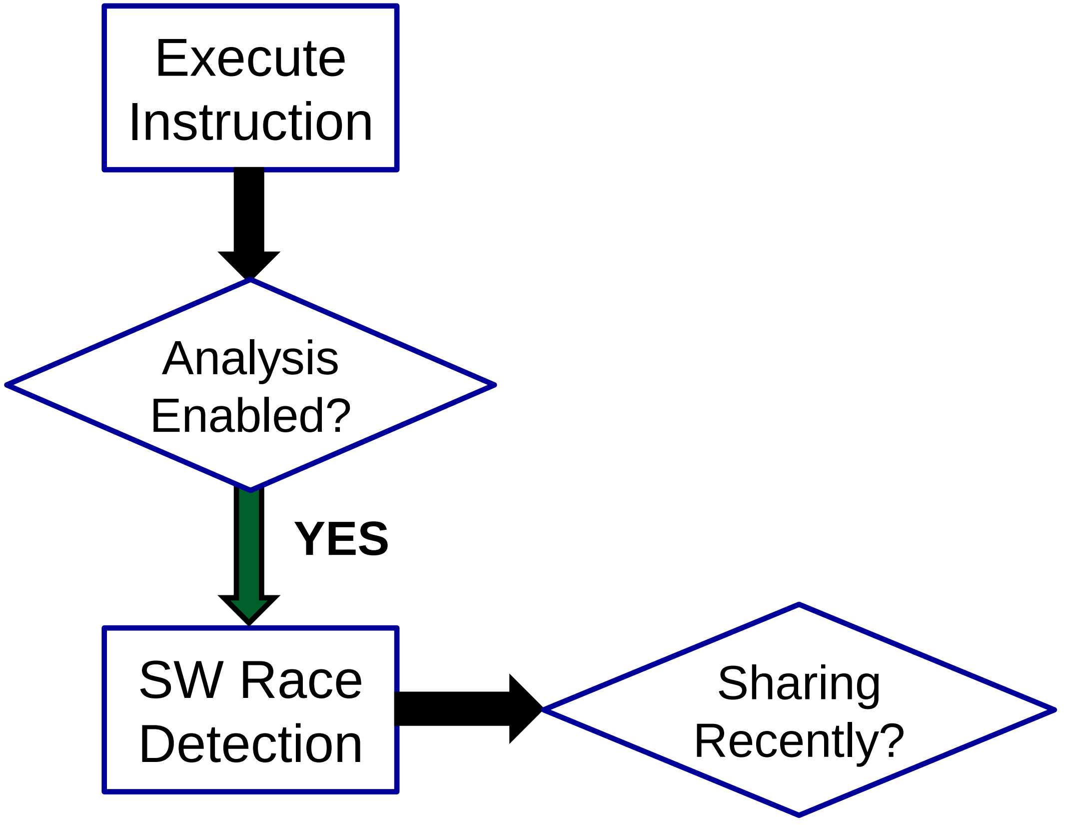
On-Demand Analysis on Real HW



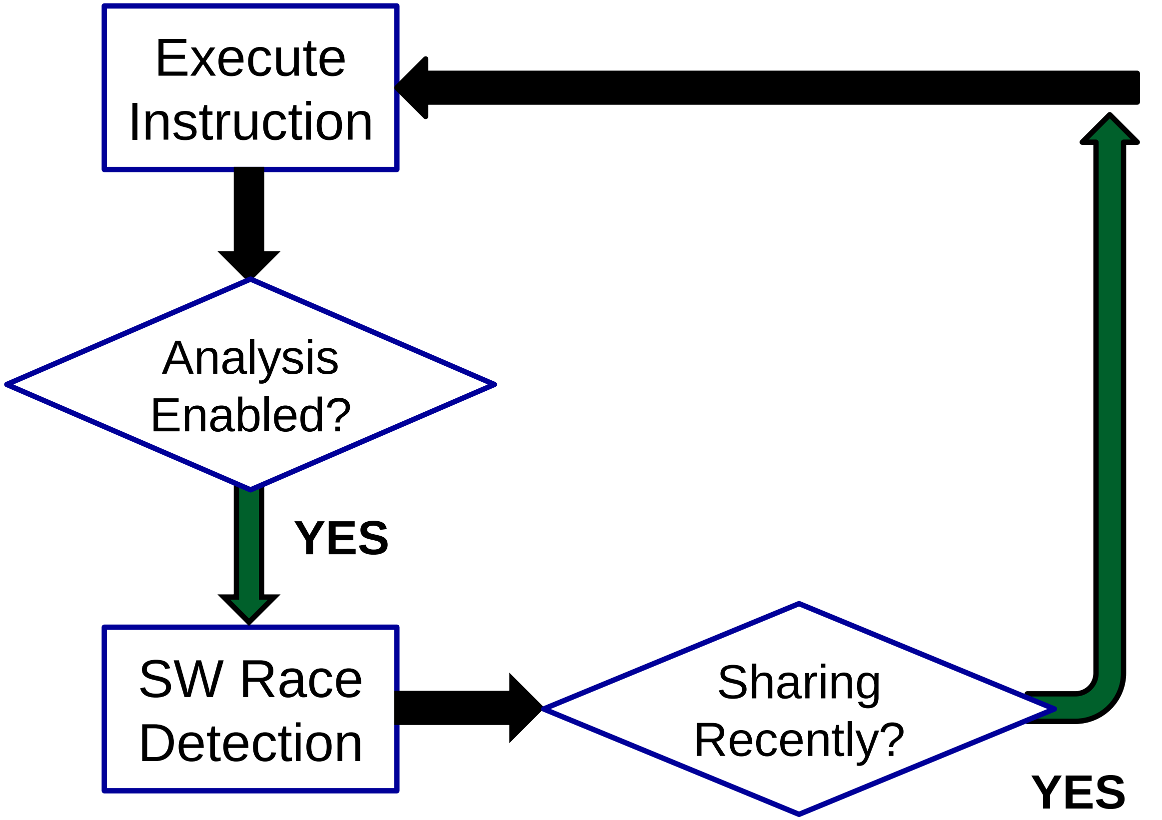
On-Demand Analysis on Real HW



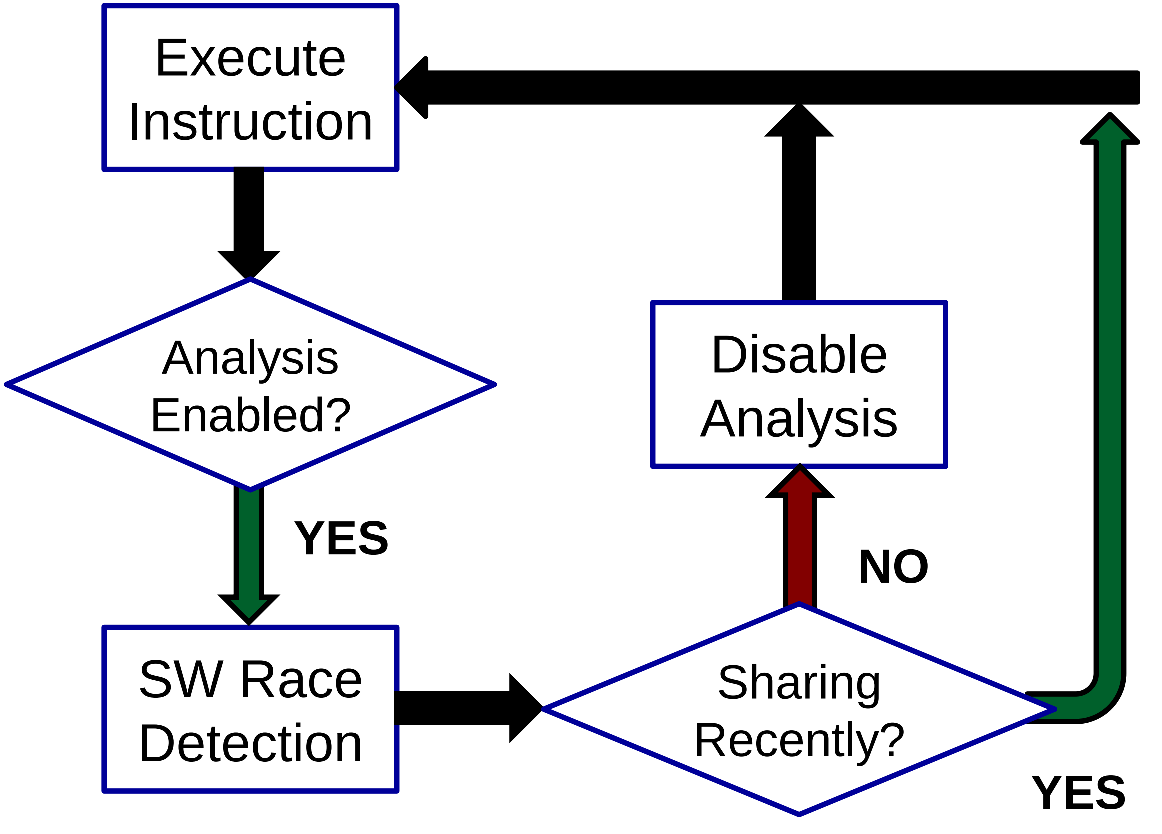
On-Demand Analysis on Real HW



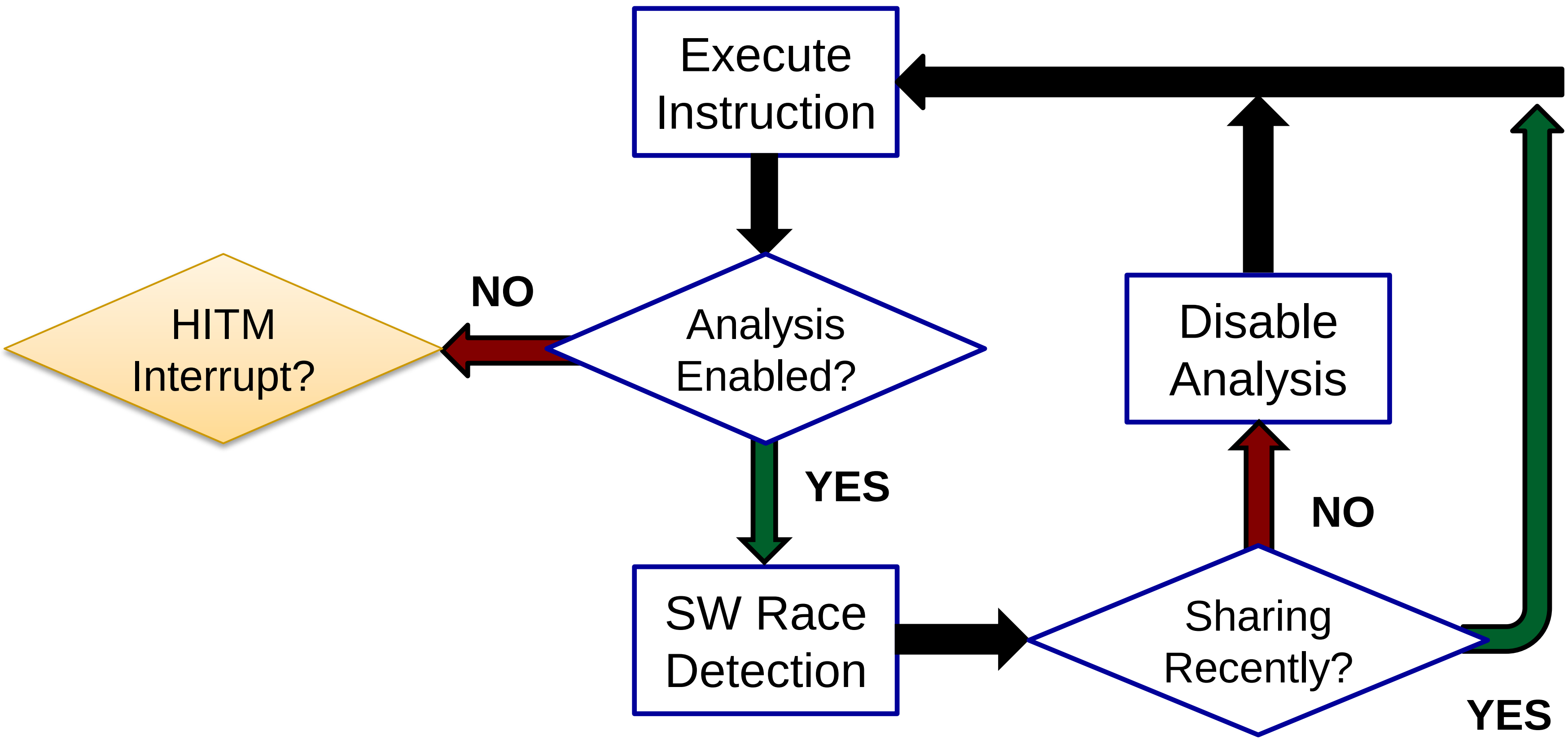
On-Demand Analysis on Real HW



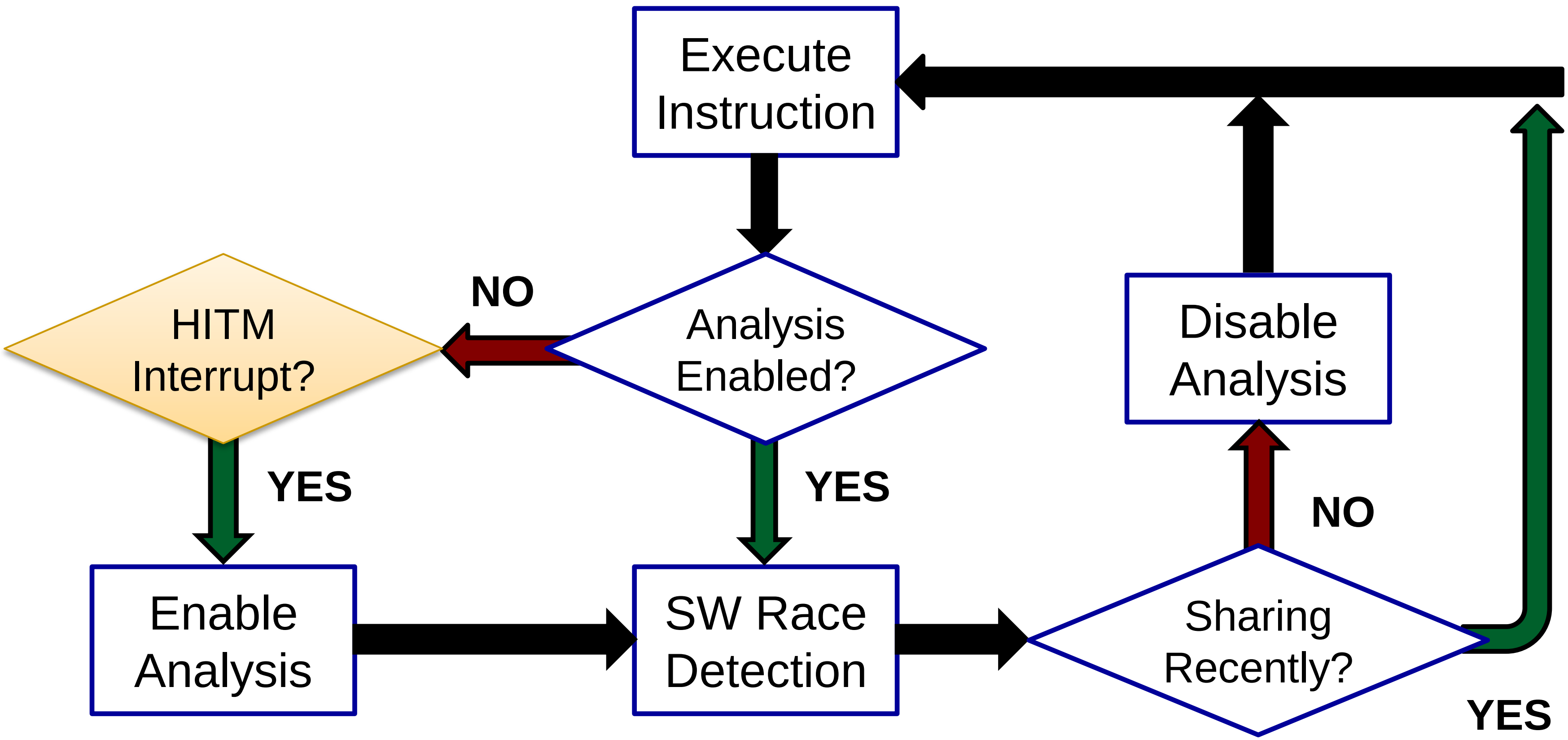
On-Demand Analysis on Real HW



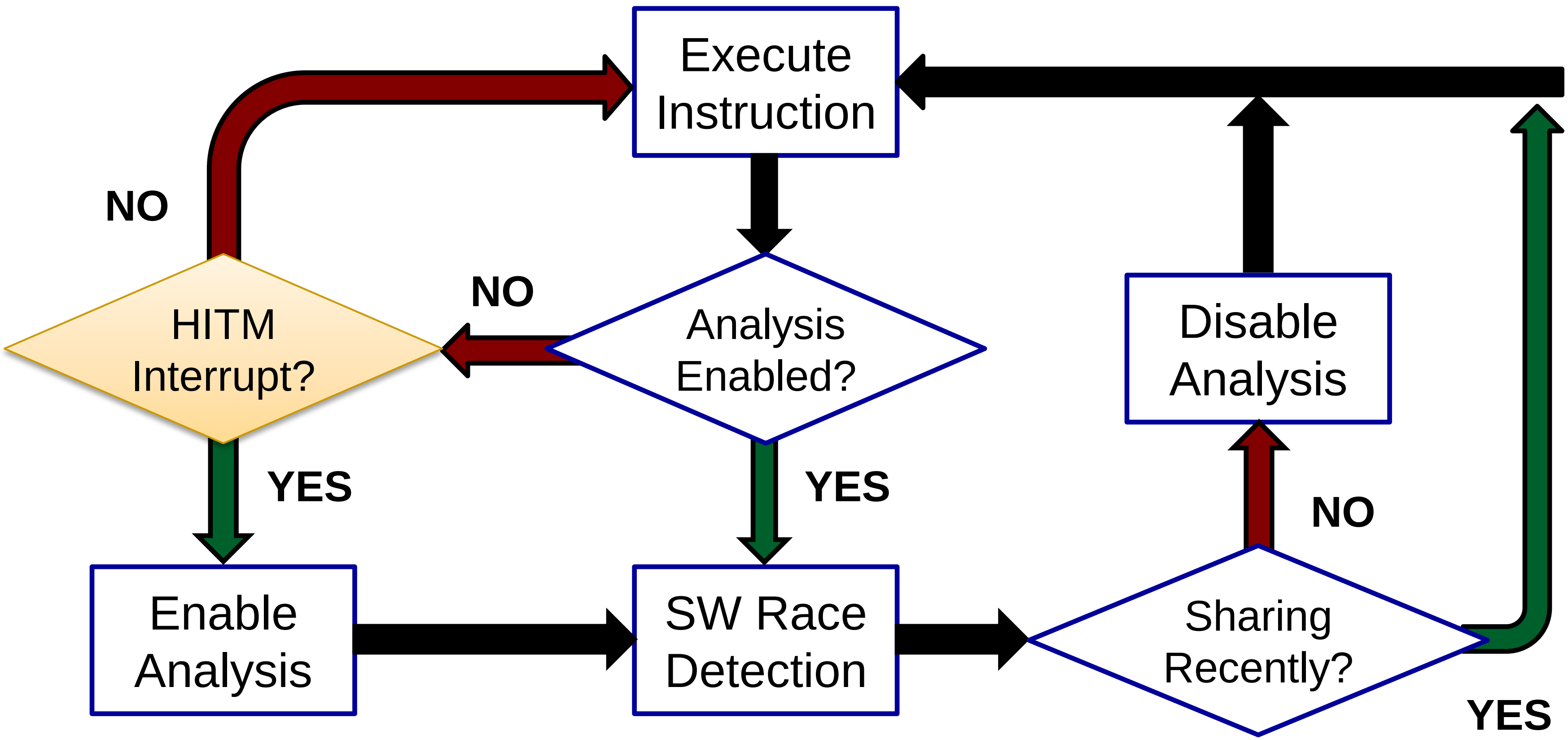
On-Demand Analysis on Real HW



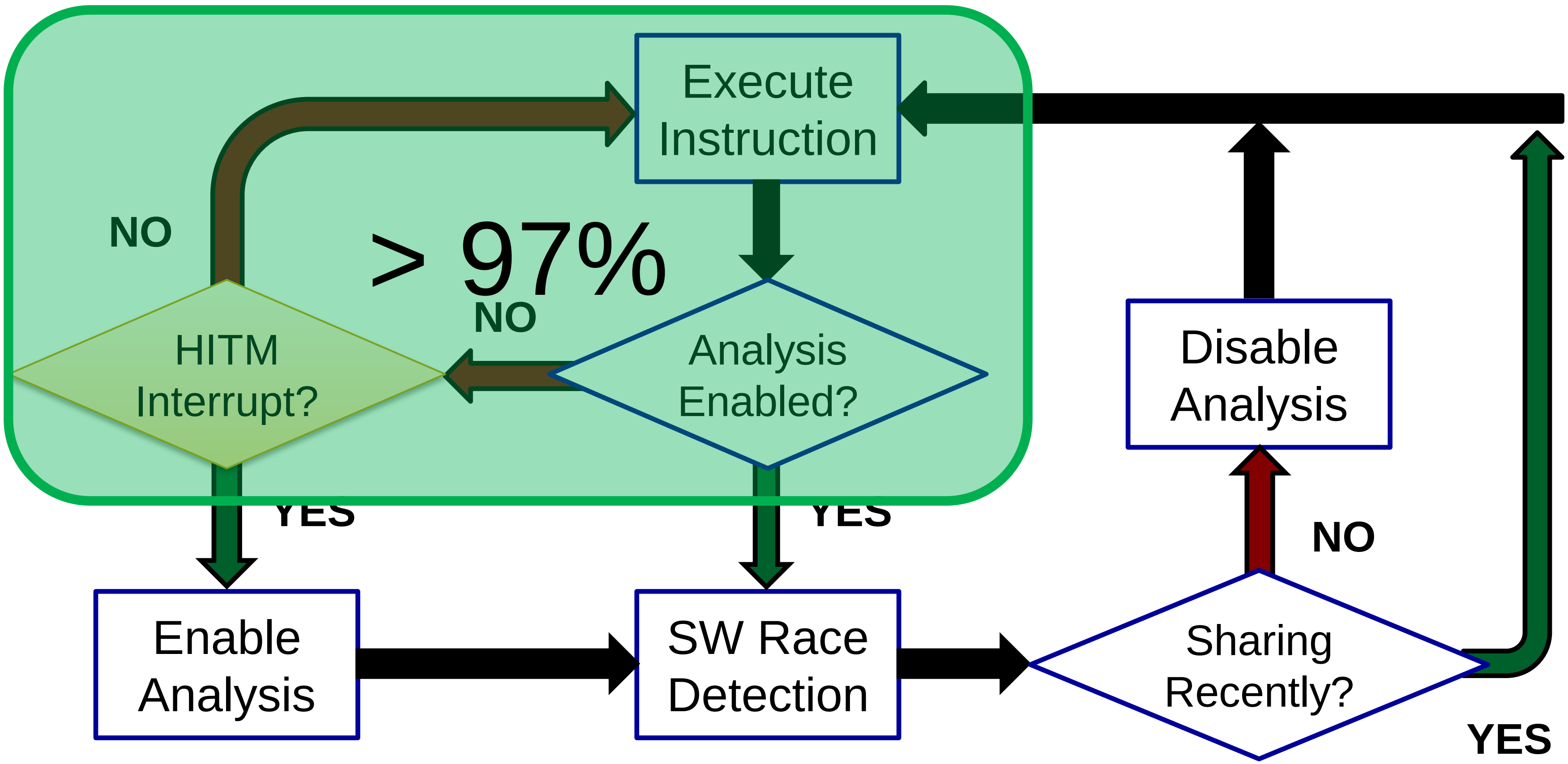
On-Demand Analysis on Real HW



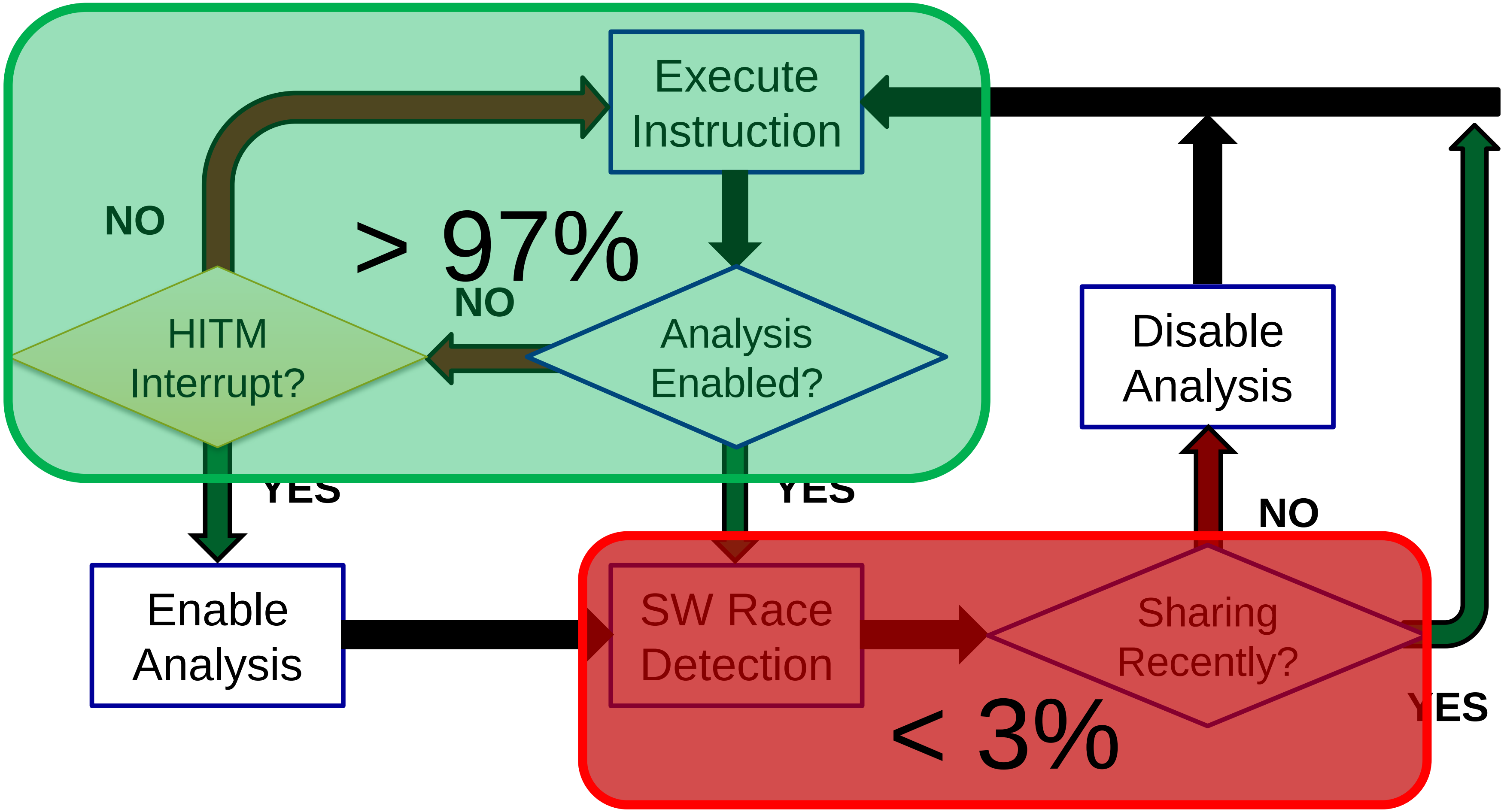
On-Demand Analysis on Real HW



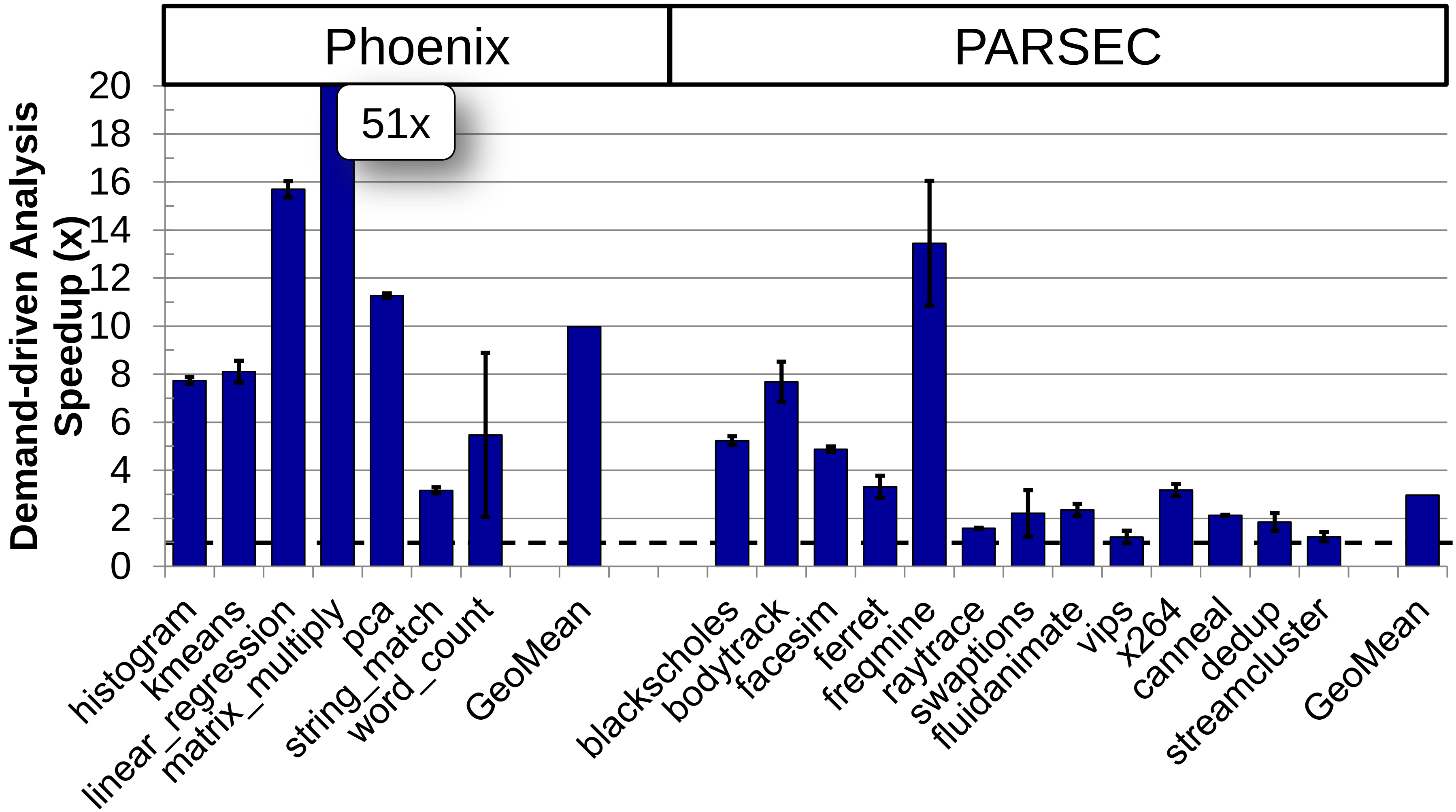
On-Demand Analysis on Real HW



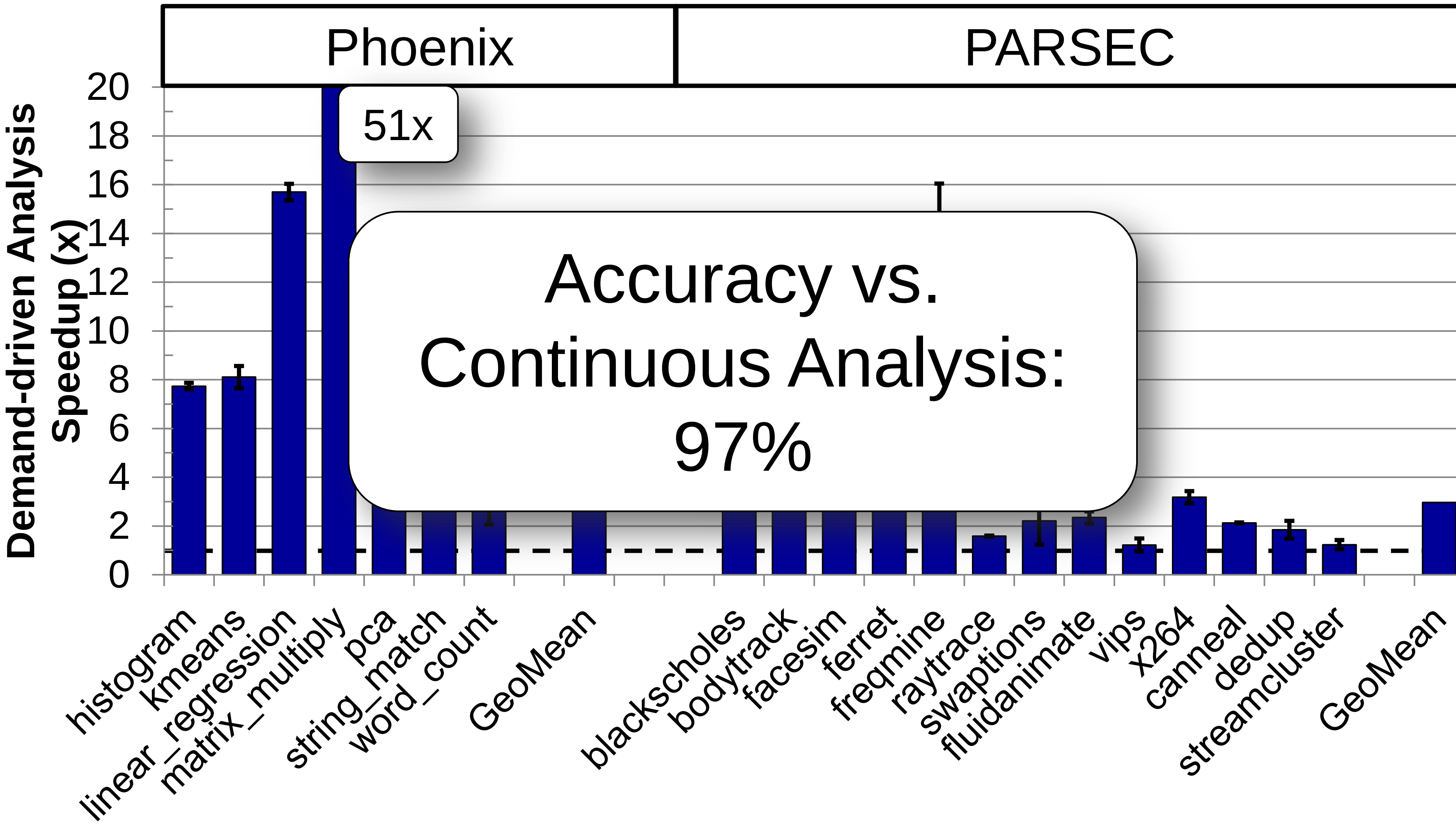
On-Demand Analysis on Real HW



Performance Increases



Performance Increases

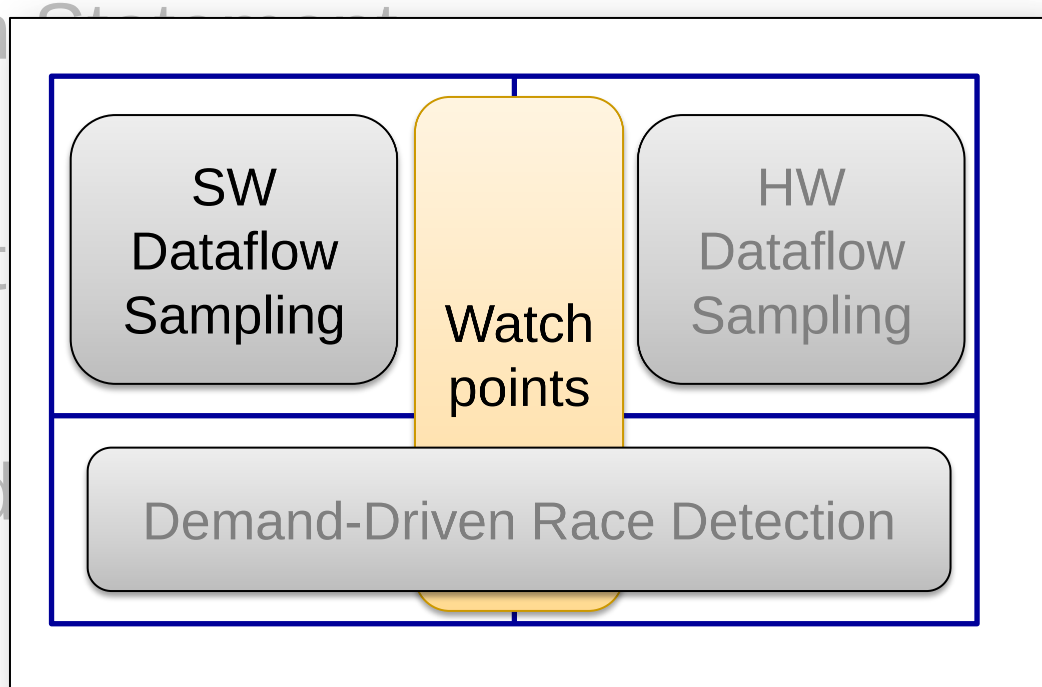


Outline

- Problem Statement
- Distributed Dynamic Dataflow Analysis
- Demand-Driven Data Race Detection
- Unlimited Watchpoints

Outline

- Problem
- Distribut... s
- Demand
- Unlimited Watchpoints



Watchpoints Work for Many Analyses

Bounds Checking

Data Race Detection

Taint Analysis

Deterministic Execution

Transactional Memory

Speculative Parallelization

Watchpoints Work for Many Analyses

Bounds Checking

Data Race Detection

Taint Analysis

Deterministic Execution

Transactional Memory

Speculative Parallelization

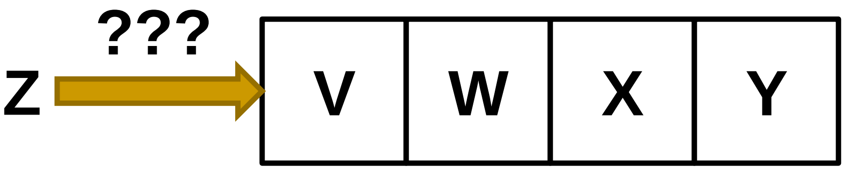
Desired Watchpoint Capabilities

- Large Number

V	**W**	**X**	**Y**

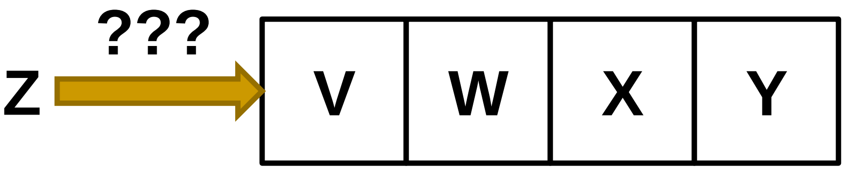
Desired Watchpoint Capabilities

- Large Number



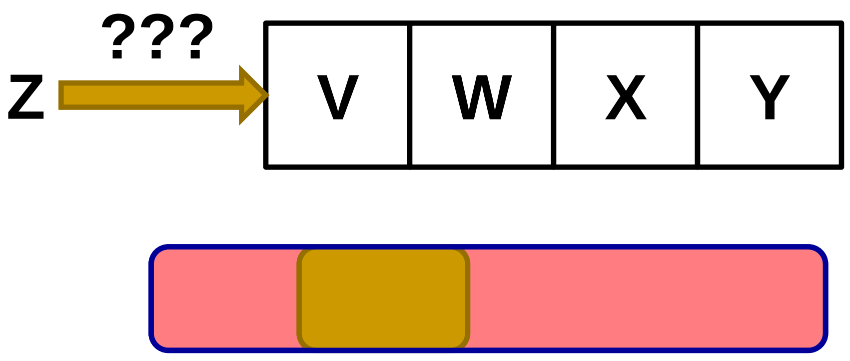
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip



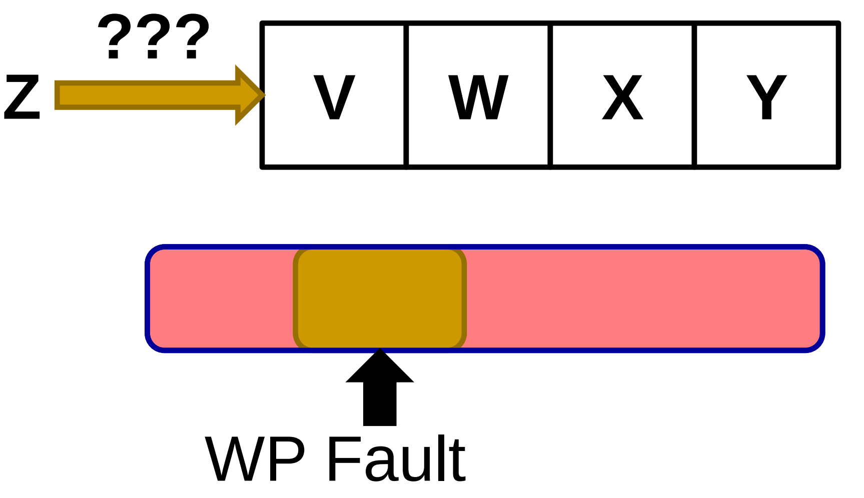
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained



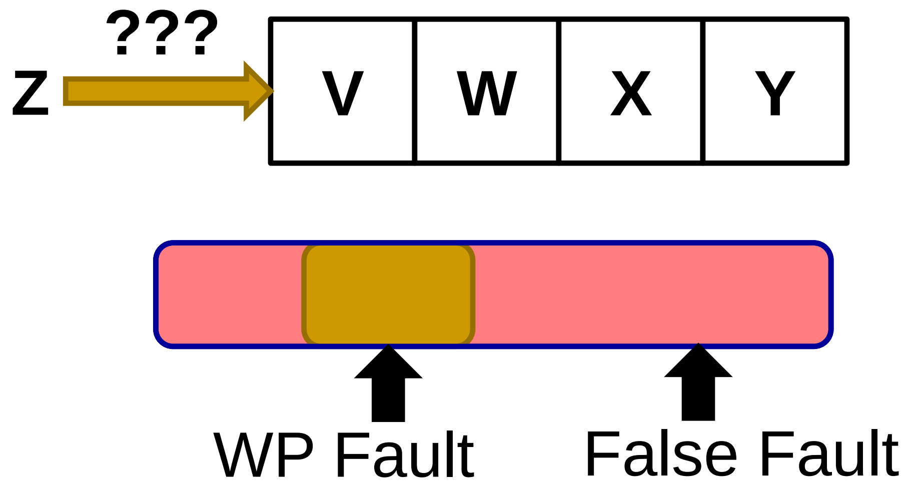
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained



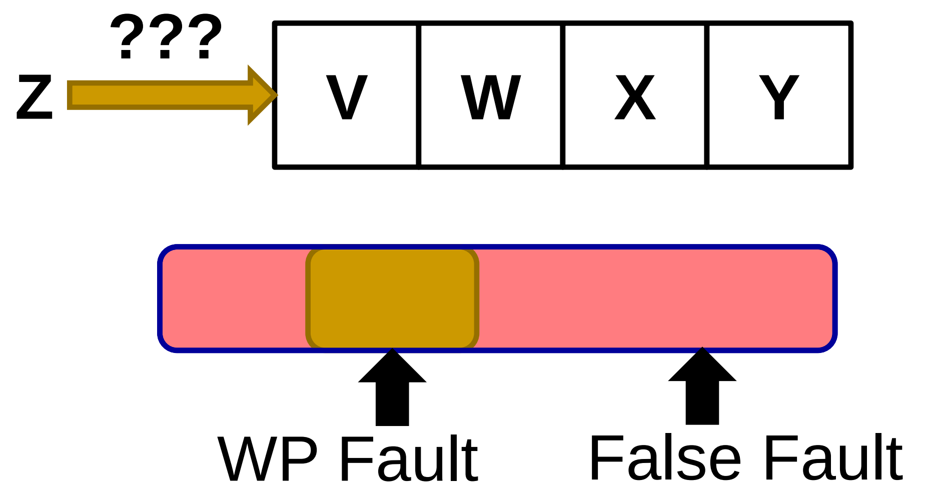
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained



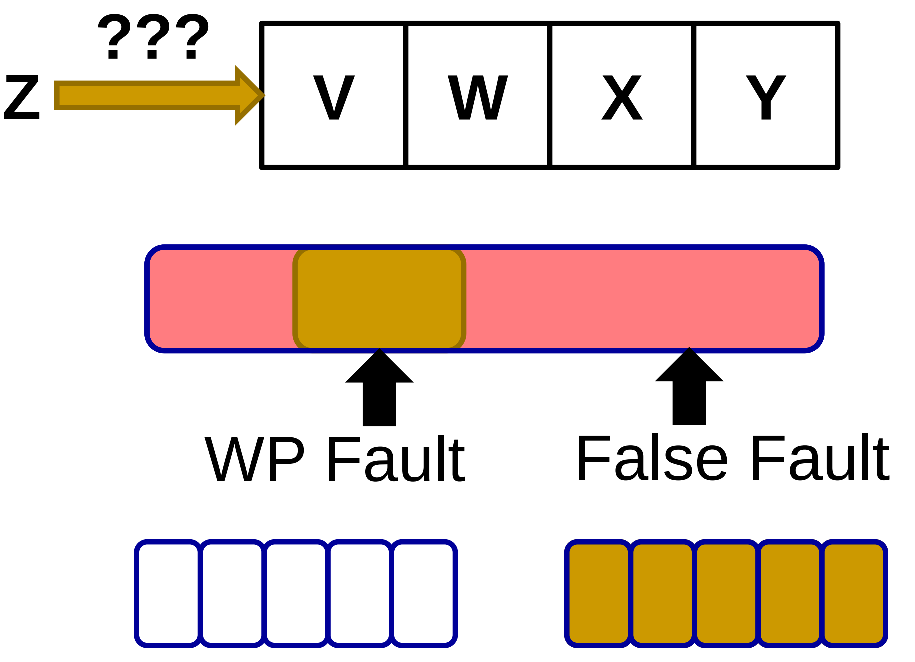
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA



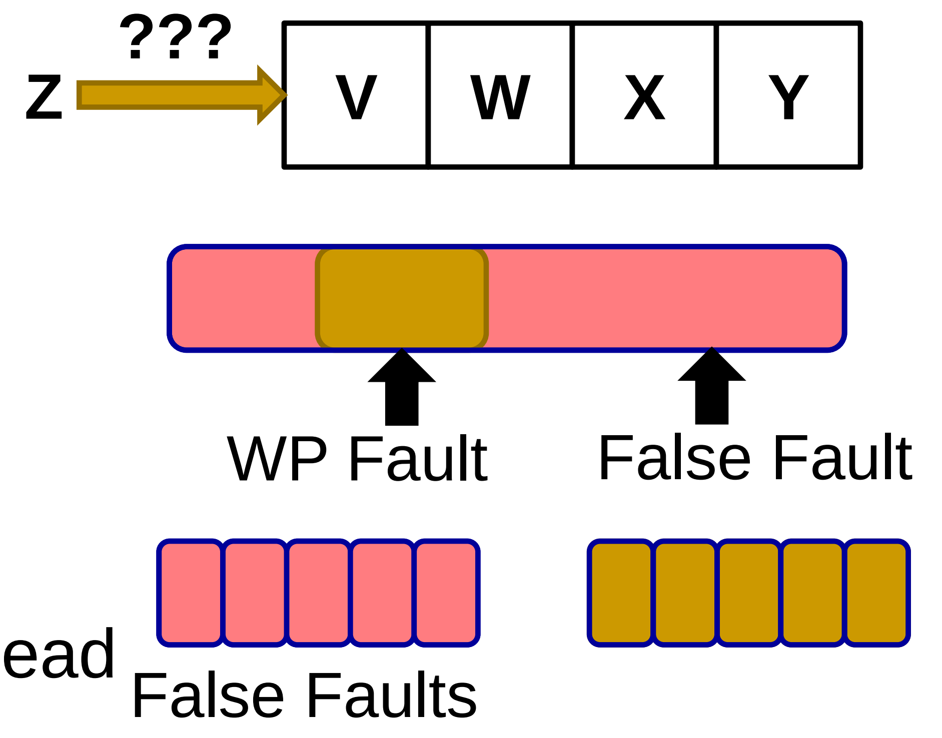
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread



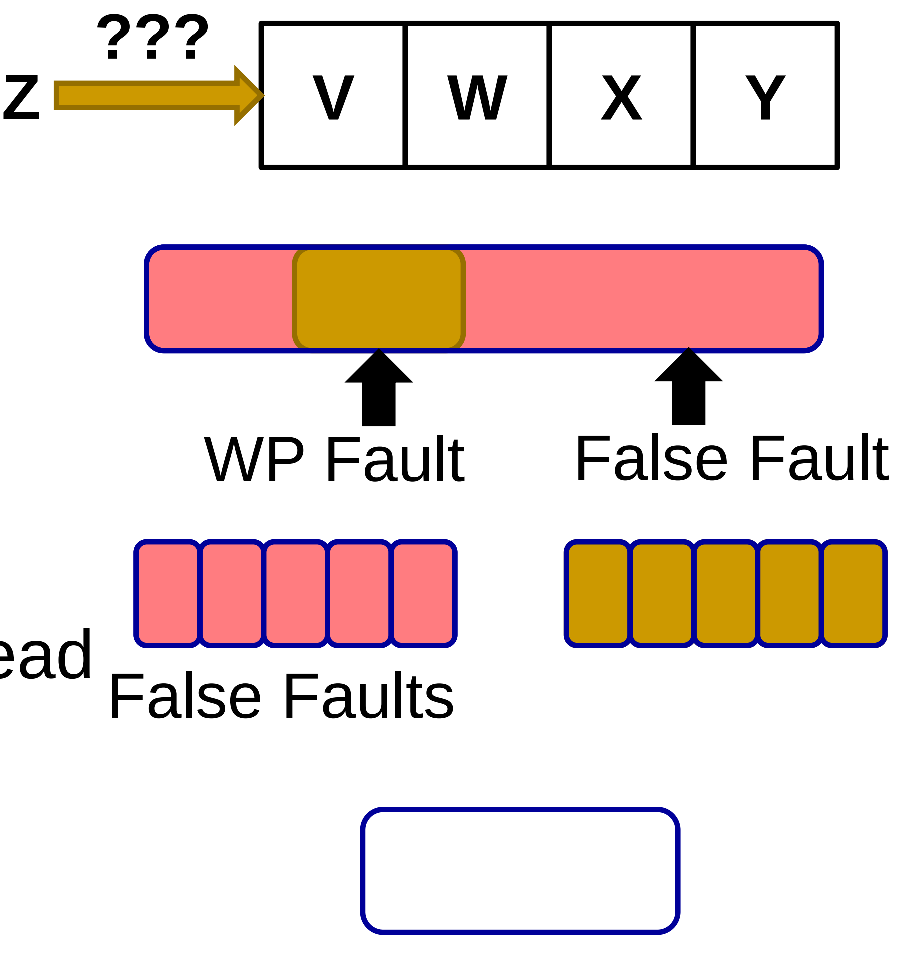
Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread



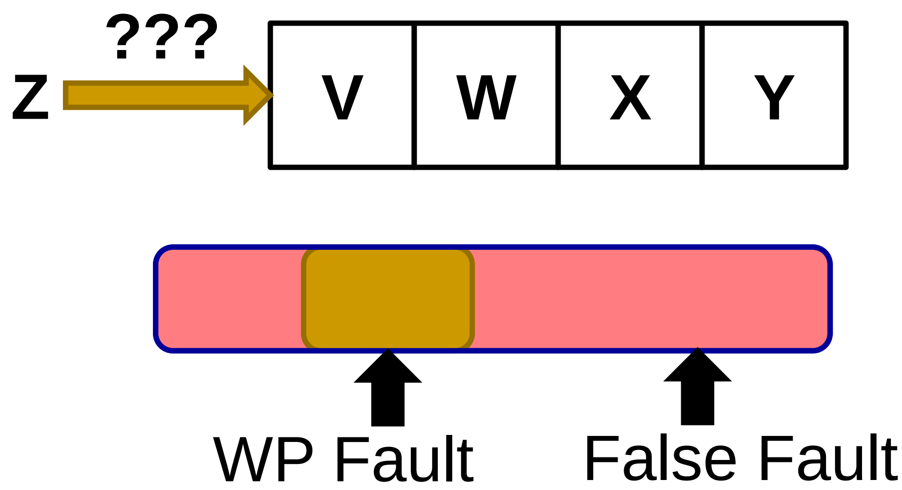
Desired Watchpoint Capabilities

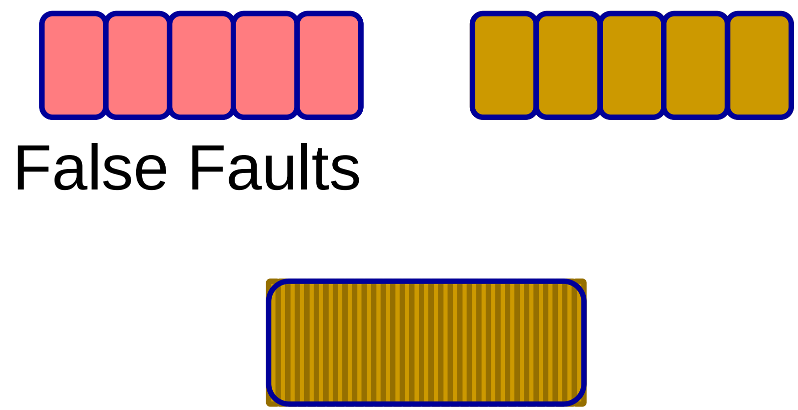
- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread
- Ranges



Desired Watchpoint Capabilities

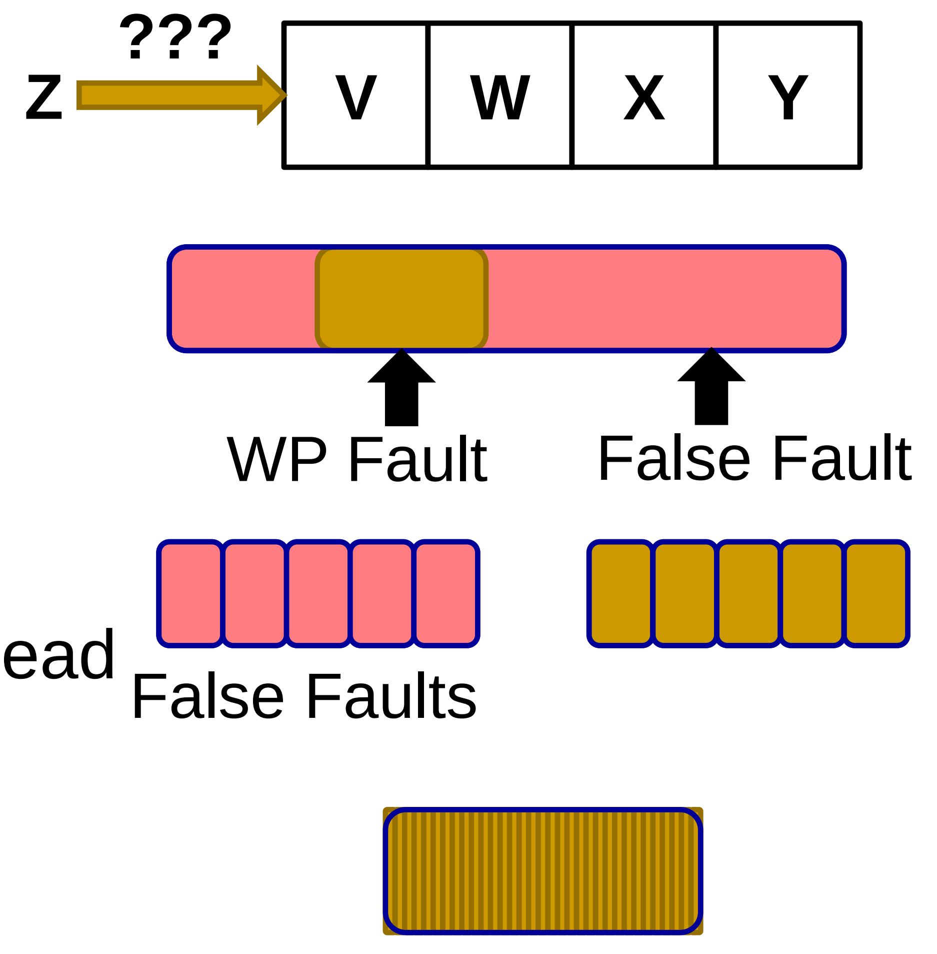
- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread
- Ranges





Desired Watchpoint Capabilities

- Large Number
  - Store in memory
  - Cache on chip
- Fine-grained
  - Watch full VA
- Per Thread
  - Cached per HW thread
- Ranges
  - Range Cache



Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0xffff_ffff	Not Watched	1
			0
			0

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0xffff_ffff	Not Watched	1
			0
			0

Set Addresses
0x5 – 0x2000
R-Watched

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
			0
			0

Set Addresses
0x5 – 0x2000
R-Watched

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
			0

Set Addresses
0x5 – 0x2000
R-Watched

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

Set Addresses
0x5 – 0x2000
R-Watched

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

Load Address
0x400

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

≤ 0x400? ≥ 0x400?

Load Address
0x400

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

≤ 0x400? ≥ 0x400?

Load Address
0x400

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

≤ 0x400?

≥ 0x400?

Load Address
0x400

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

$\leq$ 0x400?

$\geq$ 0x400?

Load Address
0x400

Range Cache

Start Address	End Address	Watchpoint?	Valid
0x0	0x4	Not Watched	1
0x5	0x2000	R Watched	1
0x2001	0xffff_ffff	Not Watched	1

≤ 0x400?

≥ 0x400?

WP Interrupt

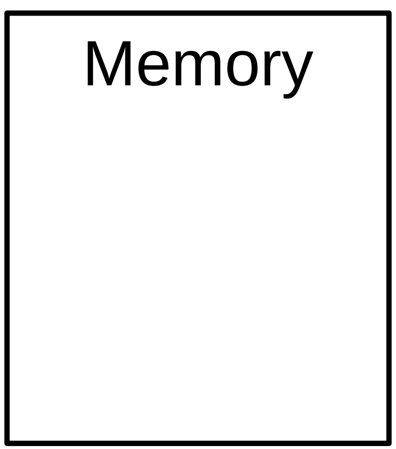
Load Address 0x400

Watchpoint System Design

- Store Ranges in Main Memory

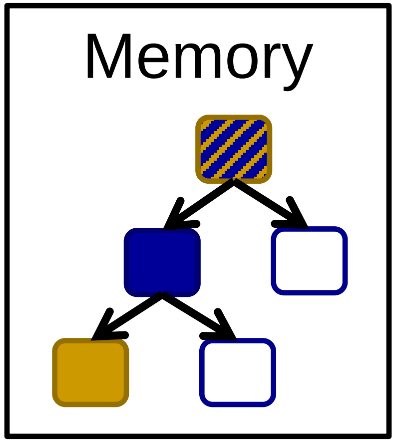
Watchpoint System Design

- Store Ranges in Main Memory



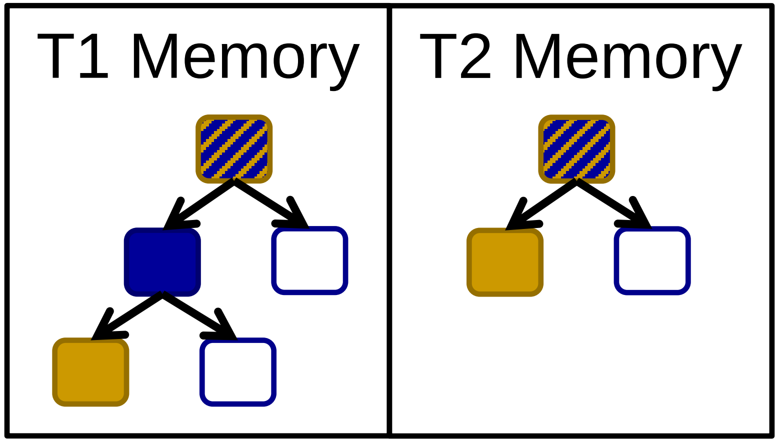
Watchpoint System Design

- Store Ranges in Main Memory



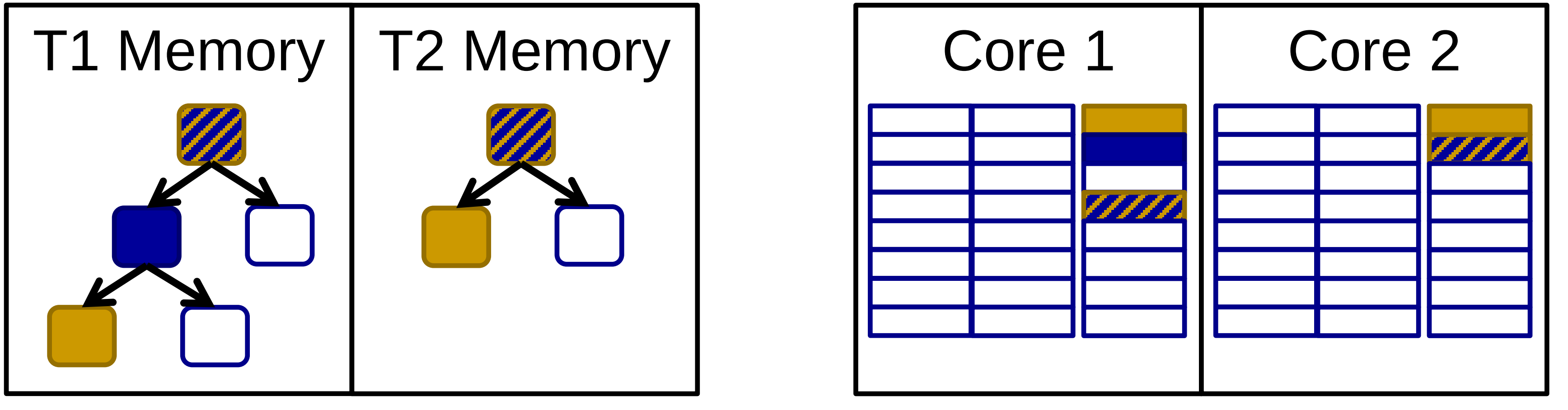
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache



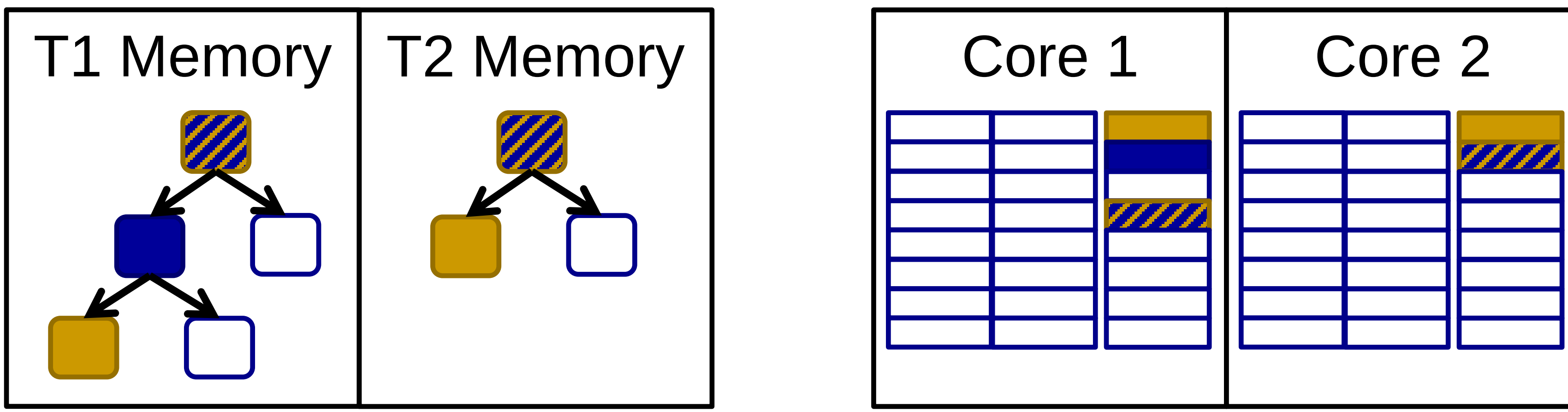
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache



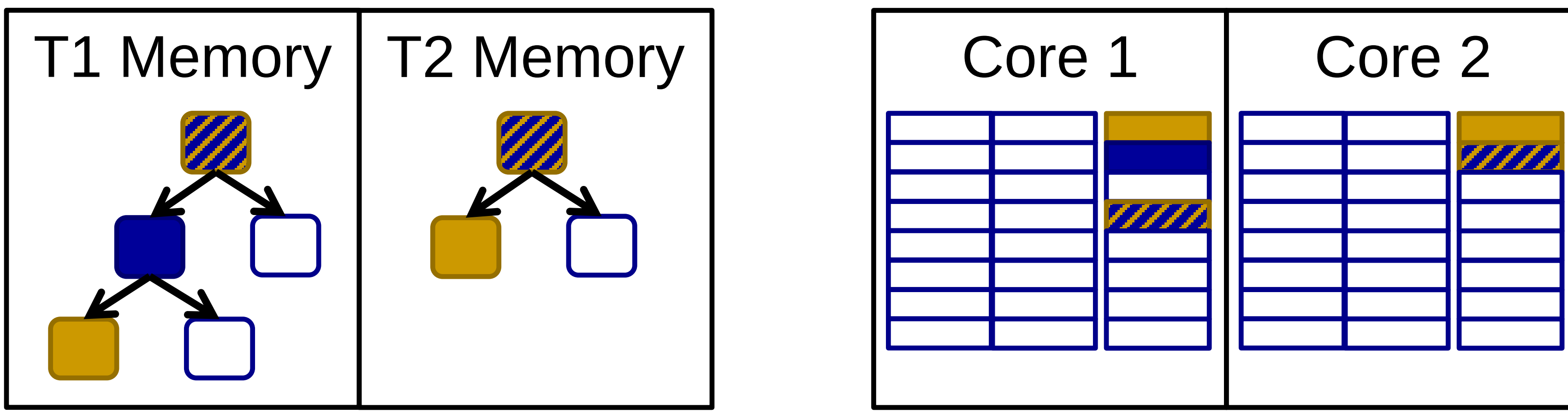
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow



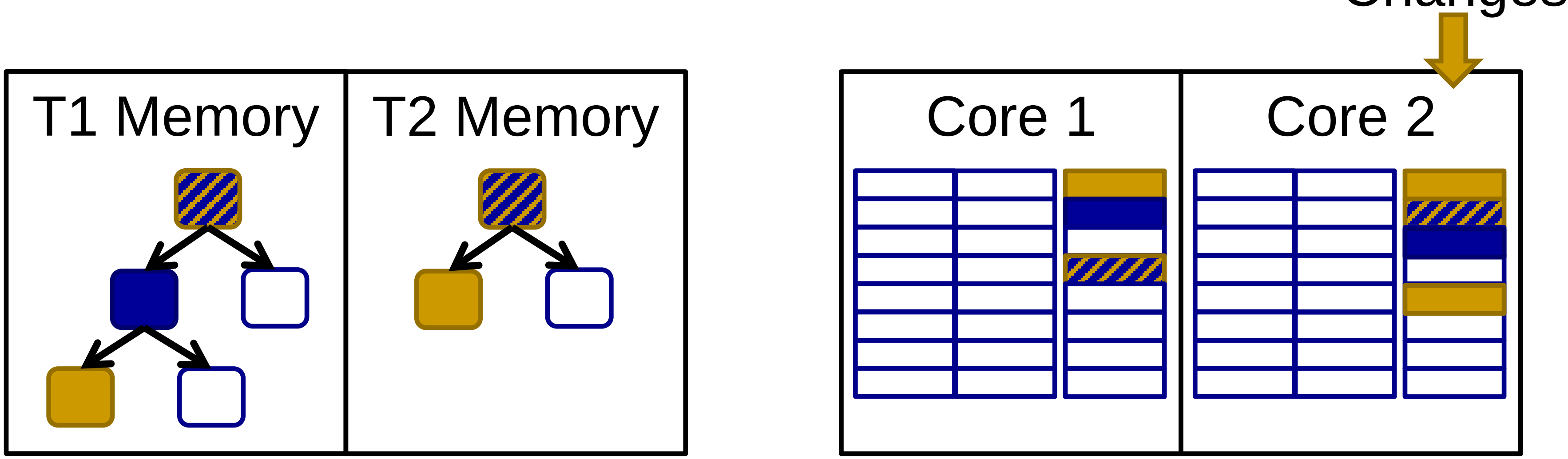
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter



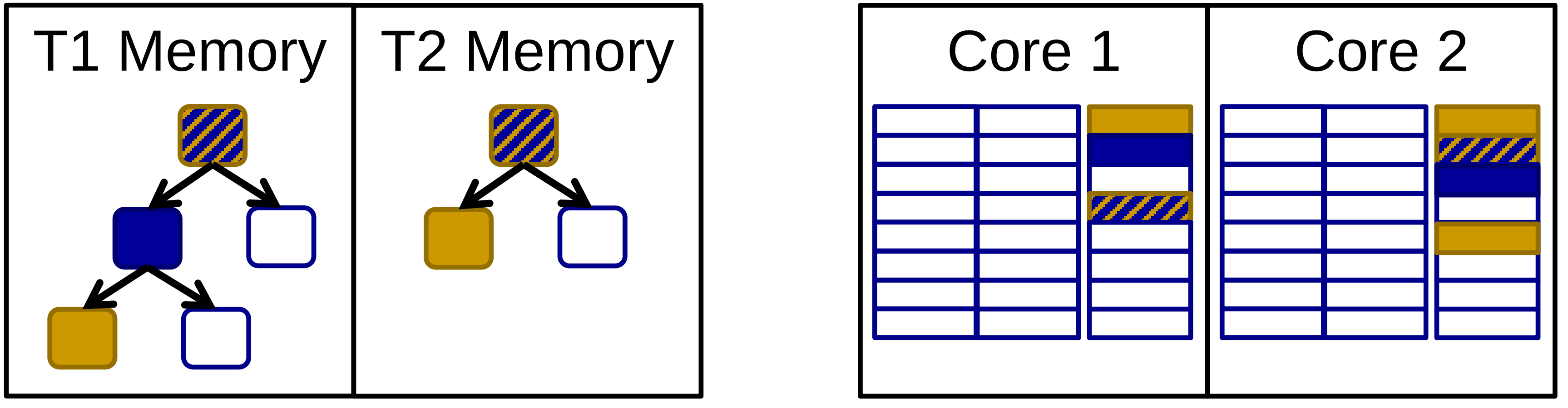
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter



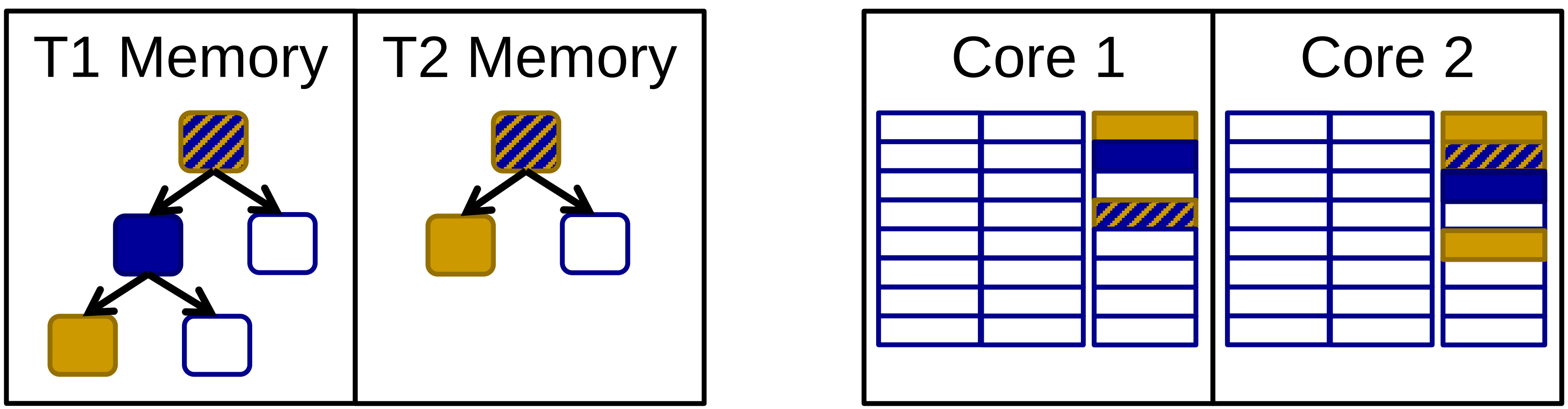
Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter



Watchpoint System Design

- Store Ranges in Main Memory
- Per-Thread Ranges, Per-Core Range Cache
- Software Handler on RC miss or overflow
- Write-back RC works as a write filter
- Precise, user-level watchpoint faults



Experimental Evaluation Setup

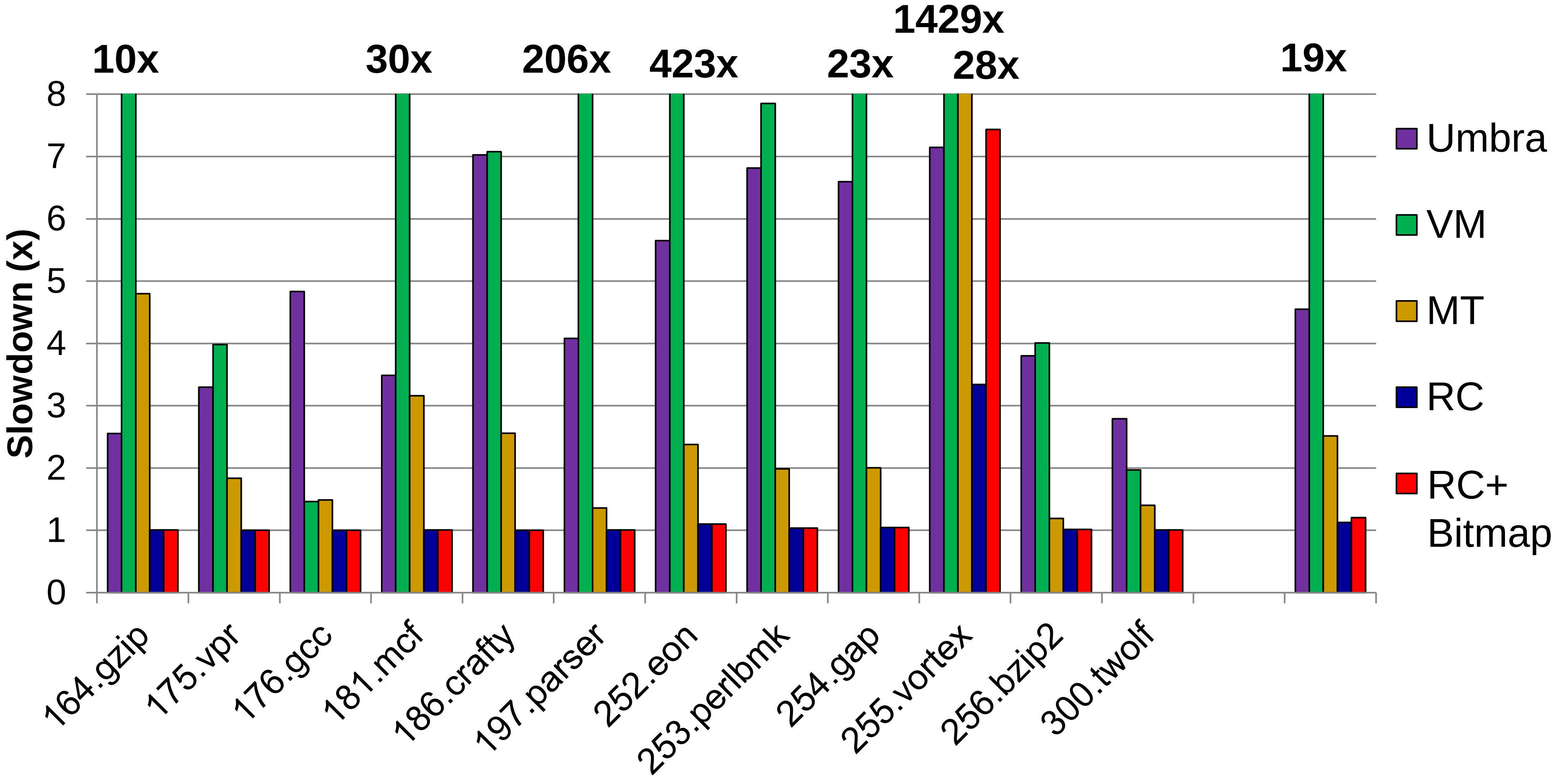
- Trace-based timing simulator using Pin

- Taint analysis on SPEC INT2000
- Race Detection on Phoenix and PARSEC

- Comparing only shadow value checks

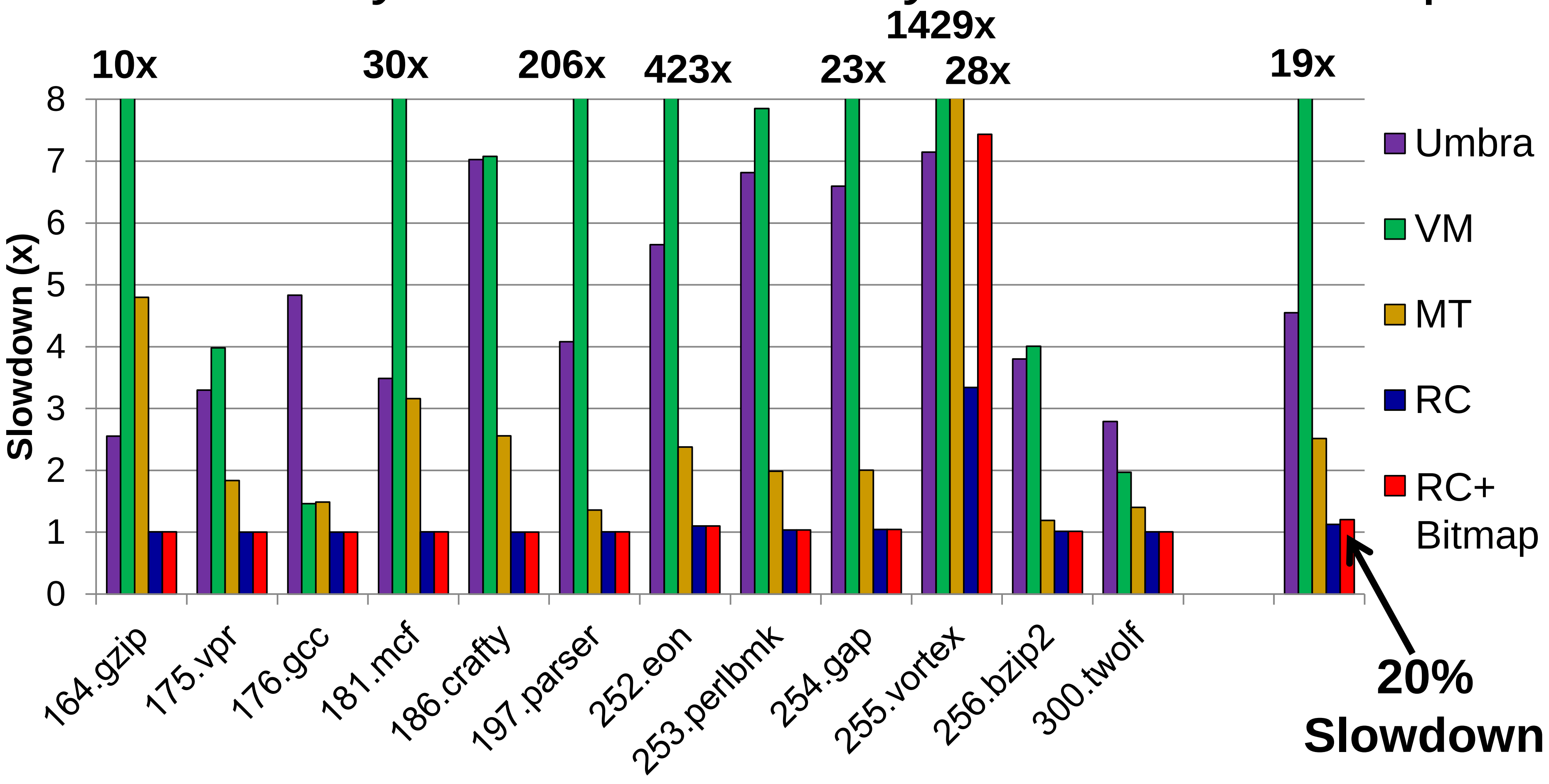
Watchpoint-Based Taint Analysis

- 128 entry RC **–or–** 64 entry RC + 2KB Bitmap

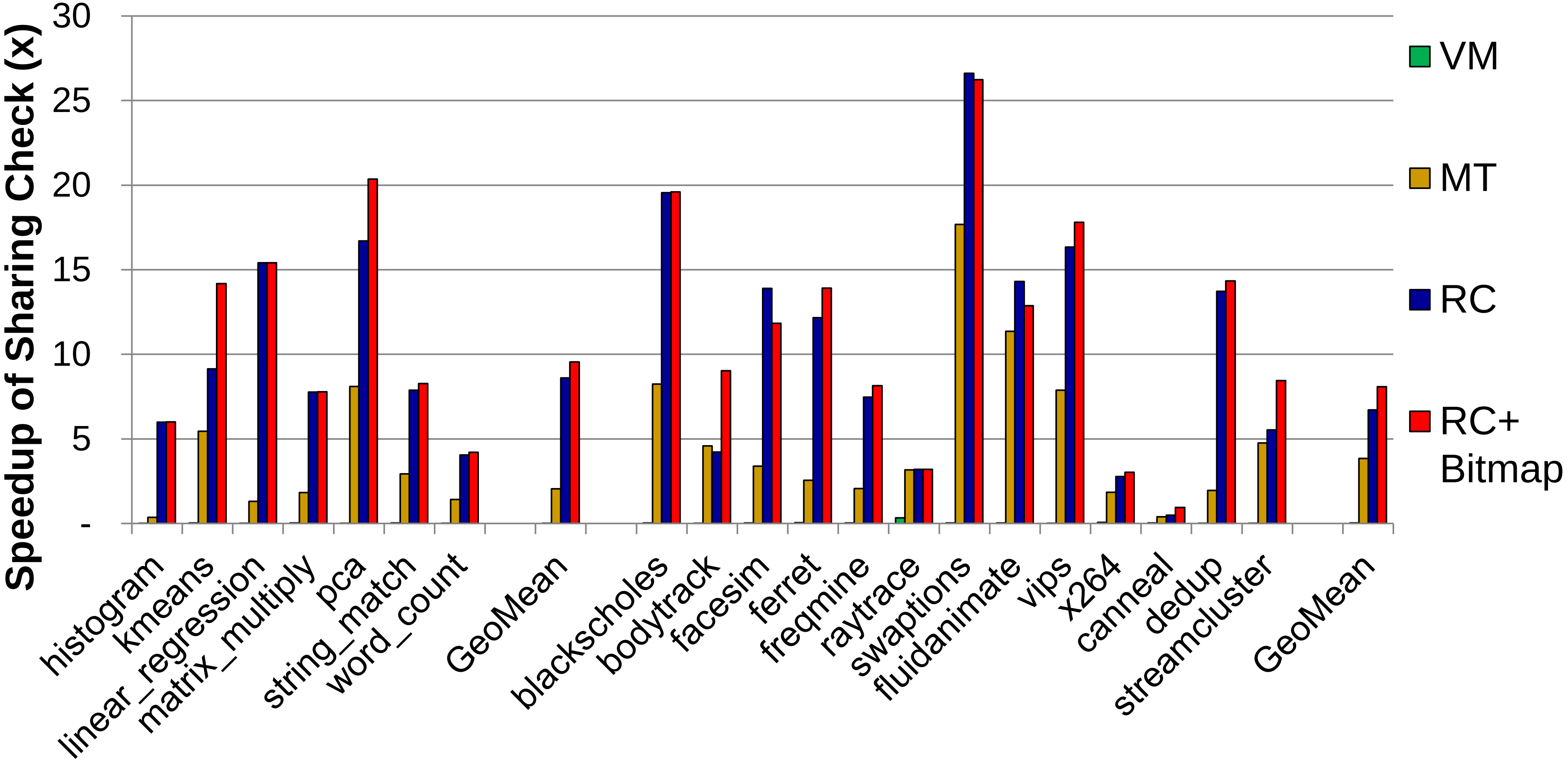


Watchpoint-Based Taint Analysis

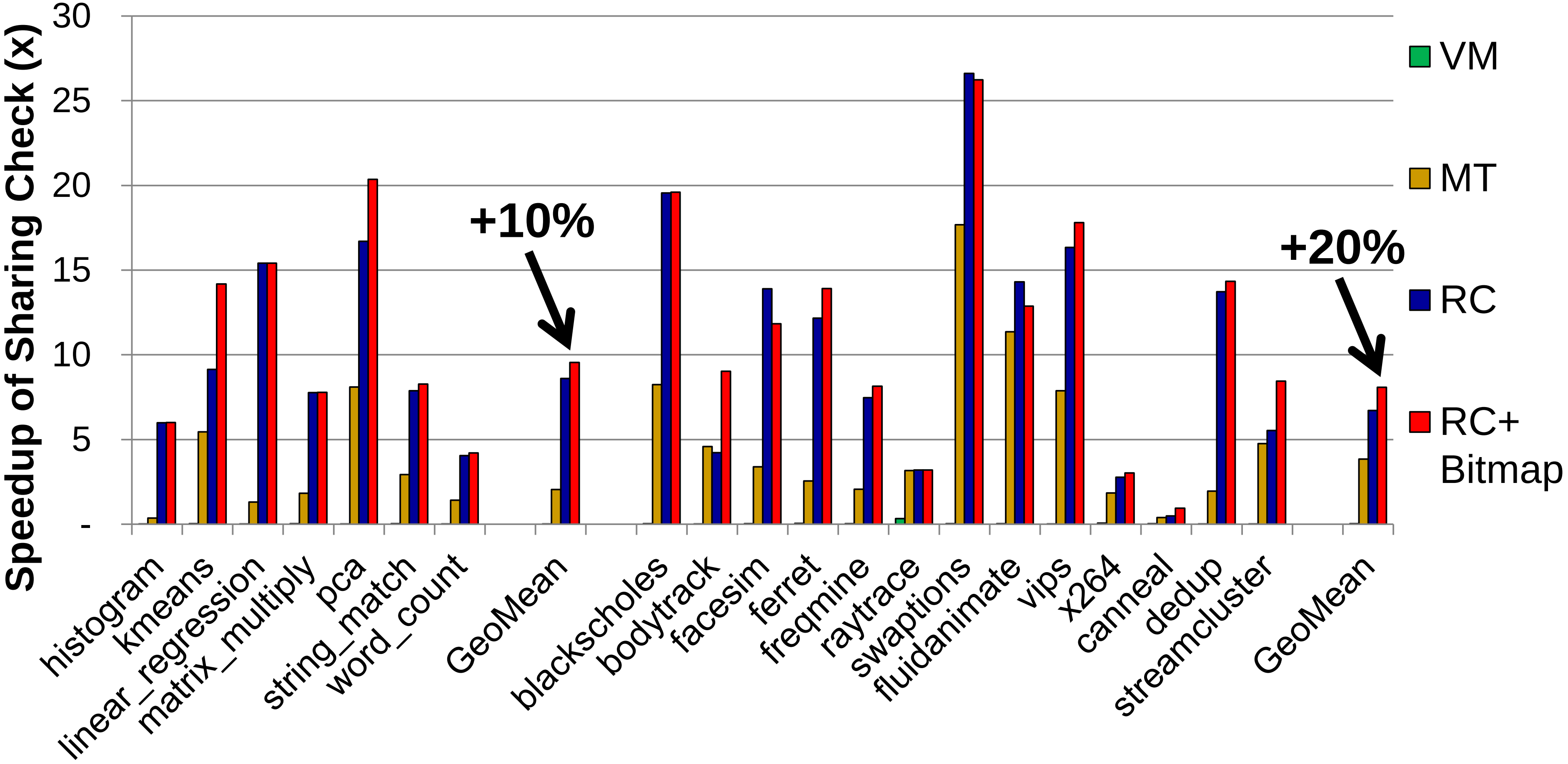
- 128 entry RC **–or–** 64 entry RC + 2KB Bitmap



Watchpoint-Based Data Race Detection



Watchpoint-Based Data Race Detection



Future Directions

- Dataflow Tests find bugs on executed code
  - What about code that is never executed?
- Sampling + Demand-Driven Race Detection
  - Good synergy between the two, like taint analysis
- Further watchpoint hardware studies:
  - Clear microarchitectural analysis
  - **More** software systems, different algorithms

Conclusions

Conclusions

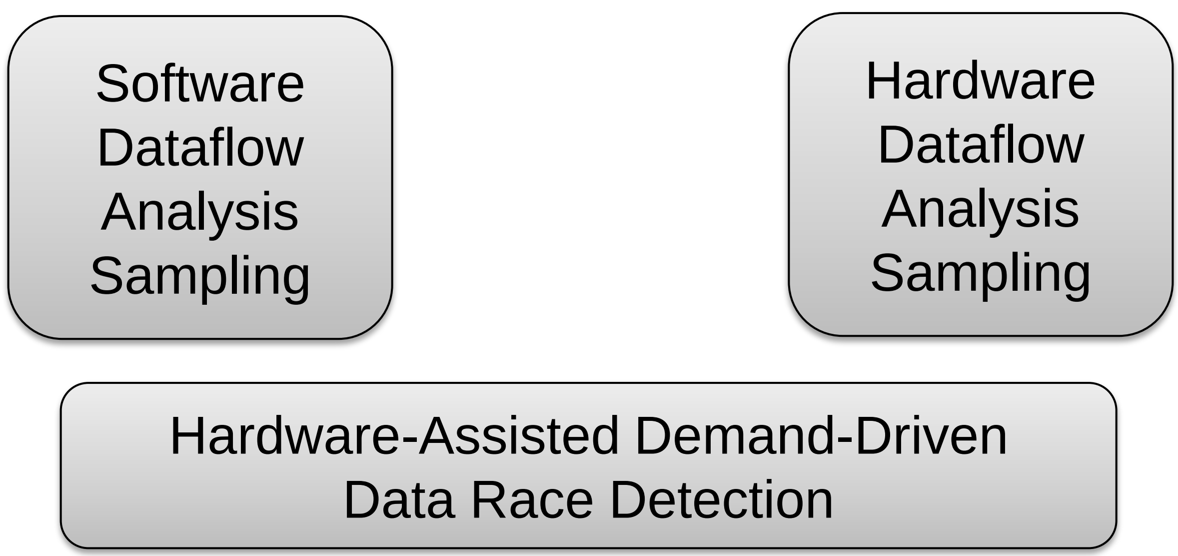
- **Sampling** allows distributed dataflow analysis

Software Dataflow Analysis Sampling

Hardware Dataflow Analysis Sampling

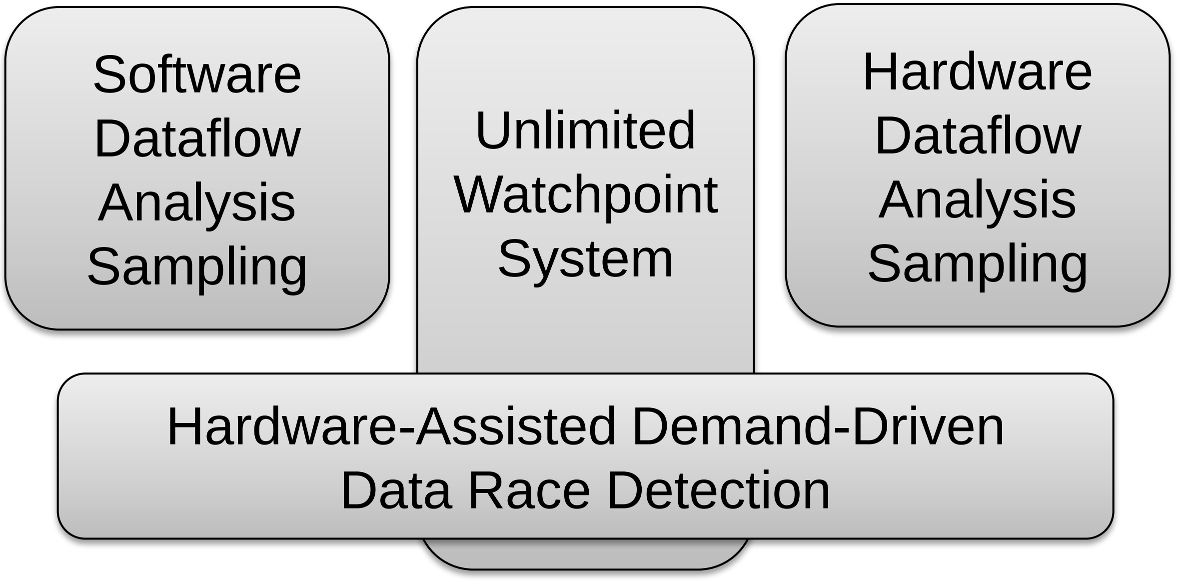
Conclusions

- **Sampling** allows distributed dataflow analysis
- **Existing hardware** can speed up race detection



Conclusions

- **Sampling** allows distributed dataflow analysis
- **Existing hardware** can speed up race detection
- **Watchpoint hardware** useful everywhere



Conclusions

- **Sampling** allows distributed dataflow analysis
- **Existing hardware** can speed up race detection
- **Watchpoint hardware** useful everywhere

Distributed Dynamic
Software Analysis

Thank You

BACKUP SLIDES

Finding Errors

- Brute Force
  - Code review, fuzz testing, whitehat/grayhat hackers
  - Time-consuming, difficult

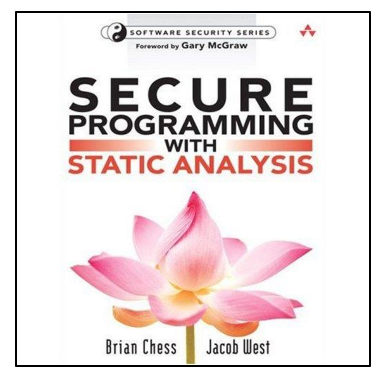
Finding Errors

- Brute Force
  - Code review, fuzz testing, whitehat/grayhat hackers
  - – Time-consuming, difficult

- Static Analysis
  - Automatically analyze source, formal reasoning, compiler checks
  - – Intractable, requires expert input, no system state

Klocwork

FORTIFY SOFTWARE

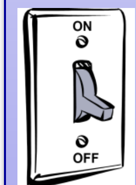


Dynamic Dataflow Analysis

- **Associate** meta-data with program values
- **Propagate/Clear** meta-data while executing
- **Check** meta-data for safety & correctness
- Forms **dataflows** of meta/shadow information

Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



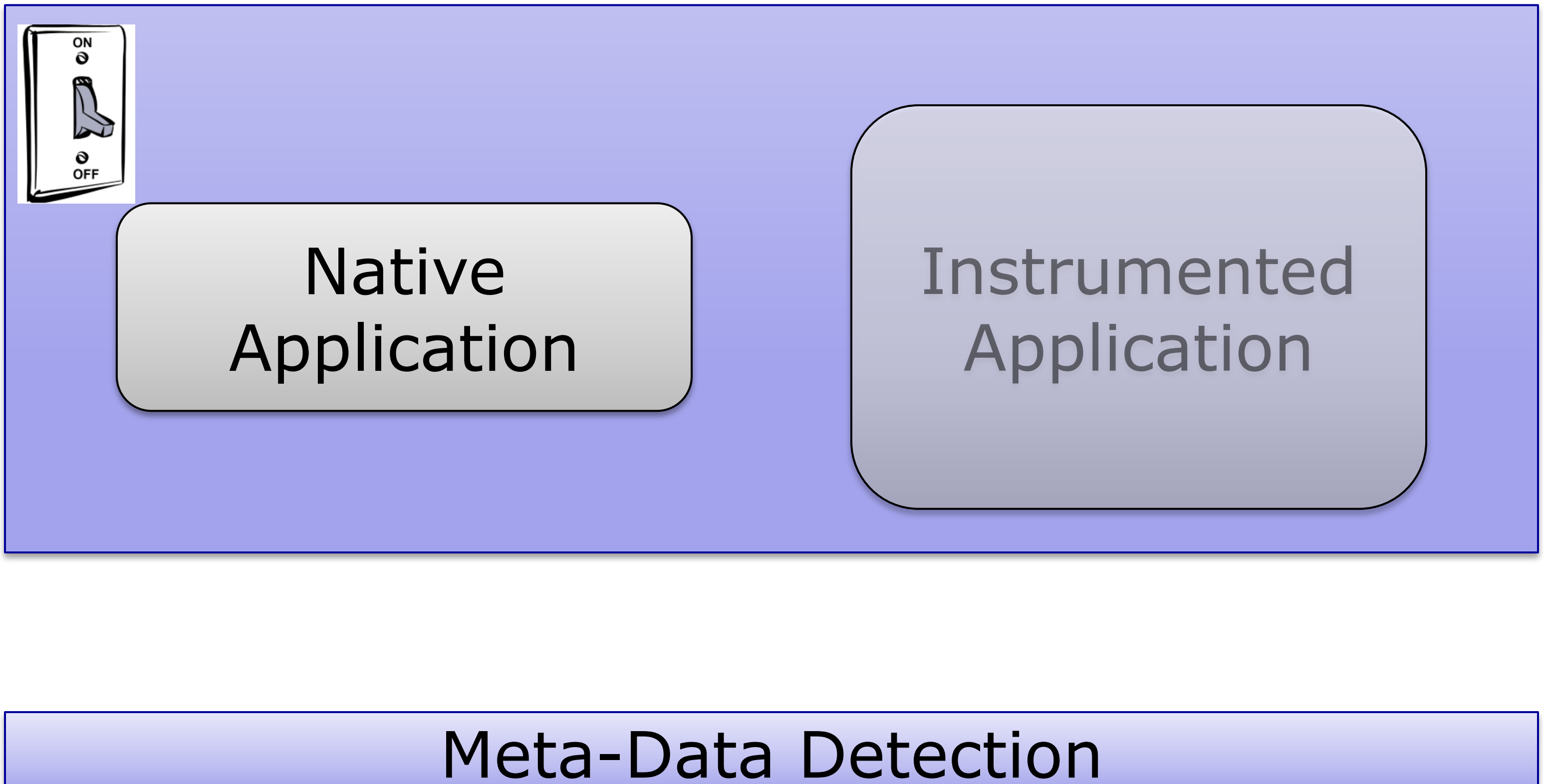
Native Application

Instrumented Application

Meta-Data Detection

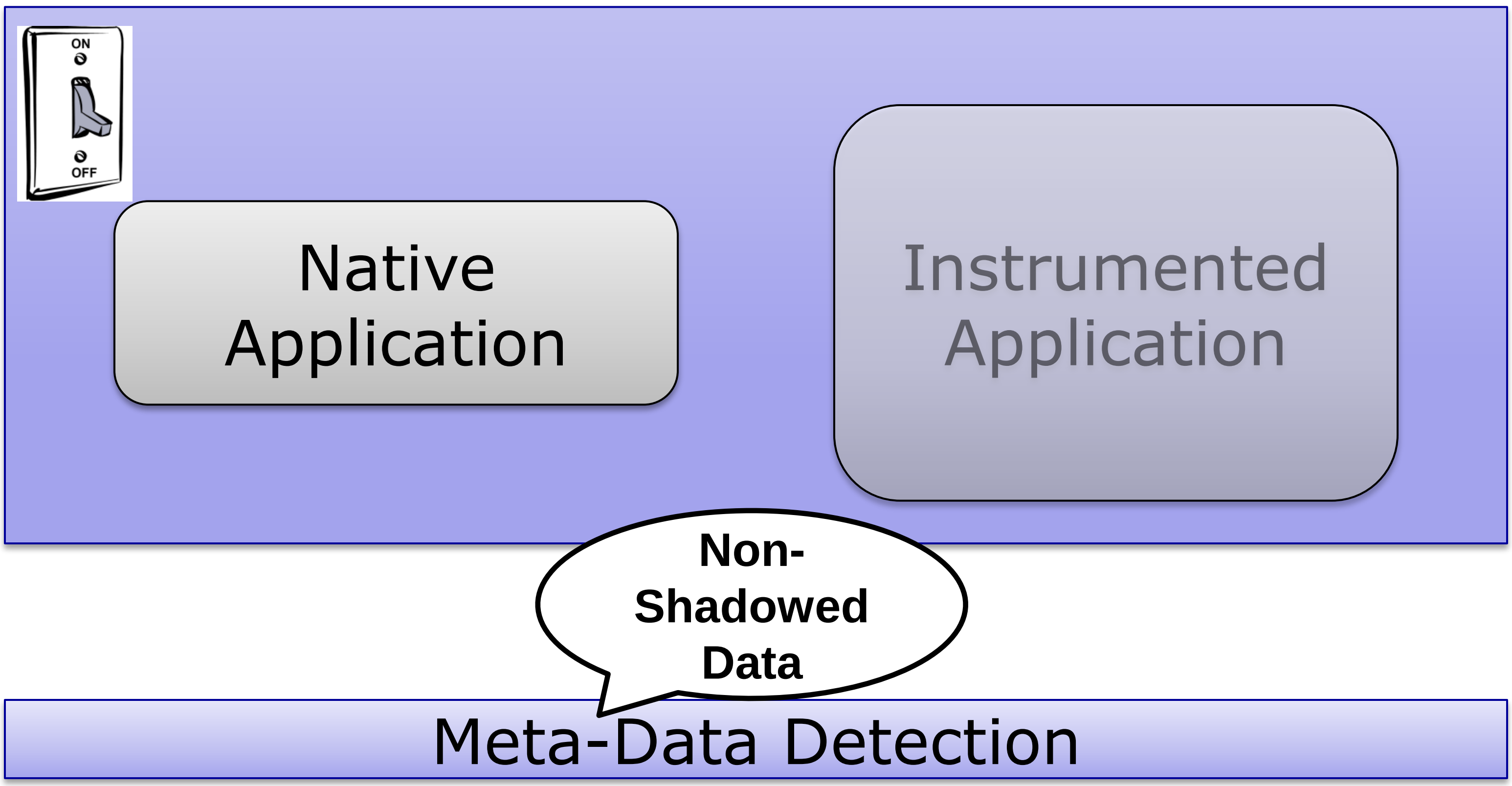
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



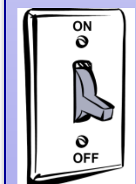
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



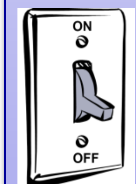
Native Application

Instrumented Application

Meta-Data Detection

Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



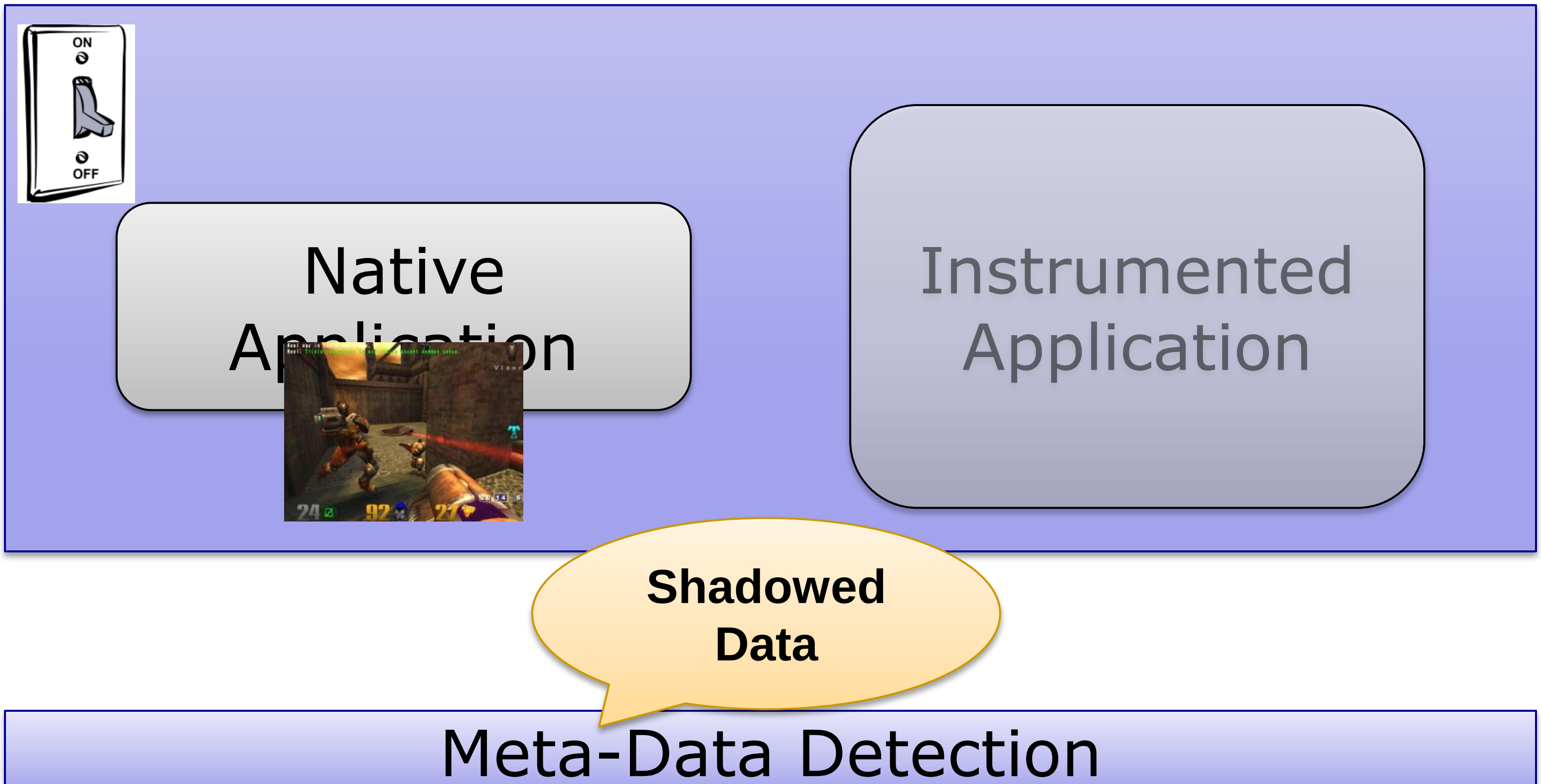
Native Application

Instrumented Application

Meta-Data Detection

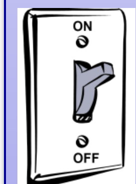
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



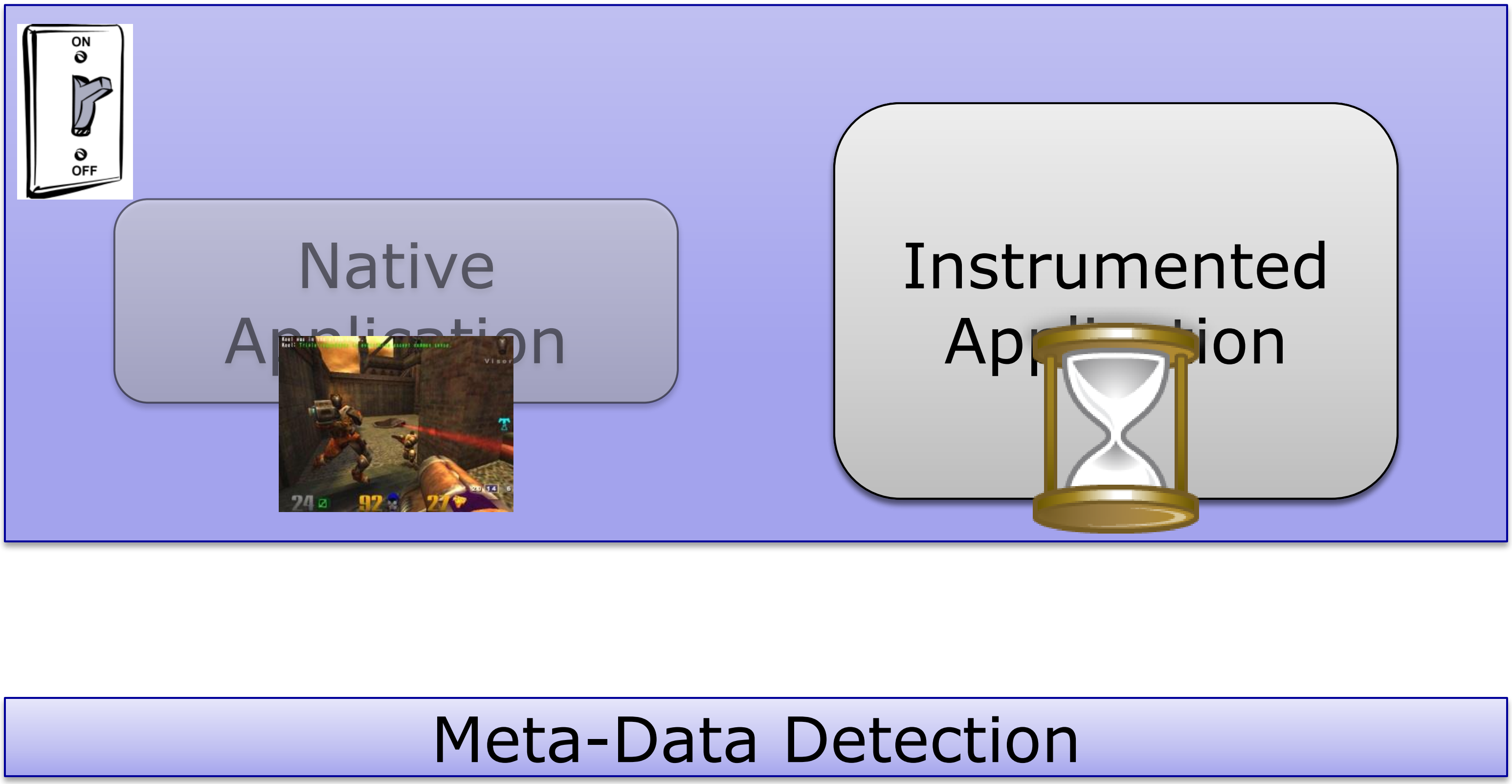
Native Application

Instrumented Application

Meta-Data Detection

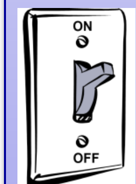
Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



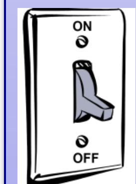
Native Application

Instrumented Application

Meta-Data Detection

Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



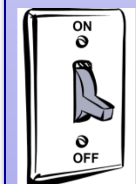
Native Application

Instrumented Application

Meta-Data Detection

Demand-Driven Dataflow Analysis

- Only Analyze Shadowed Data



Native Application

Instrumented Application

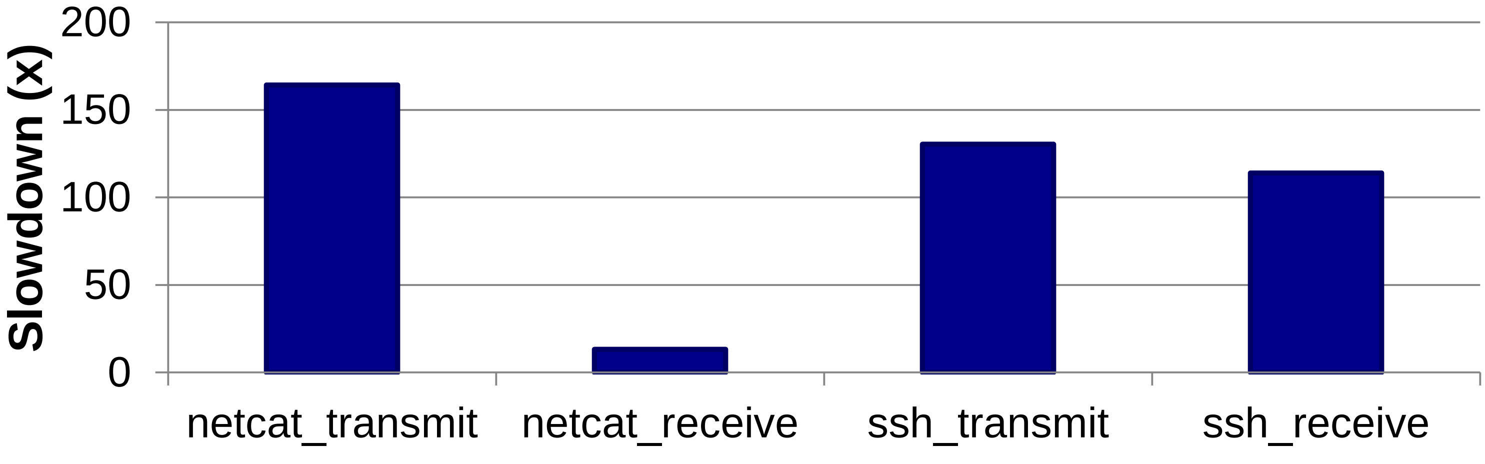
Meta-Data Detection

Results by Ho et al.

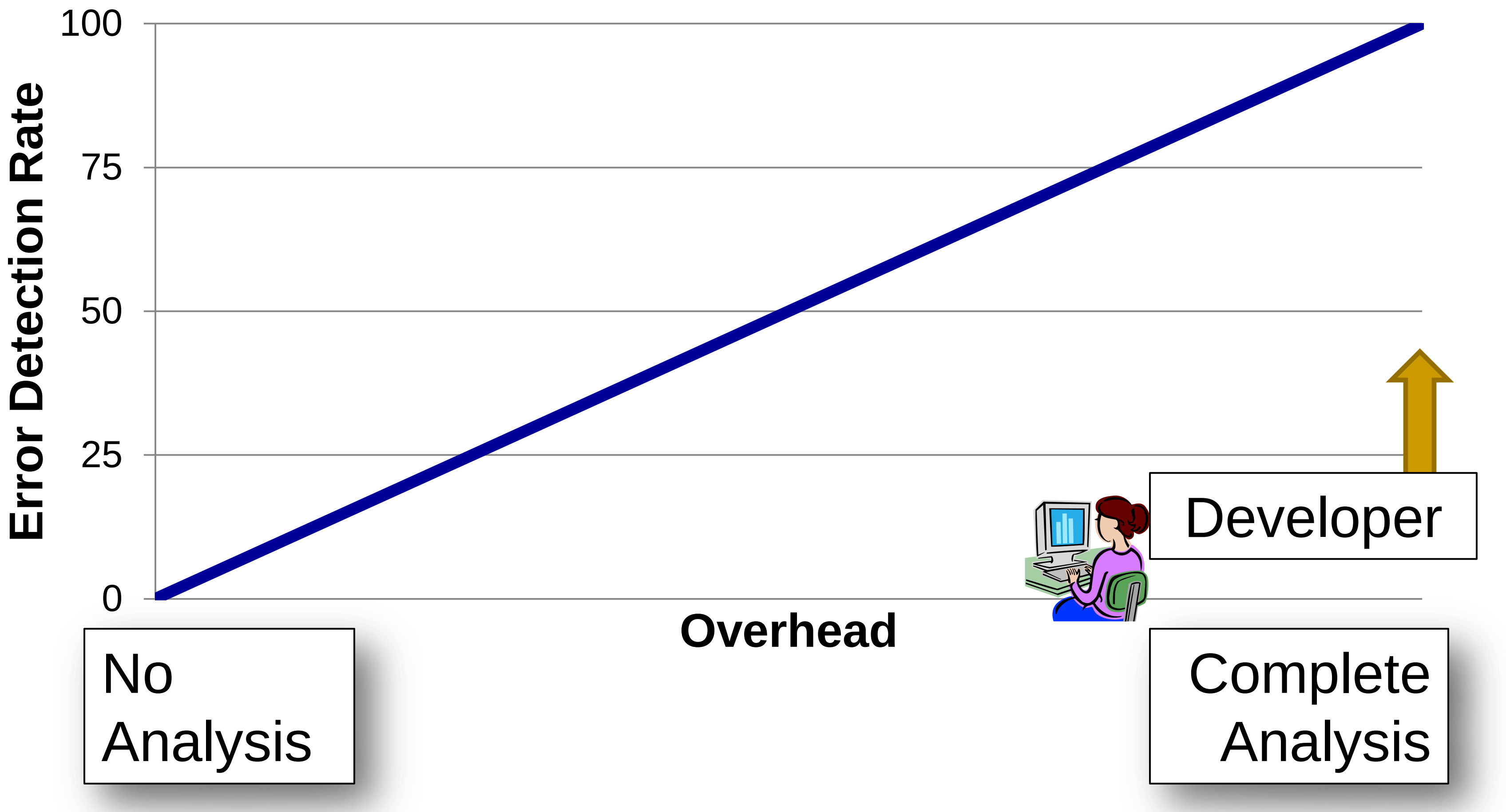
- lmbench Best Case Results:

System	Slowdown
Taint Analysis	101.7x
On-Demand Taint Analysis	1.98x

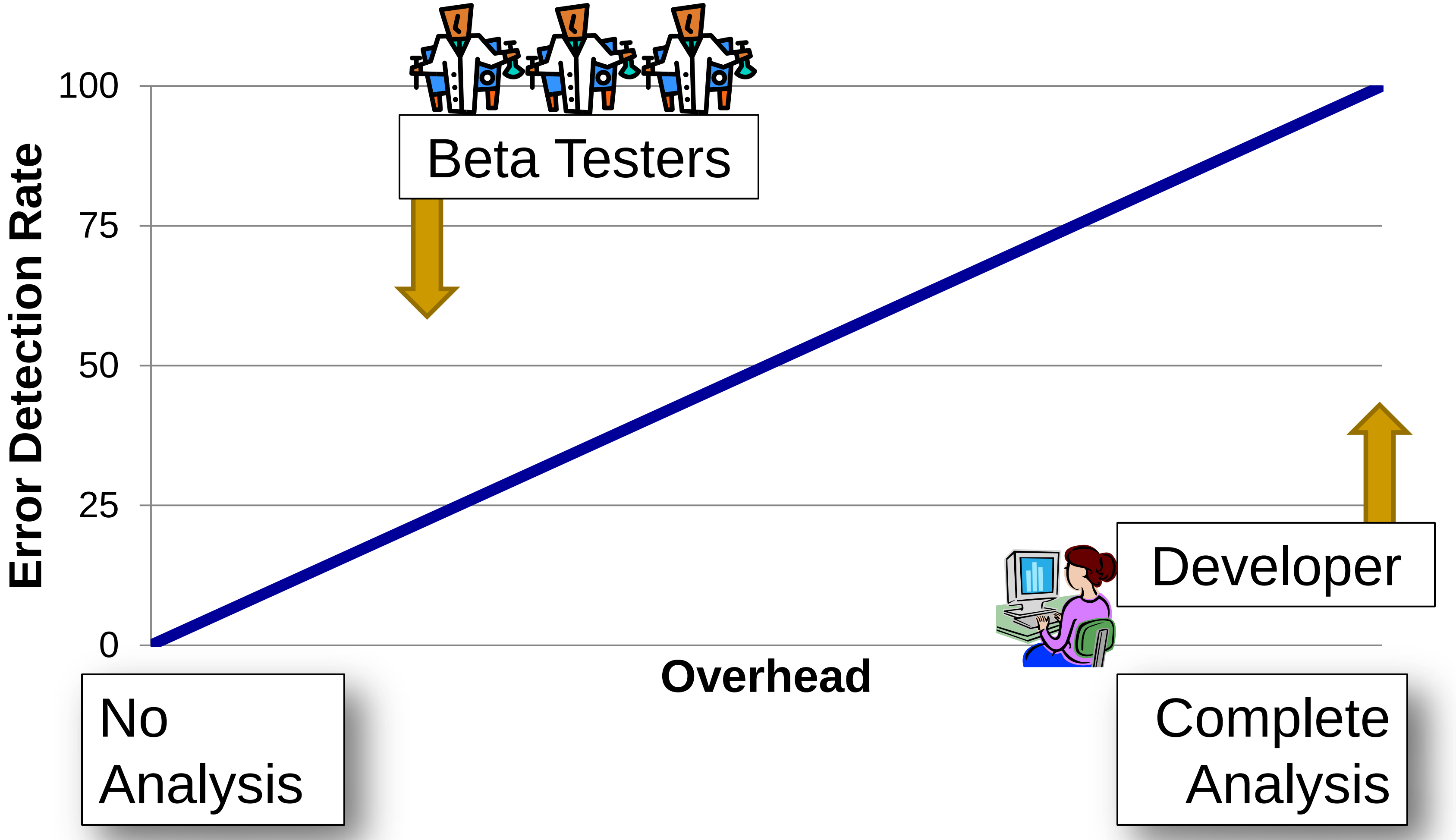
- Results when everything is tainted:



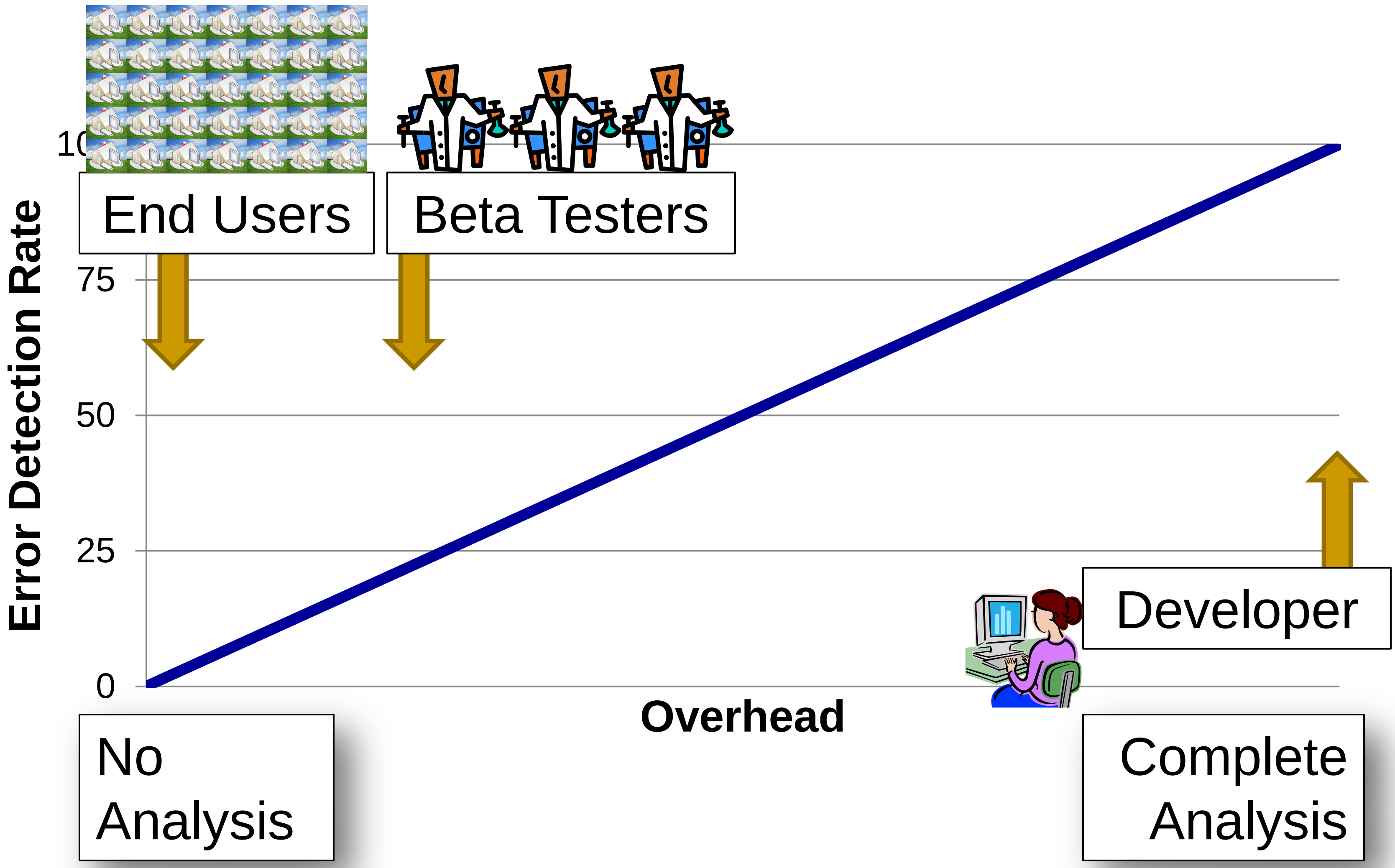
Sampling Allows Distribution



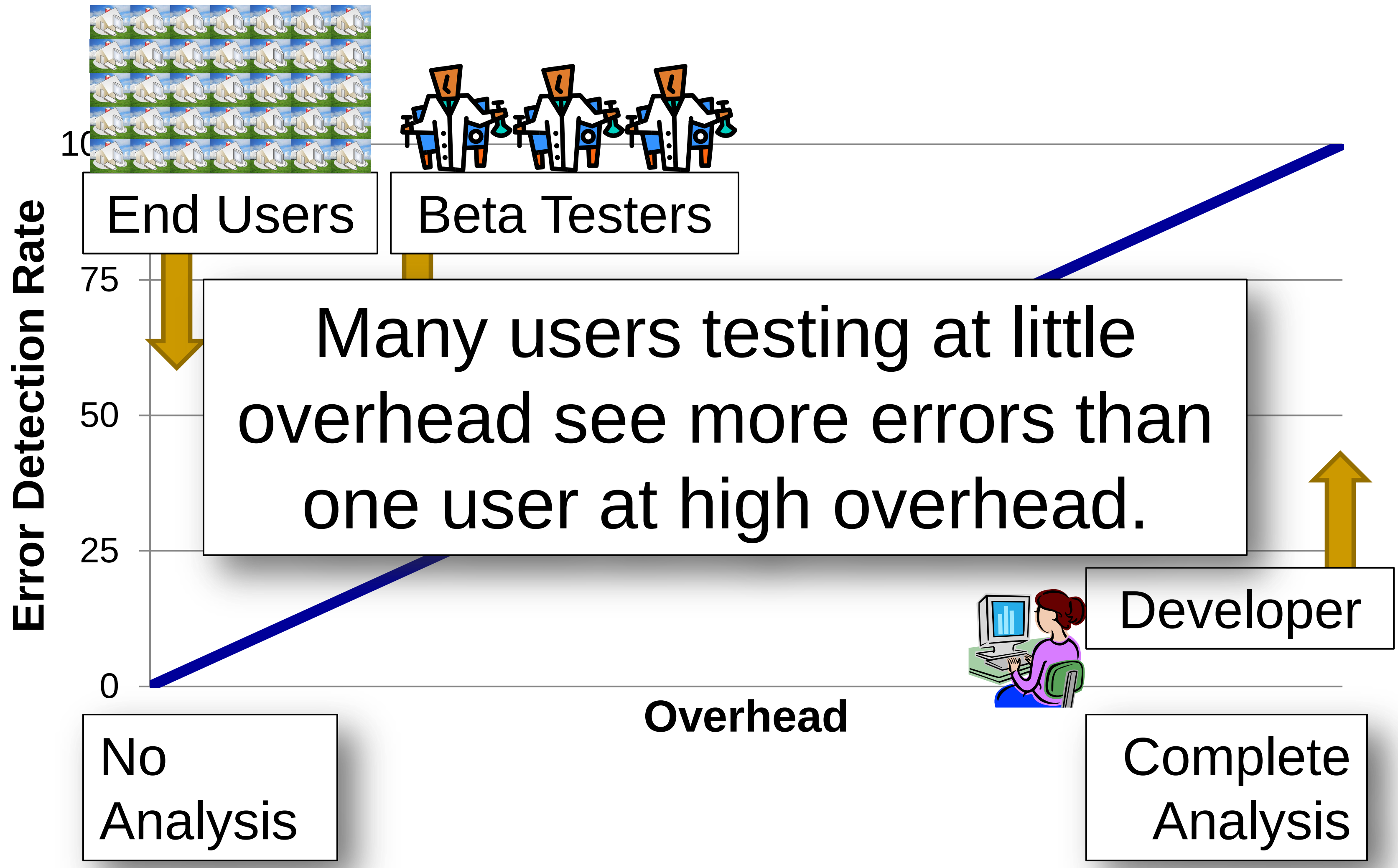
Sampling Allows Distribution



Sampling Allows Distribution

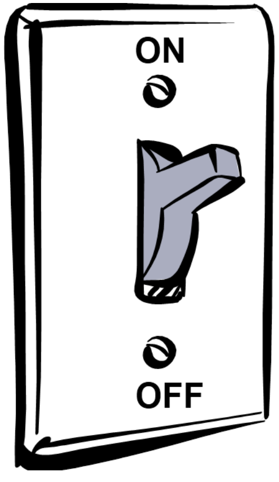


Sampling Allows Distribution

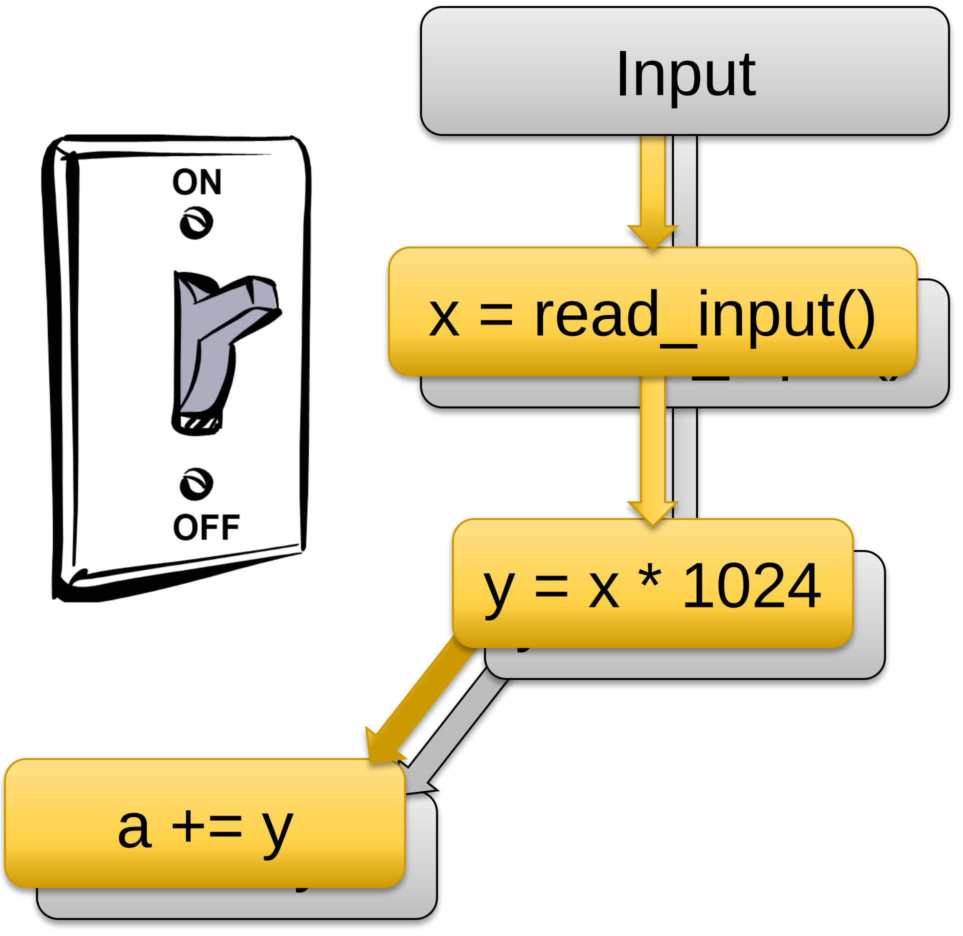


Cannot Naïvely Sample Code

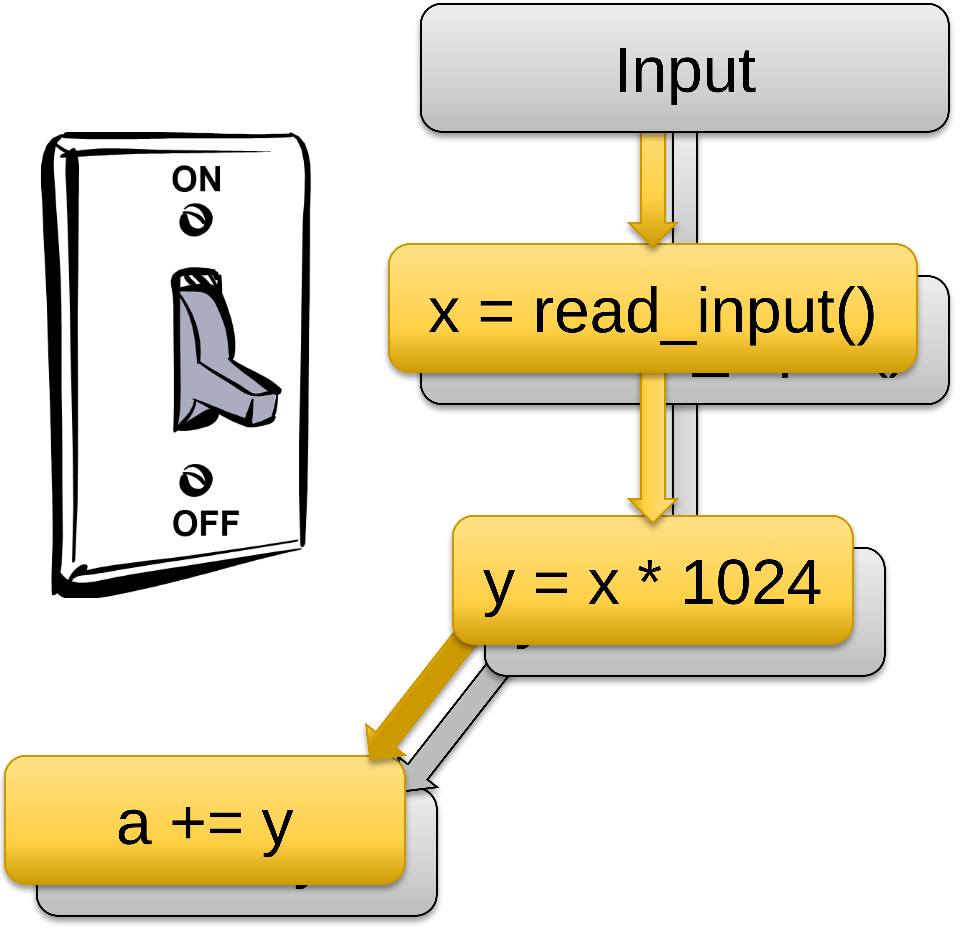
Input



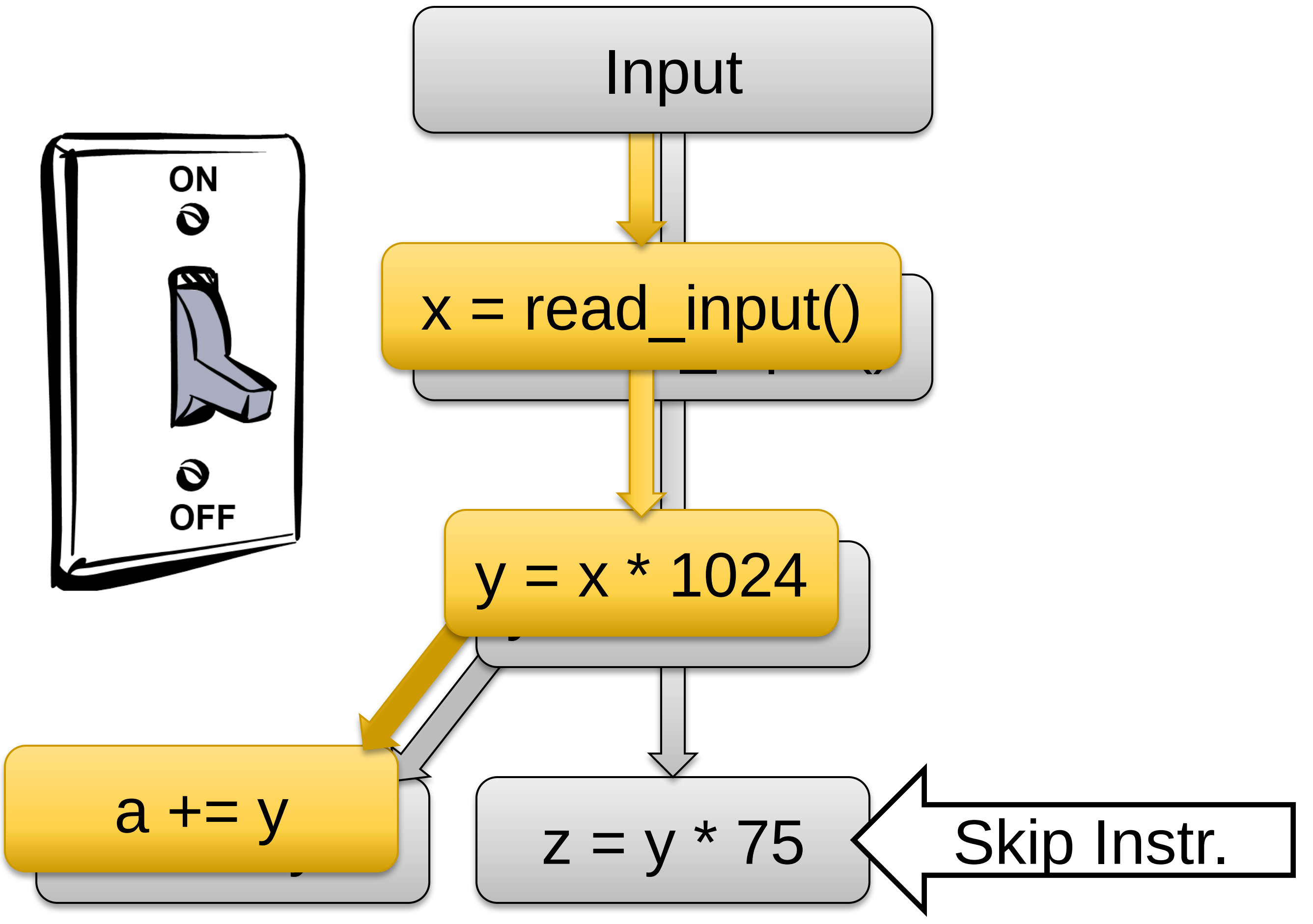
Cannot Naïvely Sample Code



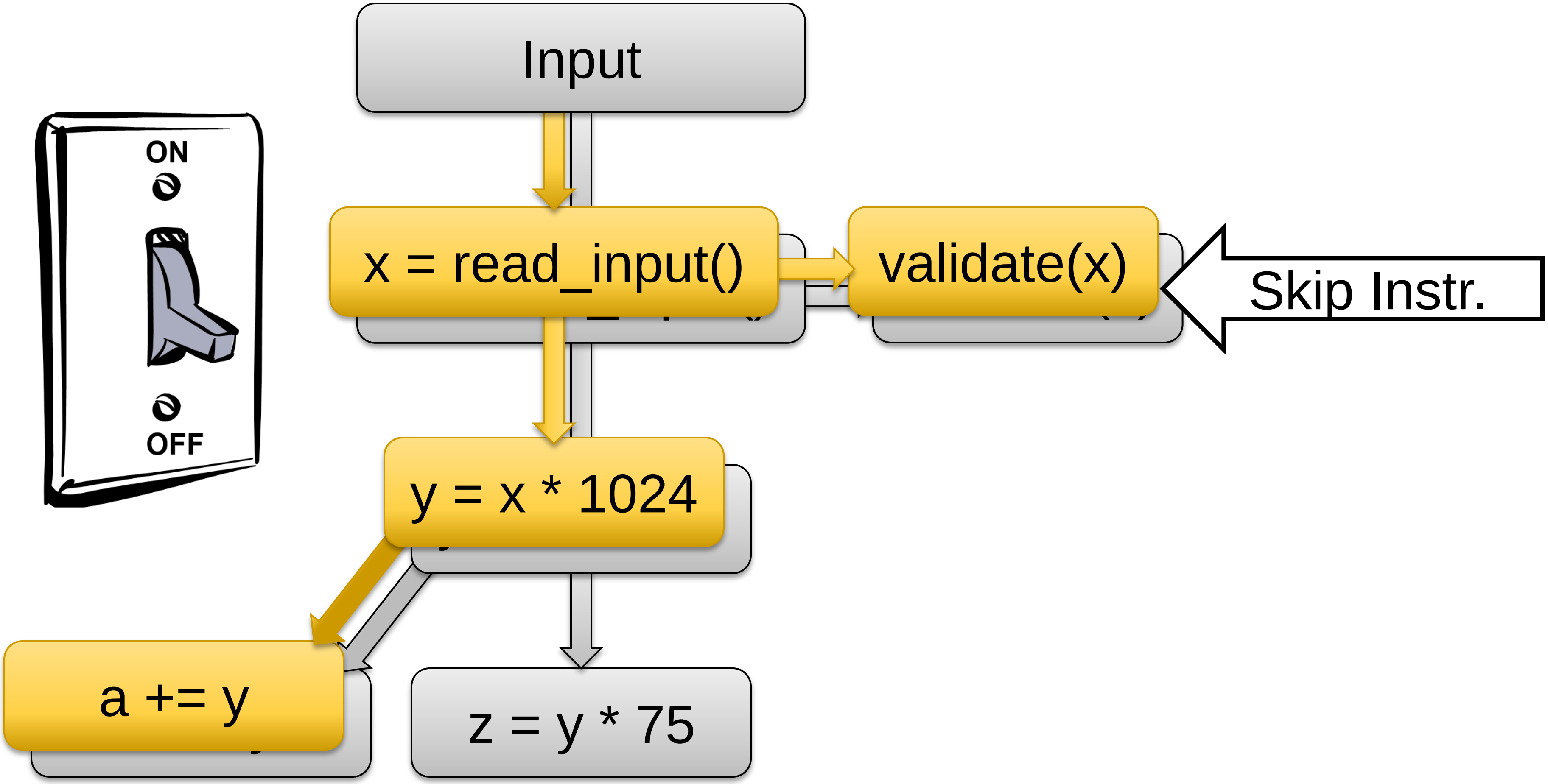
Cannot Naïvely Sample Code



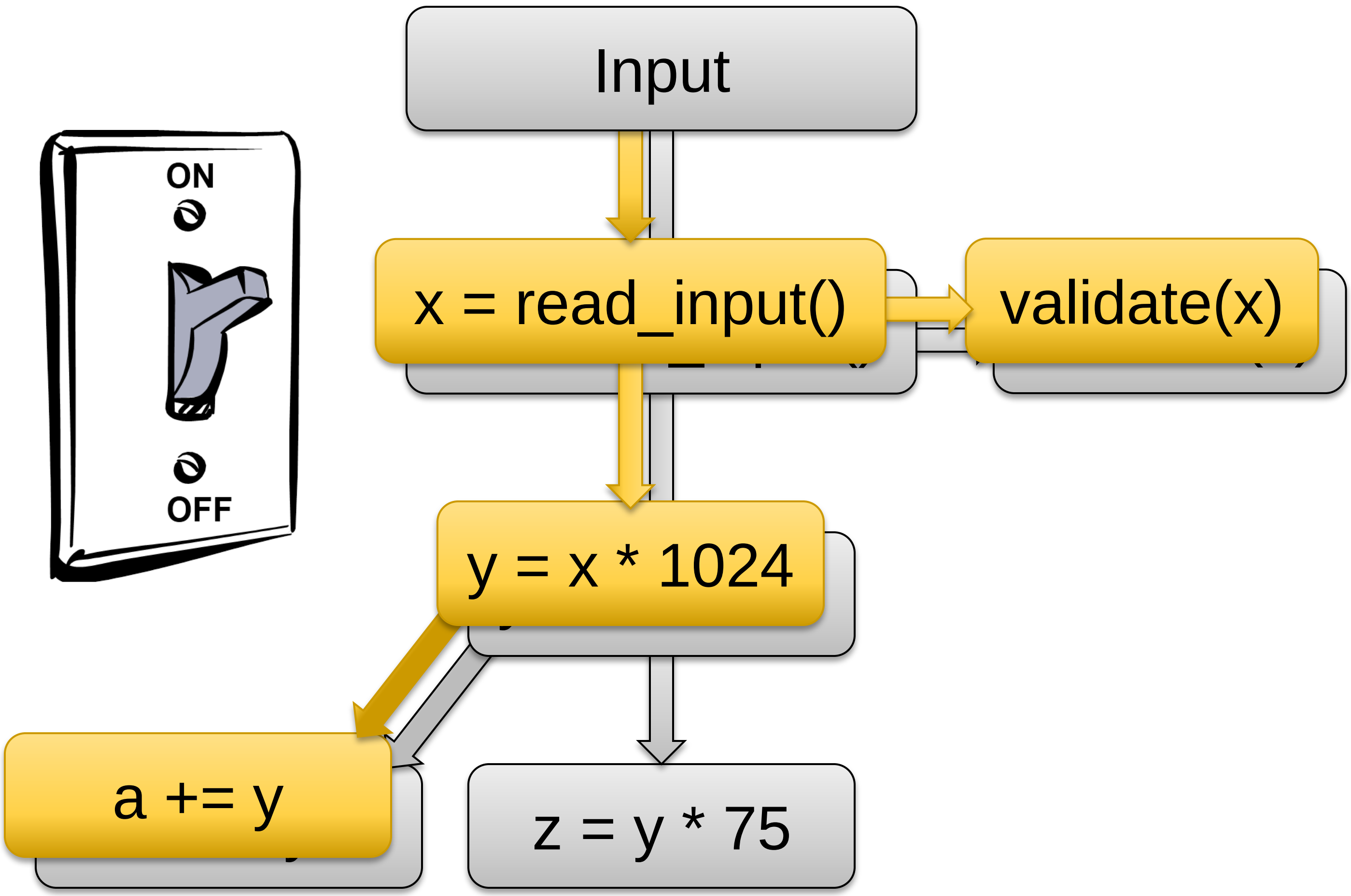
Cannot Naïvely Sample Code



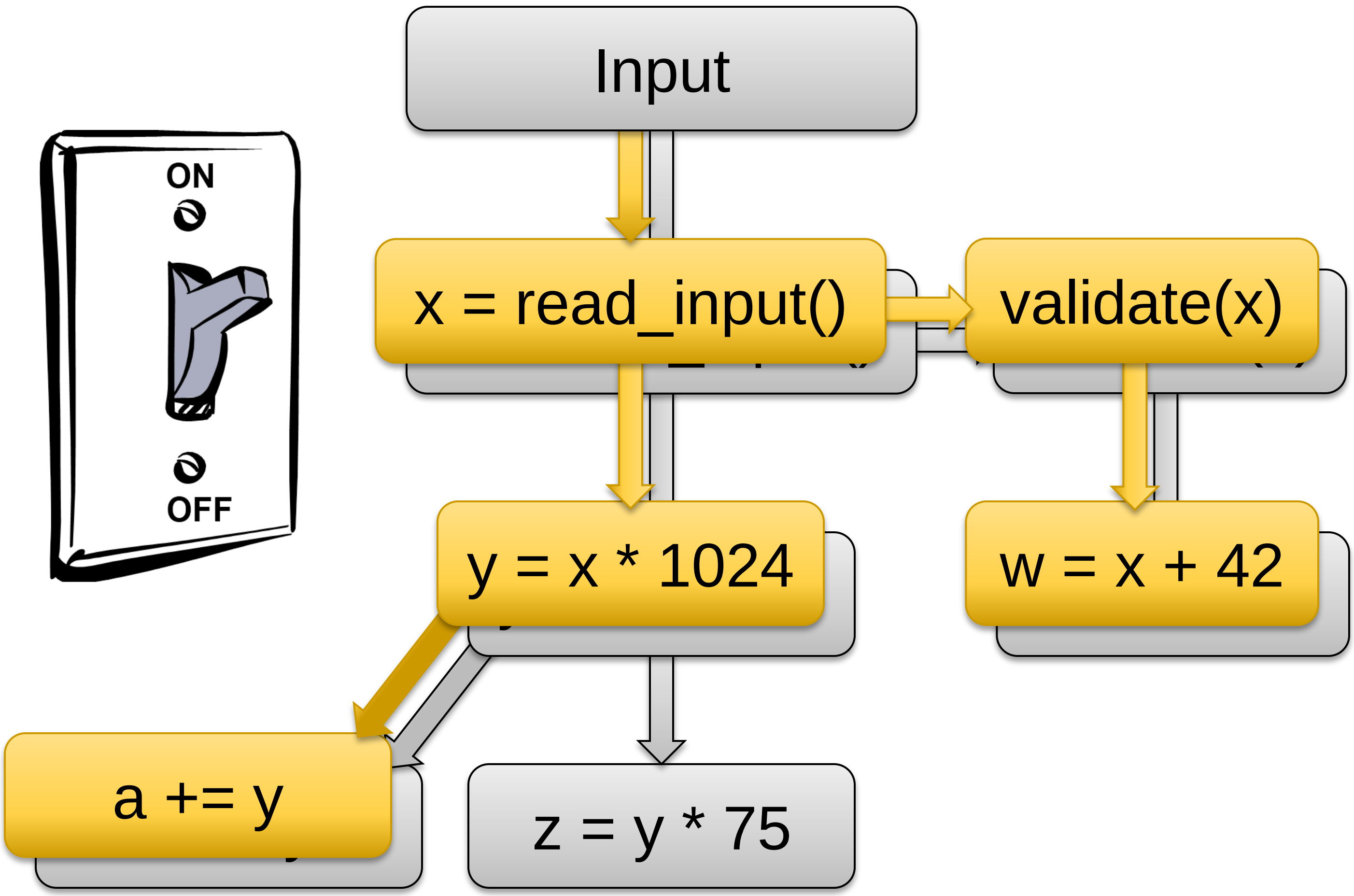
Cannot Naïvely Sample Code



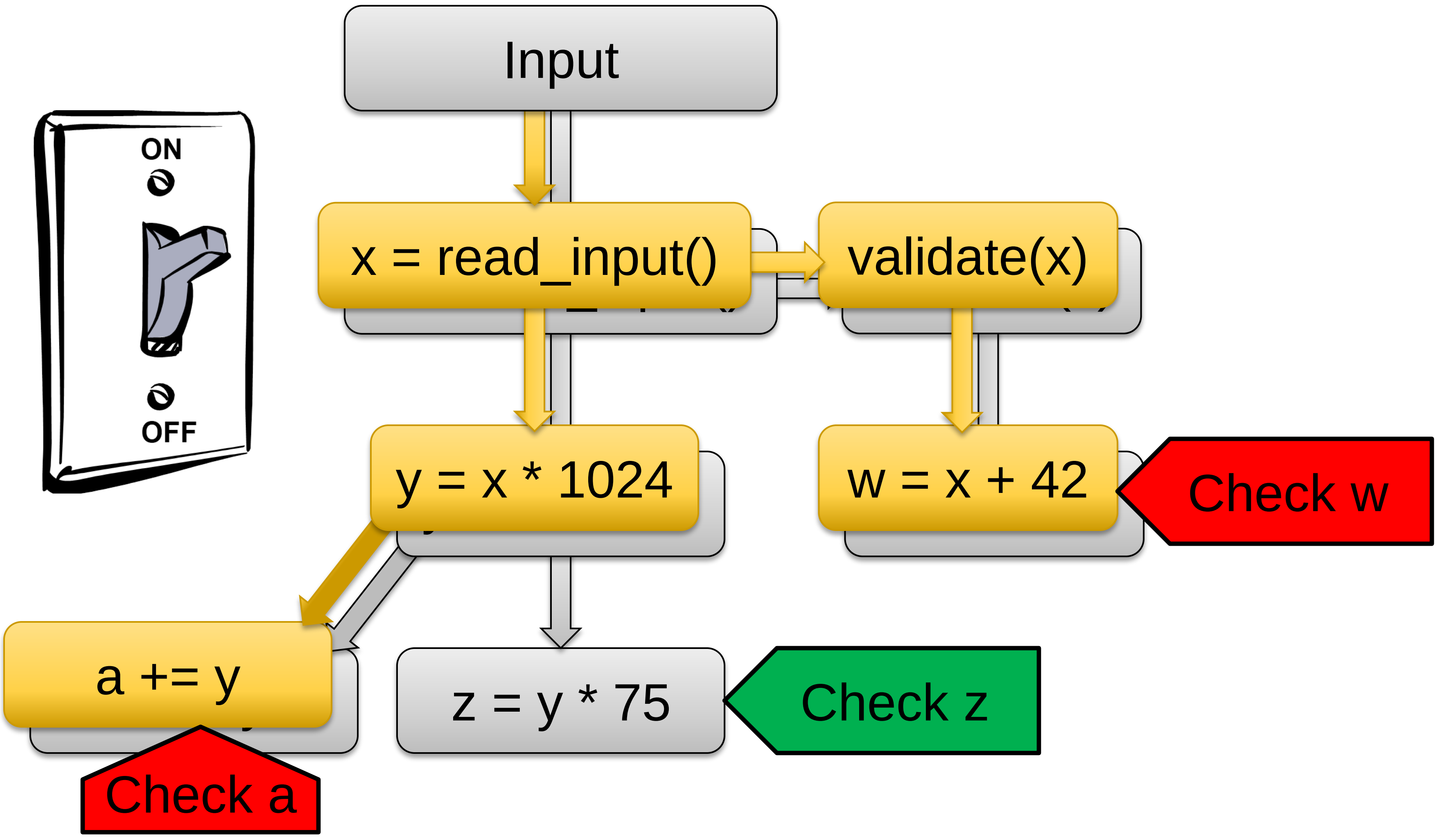
Cannot Naïvely Sample Code



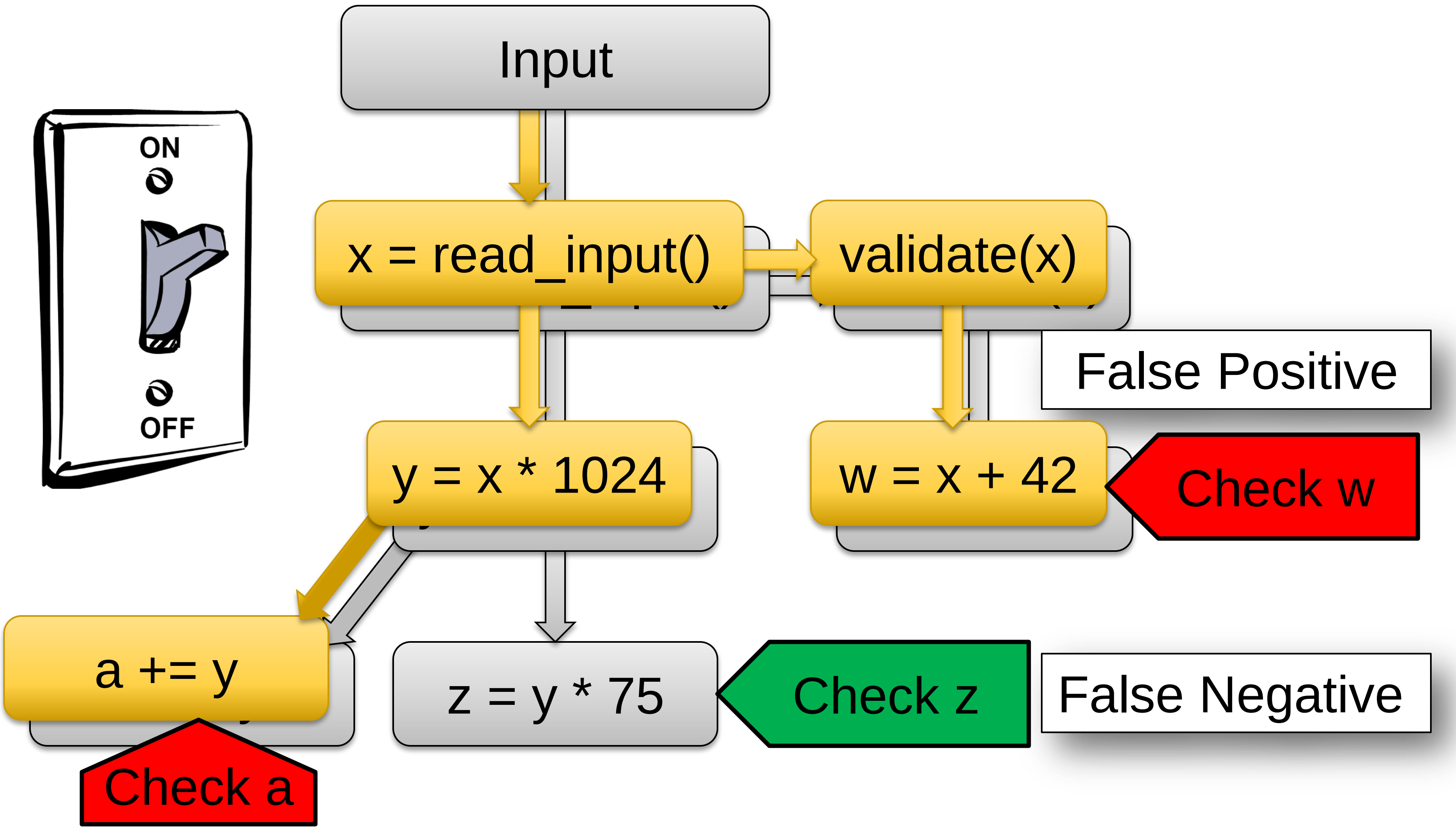
Cannot Naïvely Sample Code



Cannot Naïvely Sample Code

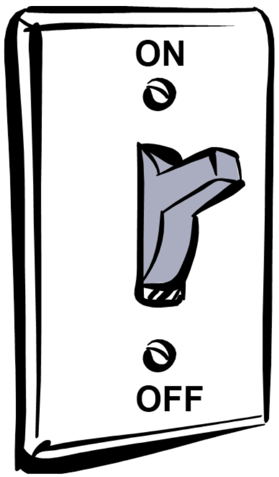


Cannot Naïvely Sample Code

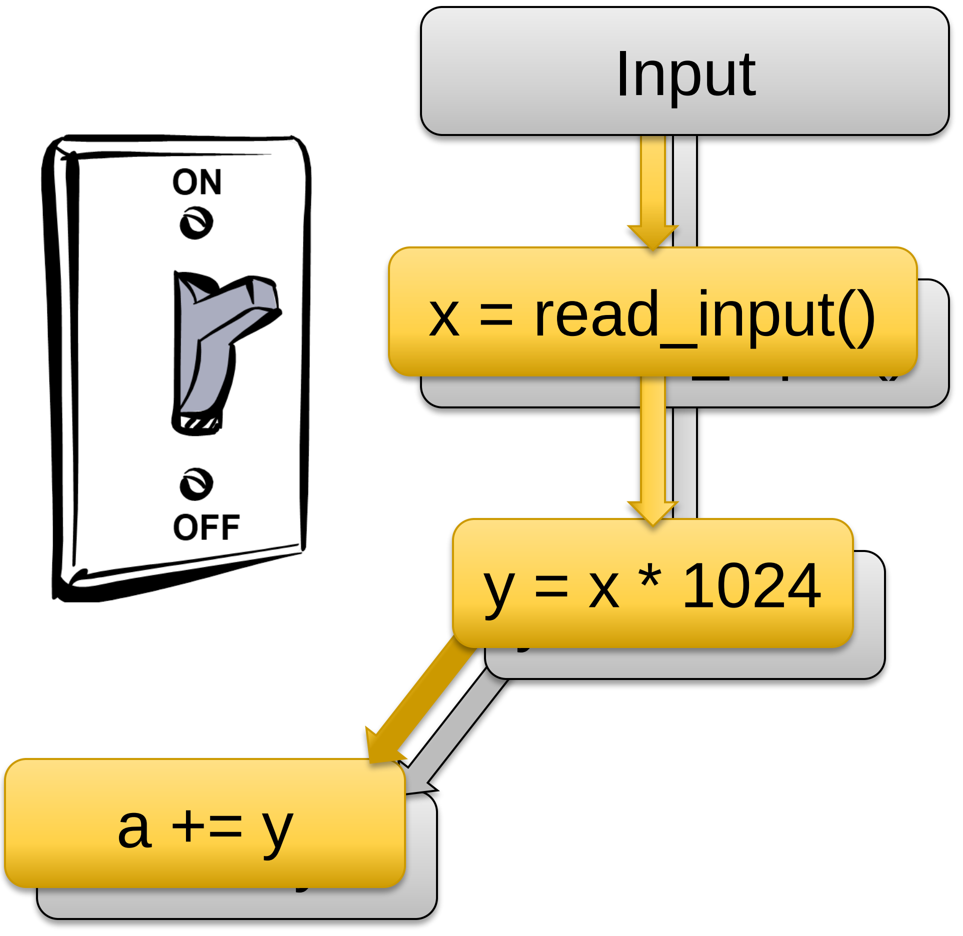


Dataflow Sampling Example

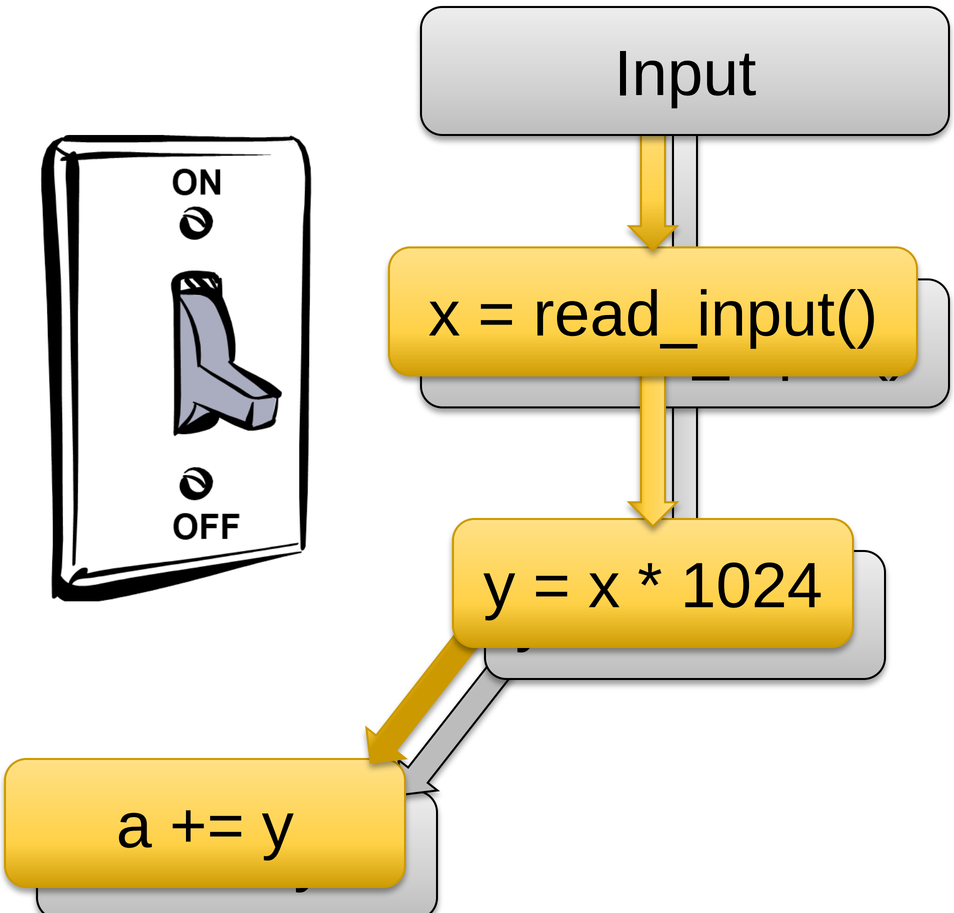
Input



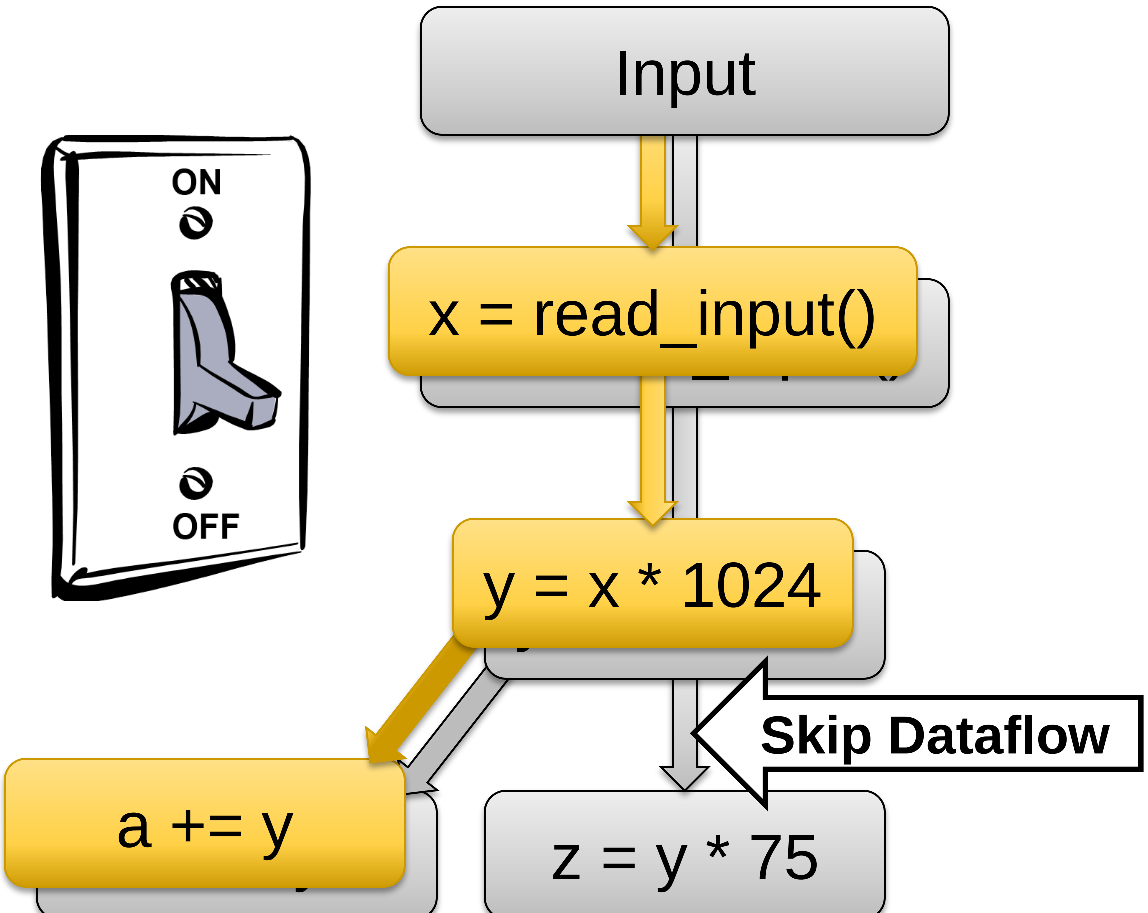
Dataflow Sampling Example



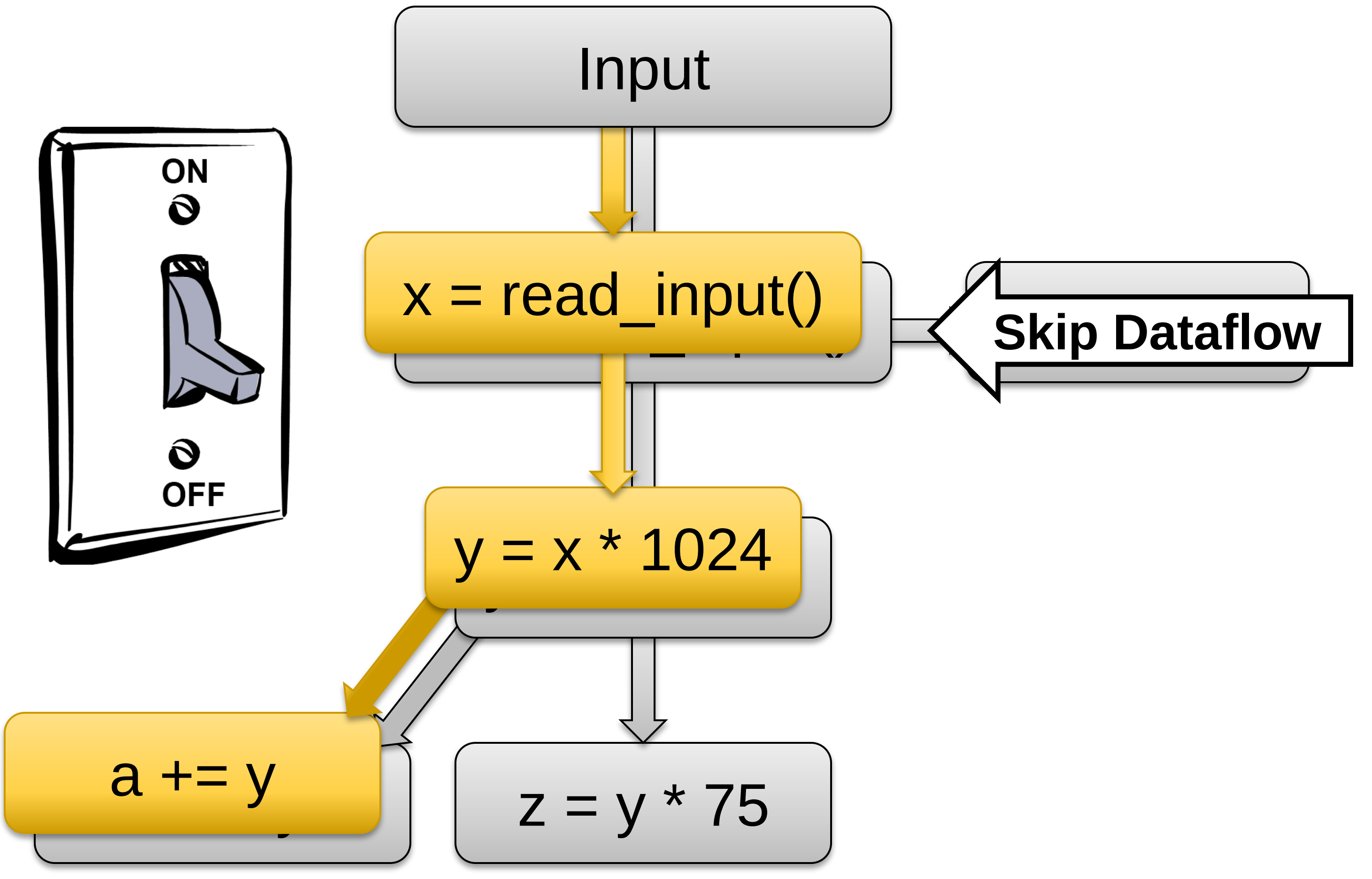
Dataflow Sampling Example



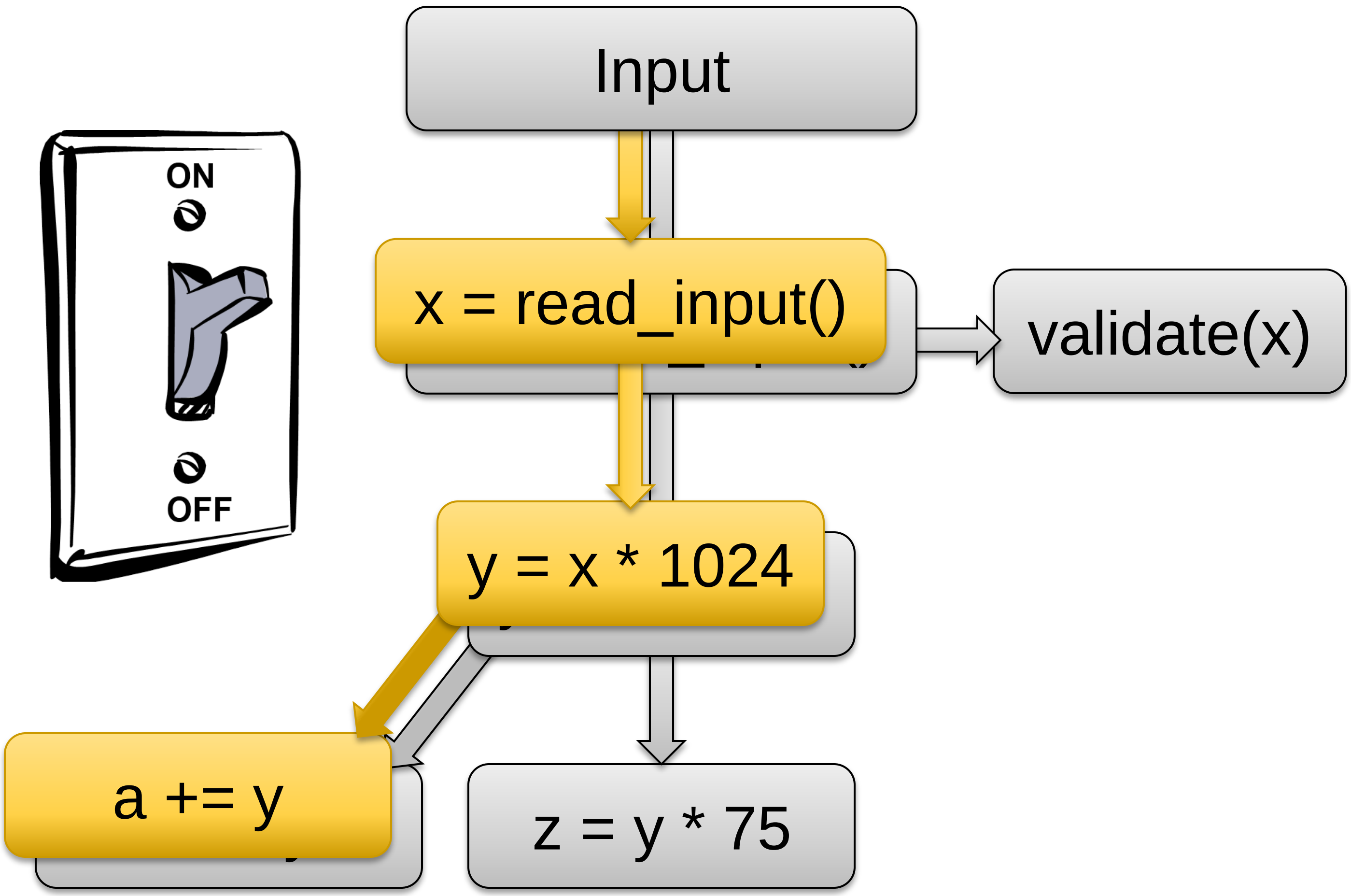
Dataflow Sampling Example



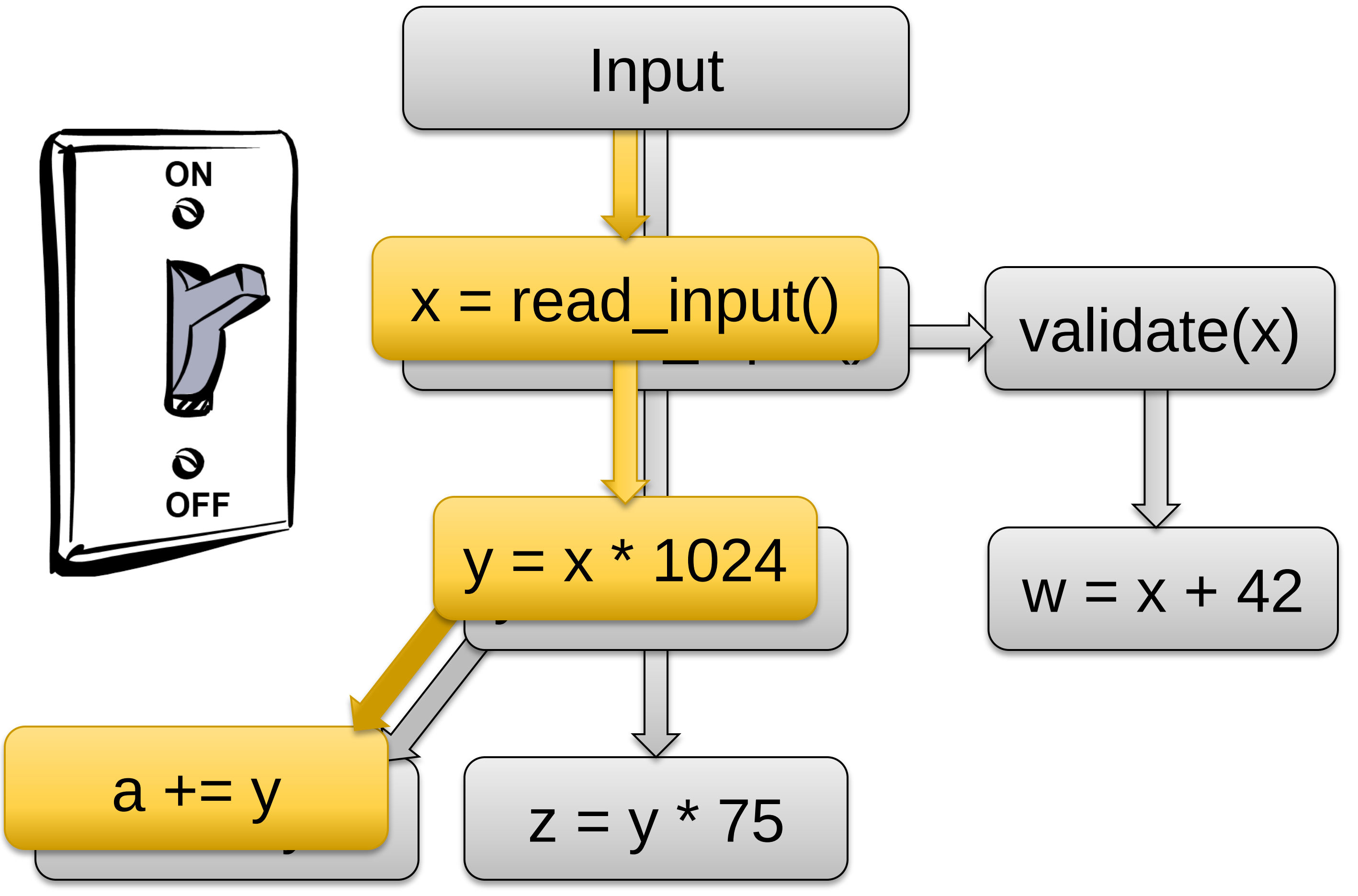
Dataflow Sampling Example



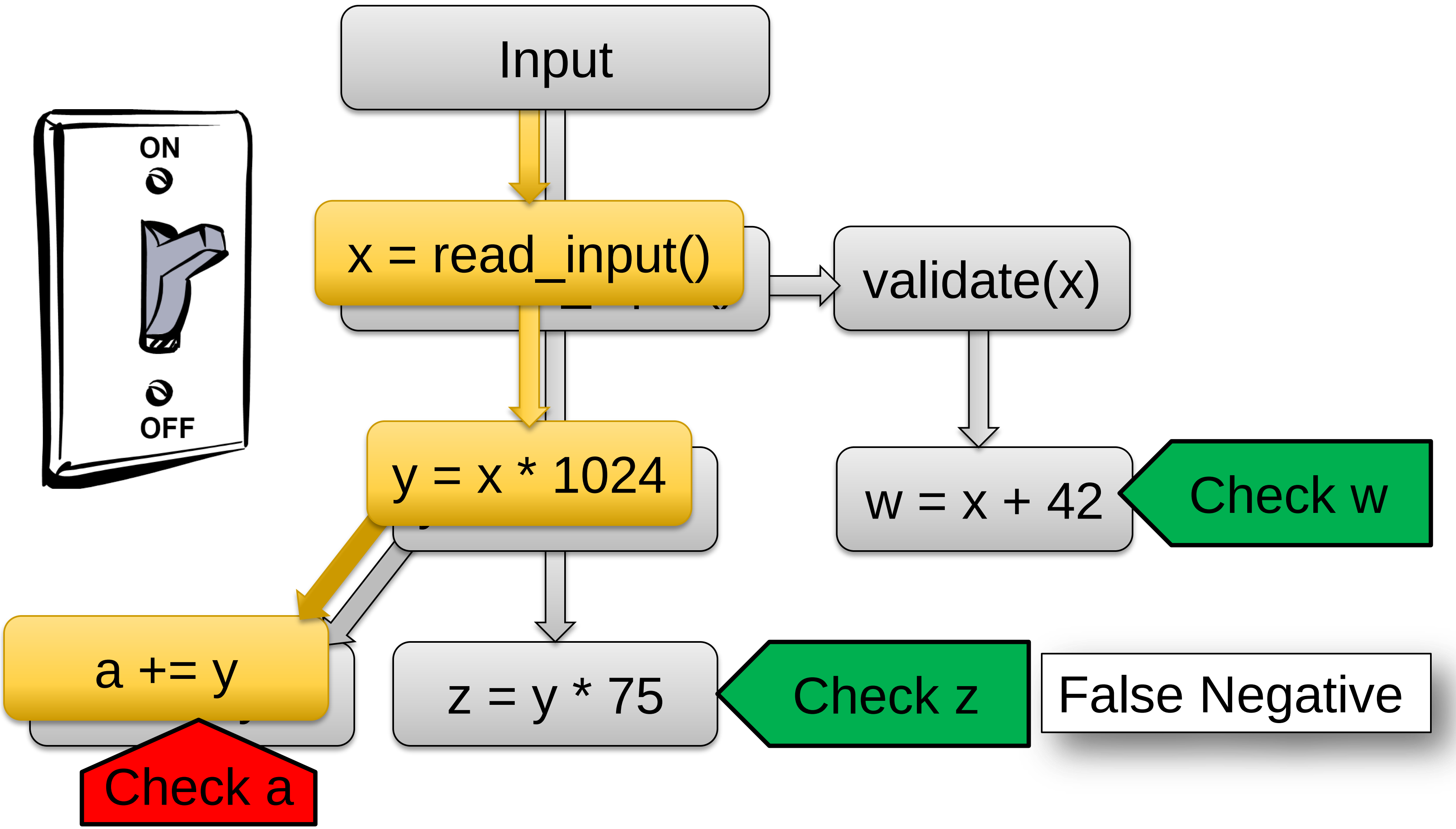
Dataflow Sampling Example



Dataflow Sampling Example



Dataflow Sampling Example

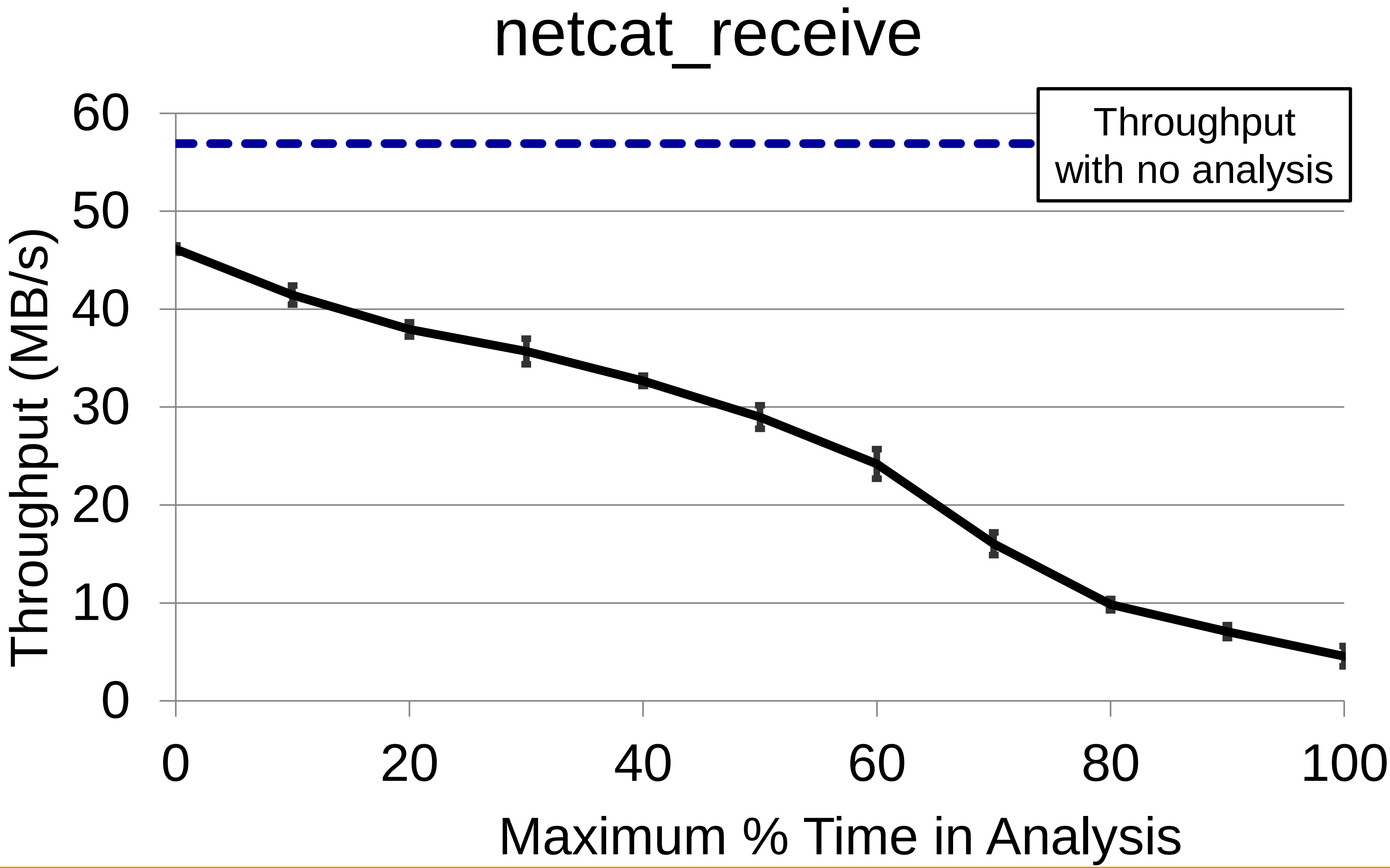


Benchmarks

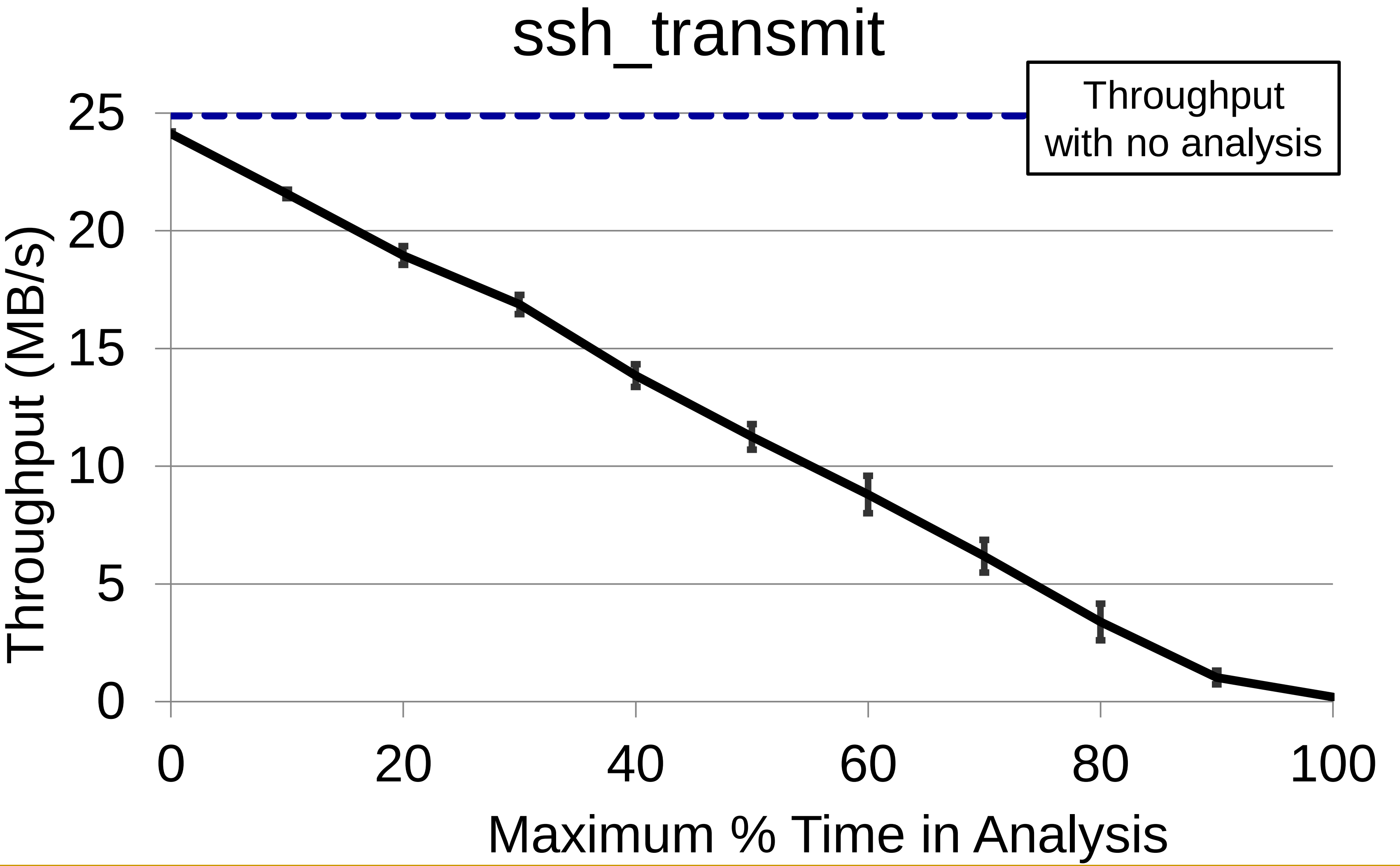
- Performance – Network Throughput
  - *Example:* ***ssh_receive***
- Accuracy of Sampling Analysis
  - Real-world Security Exploits

Name	Error Description
Apache	Stack overflow in Apache Tomcat JK Connector
Eggdrop	Stack overflow in Eggdrop IRC bot
Lynx	Stack overflow in Lynx web browser
ProFTPD	Heap smashing attack on ProFTPD Server
Squid	Heap smashing attack on Squid proxy server

Performance of Dataflow Sampling (2)



Performance of Dataflow Sampling (3)



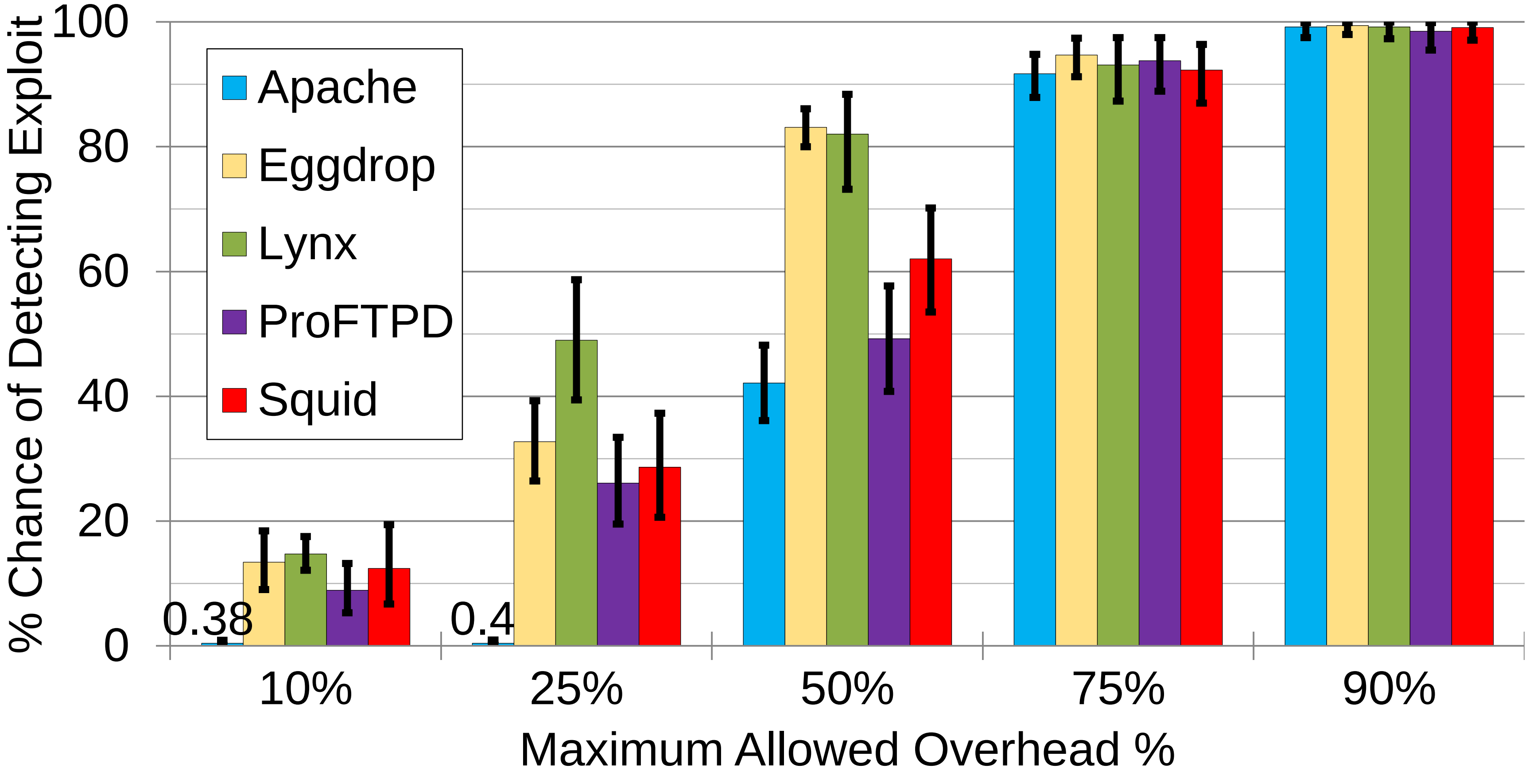
Accuracy at Very Low Overhead

- Max time in analysis: 1% every 10 seconds
- Always stop analysis after threshold
  - Lowest probability of detecting exploits

Name	Chance of Detecting Exploit
Apache	100%
Eggdrop	100%
Lynx	100%
ProFTPD	100%
Squid	100%

Accuracy with Background Tasks

netcat_receive running with benchmark

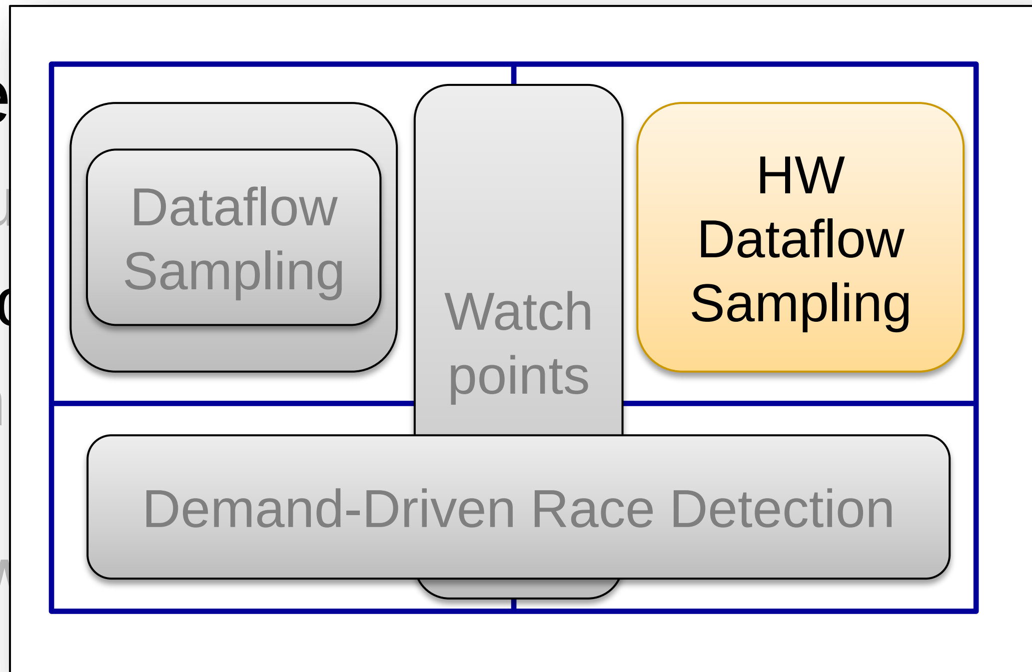


Outline

- Problem Statement
- Proposed Solutions
  - Distributed Dynamic Dataflow Analysis
  - Testudo: Hardware-Based Dataflow Sampling
  - Demand-Driven Data Race Detection
- Future Work
- Timeline

Outline

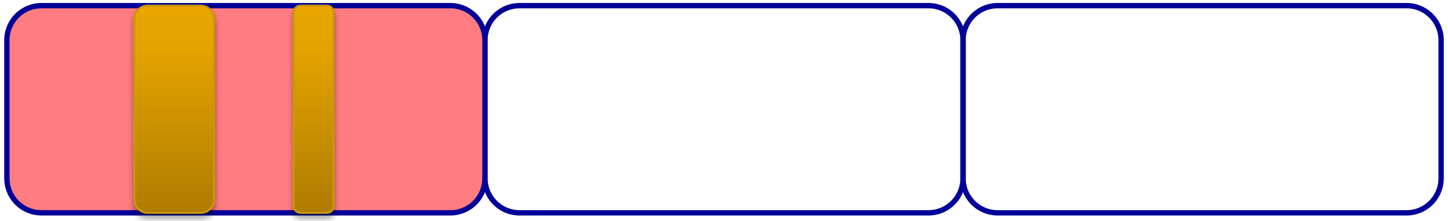
- Problem Statement
- Propose
  - Distribu
  - Testudo
  - Deman



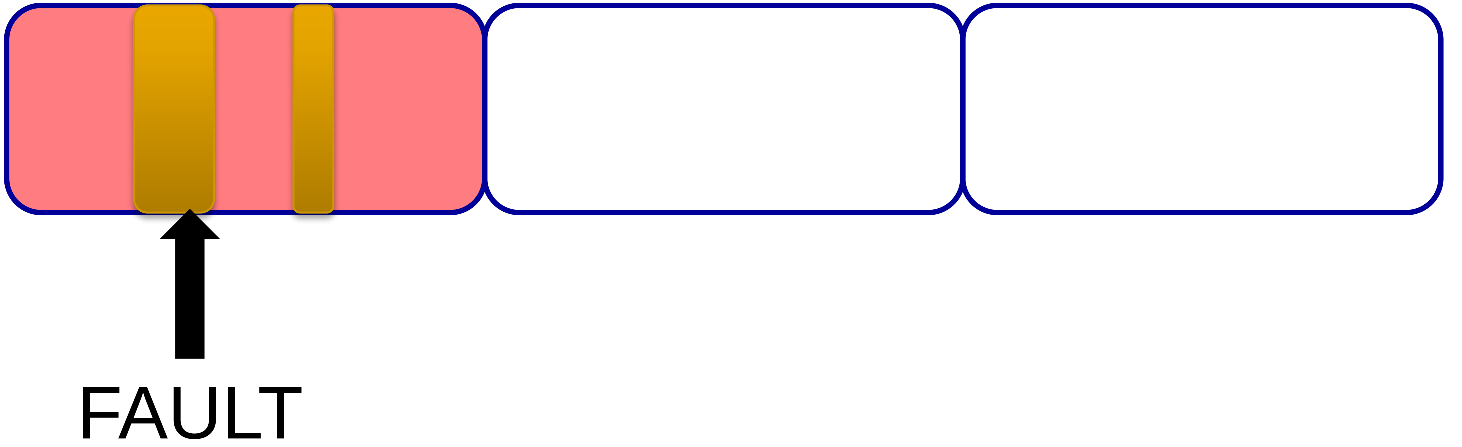
pling

- Future W
- Timeline

Virtual Memory Not Ideal

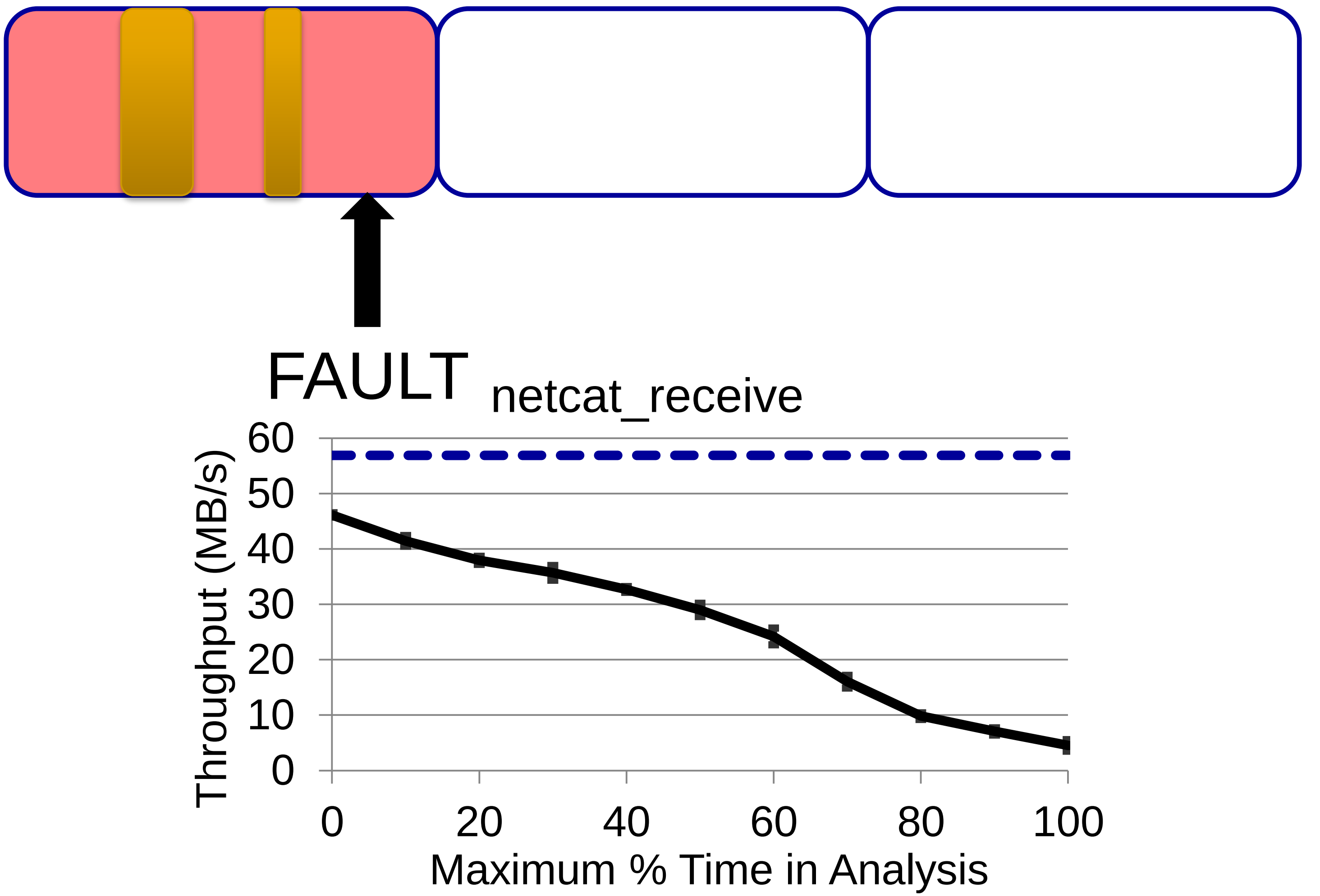


Virtual Memory Not Ideal

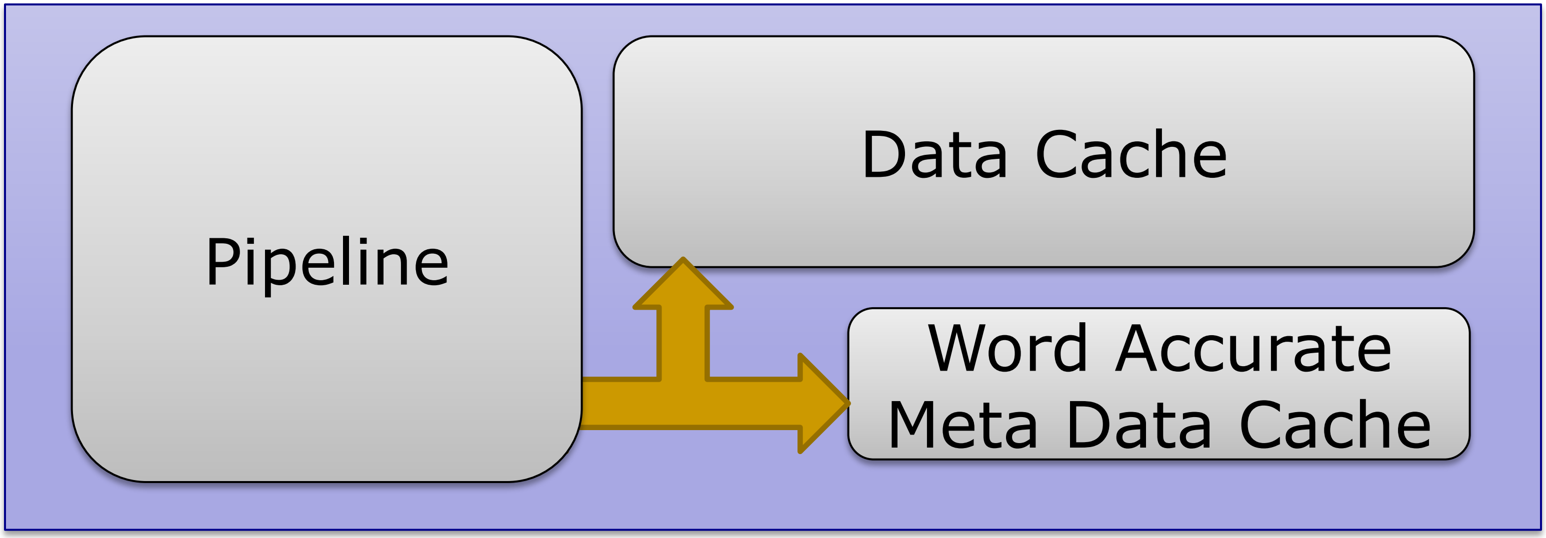


Virtual Memory Not Ideal

Virtual Memory Not Ideal

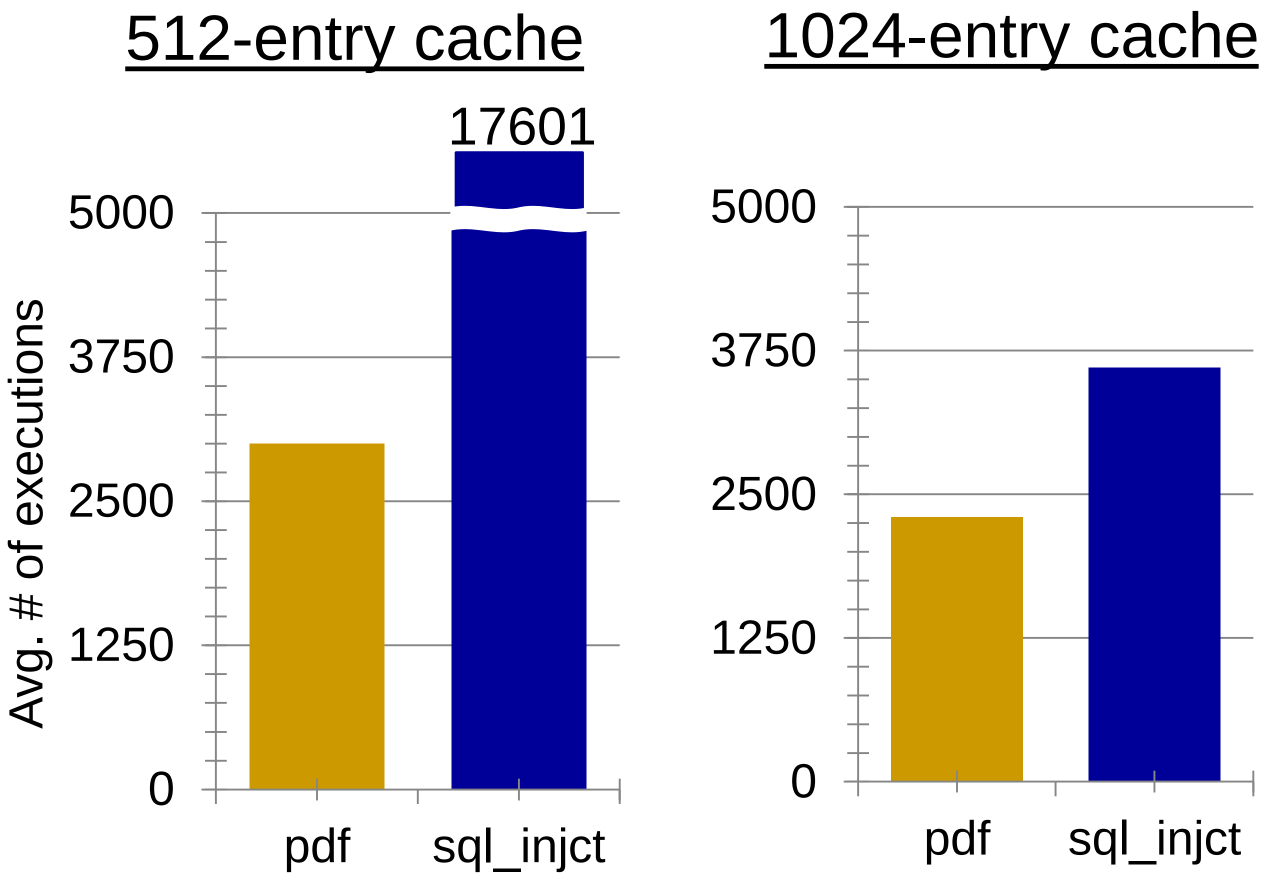


Word Accurate Meta-Data



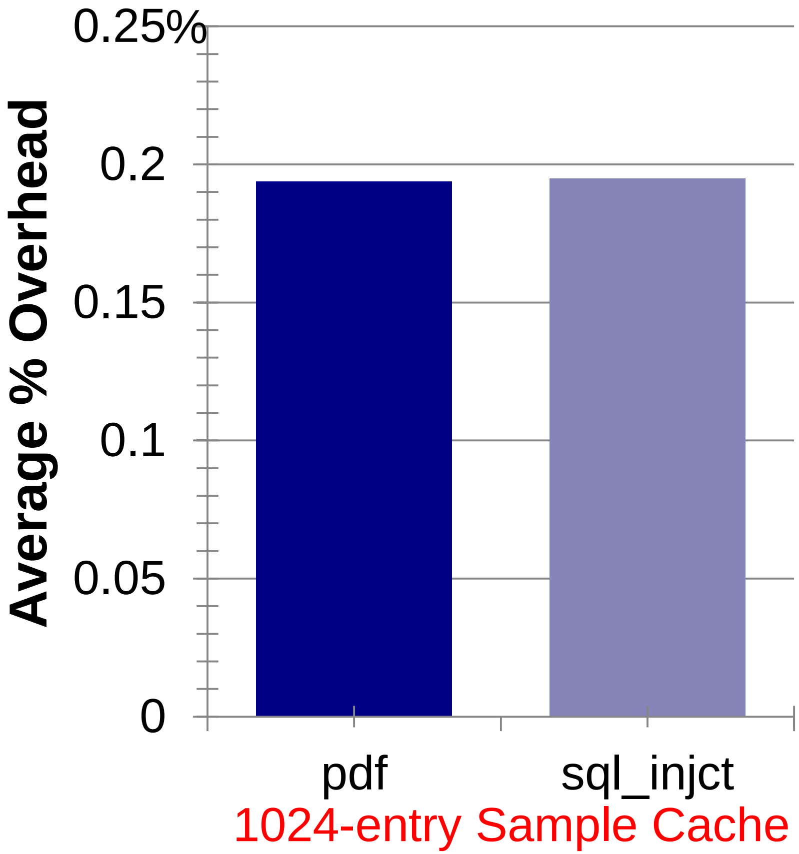
- What happens when the cache overflows?
  - Increase the size of main memory?
  - Store into virtual memory?
- **Use Sampling to Throw Away Data**

On-Chip Sampling Mechanism



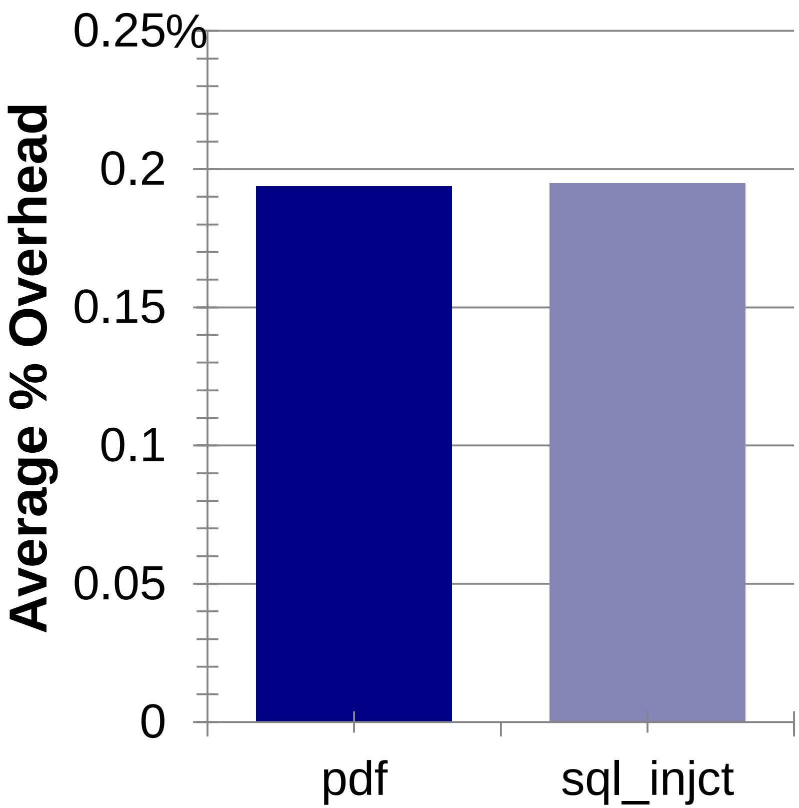
Useful for Scaling to Complex Analyses

If each shadow operation uses 1000 instructions:

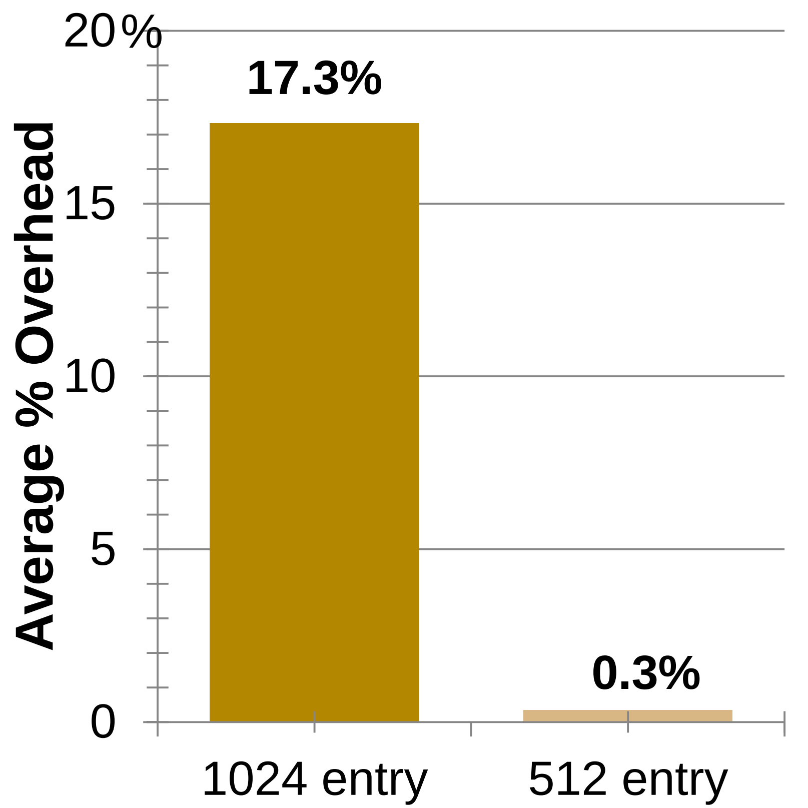


Useful for Scaling to Complex Analyses

If each shadow operation uses 1000 instructions:



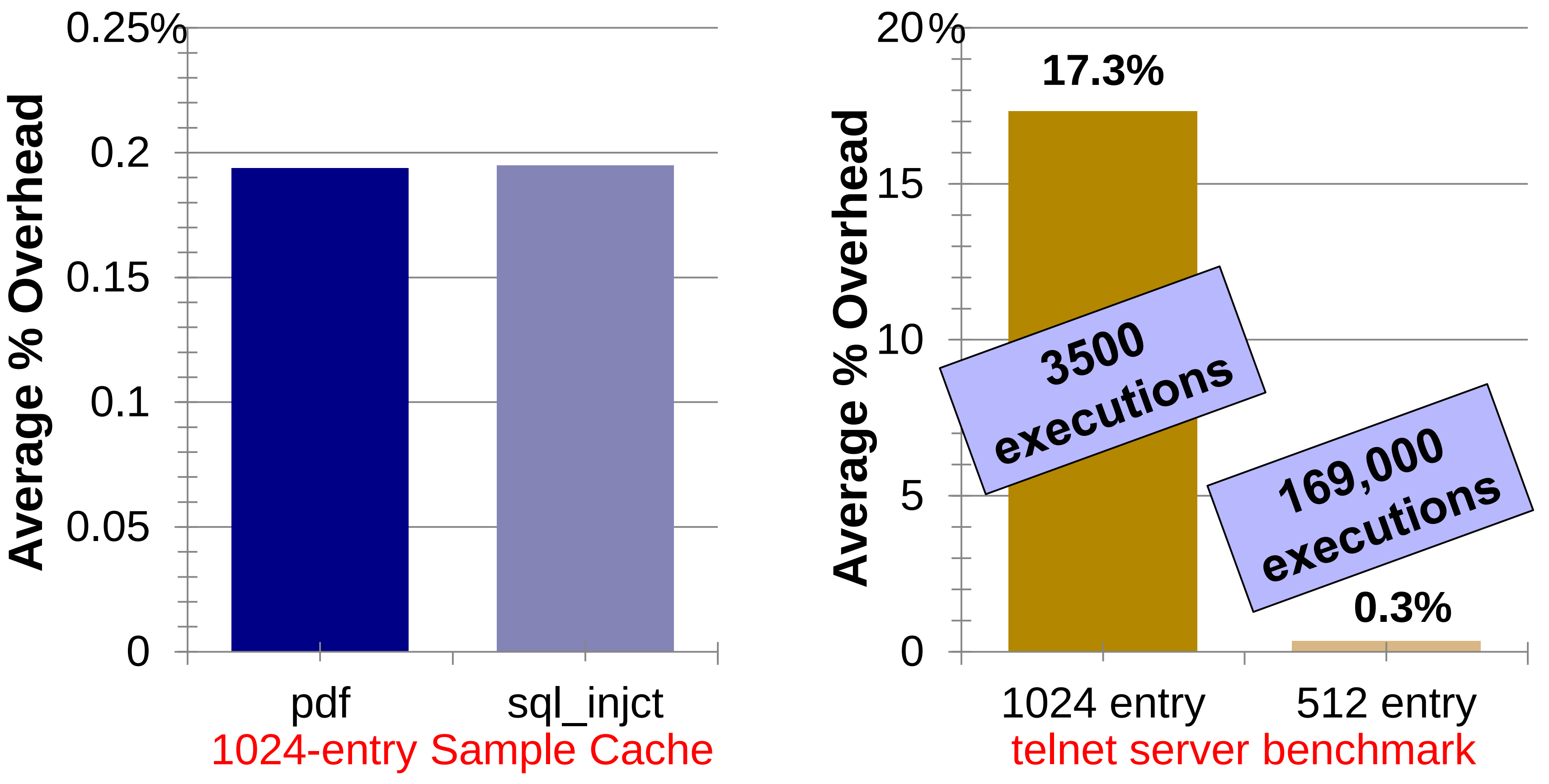
1024-entry Sample Cache



telnet server benchmark

Useful for Scaling to Complex Analyses

If each shadow operation uses 1000 instructions:



Example of Data Race Detection

TIME

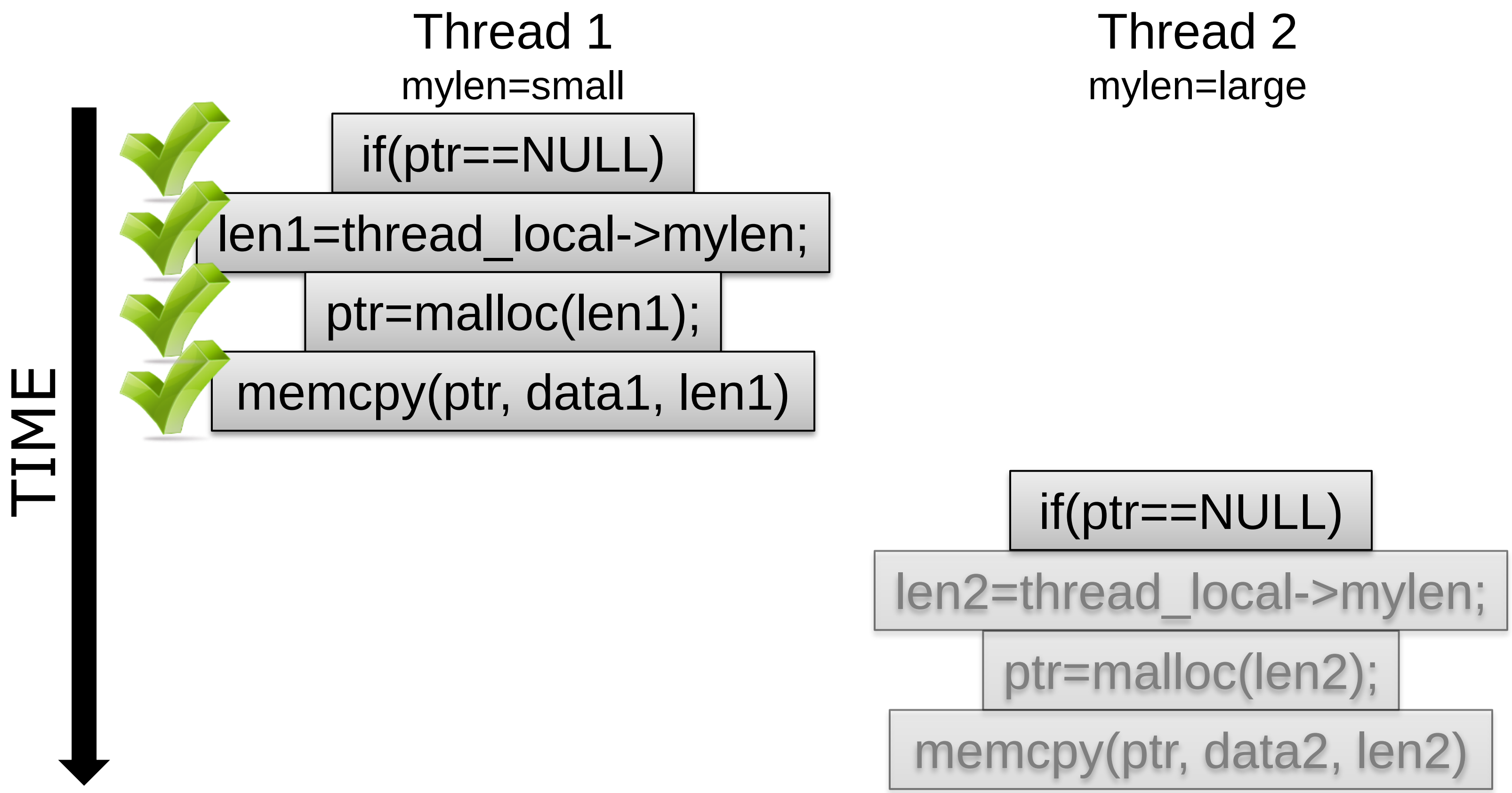
Thread 1
mylen=small

```
if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)
```

Thread 2
mylen=large

```
if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
```

Example of Data Race Detection



Example of Data Race Detection

TIME

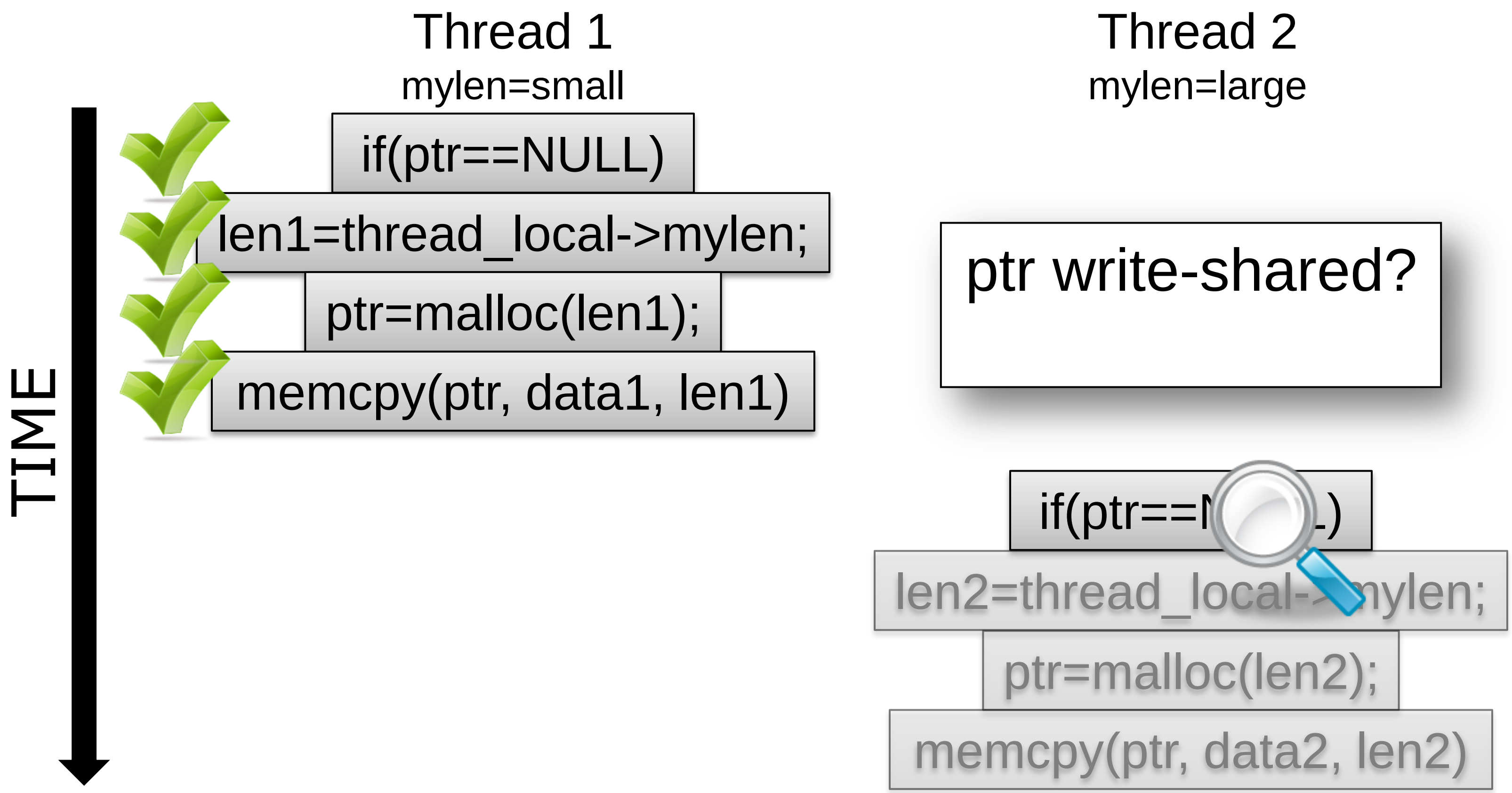
Thread 1
mylen=small

```
if(ptr==NULL)
len1=thread_local->mylen;
ptr=malloc(len1);
memcpy(ptr, data1, len1)
```

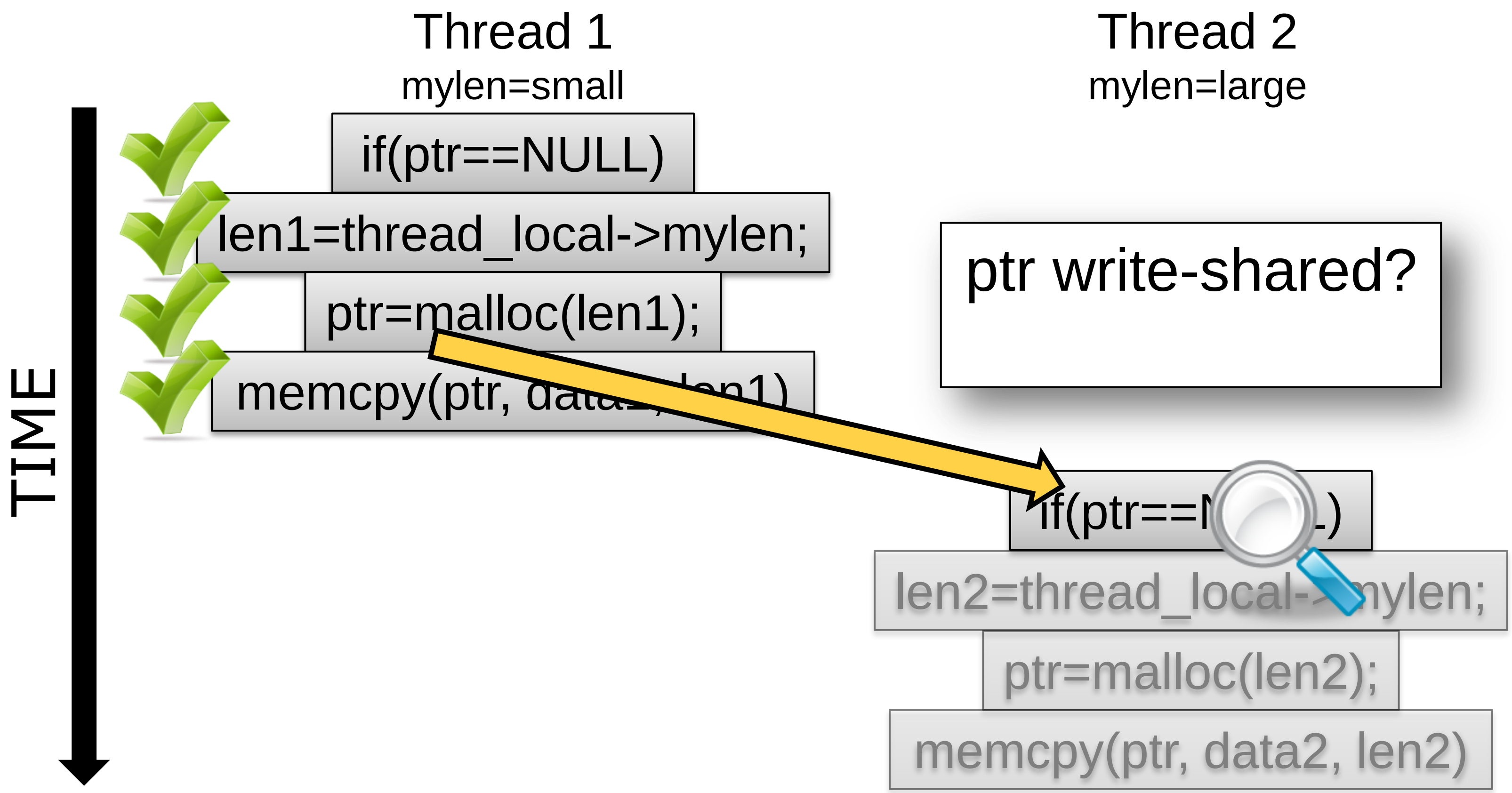
Thread 2
mylen=large

```
if(ptr==NULL)
len2=thread_local->mylen;
ptr=malloc(len2);
memcpy(ptr, data2, len2)
```

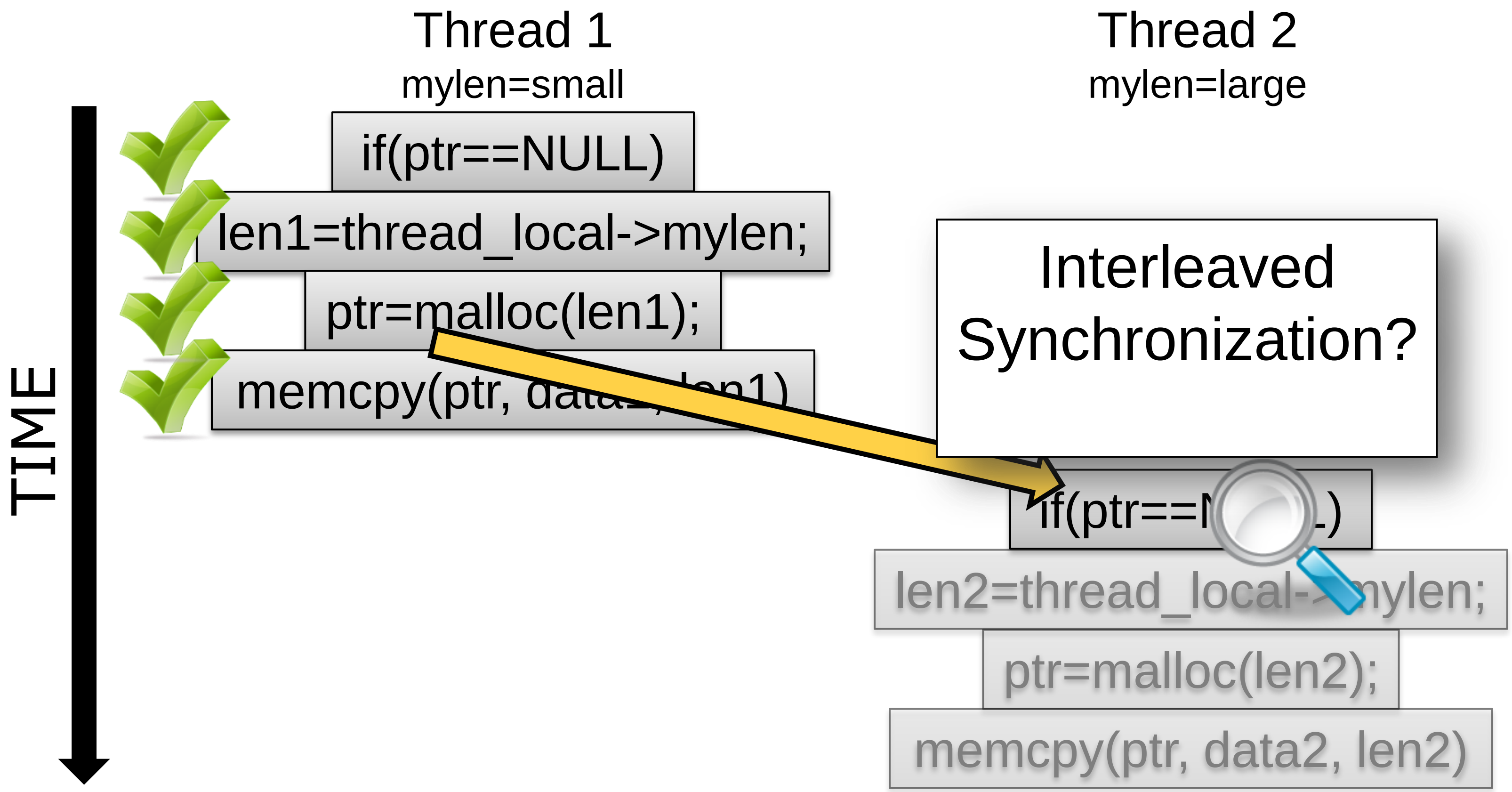
Example of Data Race Detection



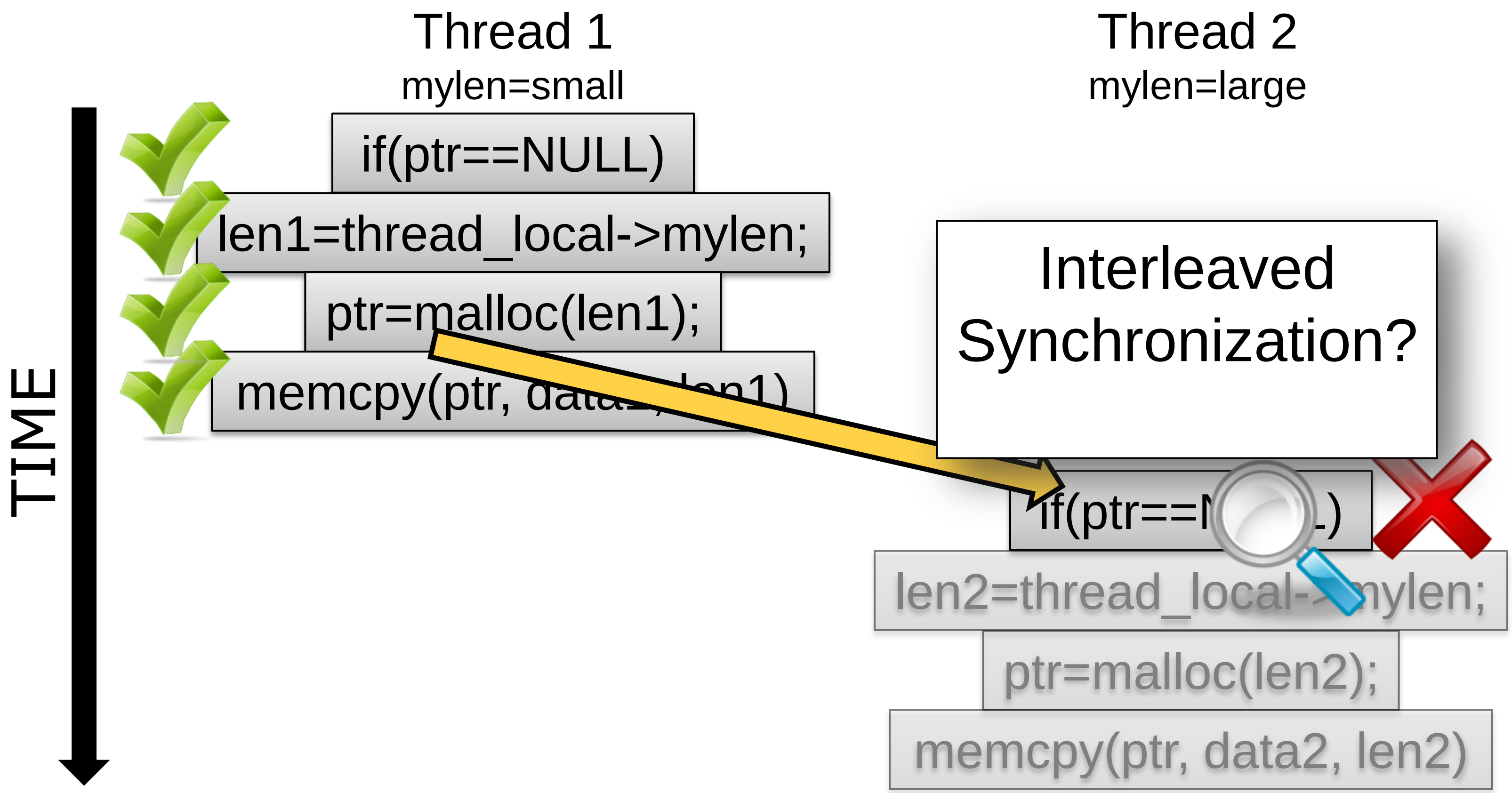
Example of Data Race Detection



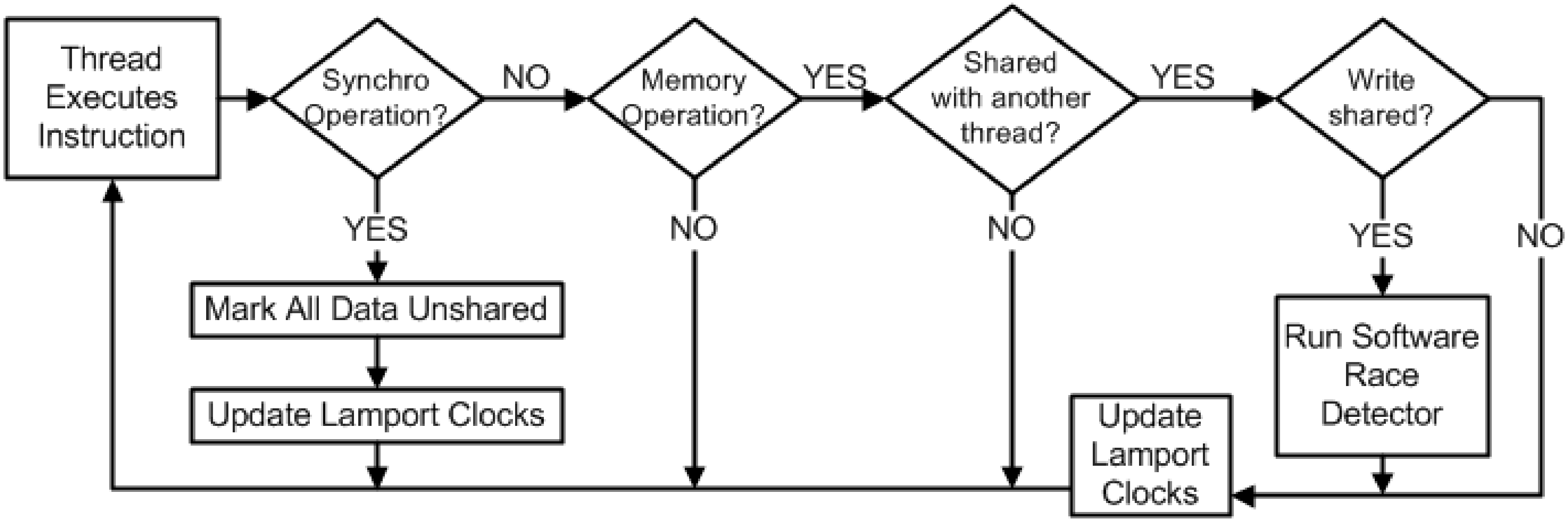
Example of Data Race Detection



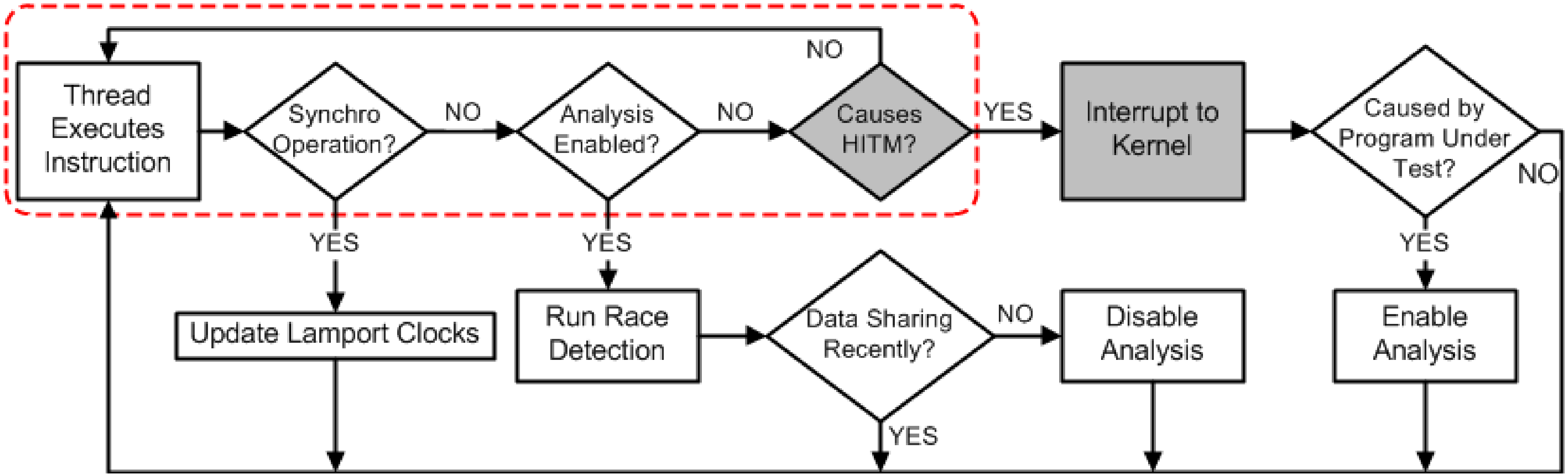
Example of Data Race Detection



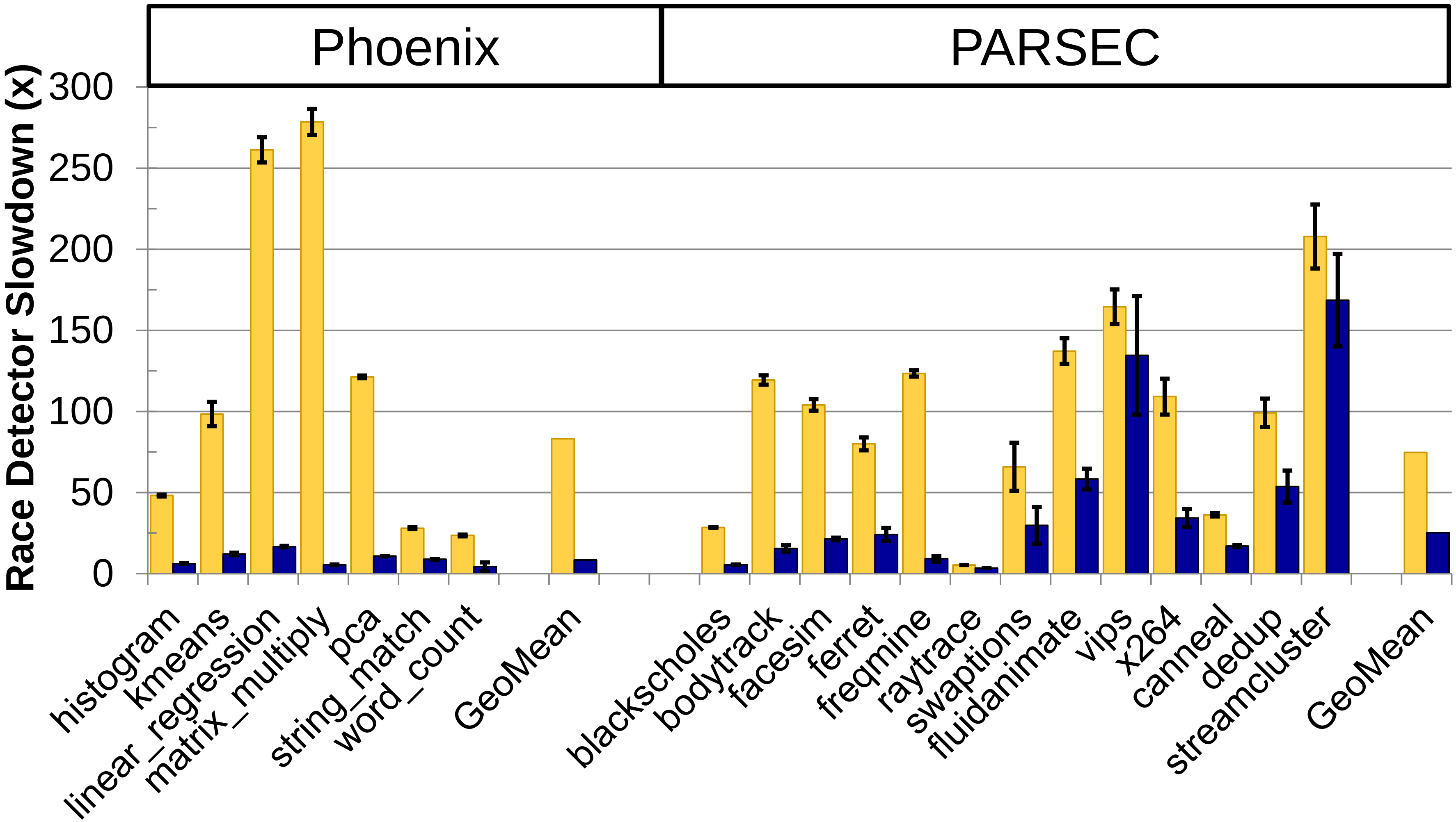
Demand-Driven Analysis Algorithm



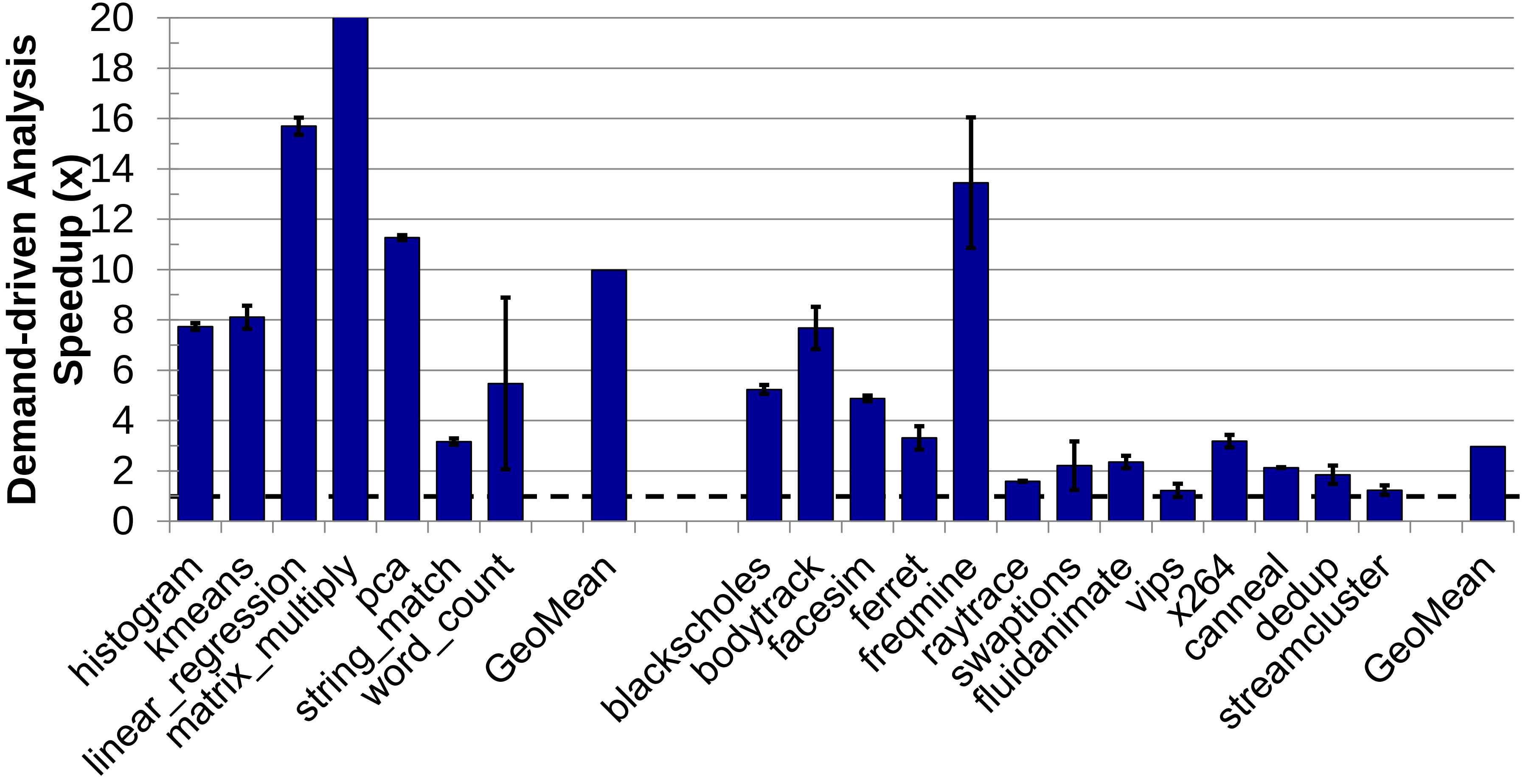
Demand-Driven Analysis on Real HW



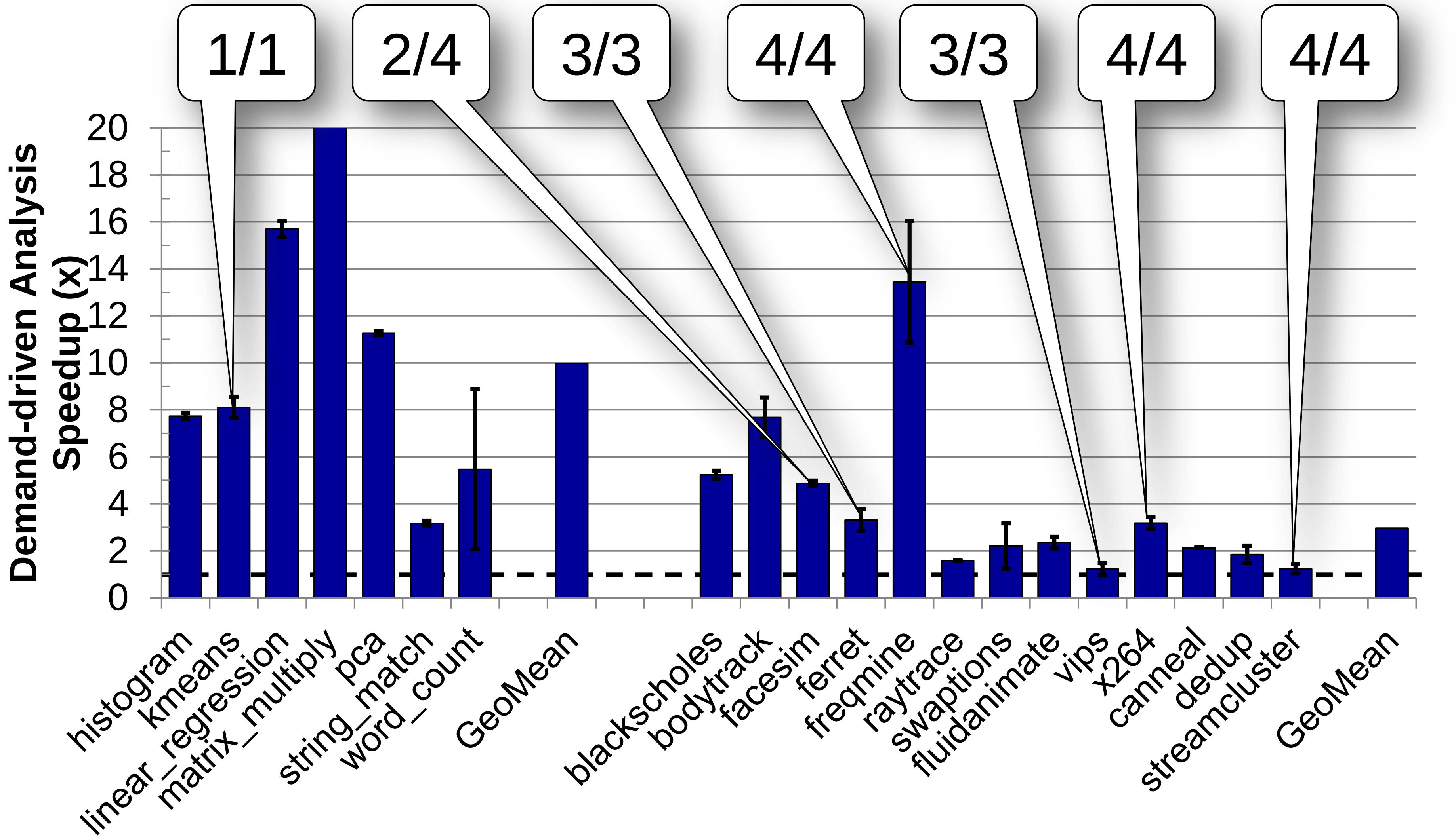
Performance Difference



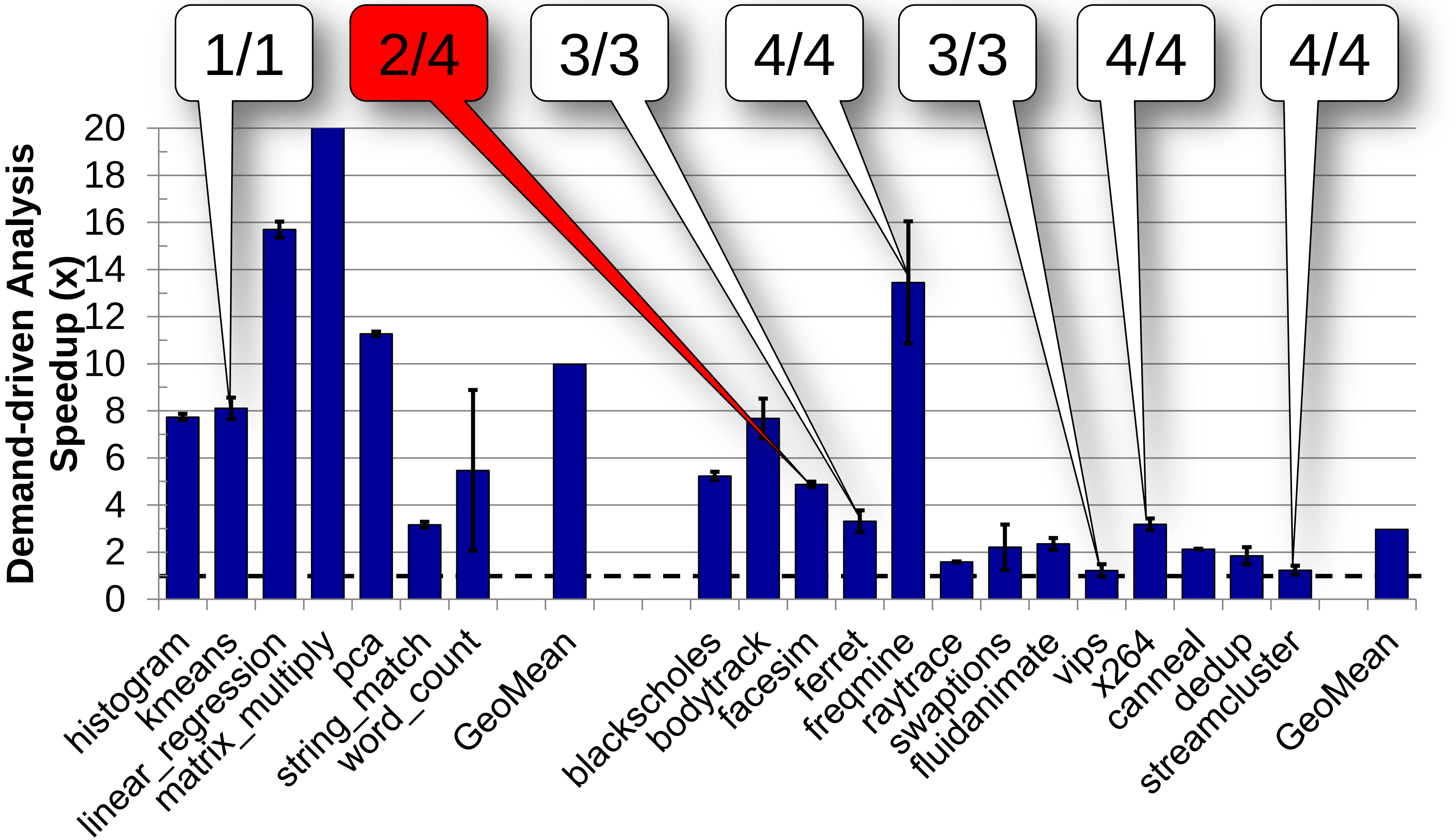
Demand-Driven Analysis Accuracy



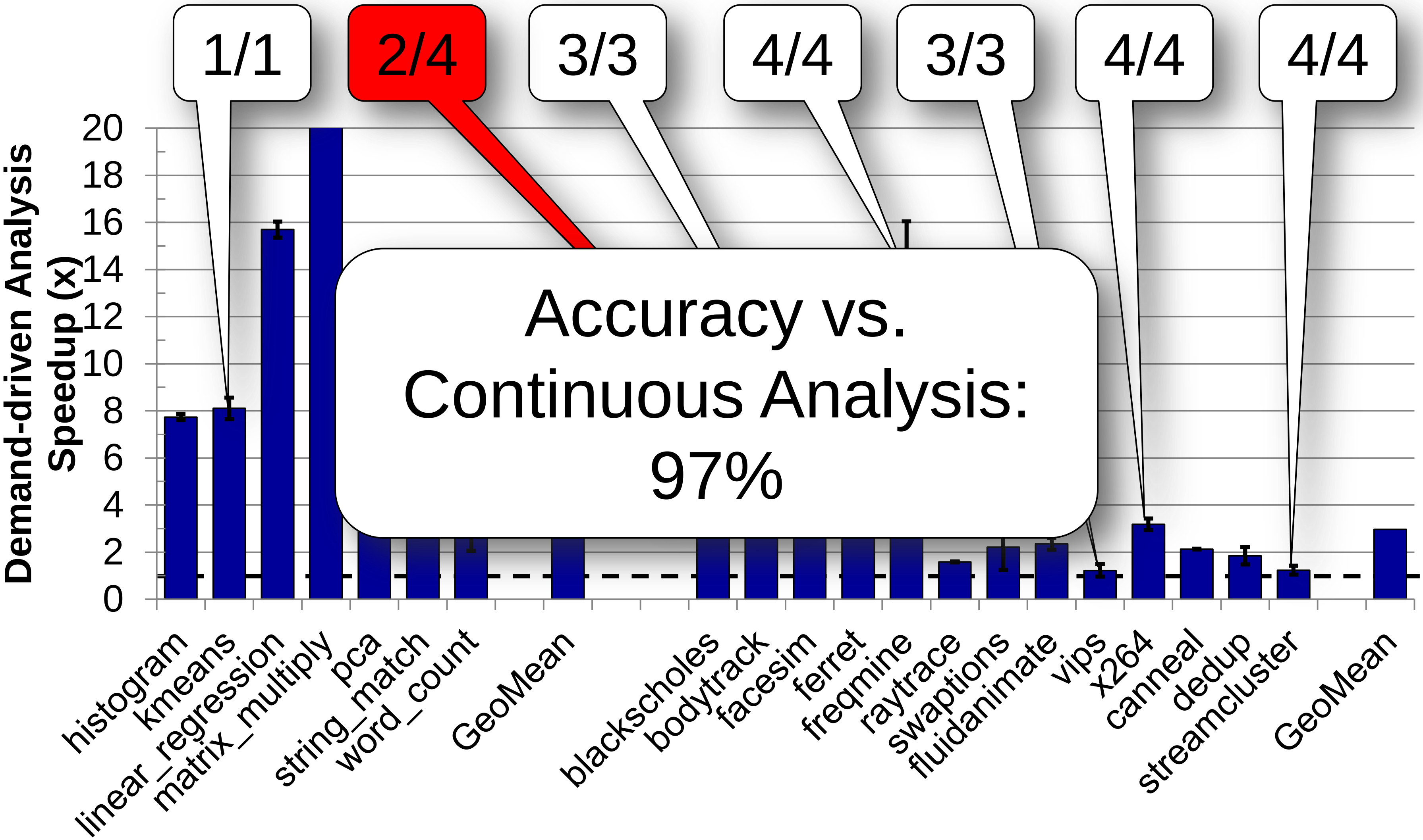
Demand-Driven Analysis Accuracy



Demand-Driven Analysis Accuracy



Demand-Driven Analysis Accuracy



Accuracy on Real Hardware

	kmeans	facesim	ferret	freqmine	vips	x264	streamcluster
W→W	1/1 (100%)	0/1 (0%)	-	-	1/1 (100%)	-	1/1 (100%)
R→W	-	0/1 (0%)	2/2 (100%)	2/2 (100%)	1/1 (100%)	3/3 (100%)	1/1 (100%)
W→R	-	2/2 (100%)	1/1 (100%)	2/2 (100%)	1/1 (100%)	3/3/ (100%)	1/1 (100%)

	Spider Monkey-0	Spider Monkey-1	Spider Monkey-2	NSPR-1	Memcached-1	Apache-1
W→W	9/9 (100%)	1/1 (100%)	1/1 (100%)	3/3 (100%)	-	1/1 (100%)
R→W	3/3 (100%)	-	1/1 (100%)	1/1 (100%)	1/1 (100%)	7/7 (100%)
W→R	8/8 (100%)	1/1 (100%)	2/2 (100%)	4/4 (100%)	-	2/2 (100%)

Accuracy on Real Hardware

	kmeans	facesim	ferret	freqmine	vips	x264	streamcluster
W→W	1/1 (100%)	0/1 (0%)	-	-	1/1 (100%)	-	1/1 (100%)
R→W	-	0/1 (0%)	2/2 (100%)	2/2 (100%)	1/1 (100%)	3/3 (100%)	1/1 (100%)
W→R	-	2/2 (100%)	1/1 (100%)	2/2 (100%)	1/1 (100%)	3/3/ (100%)	1/1 (100%)

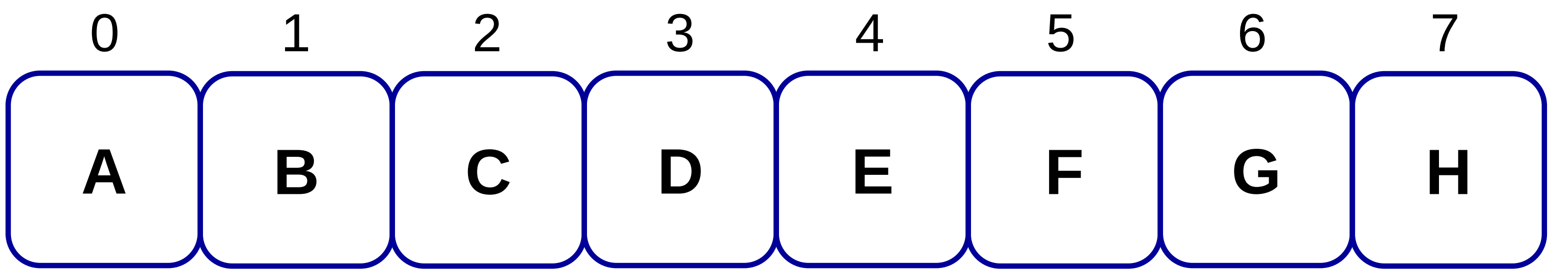
	Spider Monkey-0	Spider Monkey-1	Spider Monkey-2	NSPR-1	Memcached-1	Apache-1
W→W	9/9 (100%)	1/1 (100%)	1/1 (100%)	3/3 (100%)	-	1/1 (100%)
R→W	3/3 (100%)	-	1/1 (100%)	1/1 (100%)	1/1 (100%)	7/7 (100%)
W→R	8/8 (100%)	1/1 (100%)	2/2 (100%)	4/4 (100%)	-	2/2 (100%)

Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data

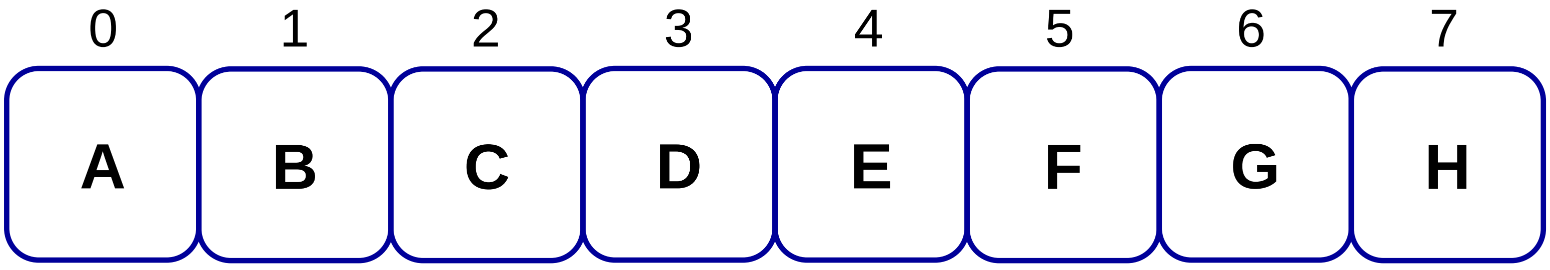
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



Hardware-Assisted Watchpoints

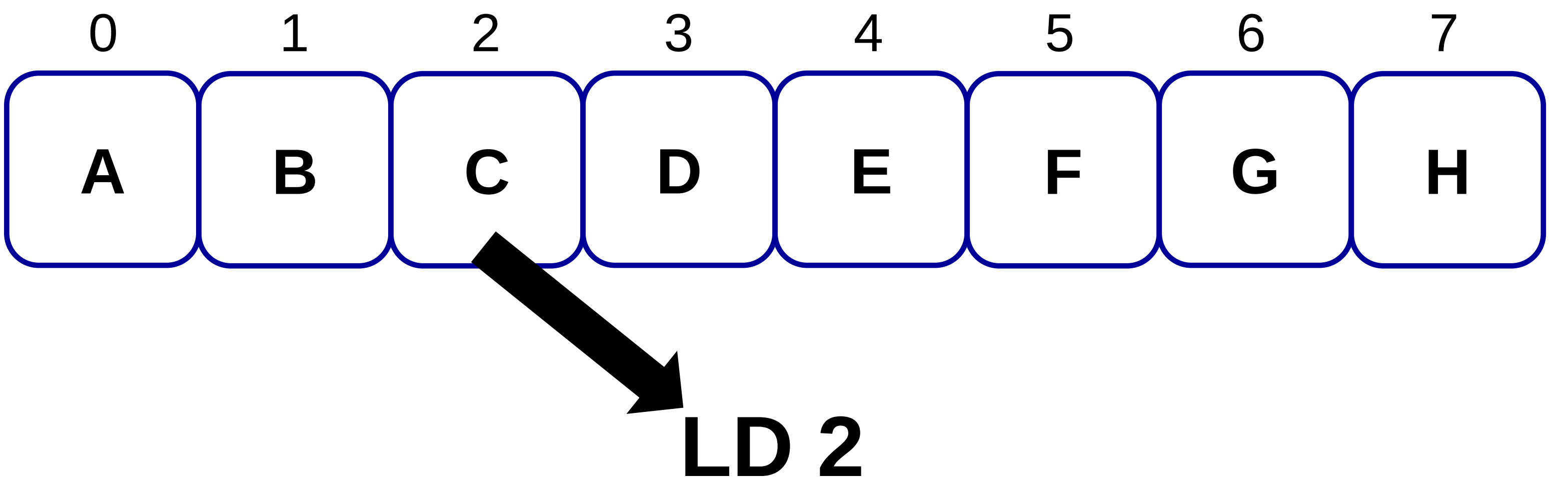
- HW Interrupt when touching watched data



LD 2

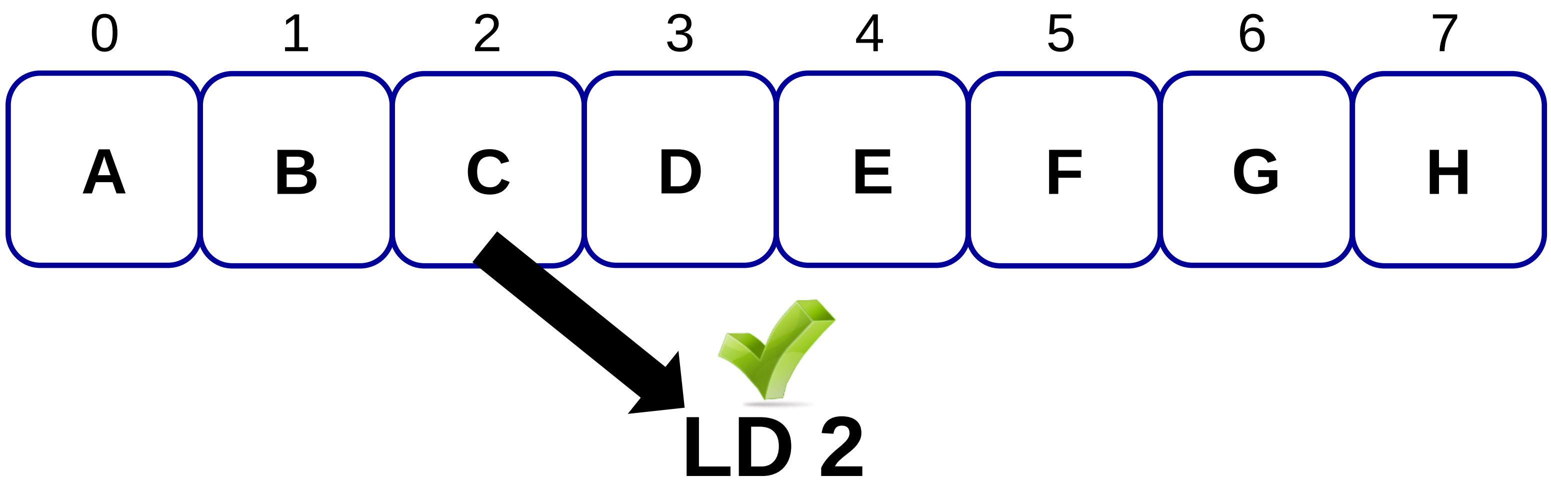
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



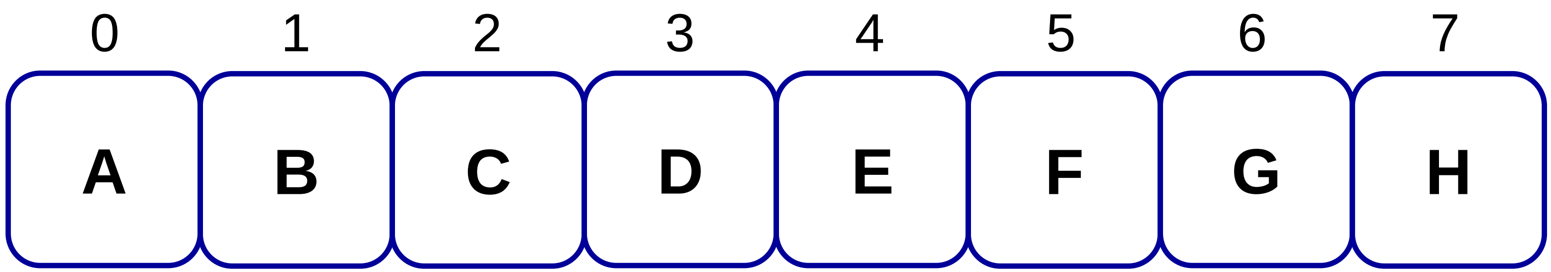
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



Hardware-Assisted Watchpoints

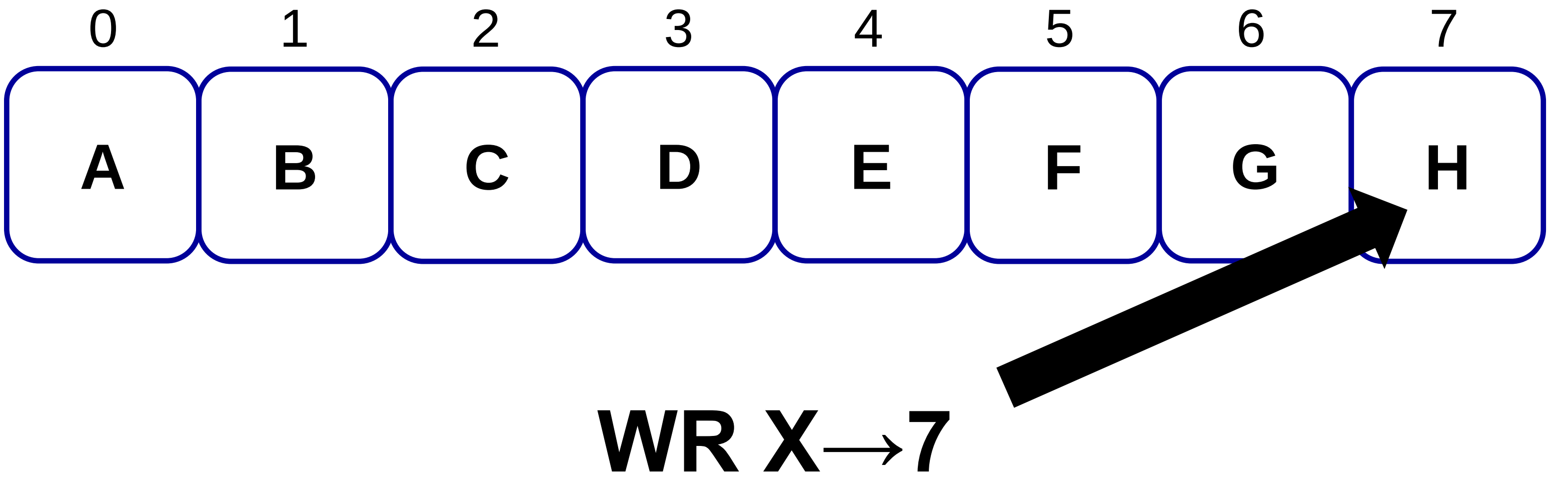
- HW Interrupt when touching watched data



WR X→7

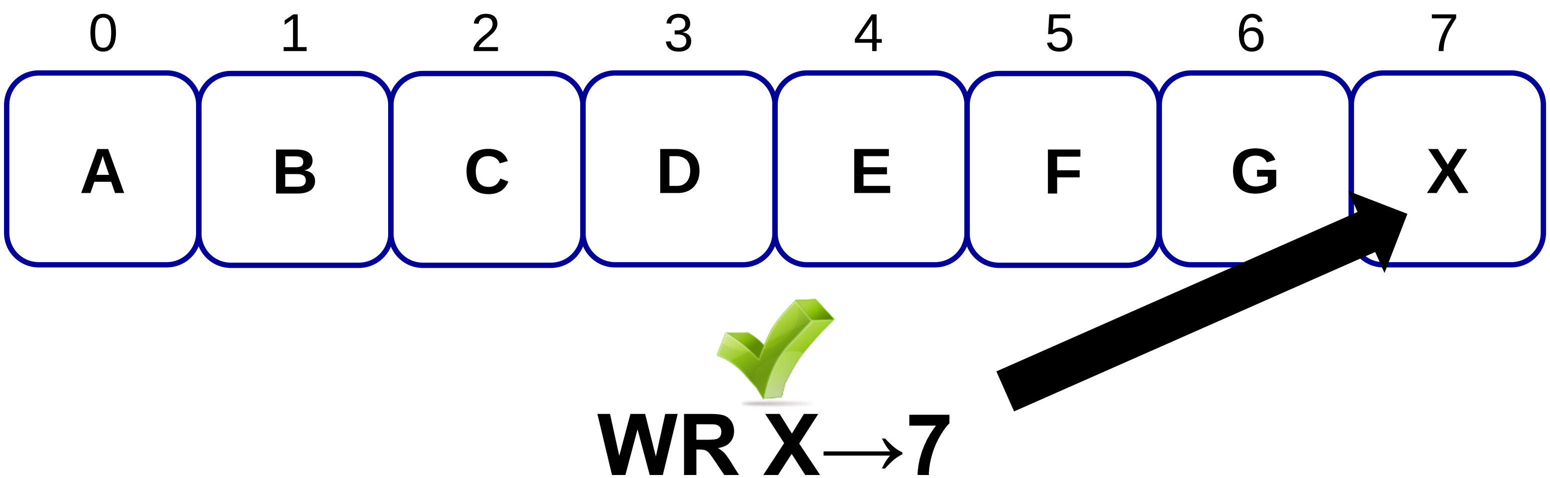
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



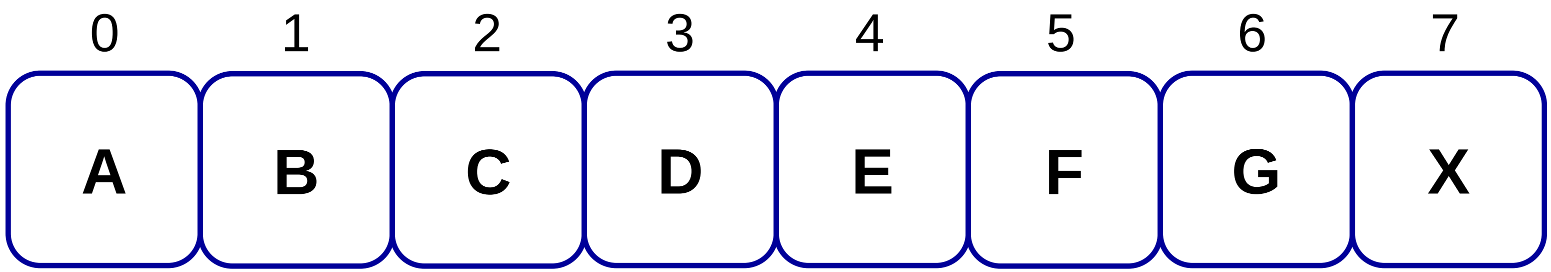
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



Hardware-Assisted Watchpoints

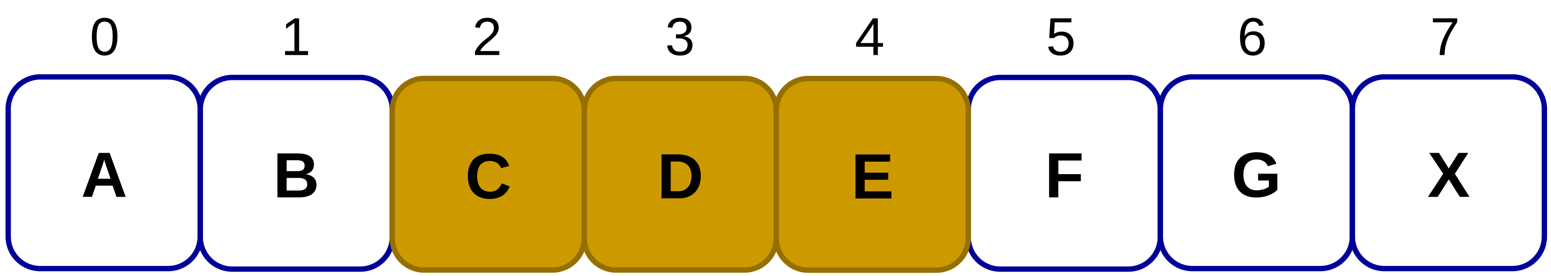
- HW Interrupt when touching watched data



R-Watch 2-4

Hardware-Assisted Watchpoints

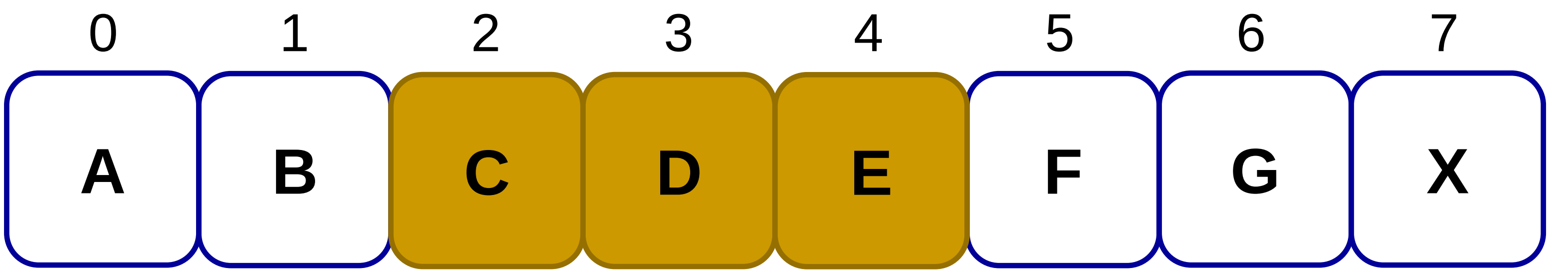
- HW Interrupt when touching watched data



R-Watch 2-4

Hardware-Assisted Watchpoints

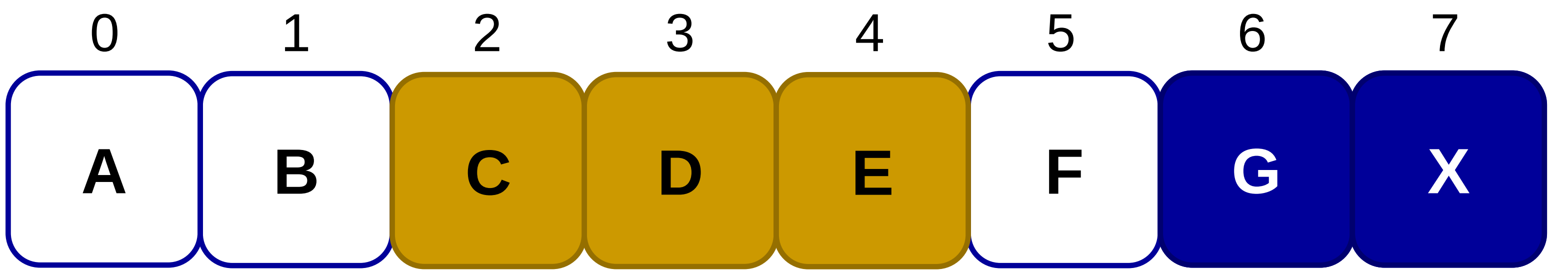
- HW Interrupt when touching watched data



W-Watch 6-7

Hardware-Assisted Watchpoints

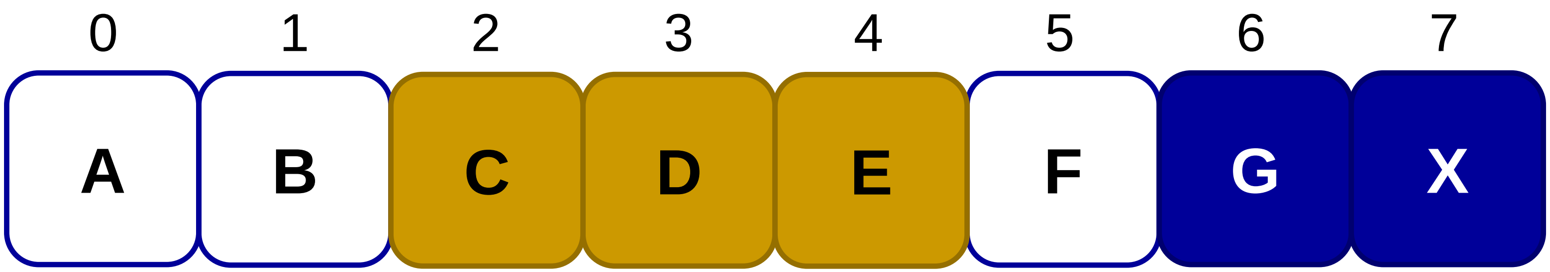
- HW Interrupt when touching watched data



W-Watch 6-7

Hardware-Assisted Watchpoints

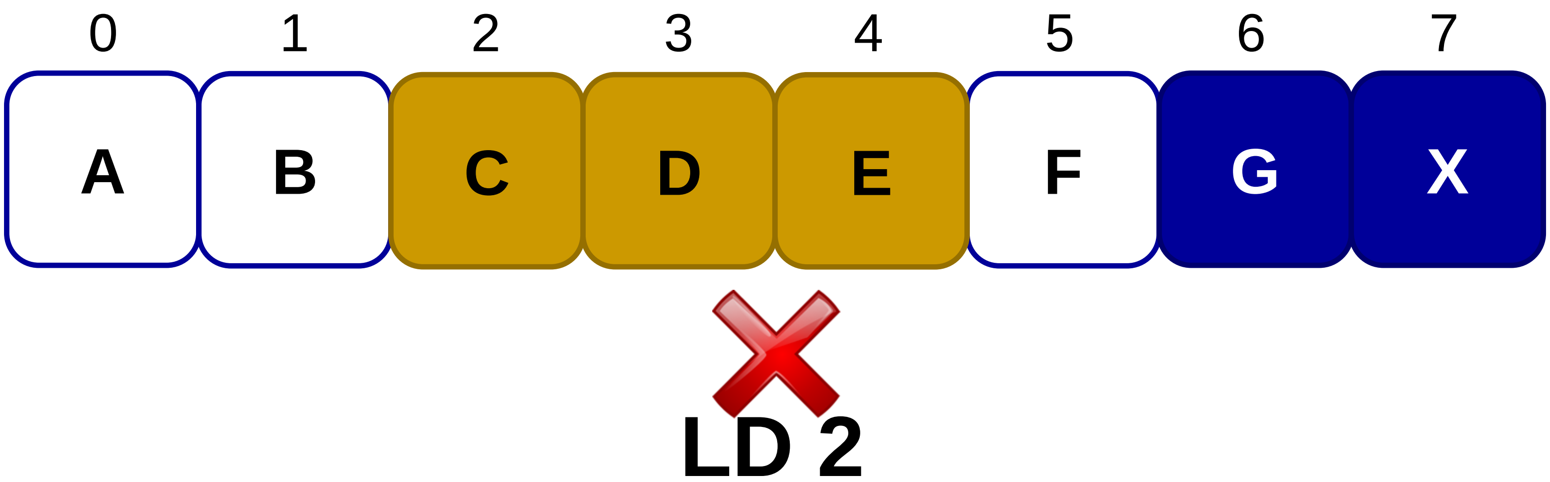
- HW Interrupt when touching watched data



LD 2

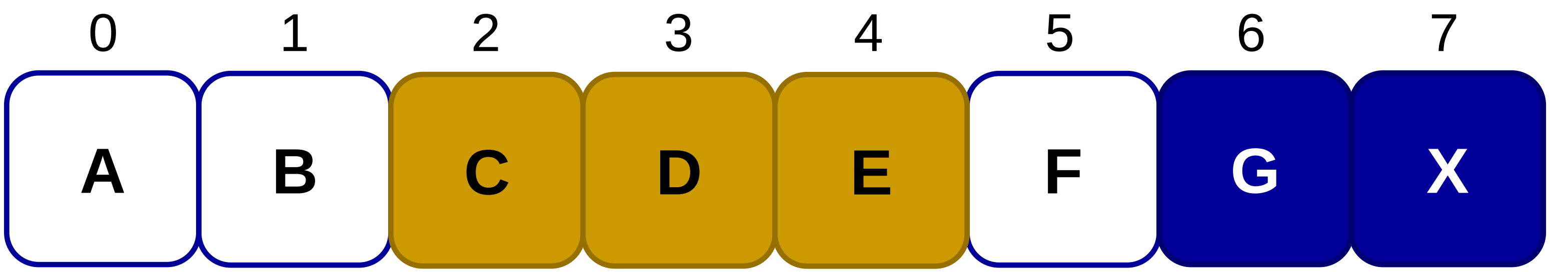
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



Hardware-Assisted Watchpoints

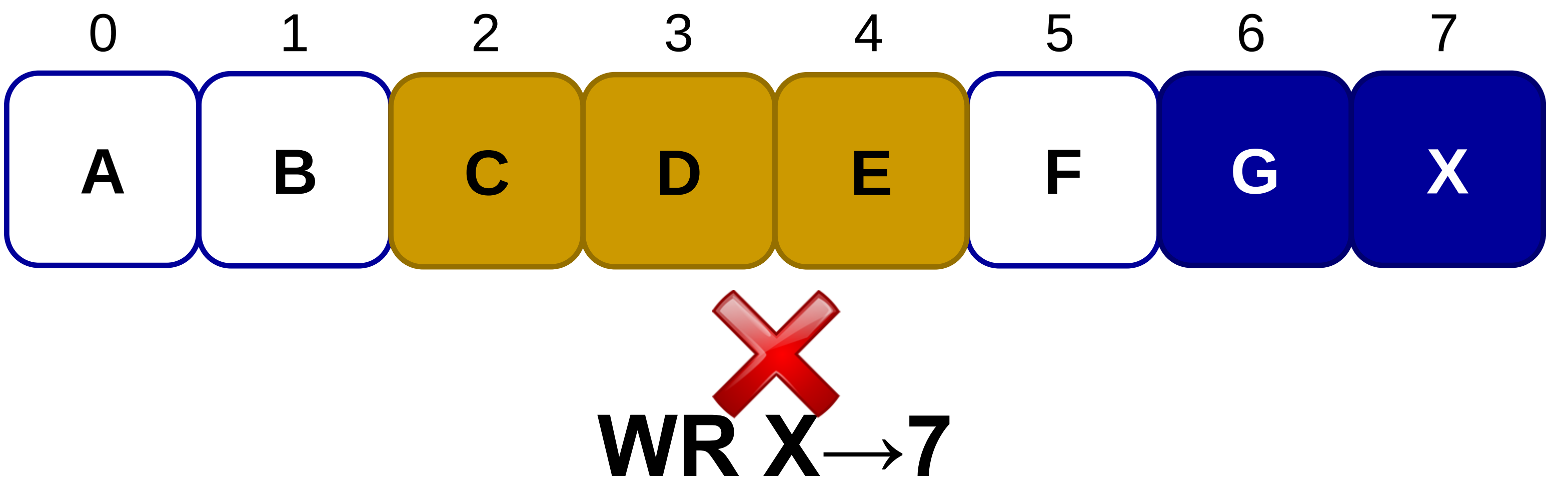
- HW Interrupt when touching watched data



WR X→7

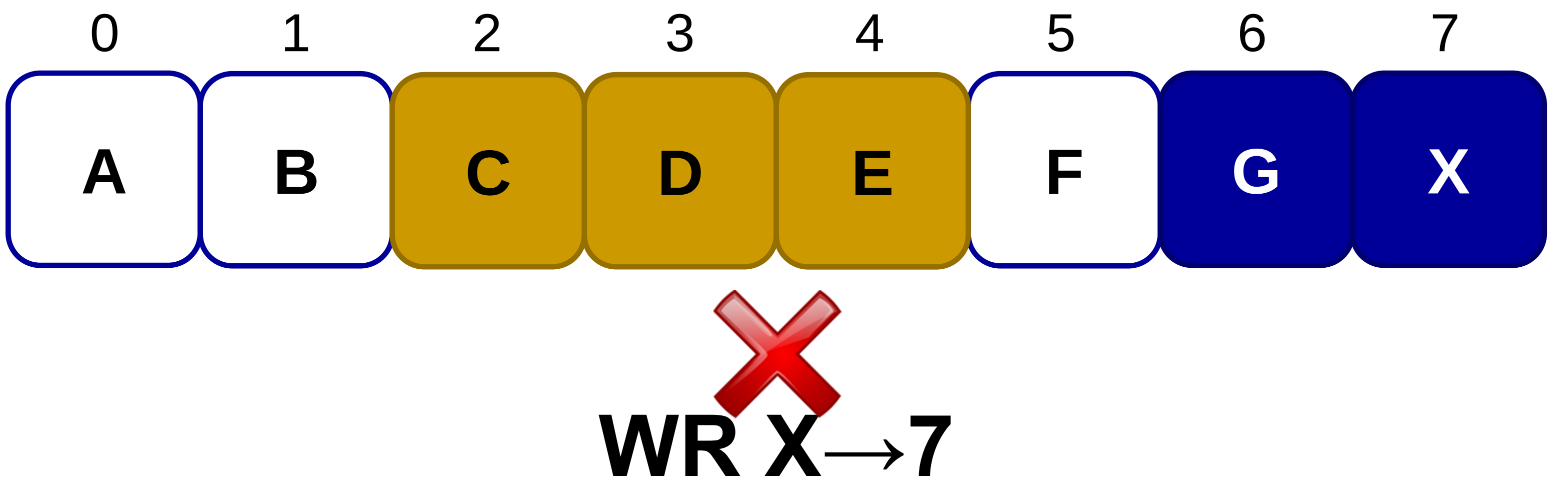
Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



Hardware-Assisted Watchpoints

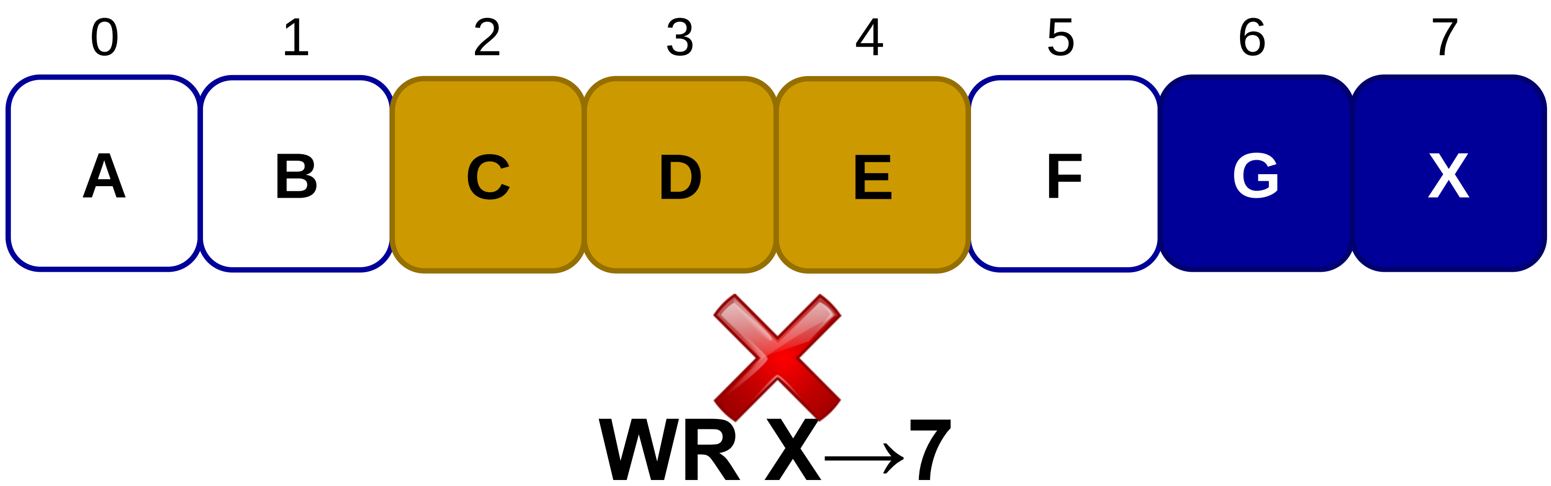
- HW Interrupt when touching watched data



- SW knows it's touching important data

Hardware-Assisted Watchpoints

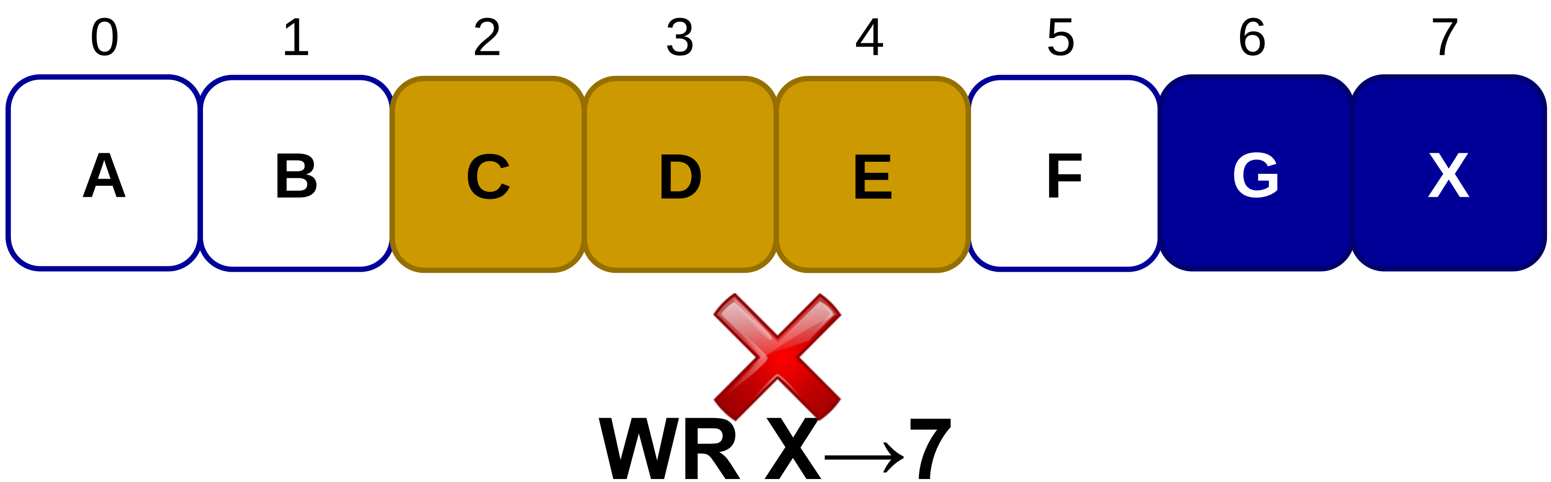
- HW Interrupt when touching watched data



- SW knows it's touching important data
  - AT NO OVERHEAD

Hardware-Assisted Watchpoints

- HW Interrupt when touching watched data



- SW knows it's touching important data
  - AT NO OVERHEAD
- Normally used for debugging

Existing Watchpoint Solutions

- Watchpoint Registers
  - Limited number (4-16), small reach (4-8 bytes)

Existing Watchpoint Solutions

- Watchpoint Registers
  - Limited number (4-16), small reach (4-8 bytes)

- Virtual Memory
  - Coarse-grained, per-process, *only* aligned ranges

Existing Watchpoint Solutions

- Watchpoint Registers
  - Limited number (4-16), small reach (4-8 bytes)
- Virtual Memory
  - Coarse-grained, per-process, *only* aligned ranges
- ECC Mangling
  - Per physical address, all cores, no ranges

Meeting These Requirements

- Unlimited Number of Watchpoints
  - Store in memory, cache on chip
- Fine-Grained
  - Watch full virtual addresses
- Per-Thread
  - Watchpoints cached per core/thread
  - TID Registers
- Ranges
  - **Range Cache**

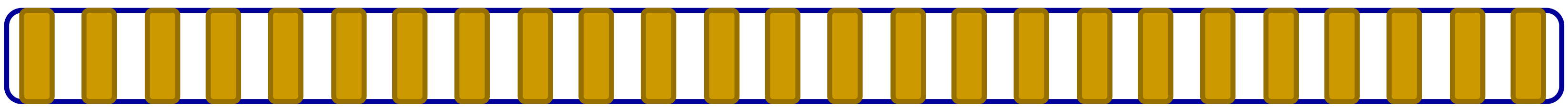
The Need for Many Small Ranges

- Some watchpoints better suited for ranges
  - 32b Addresses: 2 ranges x 64b each = **16B**

The Need for Many Small Ranges

- Some watchpoints better suited for ranges

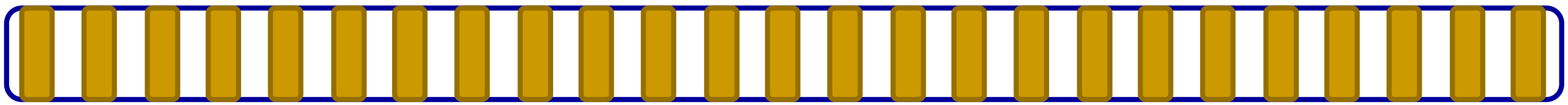
  - 32b Addresses: 2 ranges x 64b each = **16B**
- Some need large # of small watchpoints



The Need for Many Small Ranges

- Some watchpoints better suited for ranges

  - 32b Addresses: 2 ranges x 64b each = **16B**
- Some need large # of small watchpoints

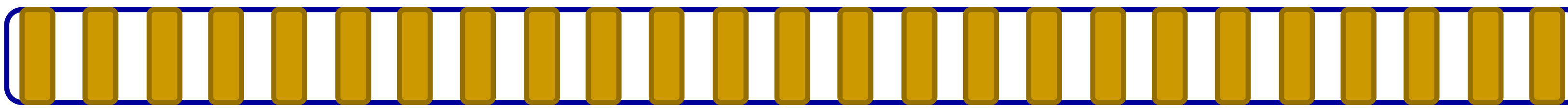


  - 51 ranges x 64b each = **408B**
  - Better stored as bitmap? 51 bits!

The Need for Many Small Ranges

- Some watchpoints better suited for ranges

  - 32b Addresses: 2 ranges x 64b each = **16B**
- Some need large # of small watchpoints

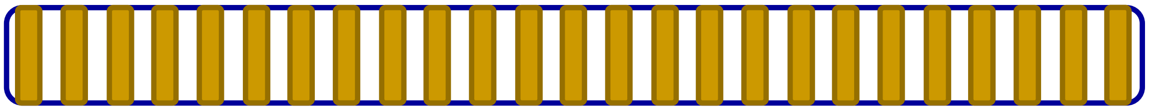


  - 51 ranges x 64b each = **408B**
  - Better stored as bitmap? 51 bits!

- Taint analysis has good ranges
- Byte-accurate race detection does not..

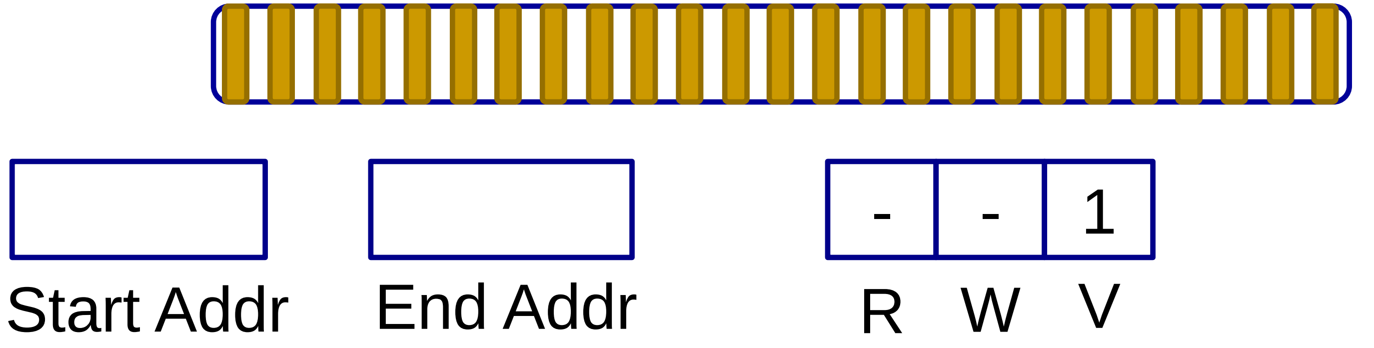
Watchpoint System Design II

- Make some RC entries point to bitmaps



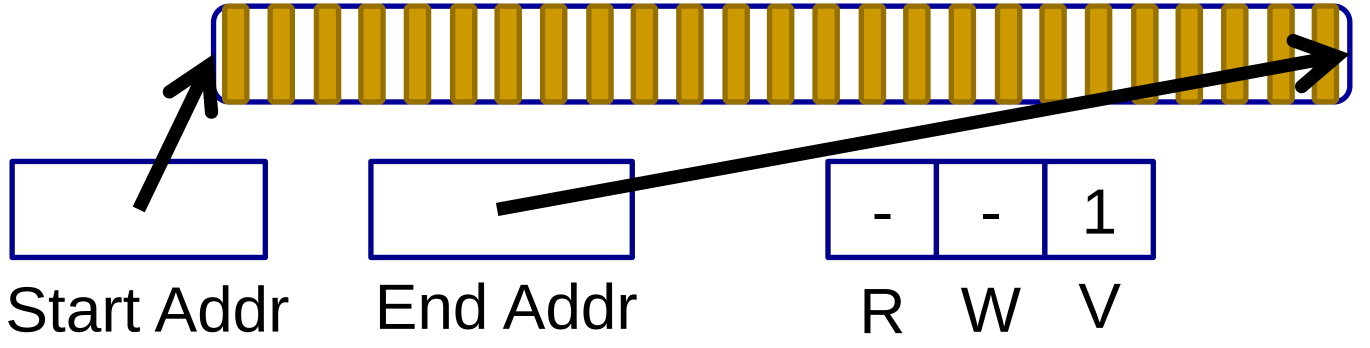
Watchpoint System Design II

- Make some RC entries point to bitmaps



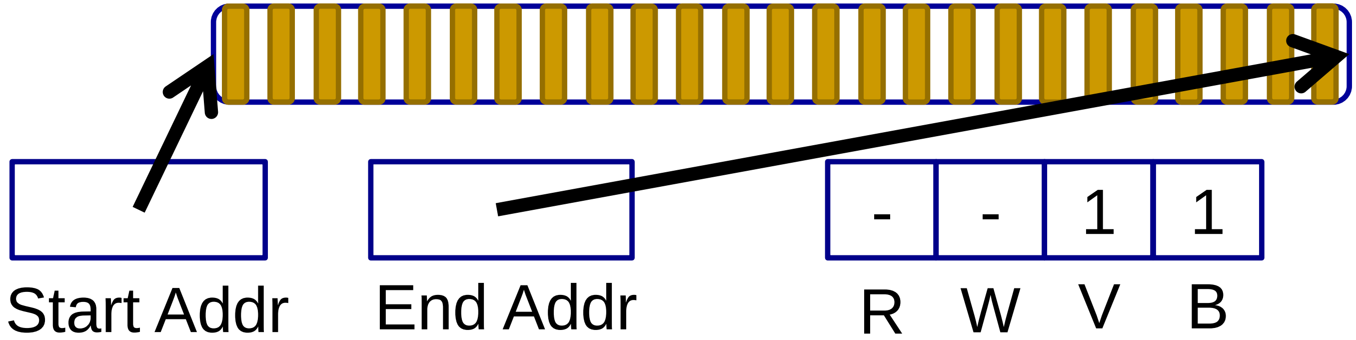
Watchpoint System Design II

- Make some RC entries point to bitmaps



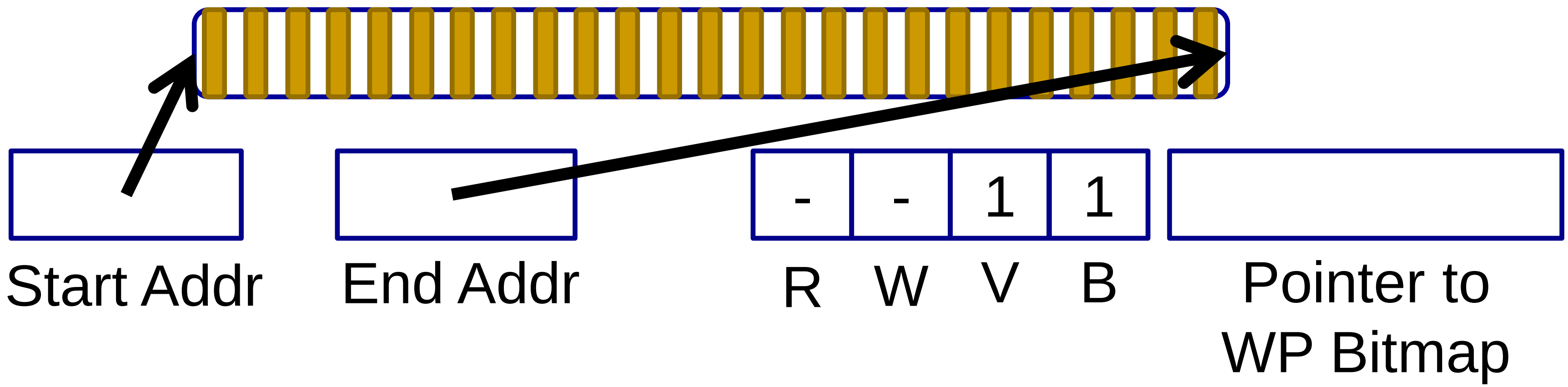
Watchpoint System Design II

- Make some RC entries point to bitmaps



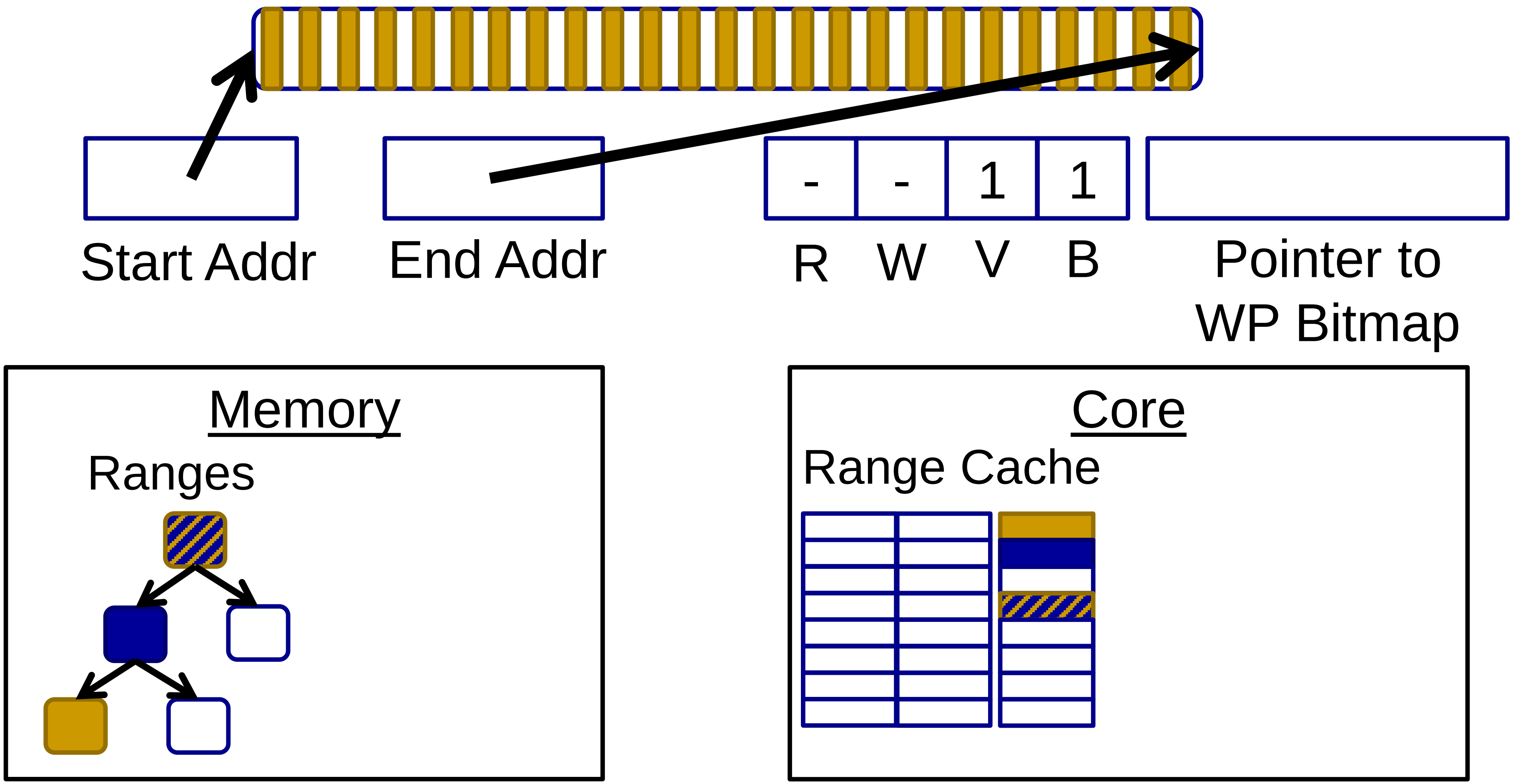
Watchpoint System Design II

- Make some RC entries point to bitmaps



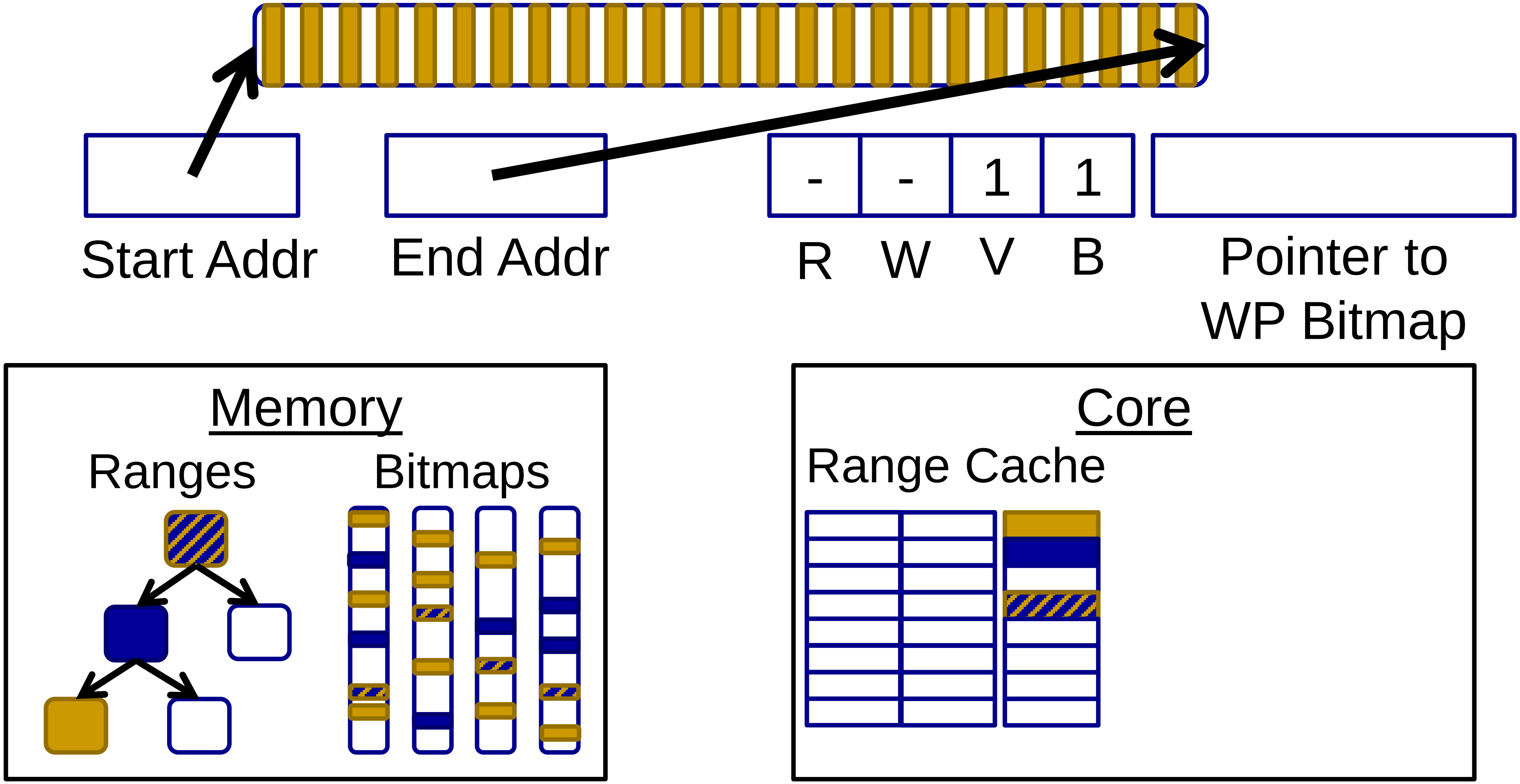
Watchpoint System Design II

- Make some RC entries point to bitmaps



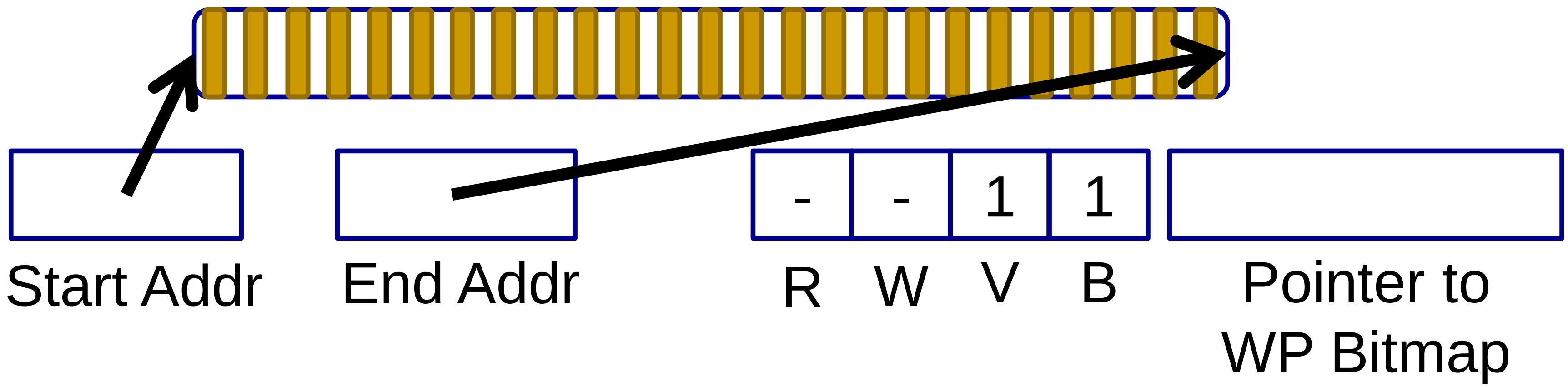
Watchpoint System Design II

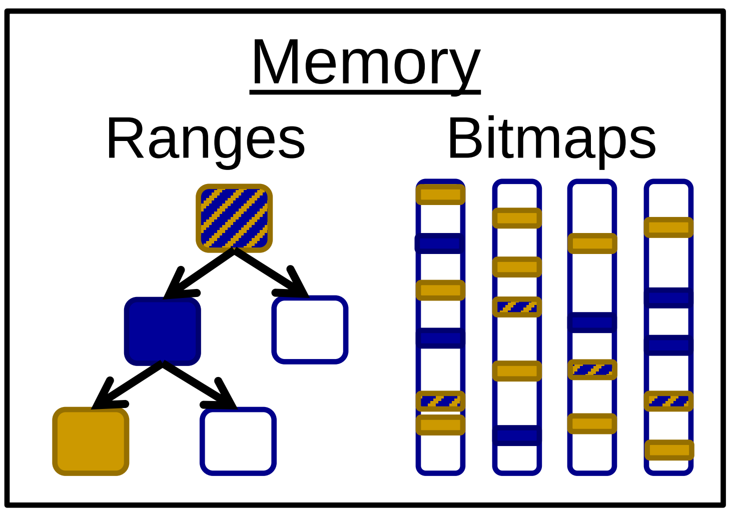
- Make some RC entries point to bitmaps

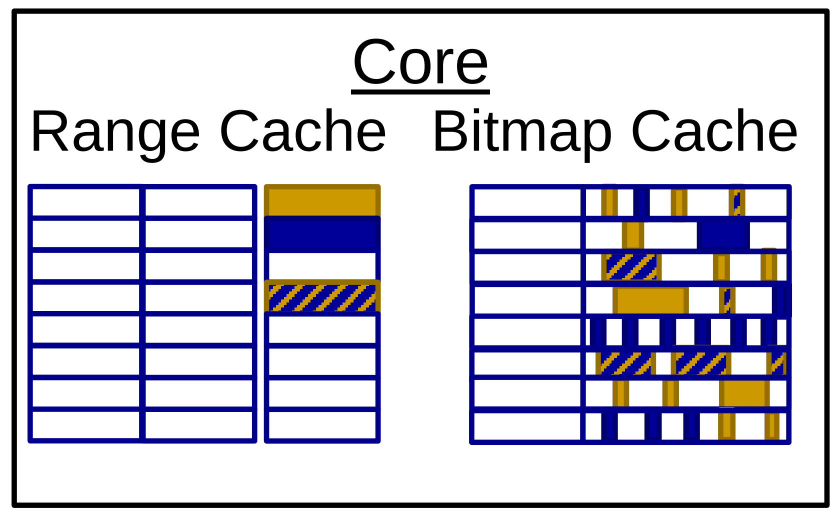


Watchpoint System Design II

- Make some RC entries point to bitmaps

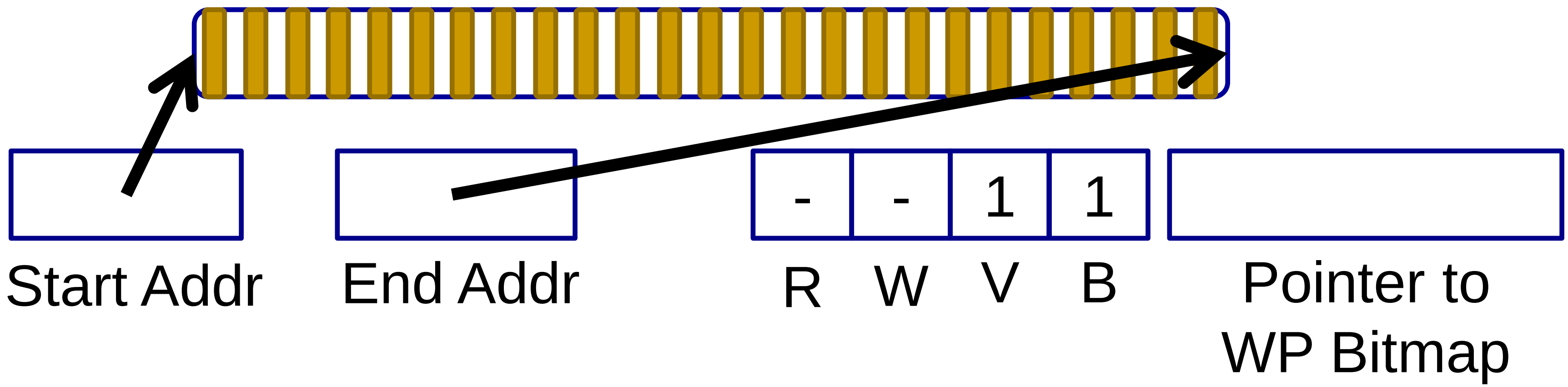


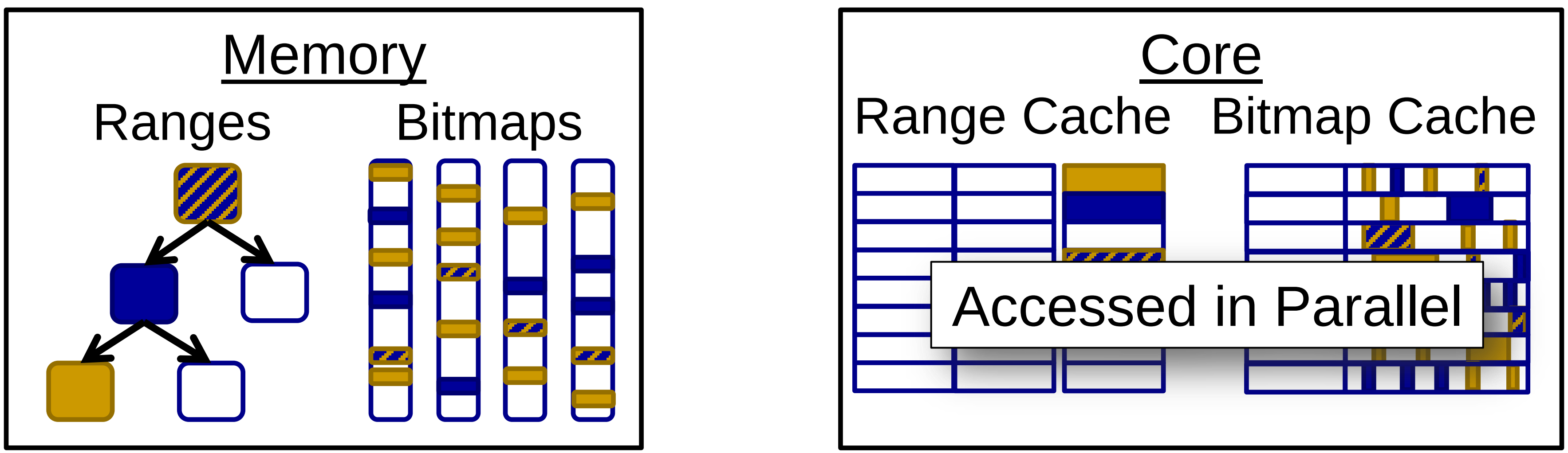




Watchpoint System Design II

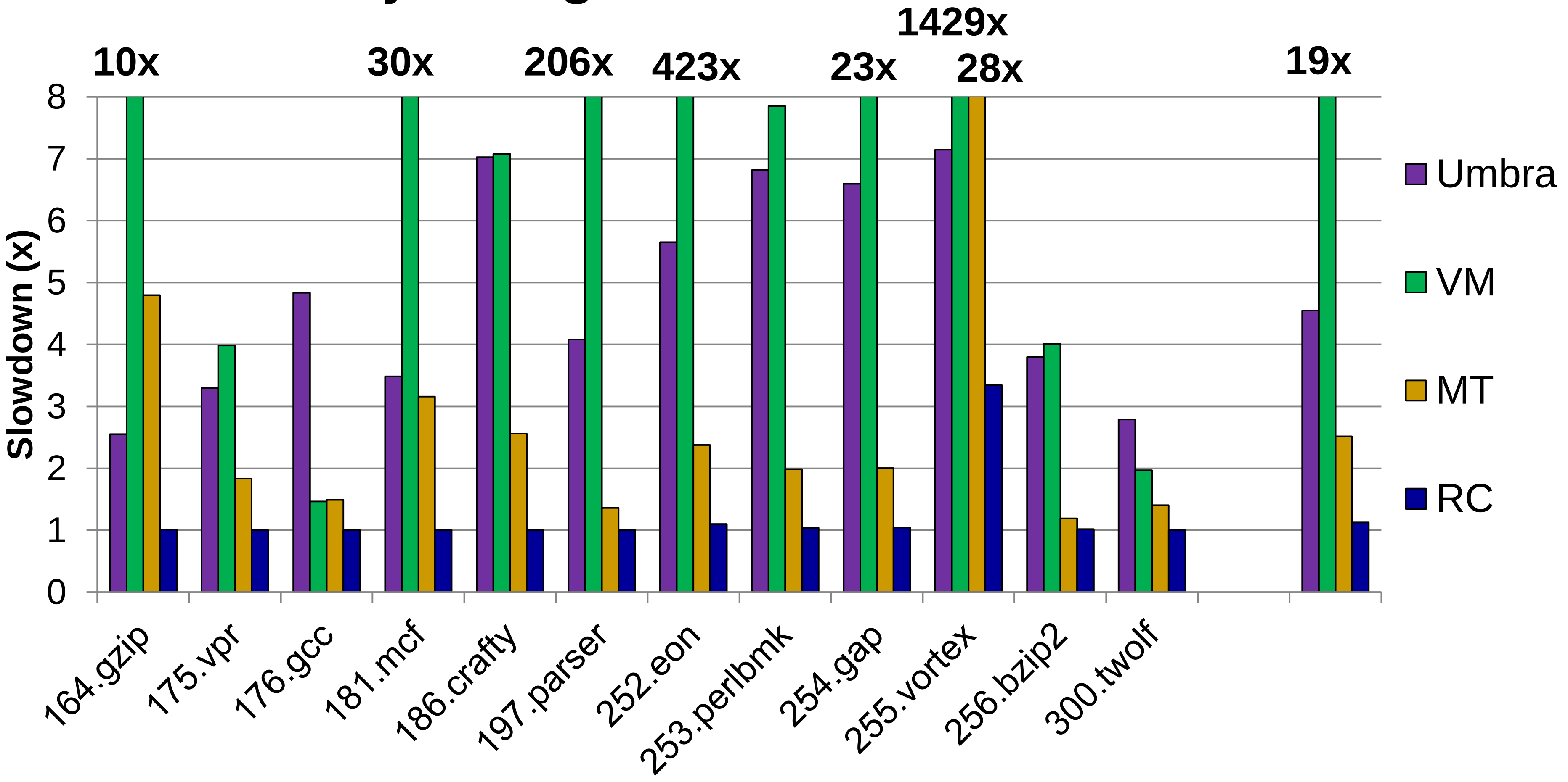
- Make some RC entries point to bitmaps





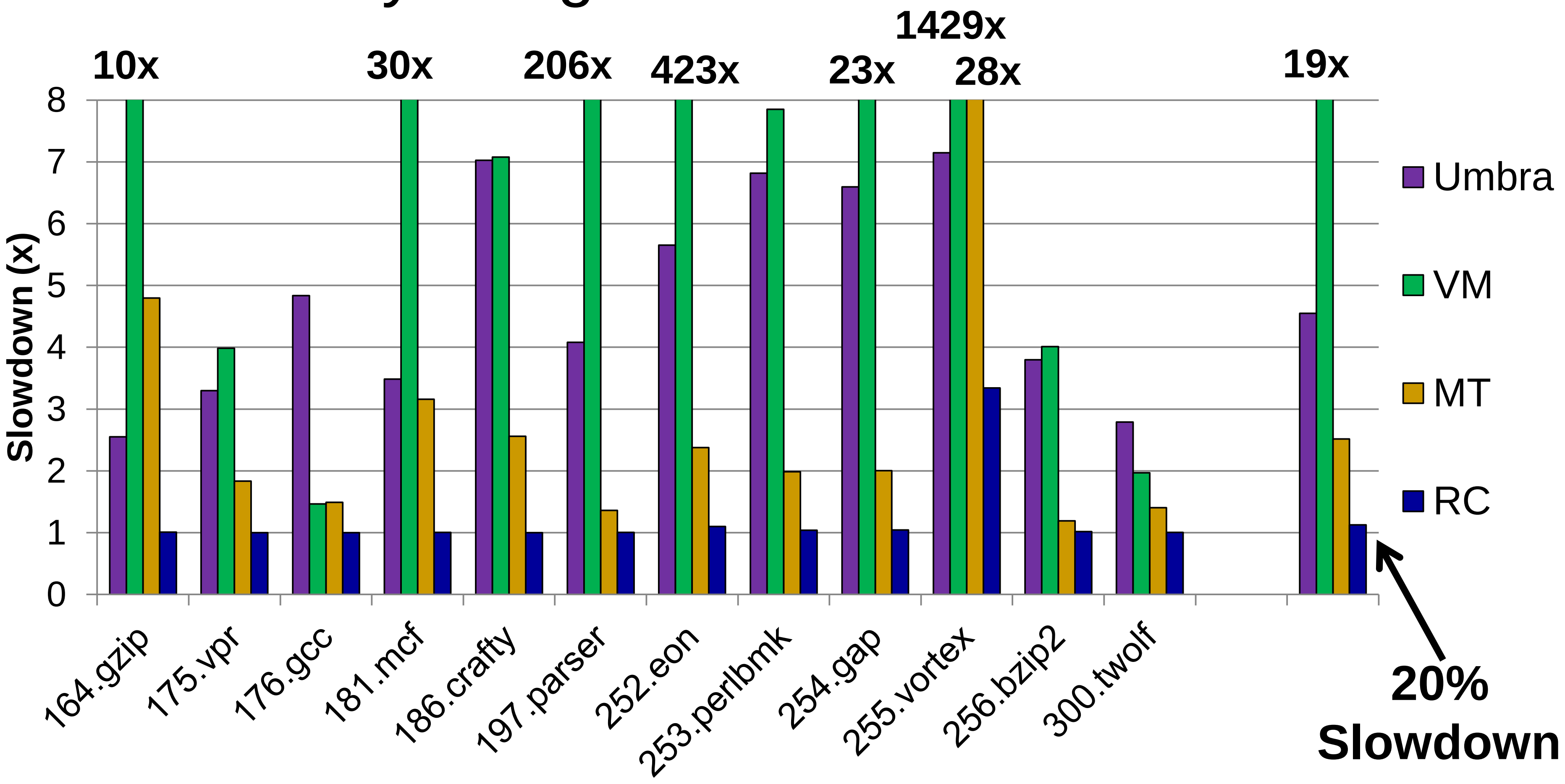
Watchpoint-Based Taint Analysis

- 128 entry Range Cache



Watchpoint-Based Taint Analysis

- 128 entry Range Cache



Width Test