Looking Beyond Microarchitectural-Only Side Channels

Mengjia Yan
mengjia@csail.mit.edu
http://people.csail.mit.edu/mengjia/

SEED Keynote
September 27, 2022

Meltdown & Spectre on the Headlines in 2018

Meltdown and Spectre: 'worst ever' CPU bugs affect virtually all computers

Everything from smartphones and PCs to cloud computing affected by major security flaw found in Intel and other processors – and fix could slow devices.

Quotes from https://www.theguardian.com/technology/2018/jan/04/meltdown-spectre-worst-cpu-bugs-ever-found-affect-computers-intel-processors-security-flaw

Current Side Channel Research Landscape

Current Side Channel Research Landscape

CACHE MISSING FOR FUN AND PROFIT

Last-Level Cache Side-Channel Attacks are Practical

DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks

Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard,
Graz University of Technology

Current Side Channel Research Landscape

CACHE MISSING FOR FUN AND PROFIT

Last-Level Cache Side-Channel Attacks are Practical

DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks

Mangard,

Port Contention for Fun and Profit

Alejandro C

Don't Mesh Around: Side-Channel Attacks and Mitigations on Mesh Interconnects

Miles Dai, *MIT;* Riccardo Paccagnella, *University of Illinois at Urbana-Champaign;* Miguel Gomez-Garcia, *MIT;* John McCalpin, *Texas Advanced Computing Center;* Mengjia Yan, *MIT*

Current Side Channel Research Landscape

CACHE MISSING FOR FUN AND PROFIT

Last-Level Cache Side-Channel Attacks are Practical

DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks

Mangard,

Port Contention for Fun and Profit

Alejandro C

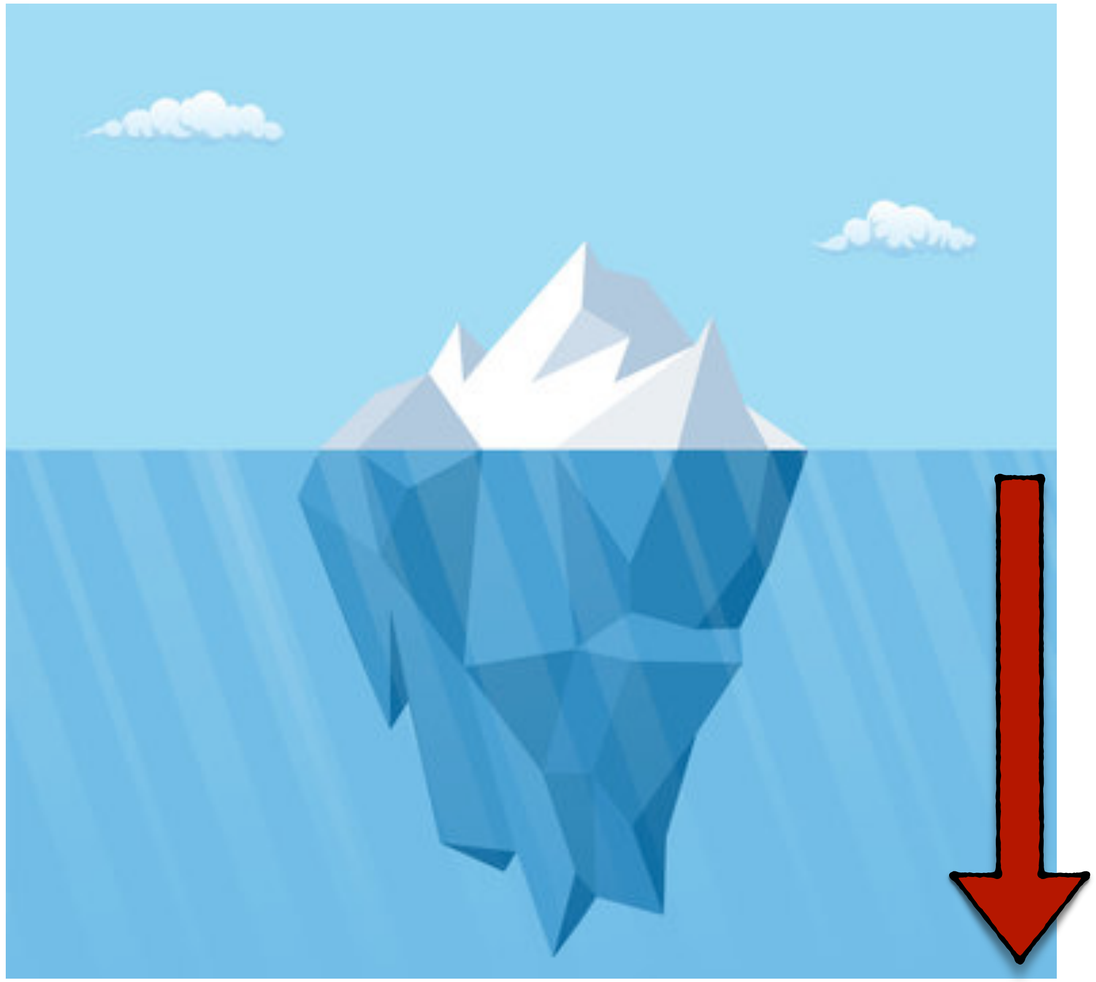
Don't Mesh Around: Side-Channel Attacks and Mitigations on Mesh Interconnects

ampaign;
g Center;

Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86

Yingchen Wang* *UT Austin*
Riccardo Paccagnella* *UIUC*
Elizabeth Tang He *UIUC*
Hovav Shacham *UT Austin*
Christopher W. Fletcher *UIUC*
David Kohlbrenner *UW*

Current Side Channel Research Landscape



CACHE MISSING FOR FUN AND PROFIT

Last-Level Cache Side-Channel Attacks are Practical

DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks

Mangard,

Port Contention for Fun and Profit

Alejandro C

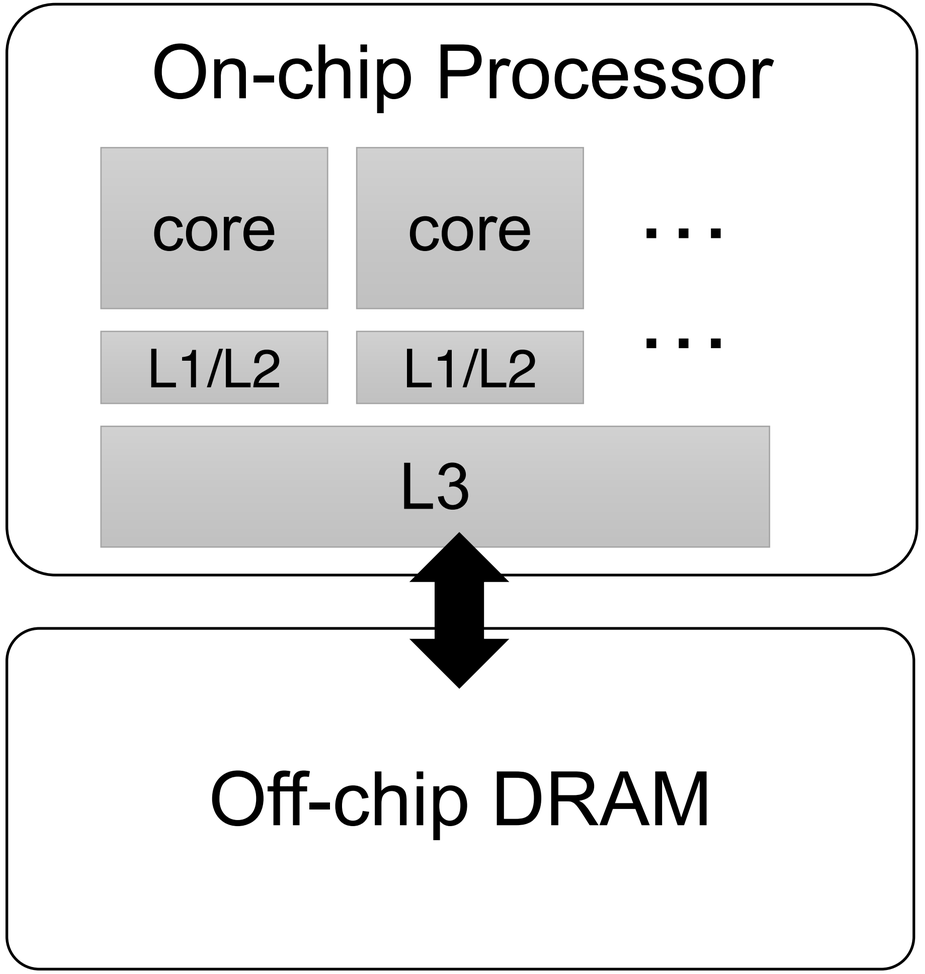
Don't Mesh Around: Side-Channel Attacks and Mitigations on Mesh Interconnects

ampaign;
g Center;

Hertzbleed: Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86

Yingchen Wang* *UT Austin*
Riccardo Paccagnella* *UIUC*
Elizabeth Tang He *UIUC*
Hovav Shacham *UT Austin*
Christopher W. Fletcher *UIUC*
David Kohlbrenner *UW*

The Age of Pervasive Hardware Attacks



The Age of Pervasive Hardware Attacks

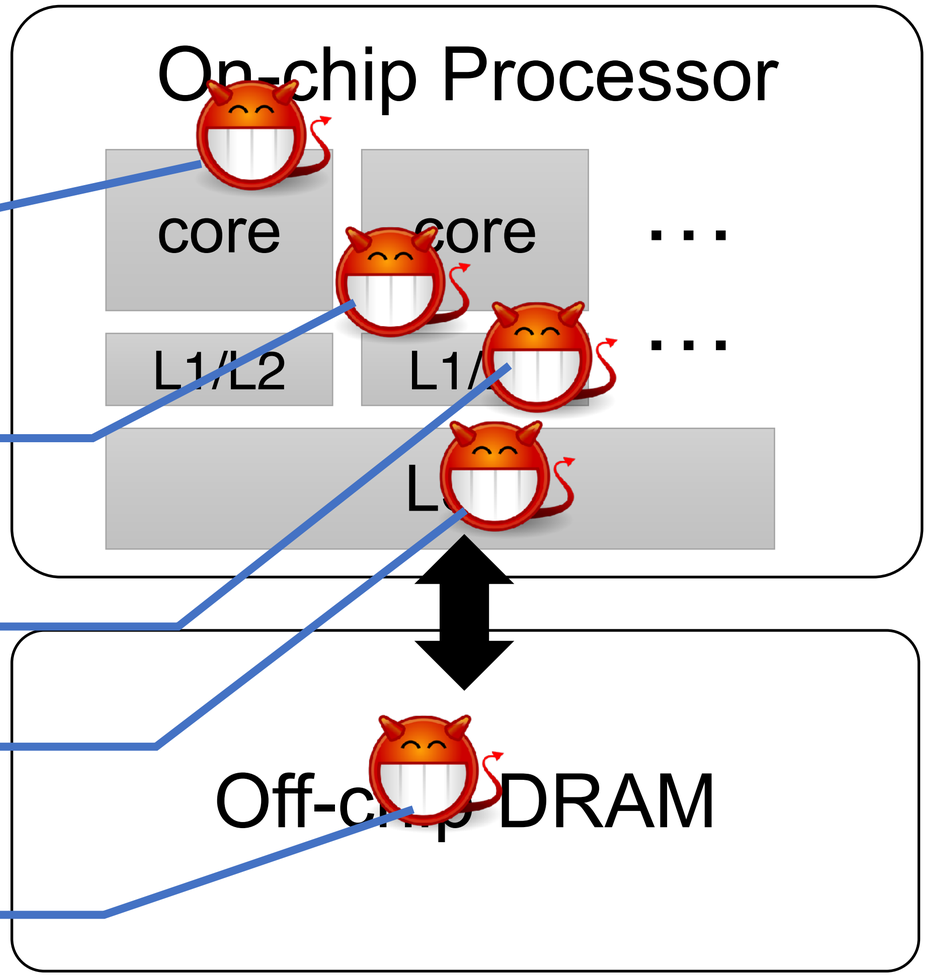
Spectre, Meltdown
Foreshadow,
Arithmetic timing

CacheOut, RIDL, Fallout
Port contention,
cache banking, 4K Alias

RNG Unit Covert Chanel

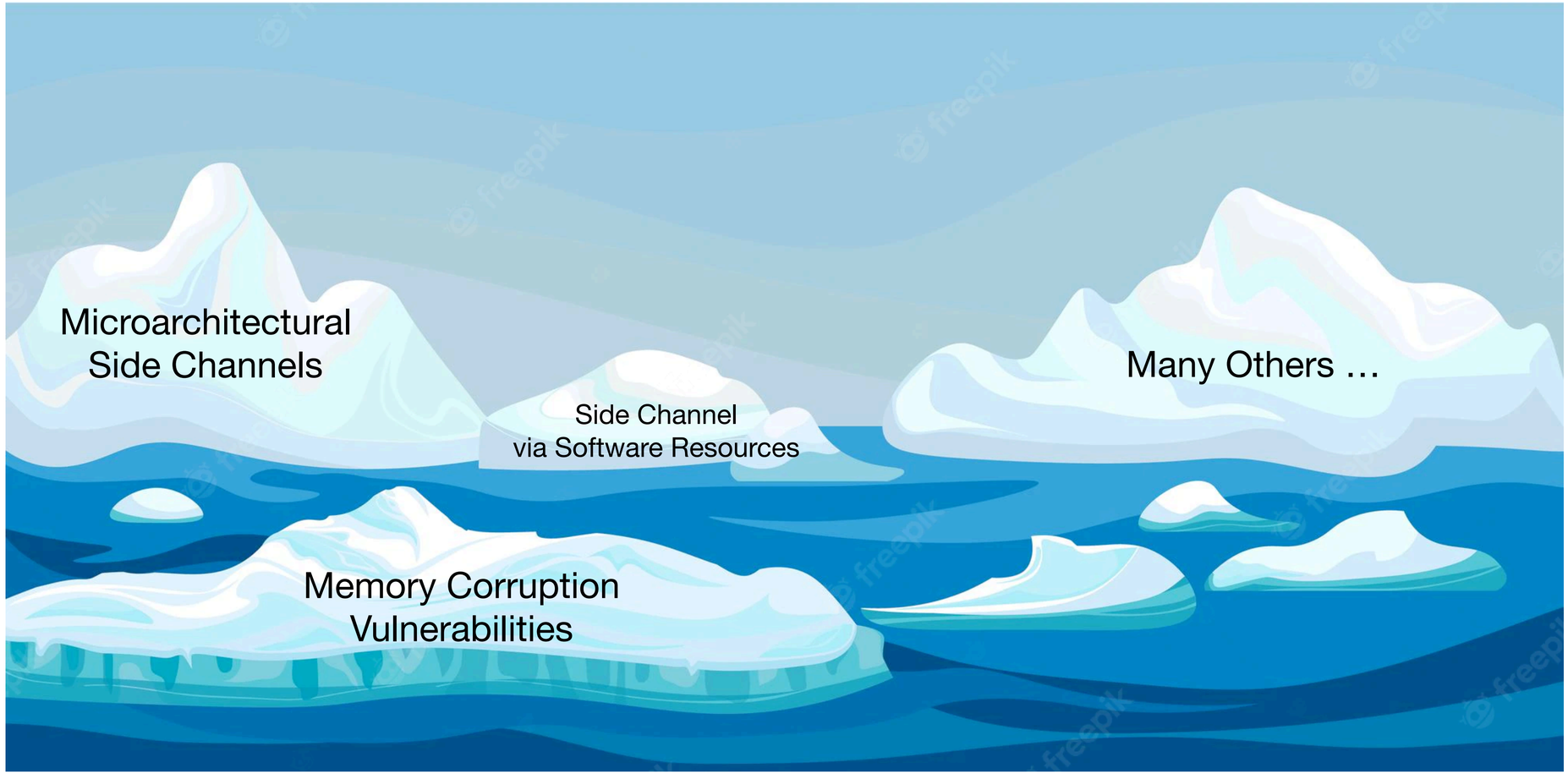
Flush+Reload, Prime+Probe
Directory Attacks

RowHammer, DRAMA
RAMBleed

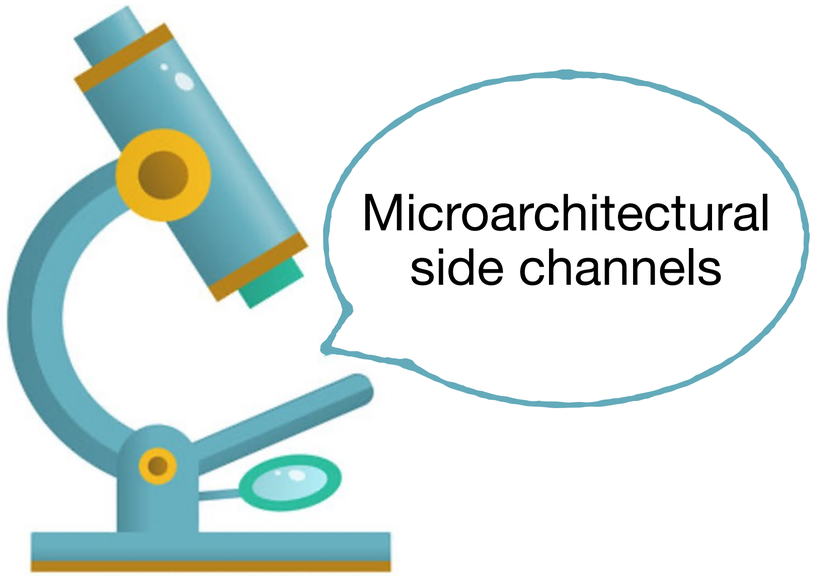


However…

However…

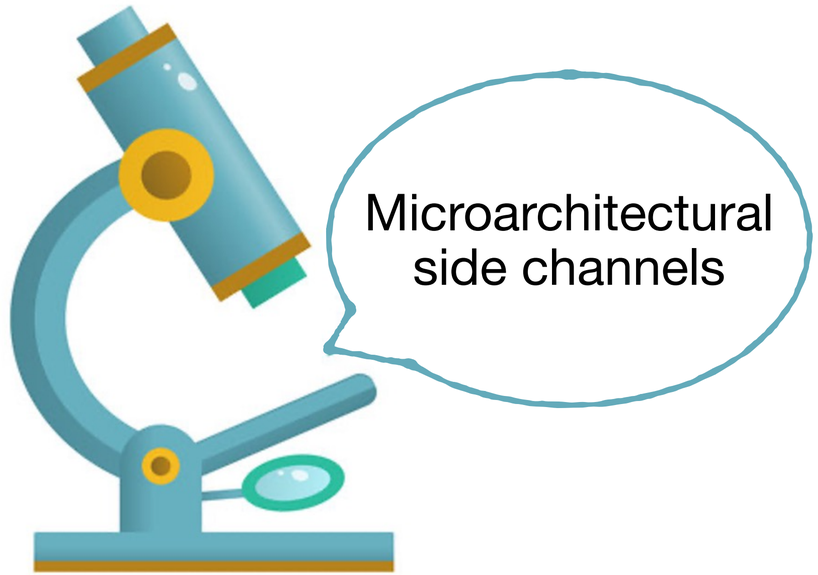

Microarchitectural
Side Channels
Side Channel
via Software Resources
Many Others …
Memory Corruption
Vulnerabilities

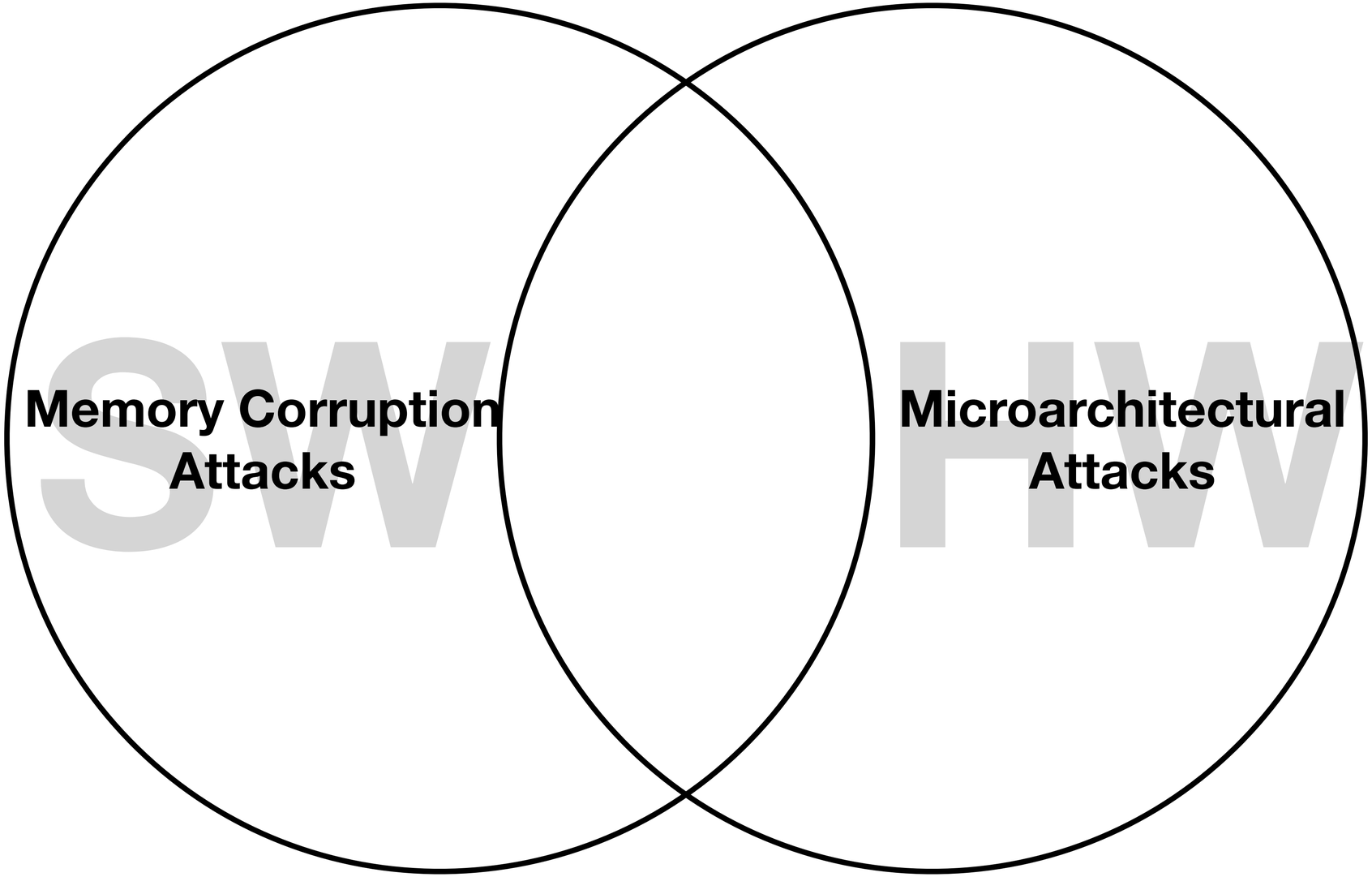
Limitations of Looking At Microarchitectural-only Side Channels

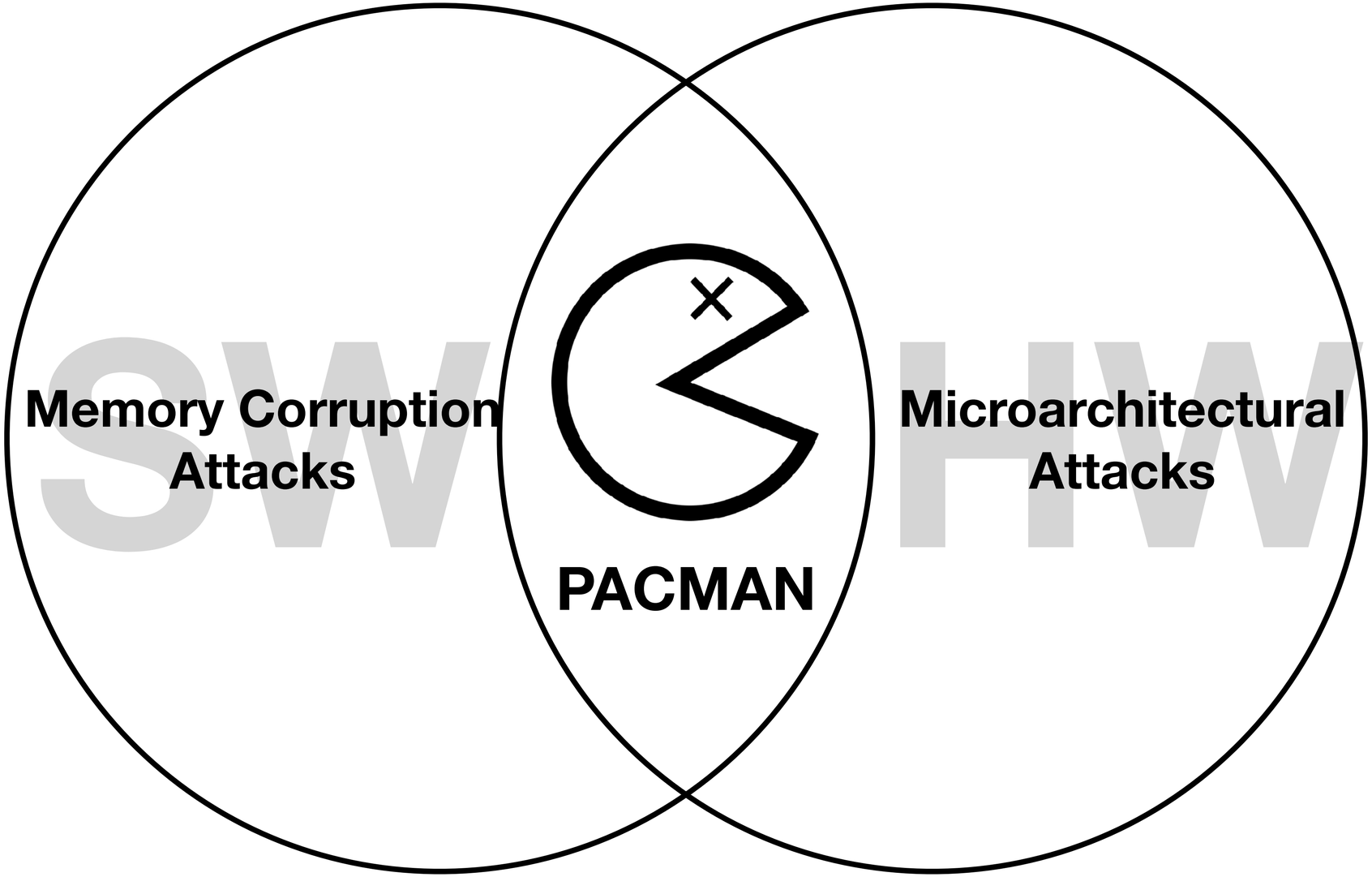


Limitations of Looking At Microarchitectural-only Side Channels

- Part 1: Miss threats that arise from compound threat models
- Part 2: Misunderstand root causes of existing side channel attacks

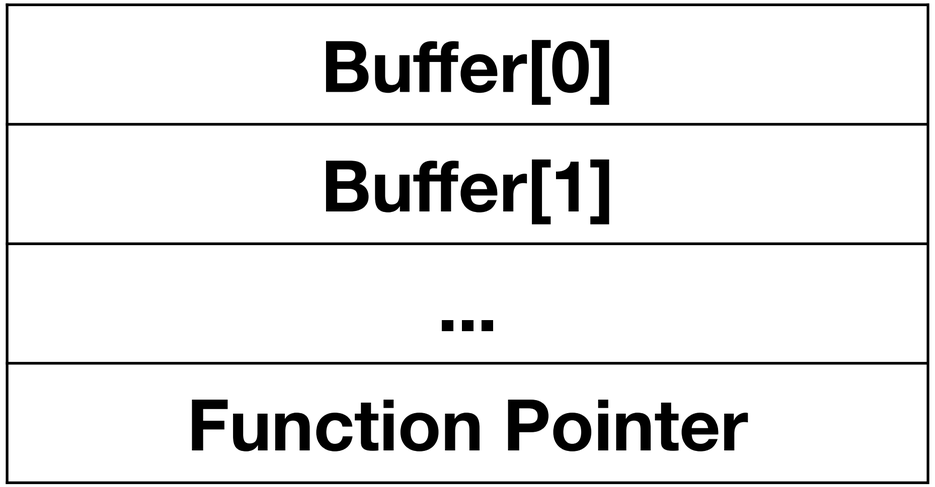



SW
HW
Memory Corruption Attacks
Microarchitectural Attacks

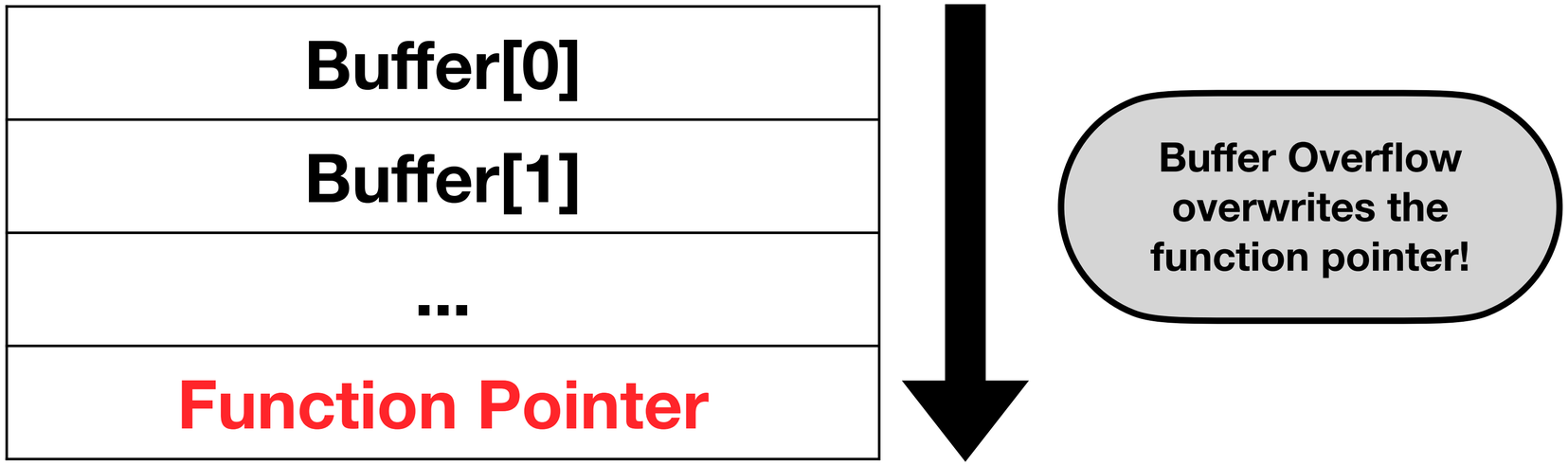


PACMAN: Attacking ARM Pointer Authentication with Speculative Execution;
Joseph Ravichandran, Weon Taek Na*, Jay Lang, Mengjia Yan; ISCA, 2022.*

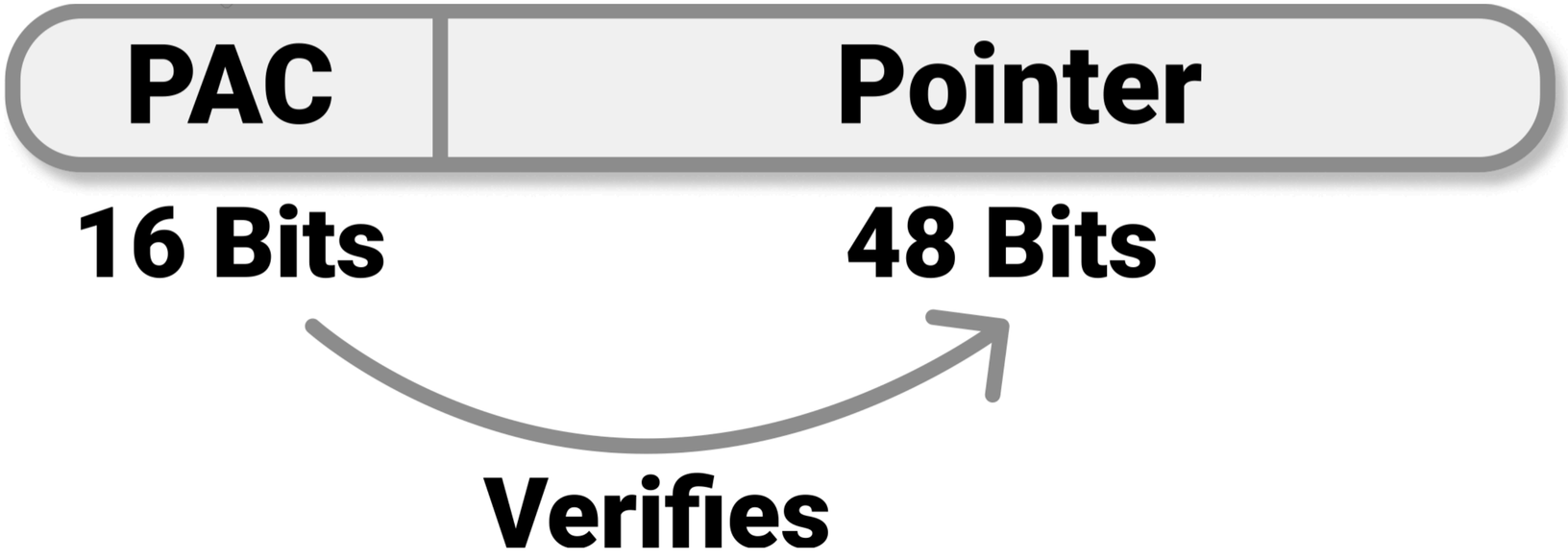
Buffer Overflow



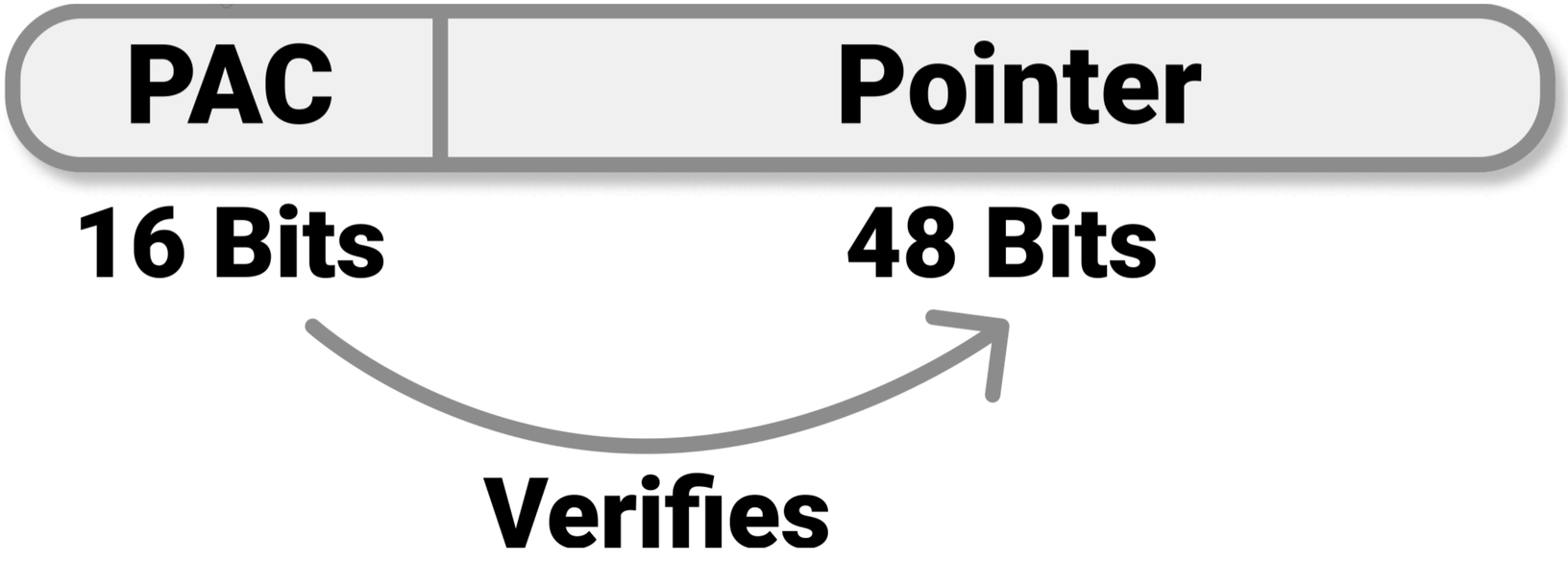
Buffer Overflow



ARM Pointer Authentication



ARM Pointer Authentication



PAC = crypto_func(pointer, salt, key)

Two Operations

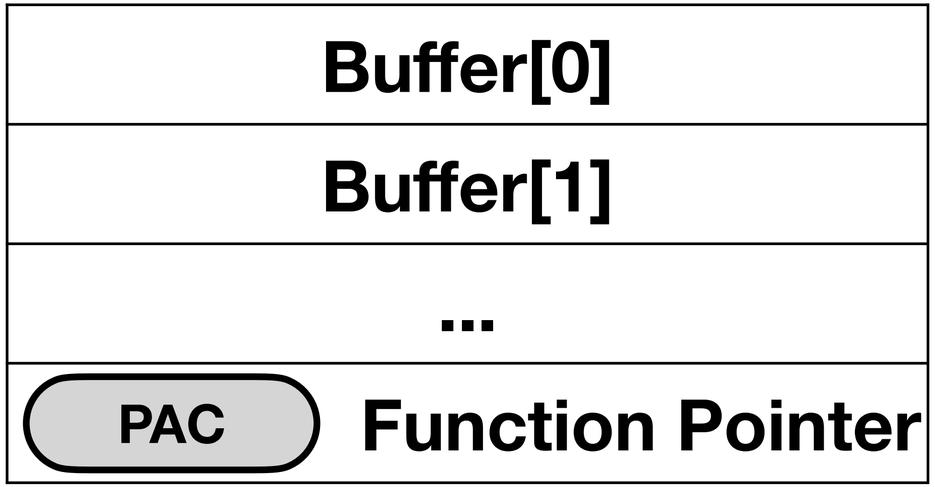
Sign

Before saving a pointer to memory, compute the PAC

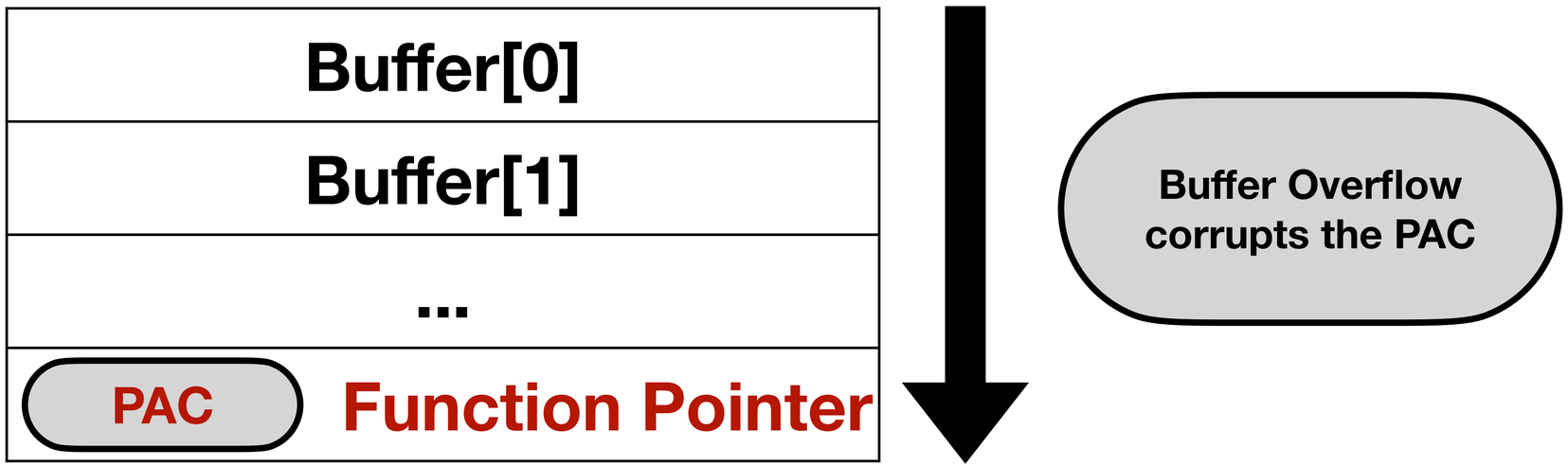
Verify

Before using a pointer, check the pointer's PAC

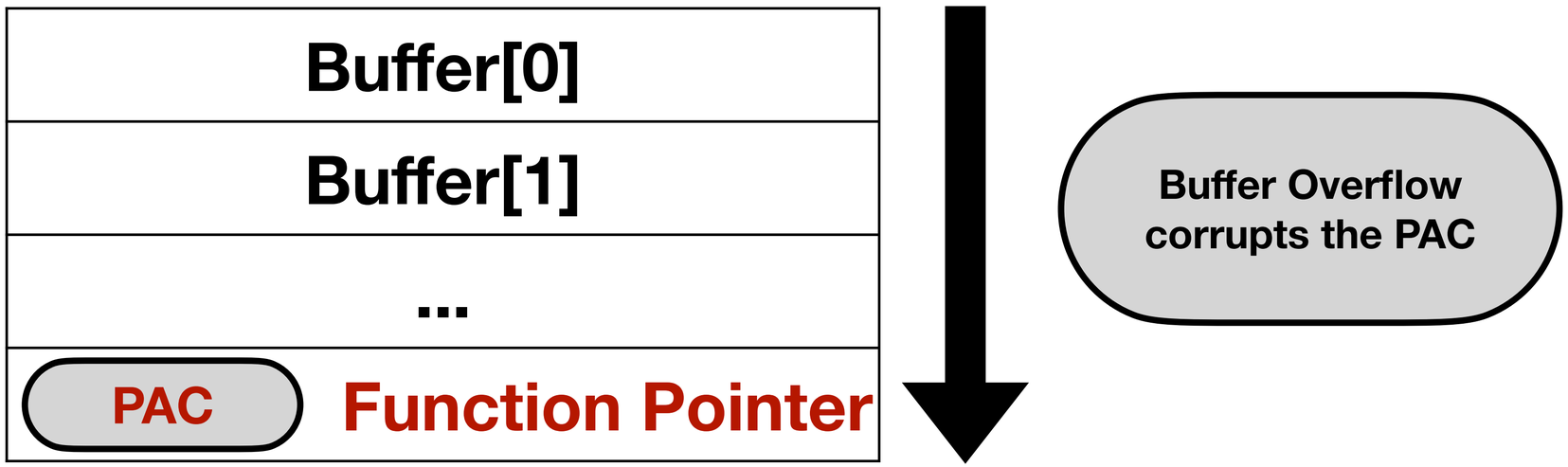
Buffer Overflow



Buffer Overflow



Buffer Overflow



Invalid PAC means we crash!

Extending ARM Pointer Authentication

Extending ARM Pointer Authentication

PAC it up: Towards Pointer Integrity using ARM Pointer Authentication

Hans Liljestrand, *Aalto University, Huawei Technologies Oy;* Thomas Nyman,

PTAuth: Temporal Memory Safety via Robust Points-to Authentication

Reza Mirzazade Farkhani, Mansour Ahmadi, and Long Lu, *Northeastern*

Protecting Indirect Branches against Fault Attacks using ARM Pointer Authentication

Pascal Nasahl
Graz University of Technology
pascal.nasahl@iaik.tugraz.at

Robert Schilling
Graz University of Technology
robert.schilling@iaik.tugraz.at

Stefan Mangard
Graz University of Technology
Lamarr Security Research
stefan.mangard@iaik.tugraz.at

Hardware-based Always-On Heap Memory Safety

Yonghae Kim
Georgia Institute of Technology
yonghae@gatech.edu

Jaekyu Lee
Arm Research
jaekyu.lee@arm.com

Hyesoon Kim
Georgia Institute of Technology
hyesoon@cc.gatech.edu

Extending ARM Pointer Authentication

The security properties of these mechanisms
have been examined **solely** under the memory safety threat model.

Threat Model

Memory
corruption
vulnerability

Threat Model

Memory corruption vulnerability

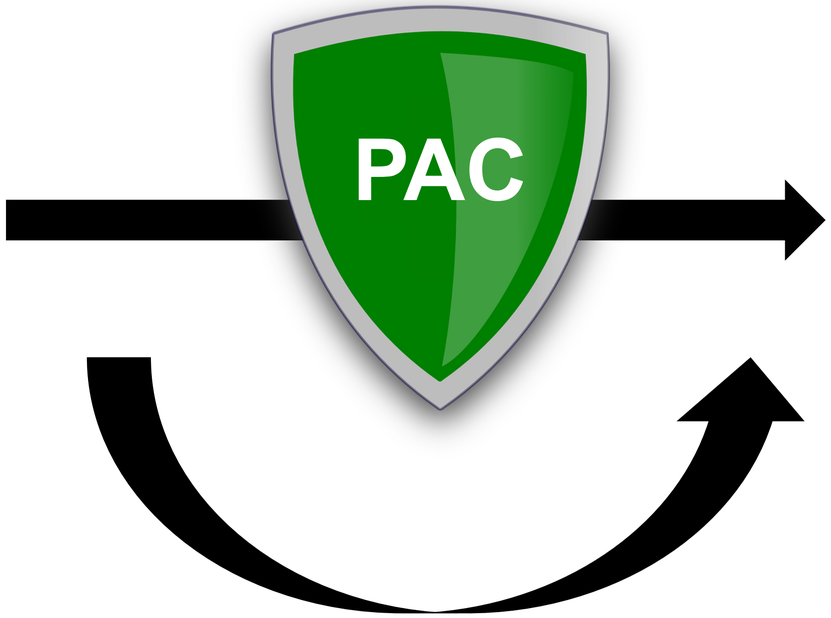
Control-flow hijacking attack

Threat Model

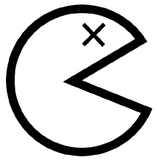
Threat Model

last line of defense for software attacks

Memory corruption vulnerability



Control-flow hijacking attack



PACMAN
Bypass defenses using microarchitectural attacks

Key Insight

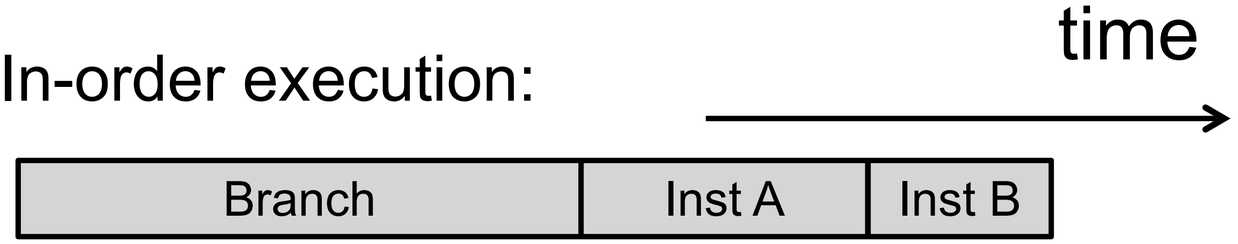
Break PAC with Microarchitectural Attacks

1. Guess a PAC **speculatively** to prevent crashes
2. Leak verification results via side channel

Speculative Execution

```
if (…) {   //Branch
   Inst A
   Inst B
}
```

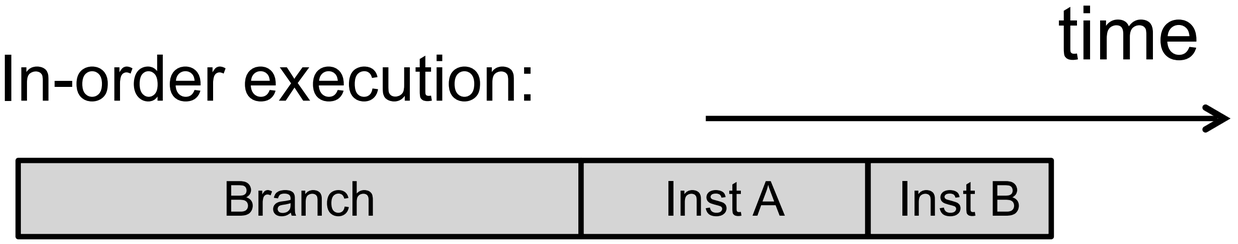
In-order execution:



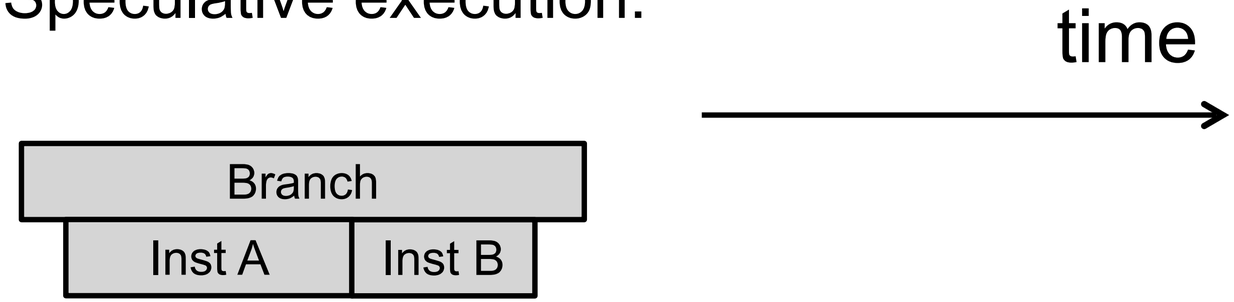
Speculative Execution

```
if (…) {   //Branch
   Inst A
   Inst B
}
```

In-order execution:



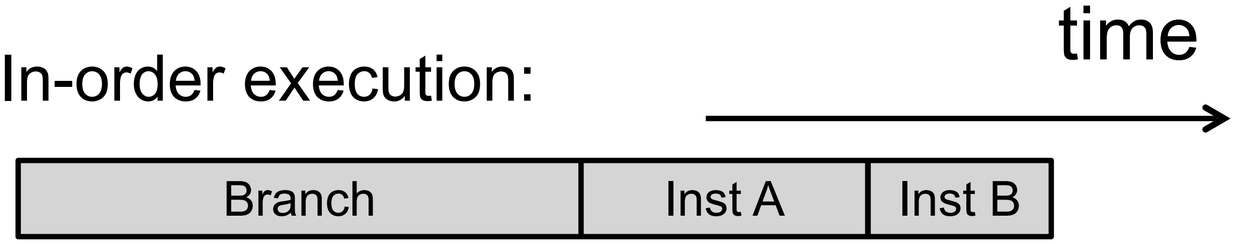
Speculative execution:



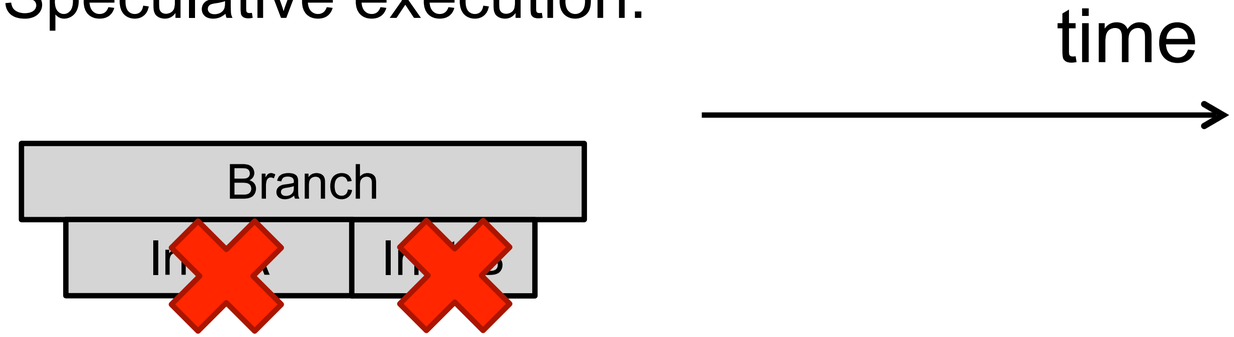
Speculative Execution

```
if (…) {   //Branch
   Inst A
   Inst B
}
```

In-order execution:



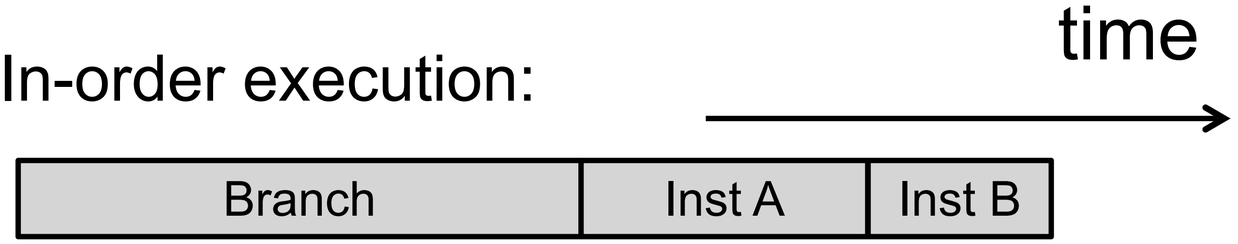
Speculative execution:



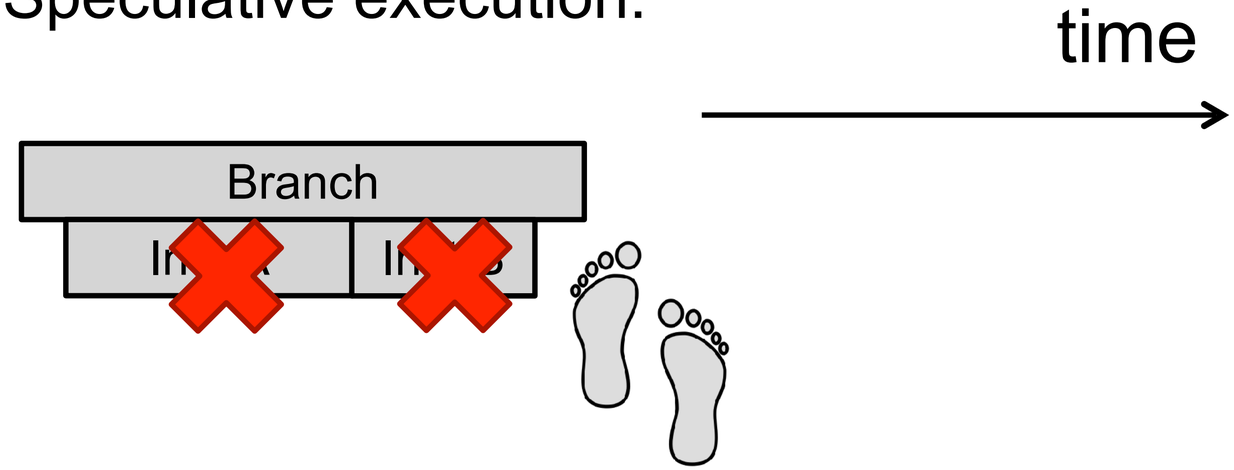
Speculative Execution

```
if (…) {   //Branch
   Inst A
   Inst B
}
```

In-order execution:



Speculative execution:



Micro-architecture side effects are not rolled back

PACMAN Gadgets

```
if (condition):
  verified_ptr = AUT(guess_ptr) // AUT
  load(verified_ptr)            // LD
```

Data Gadget

Attack Procedure

```
if (condition):
  verified_ptr = AUT(guess_ptr) // AUT
  load(verified_ptr)            // LD
```

Correct PAC

Incorrect PAC

Attack Procedure

```
if (condition):
  verified_ptr = AUT(guess_ptr) // AUT
  load(verified_ptr)            // LD
```

time →

Correct PAC

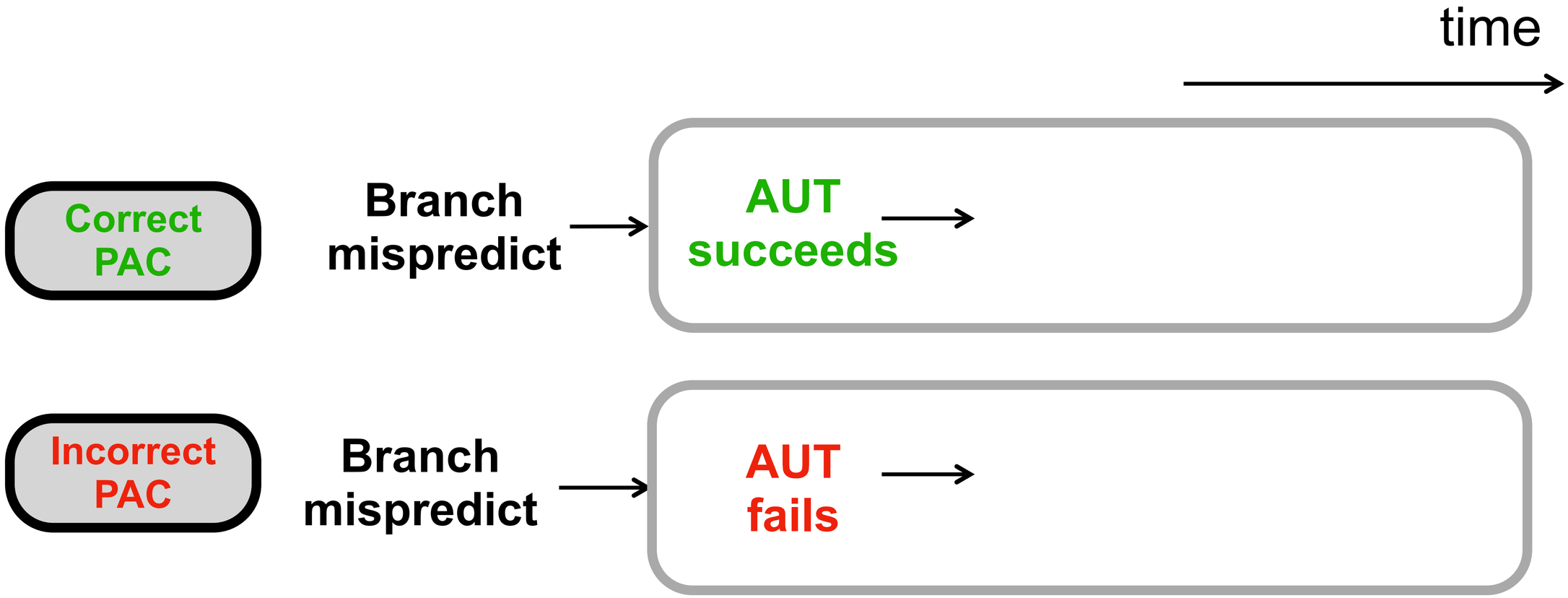
Branch mispredict →

Incorrect PAC

Branch mispredict →

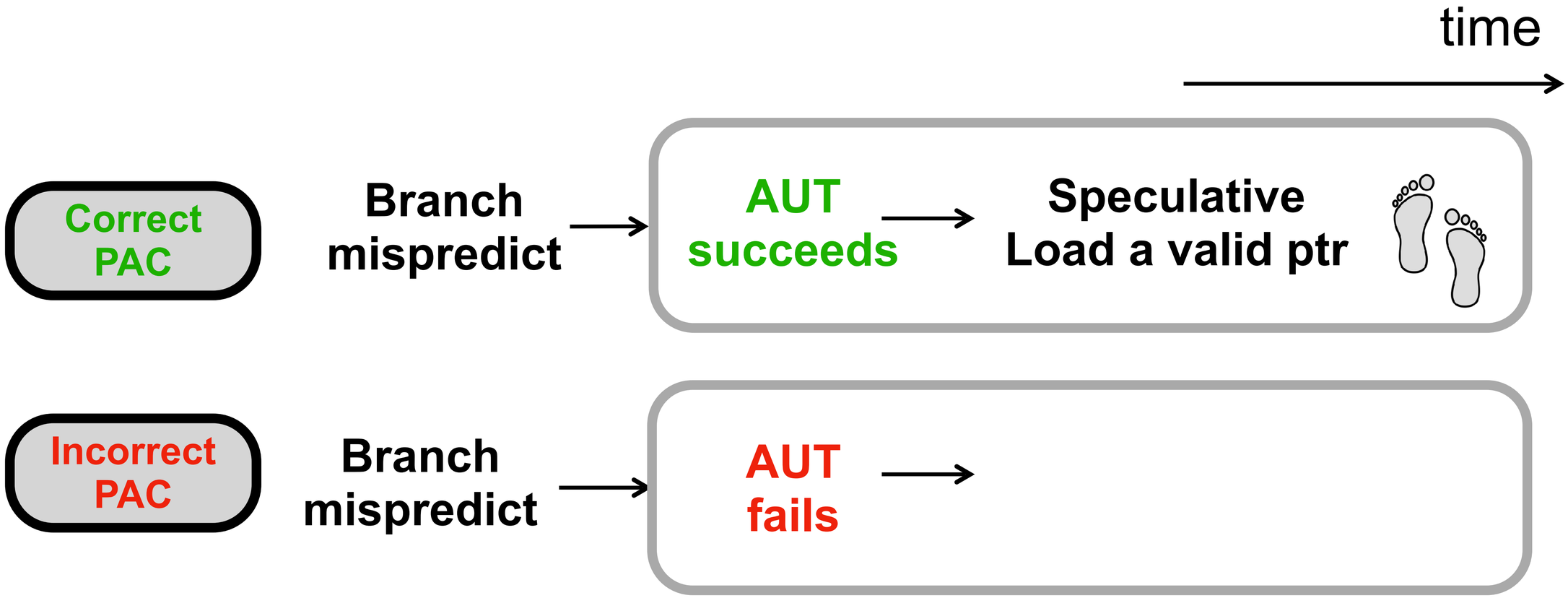
Attack Procedure

```
if (condition):
  verified_ptr = AUT(guess_ptr) // AUT
  load(verified_ptr)            // LD
```



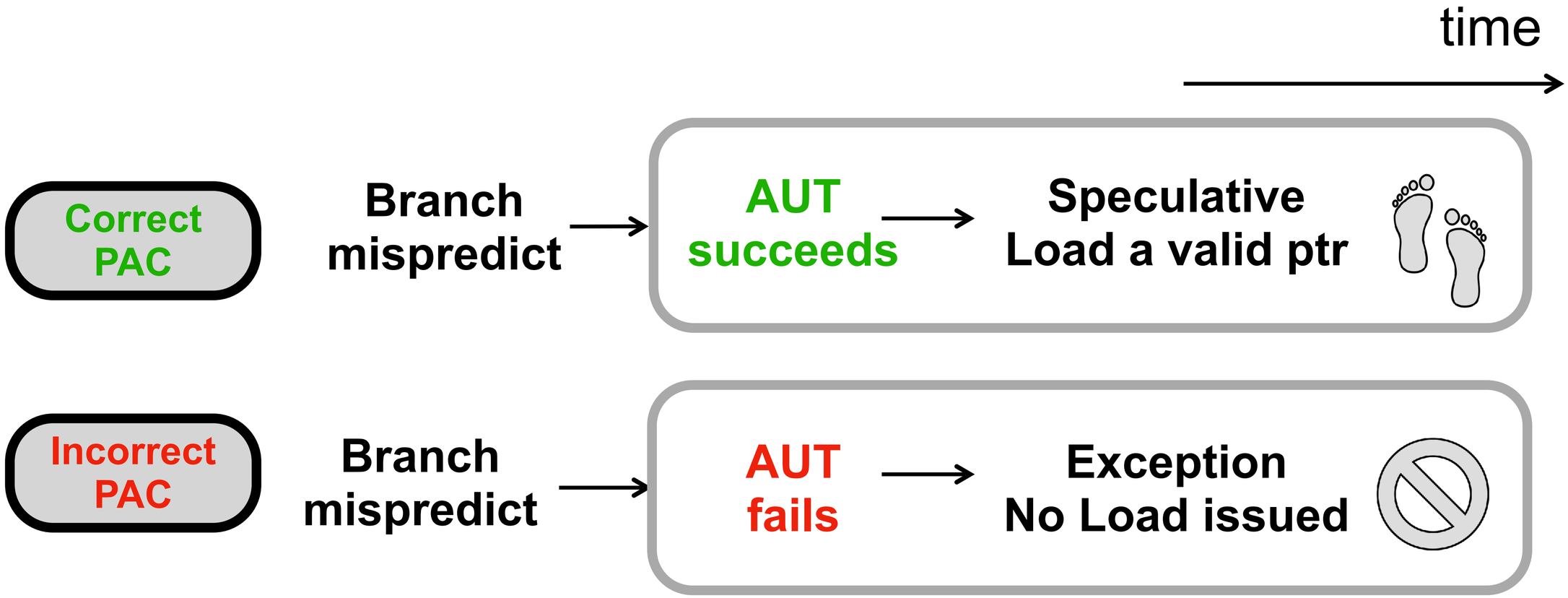
Attack Procedure

```
if (condition):
  verified_ptr = AUT(guess_ptr) // AUT
  load(verified_ptr)            // LD
```



Attack Procedure

```
if (condition):
  verified_ptr = AUT(guess_ptr) // AUT
  load(verified_ptr)            // LD
```



TARGET

The world's first desktop CPU
that supports Pointer Authentication.

Image: Apple ("Apple Unleashes M1")

TARGET

The world's first desktop CPU
that supports Pointer Authentication.

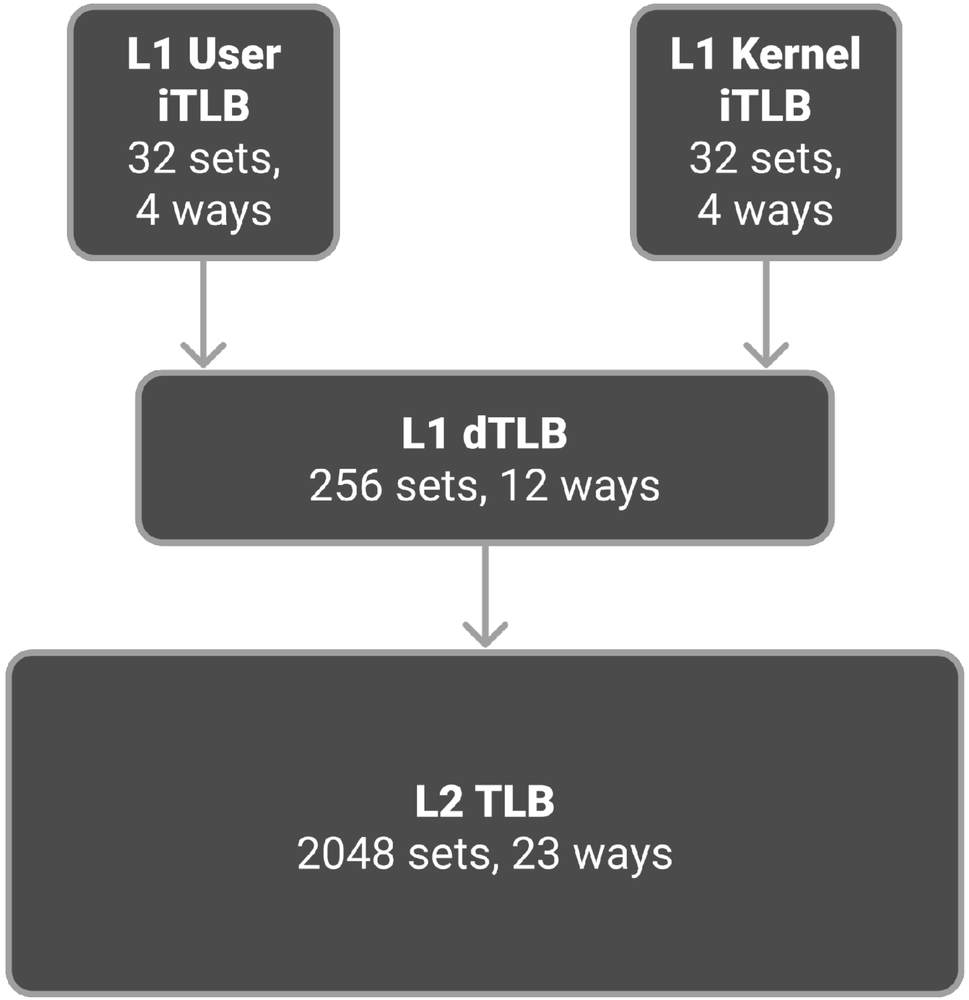
Image: Apple ("Apple Unleashes M1")

Challenges of Real World Hardware

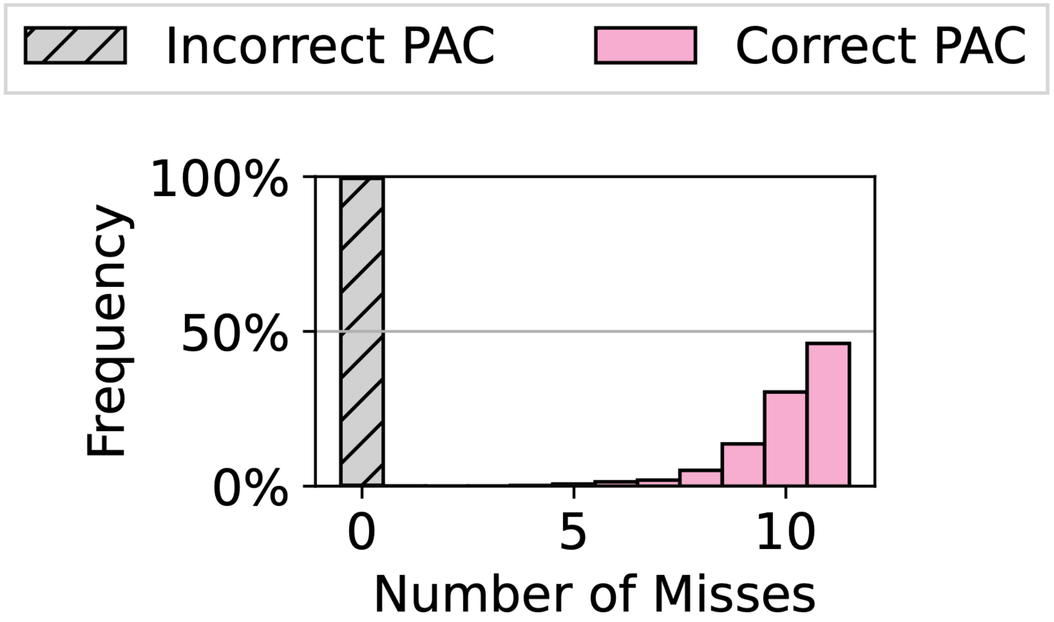
- No documentation of microarchitectural details.
- No high resolution timer.
- macOS is a difficult system to integrate attacks on.

Essentially, we had to reinvent the wheel.

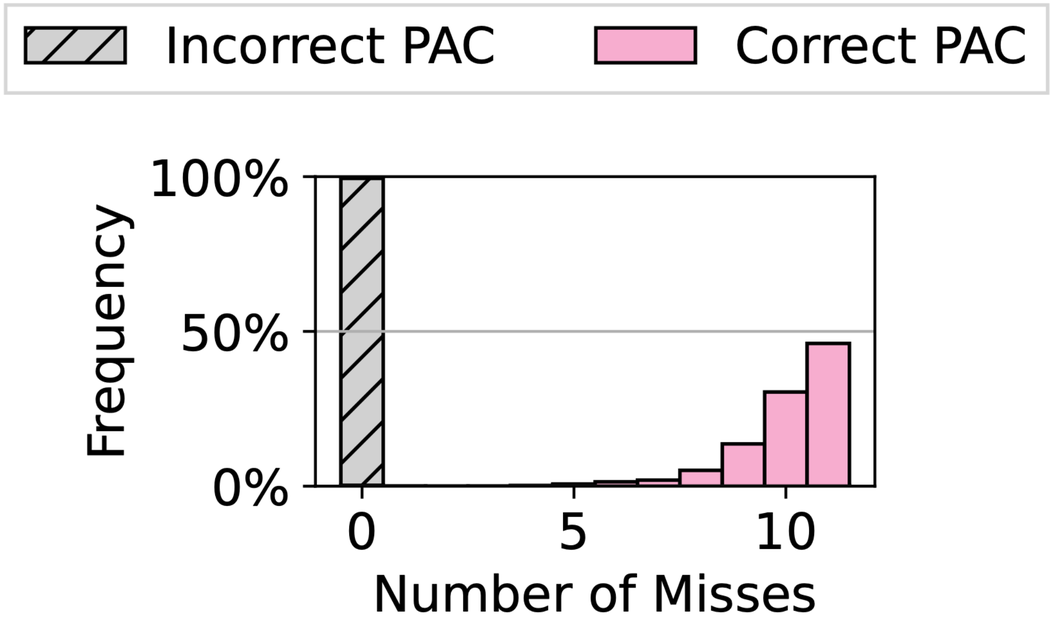
Conjectured TLB Hierarchy



PAC Oracle Accuracy



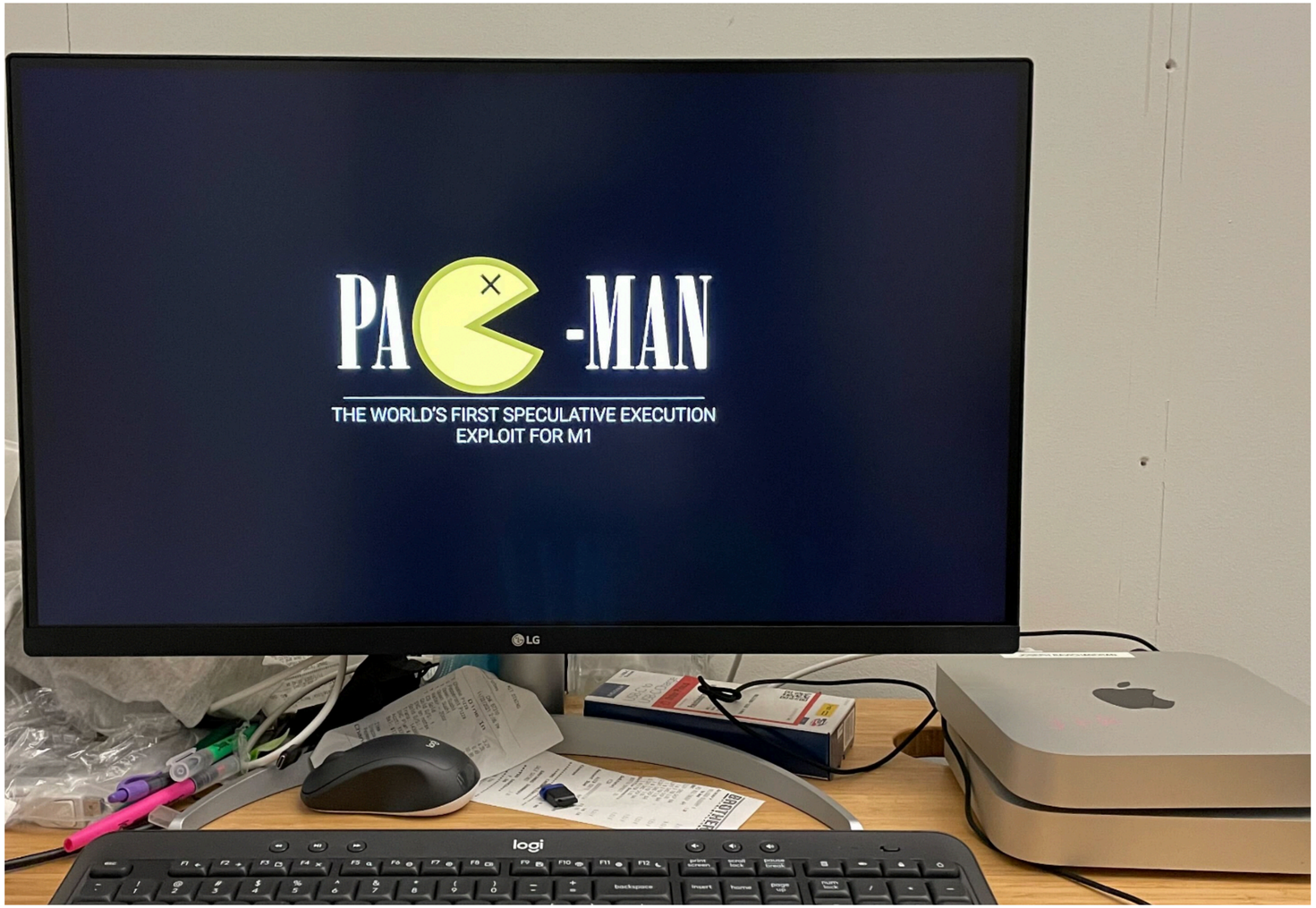
PAC Oracle Accuracy



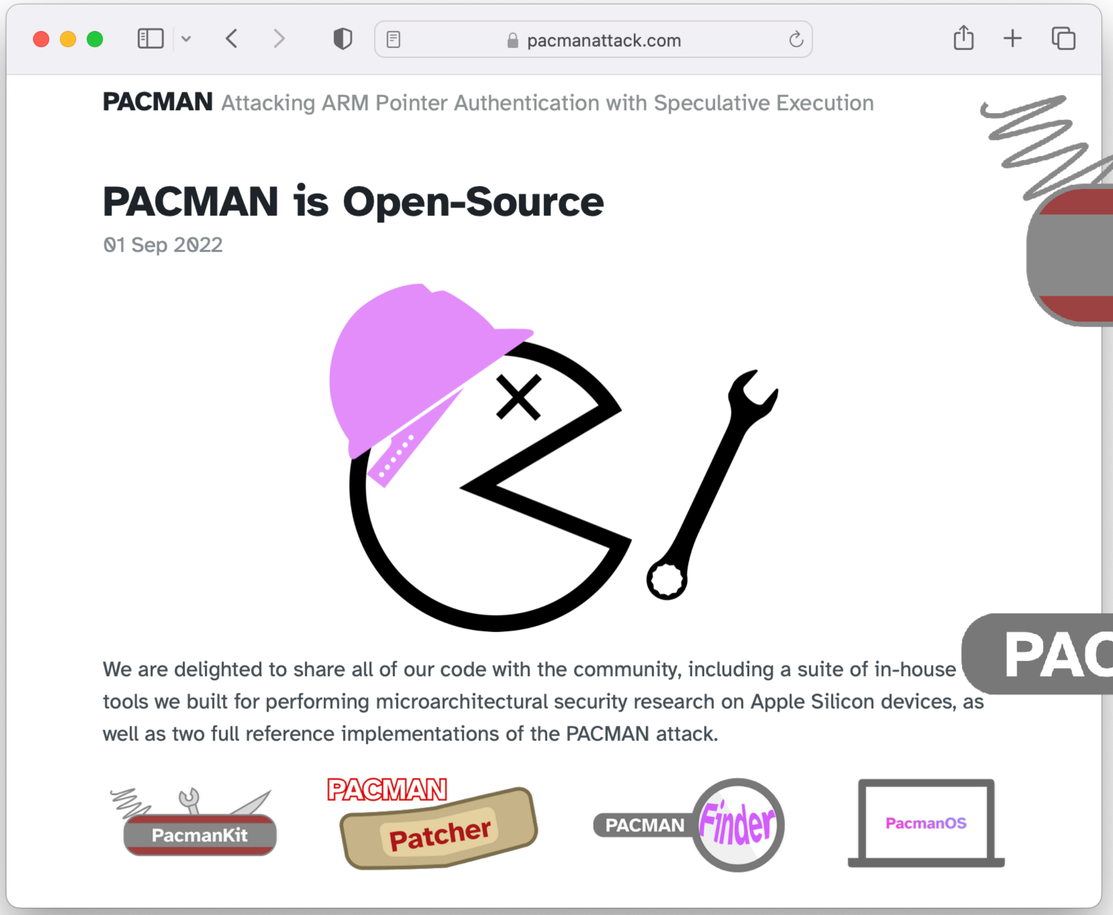
With a highly reliable PAC oracle, the attacker can brute-force the PAC value.

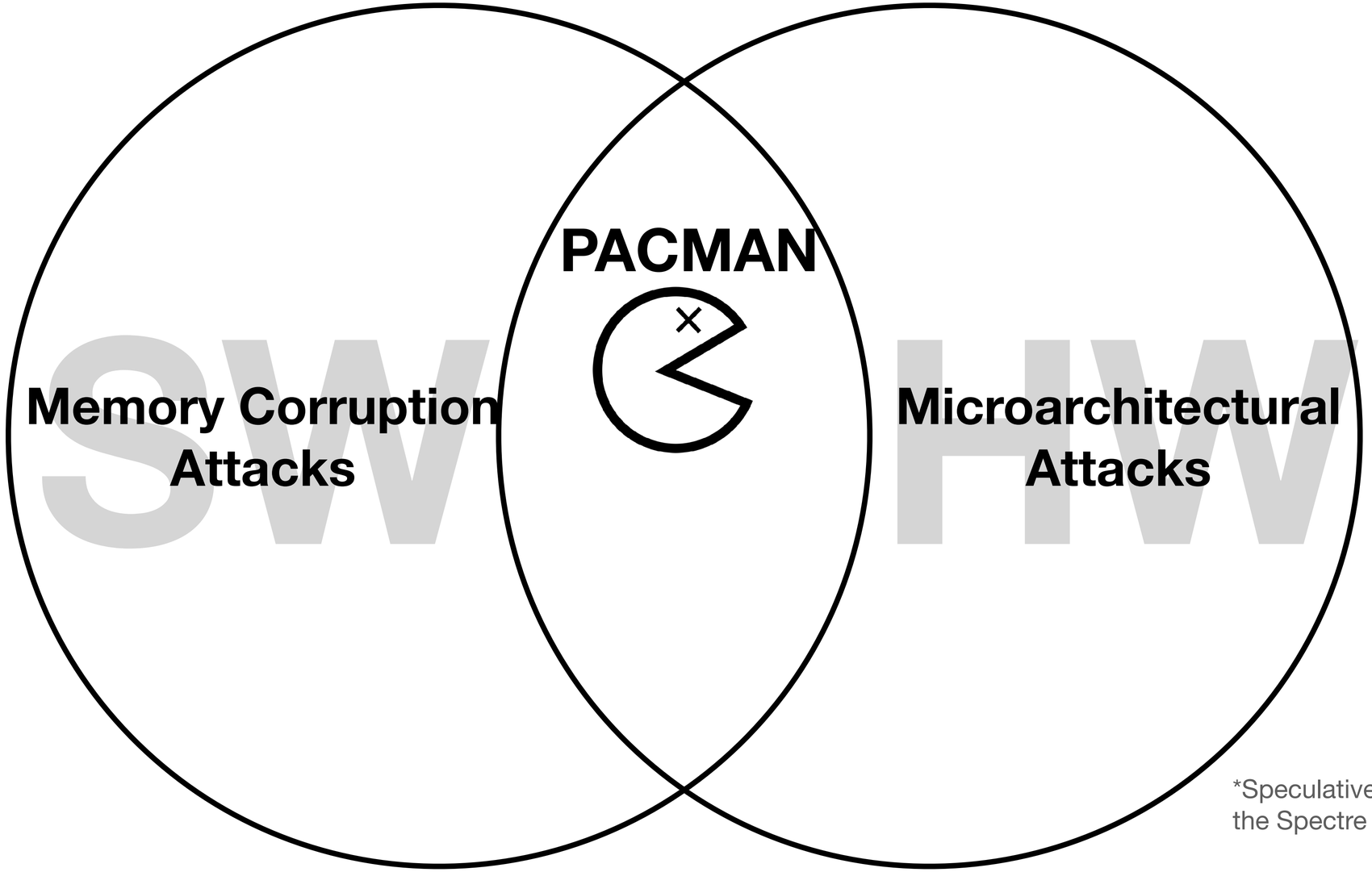
PacmanOS

A Rust-based bare metal
environment for performing experiments.

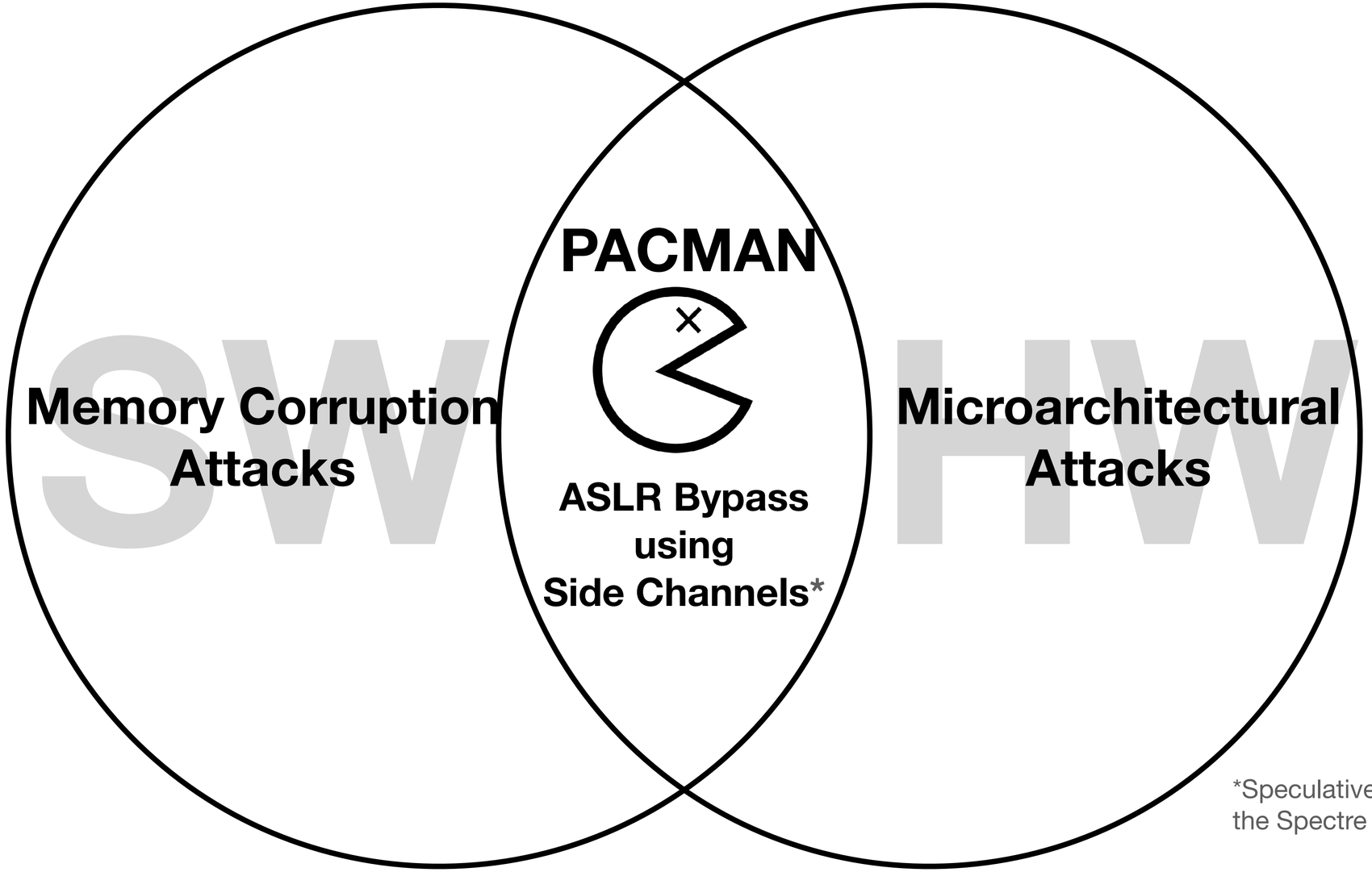


PACMAN @DEFCON

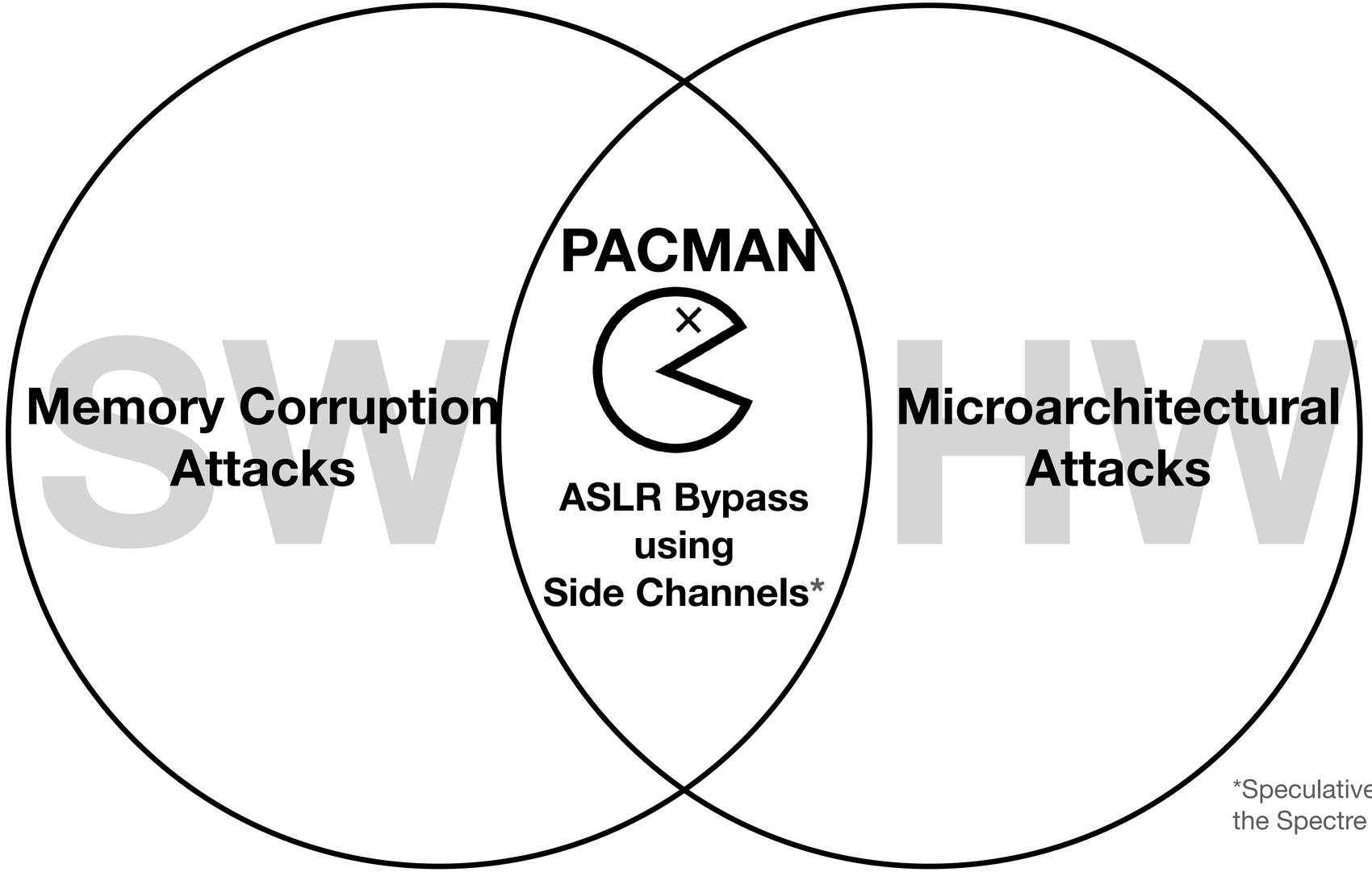




*Speculative Probing: Hacking Blind in the Spectre Era; Göktaş et al; CCS'20



*Speculative Probing: Hacking Blind in the Spectre Era; Göktaş et al; CCS'20

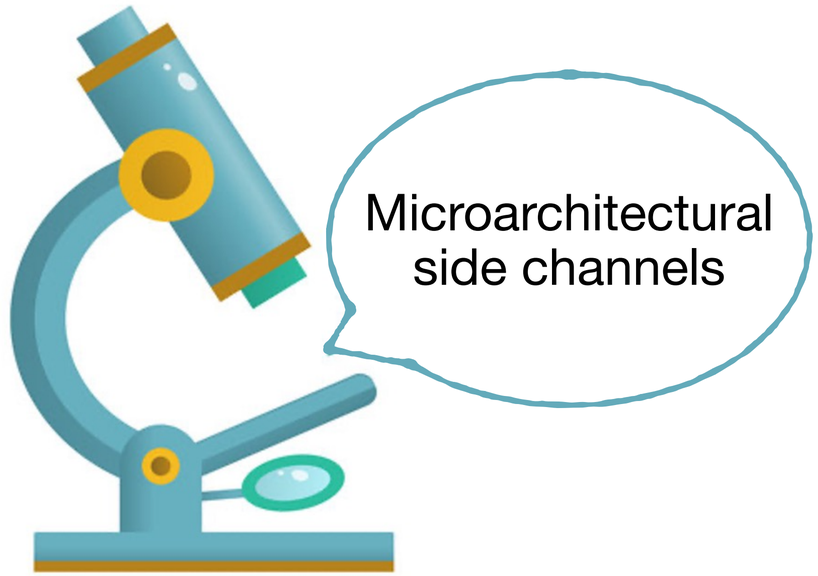


*Speculative Probing: Hacking Blind in the Spectre Era; Göktaş et al; CCS'20

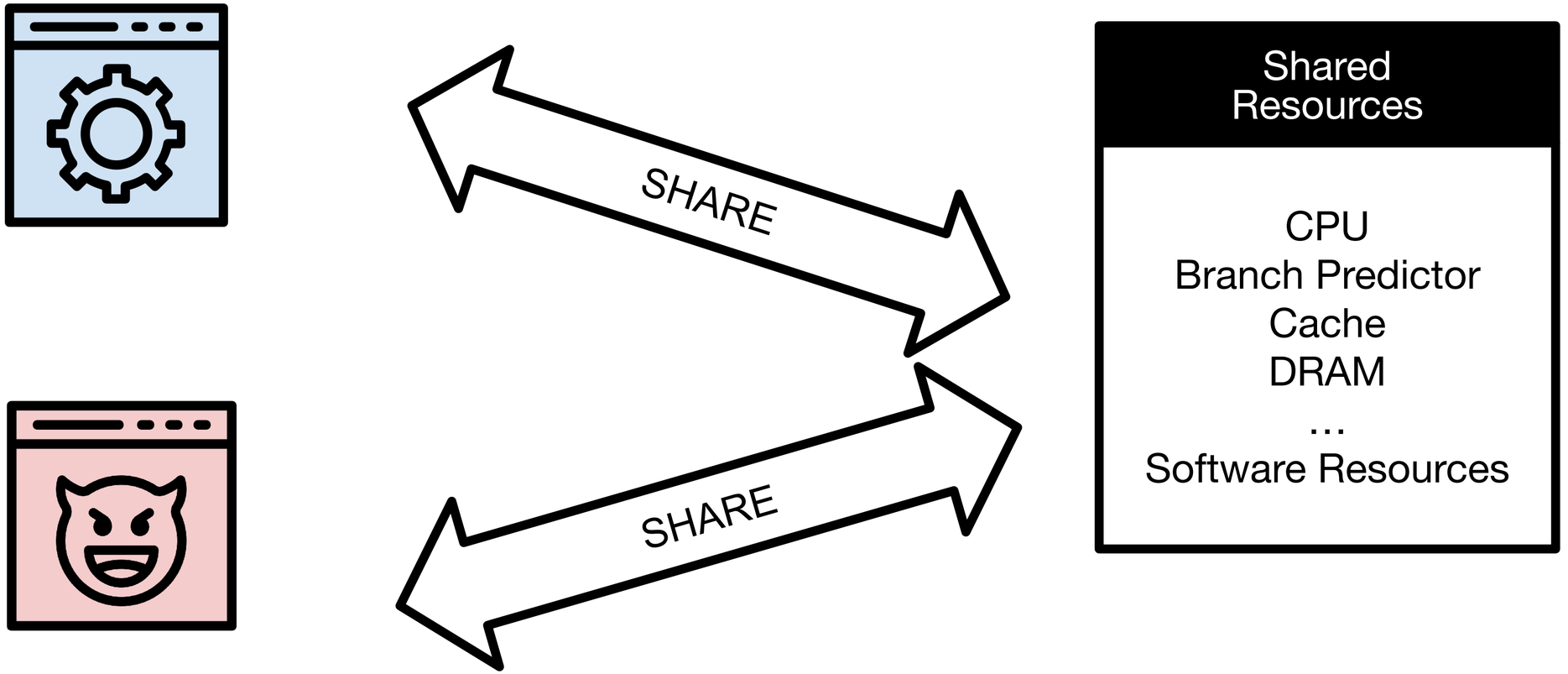
Takeaway 1: New threats arise from compound threat models

Limitations of Looking At Microarchitectural-only Side Channels

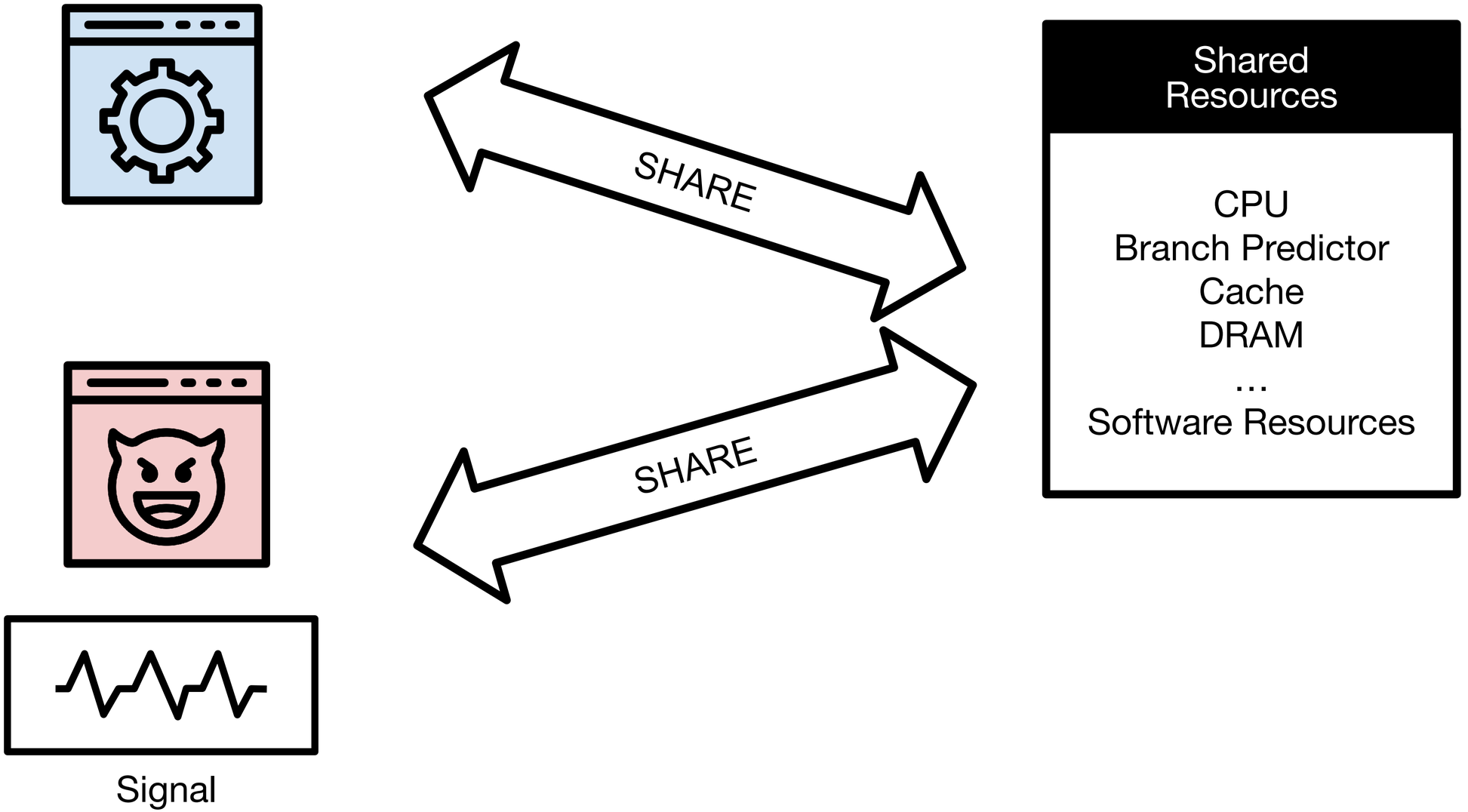
- Part 1: Miss threats that arise from compound threat models
- Part 2: Misunderstand root causes of existing side channel attacks



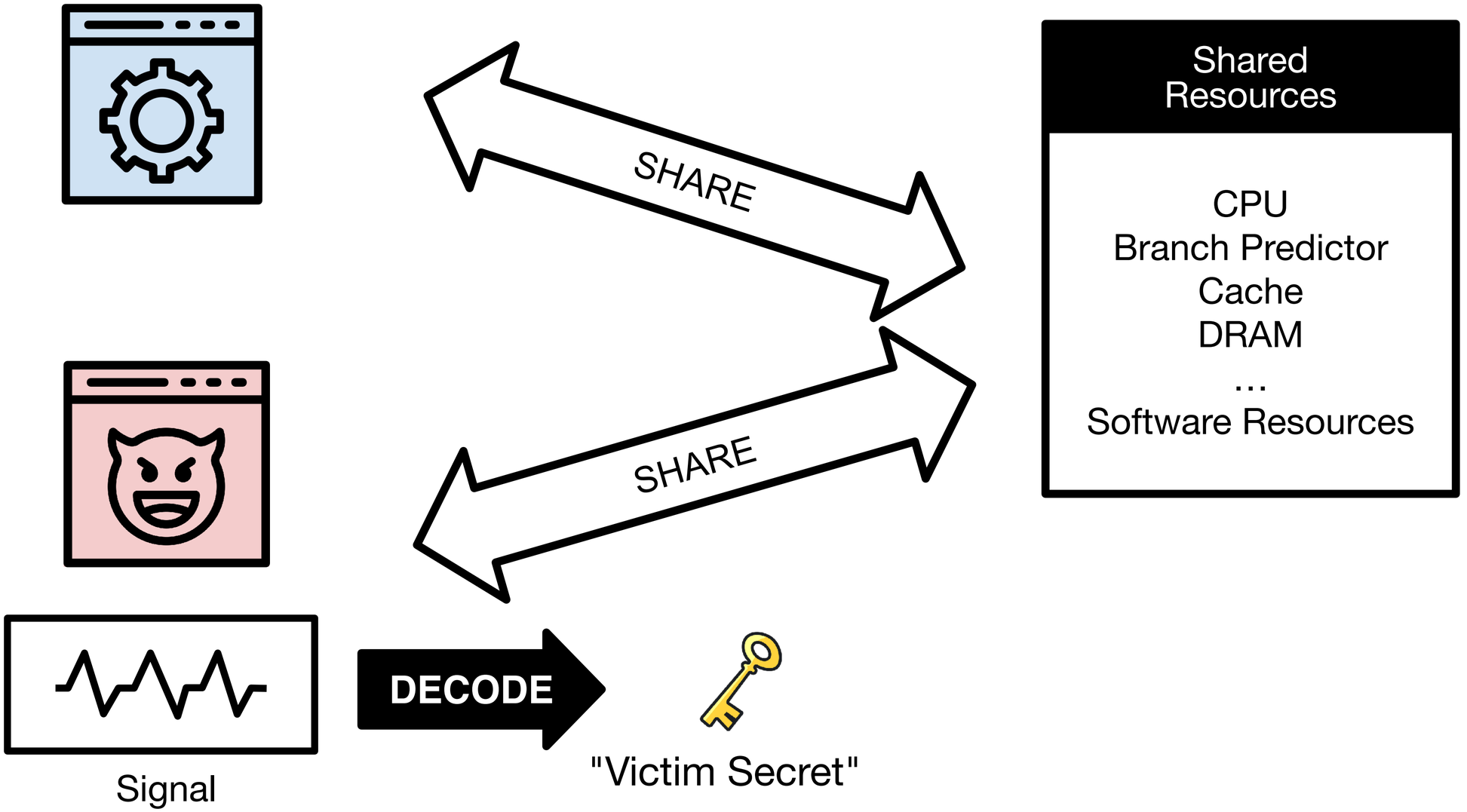
Microarchitectural Timing Side Channels



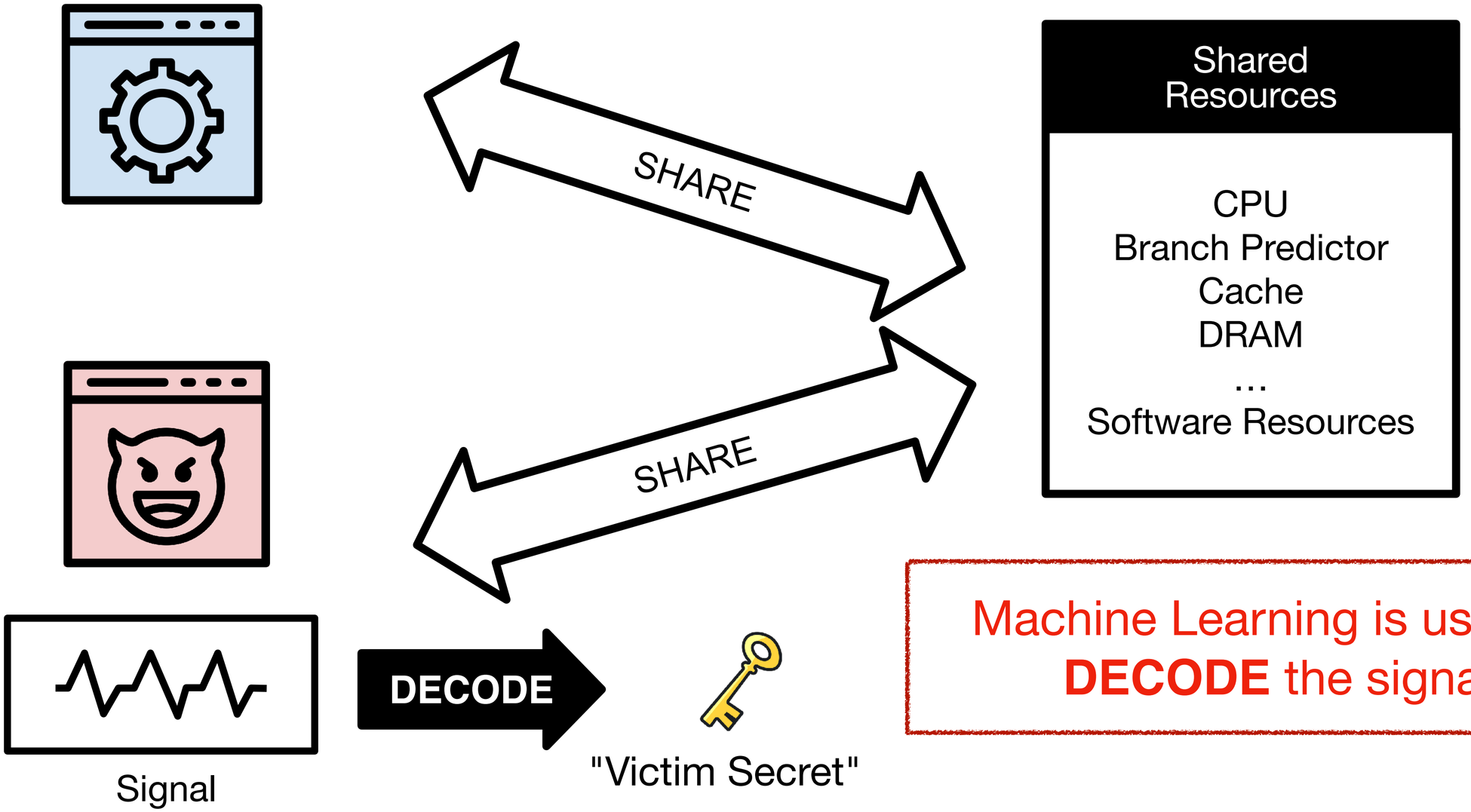
Microarchitectural Timing Side Channels



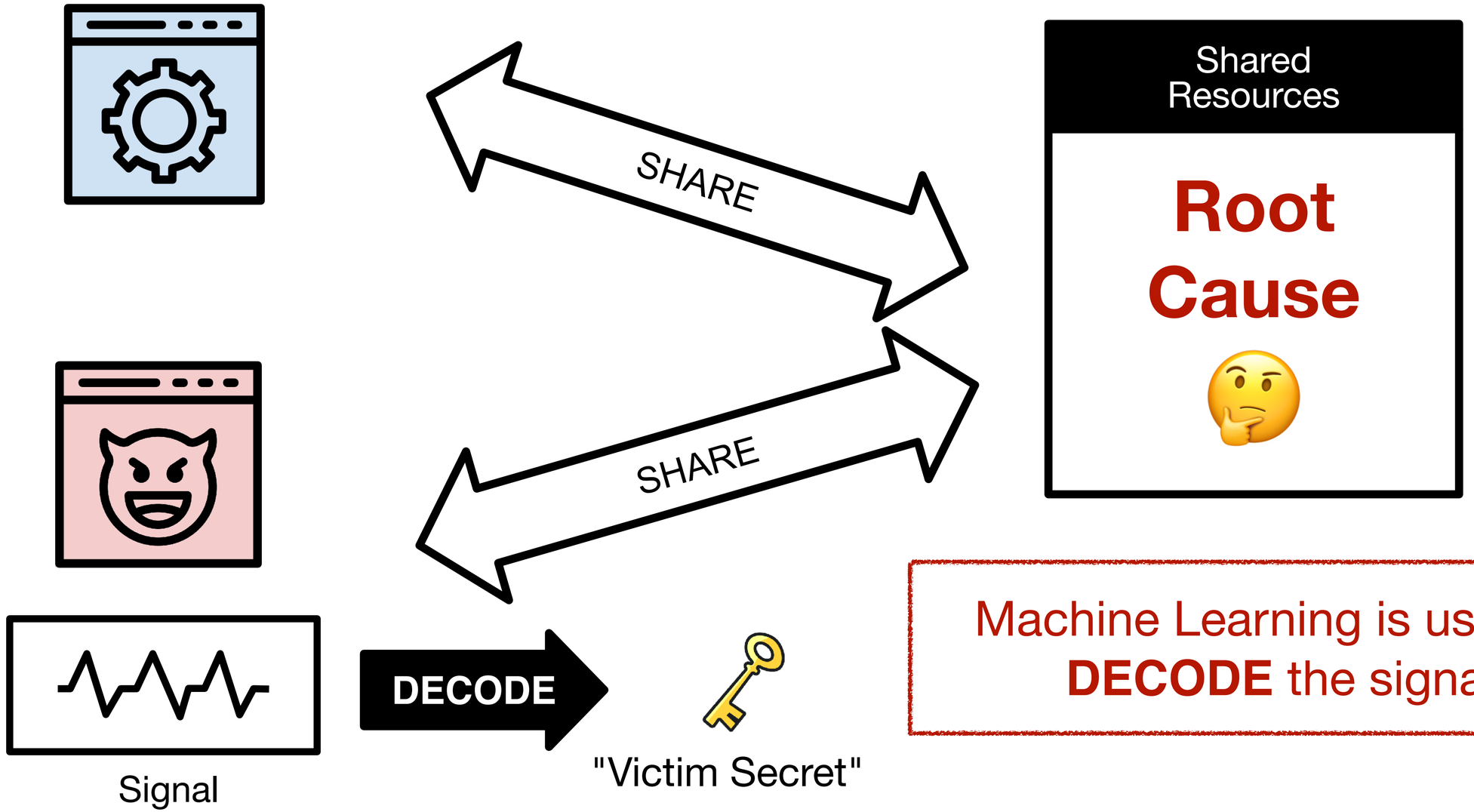
Microarchitectural Timing Side Channels



Microarchitectural Timing Side Channels



Microarchitectural Timing Side Channels



A Cache-Occupancy Attack*

ATTACKER'S CODE

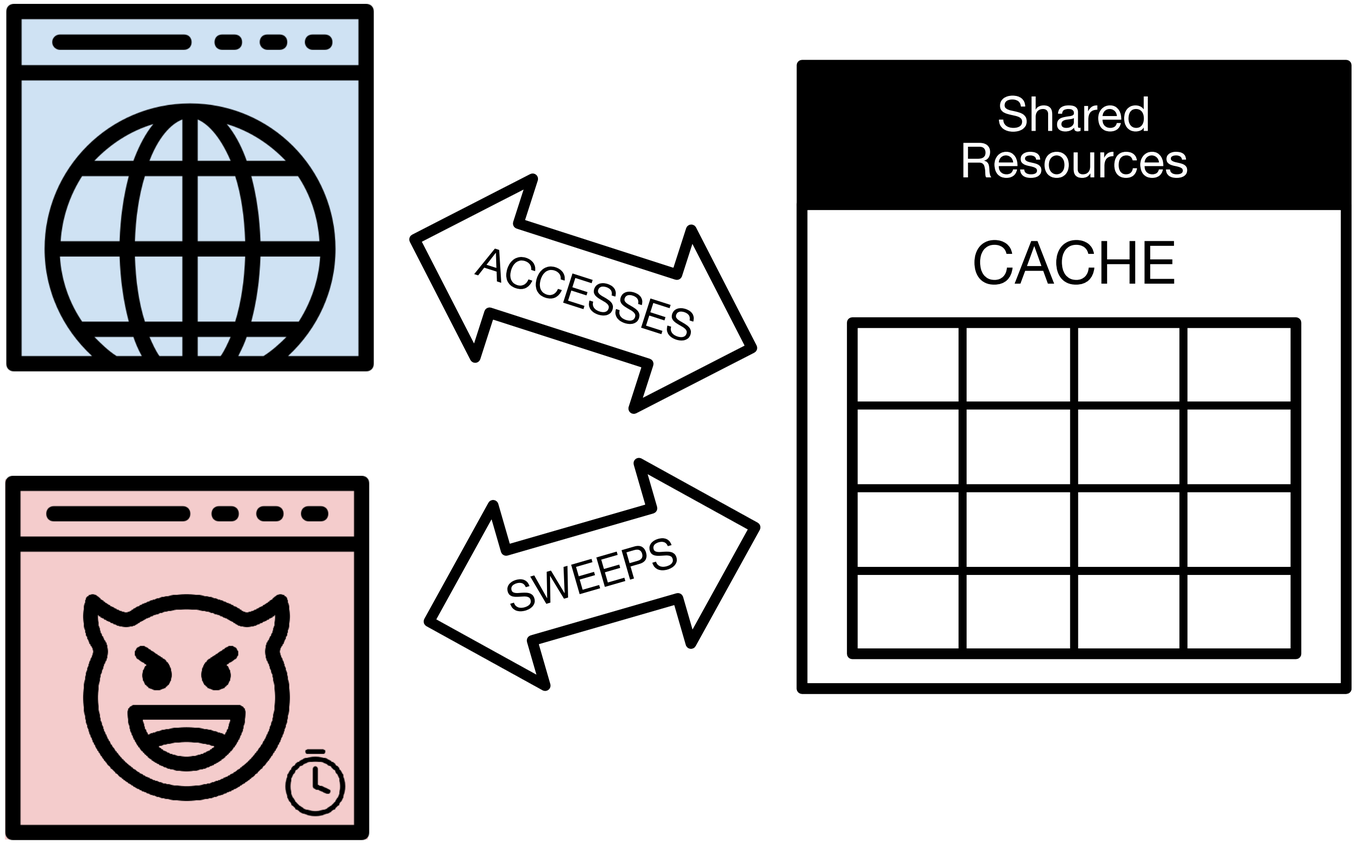
```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
    SWEEP_CACHE();
  }
  Trace[start] = counter;
}
```

* Shusterman, et al. "Prime+Probe 1,JavaScript 0: Overcoming Browser-based Side-Channel Defenses." *USENIX Security'21*

A Cache-Occupancy Attack*

ATTACKER'S CODE

```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
    SWEEP_CACHE();
  }
  Trace[start] = counter;
}
```

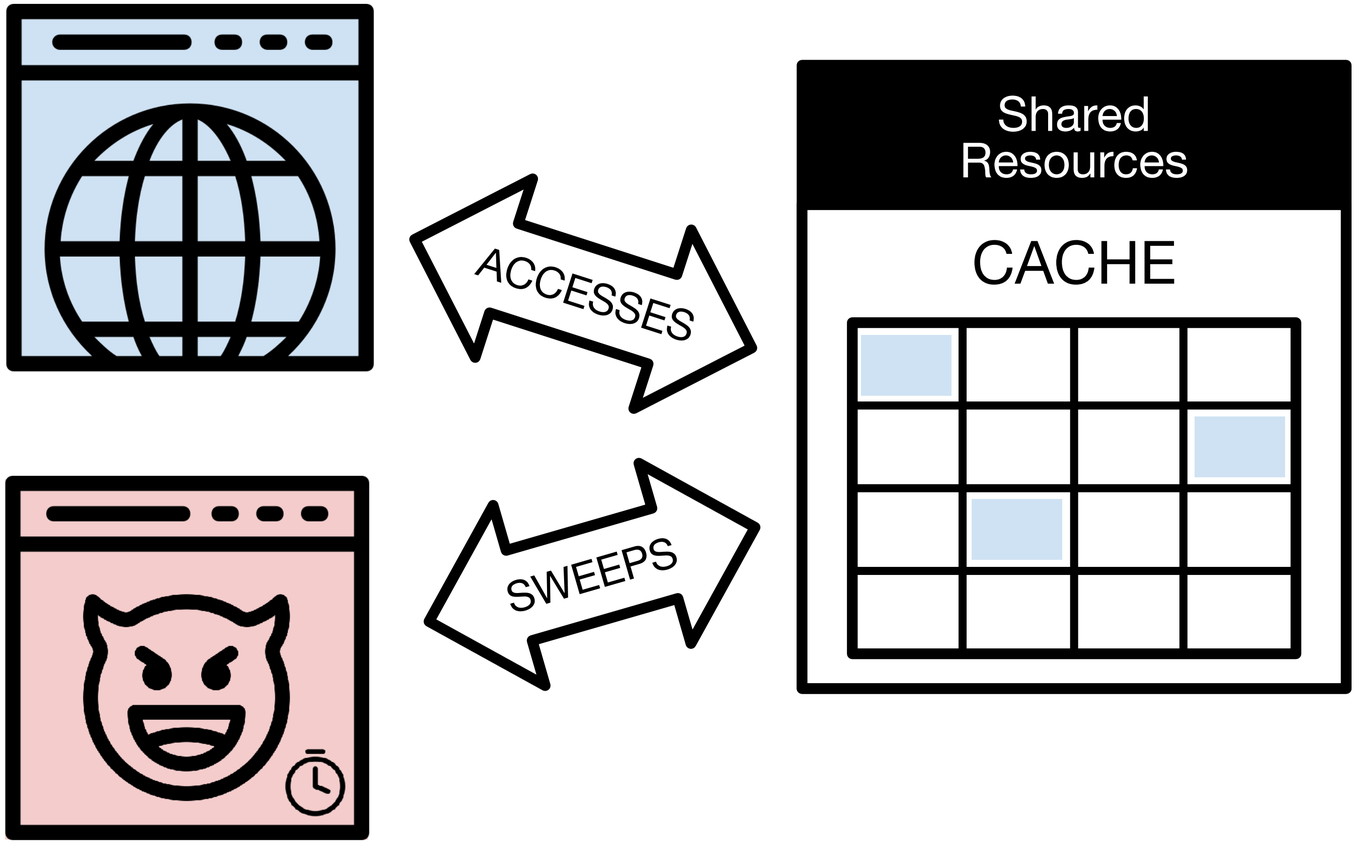


* Shusterman, et al. "Prime+Probe 1,JavaScript 0: Overcoming Browser-based Side-Channel Defenses." *USENIX Security'21*

A Cache-Occupancy Attack*

ATTACKER'S CODE

```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
    SWEEP_CACHE();
  }
  Trace[start] = counter;
}
```

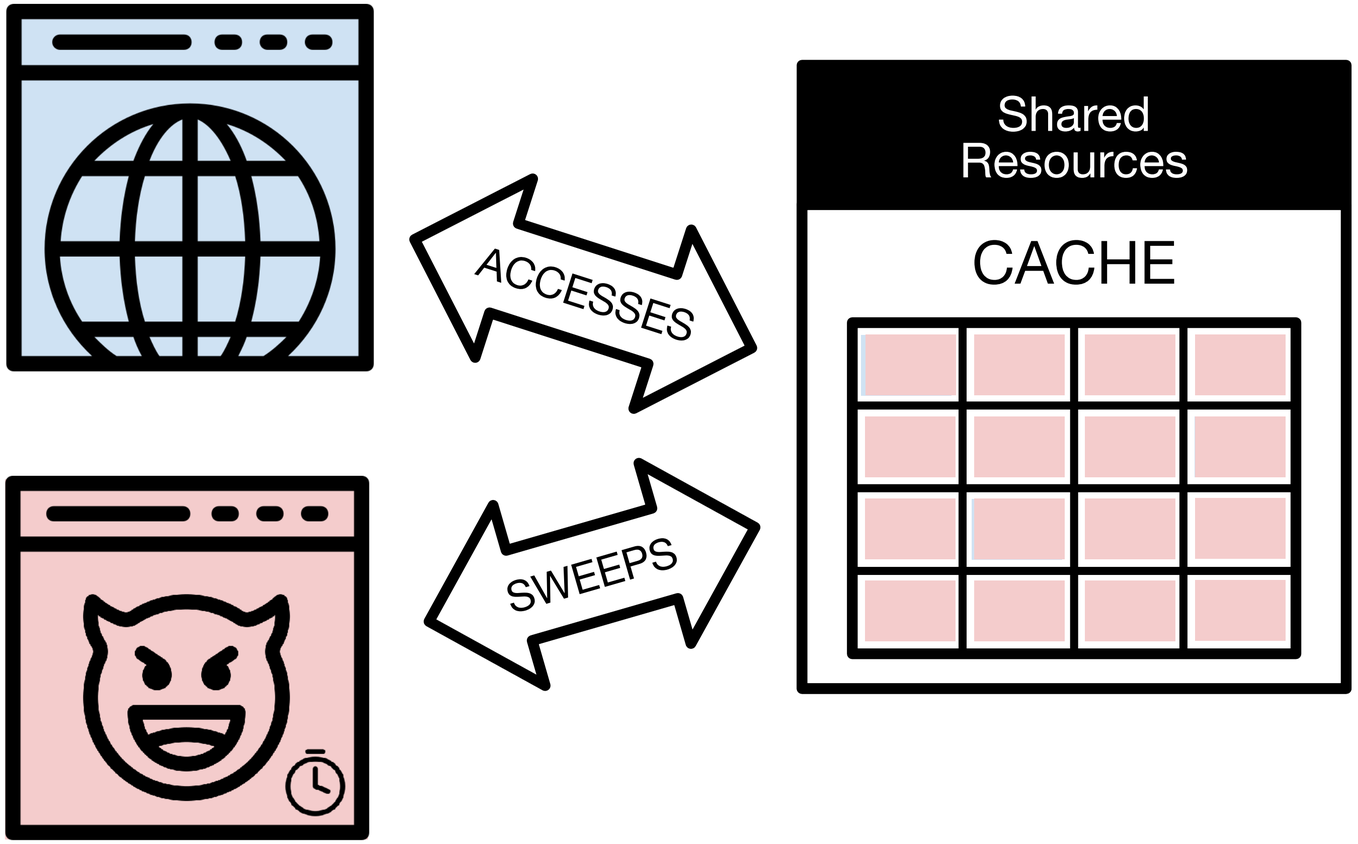


* Shusterman, et al. "Prime+Probe 1,JavaScript 0: Overcoming Browser-based Side-Channel Defenses." *USENIX Security'21*

A Cache-Occupancy Attack*

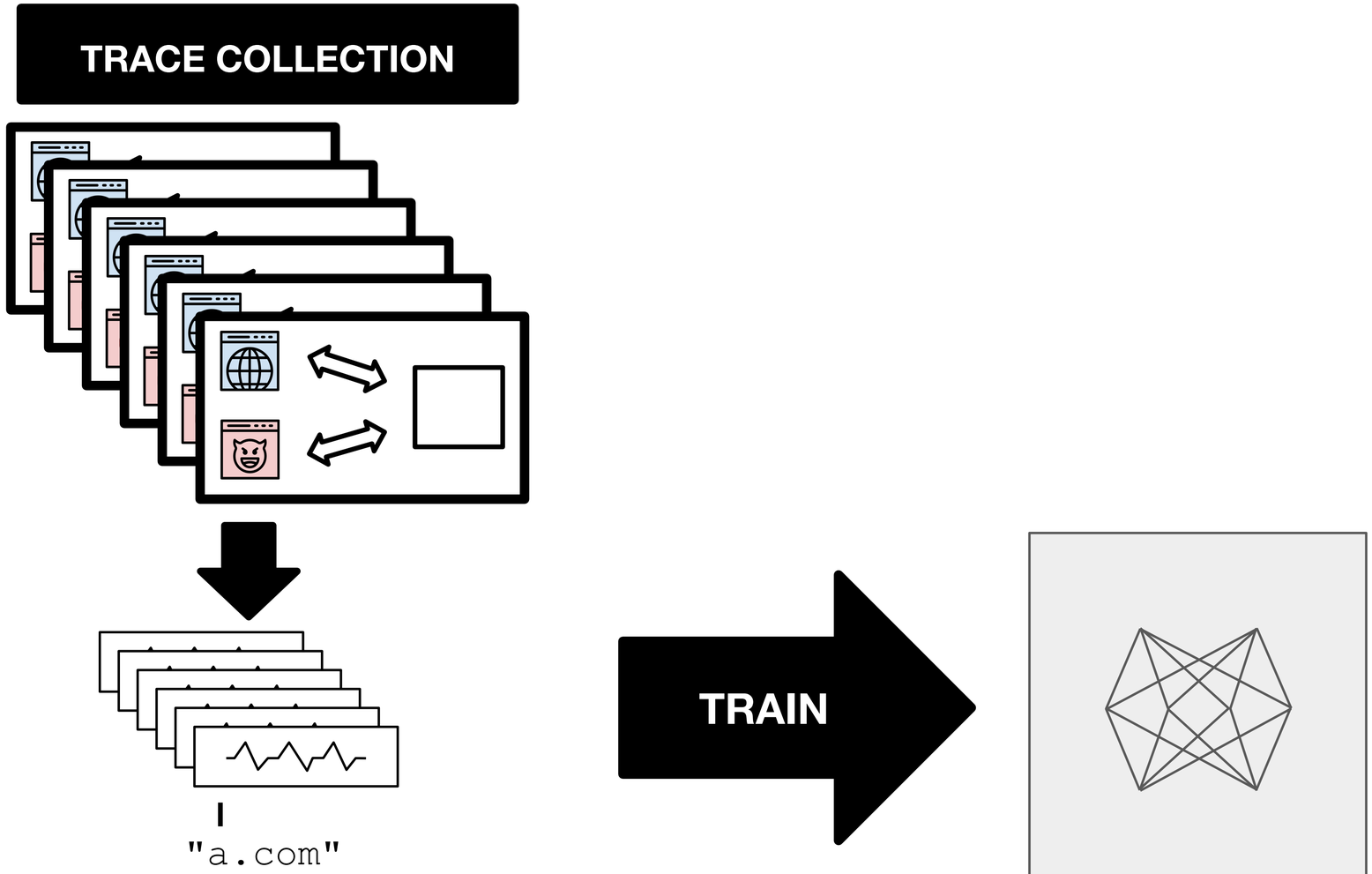
ATTACKER'S CODE

```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
    SWEEP_CACHE();
  }
  Trace[start] = counter;
}
```

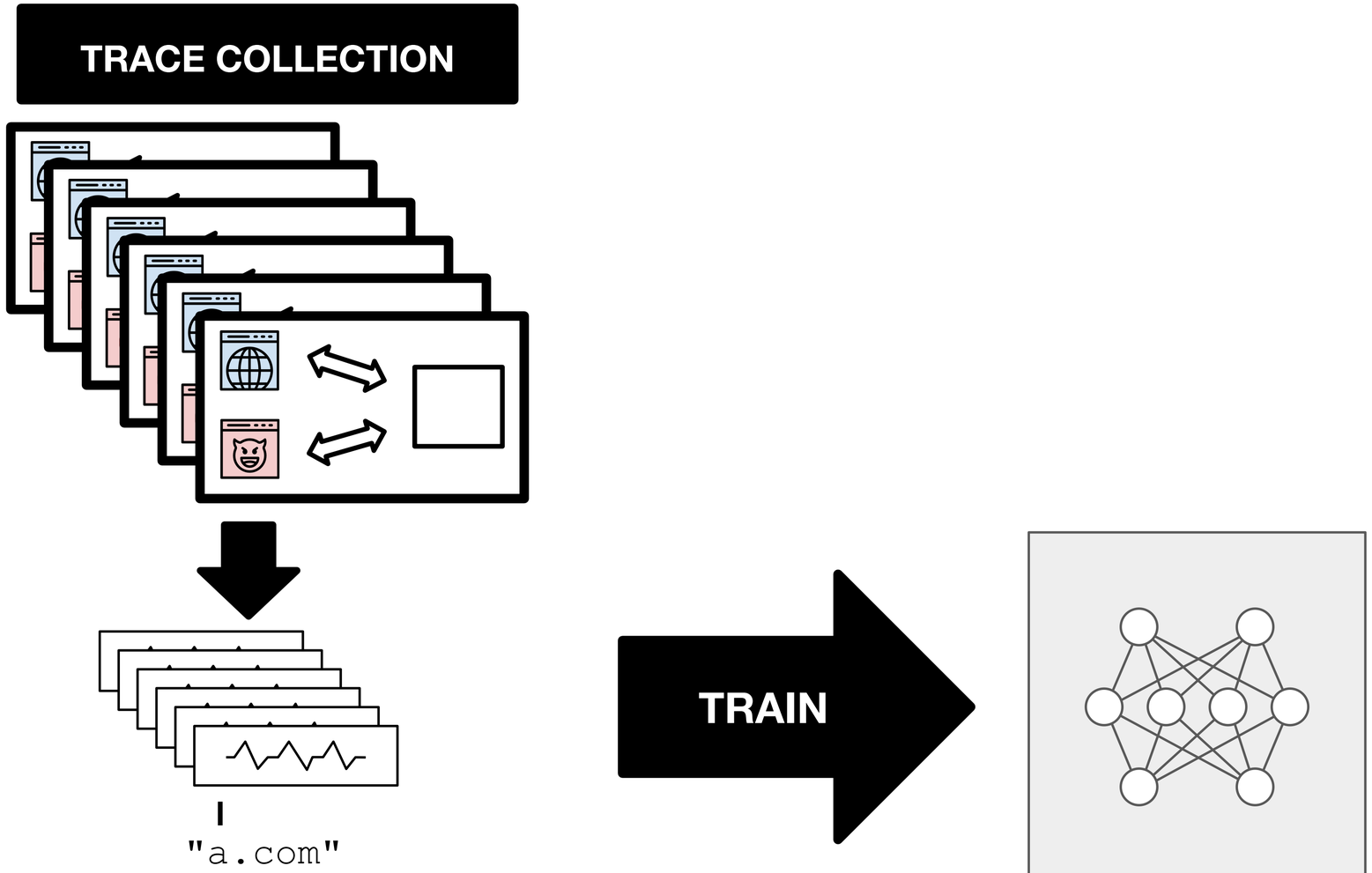


* Shusterman, et al. "Prime+Probe 1,JavaScript 0: Overcoming Browser-based Side-Channel Defenses." *USENIX Security'21*

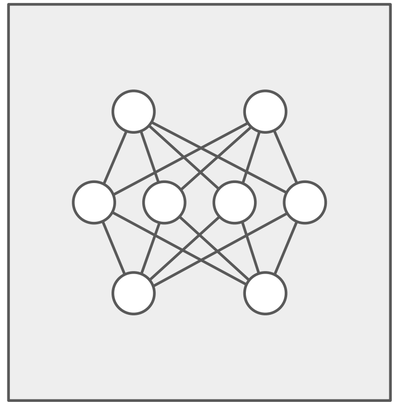
Website Fingerprinting



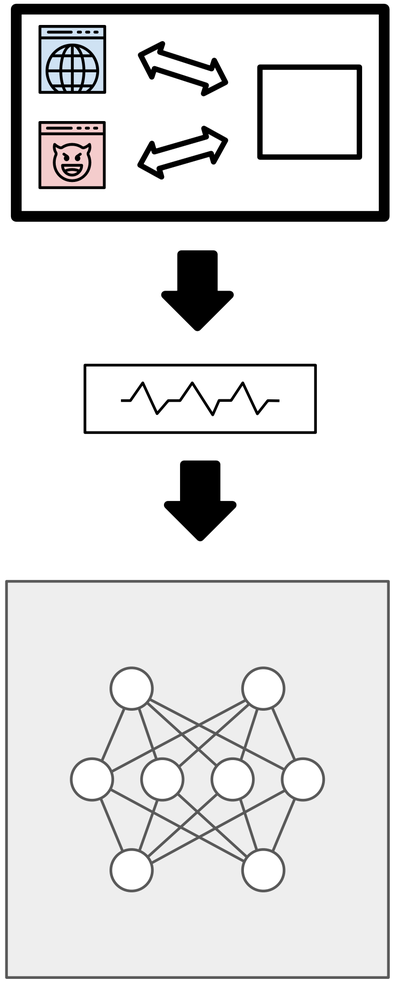
Website Fingerprinting



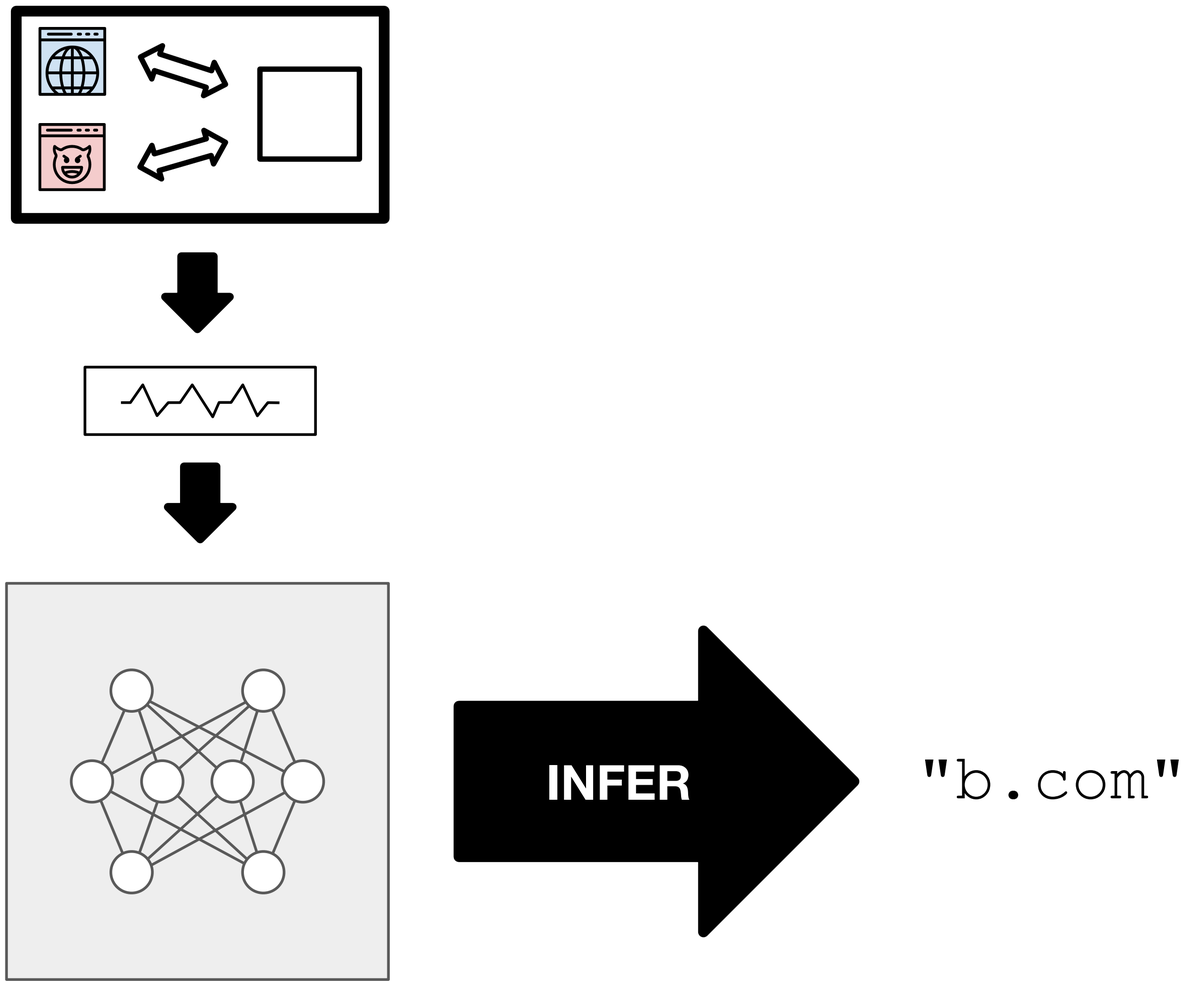
Website Fingerprinting



Website Fingerprinting



Website Fingerprinting



A Surprising Experiment

ATTACKER'S CODE

```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
    SWEEP_CACHE();
  }
  Trace[start] = counter;
}
```

A Surprising Experiment

ATTACKER'S CODE

```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
  }
  Trace[start] = counter;
}
```

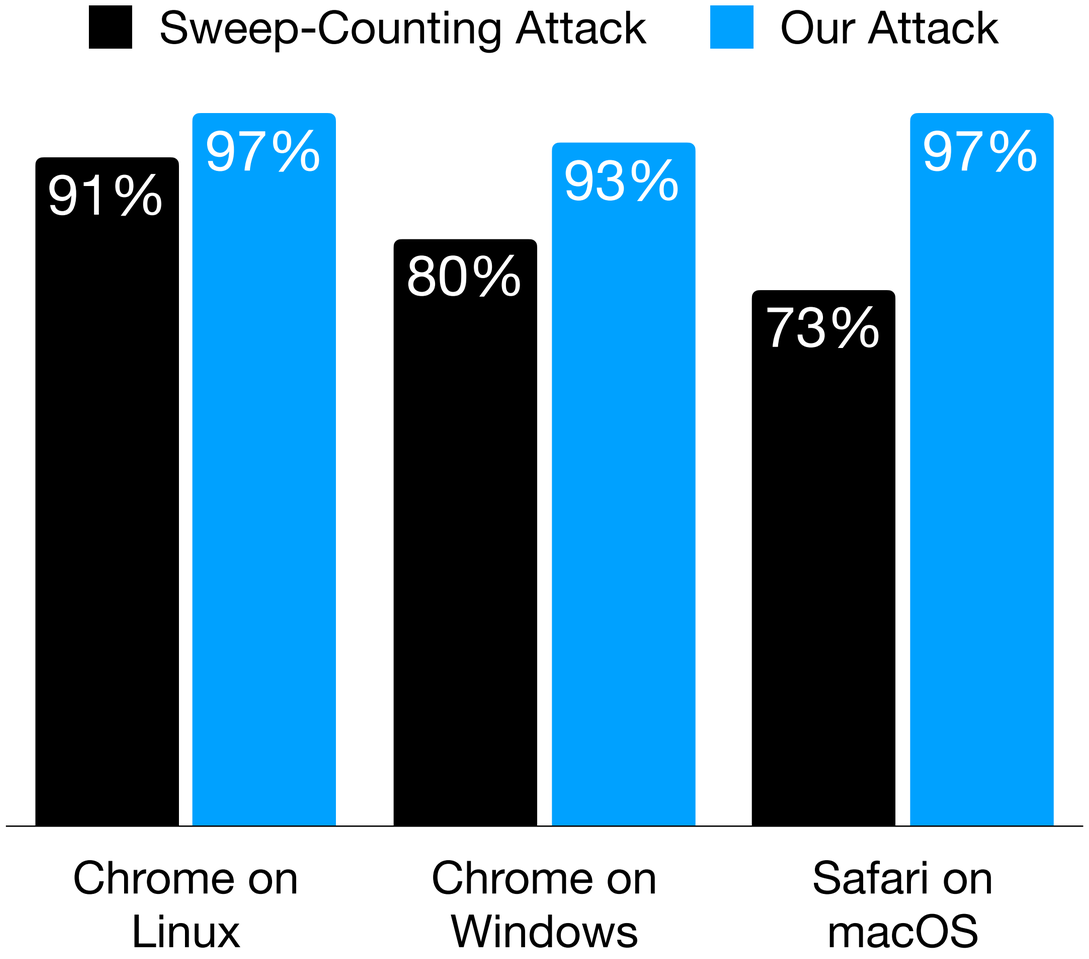
REMOVE MEMORY ACCESSES

A Surprising Experiment

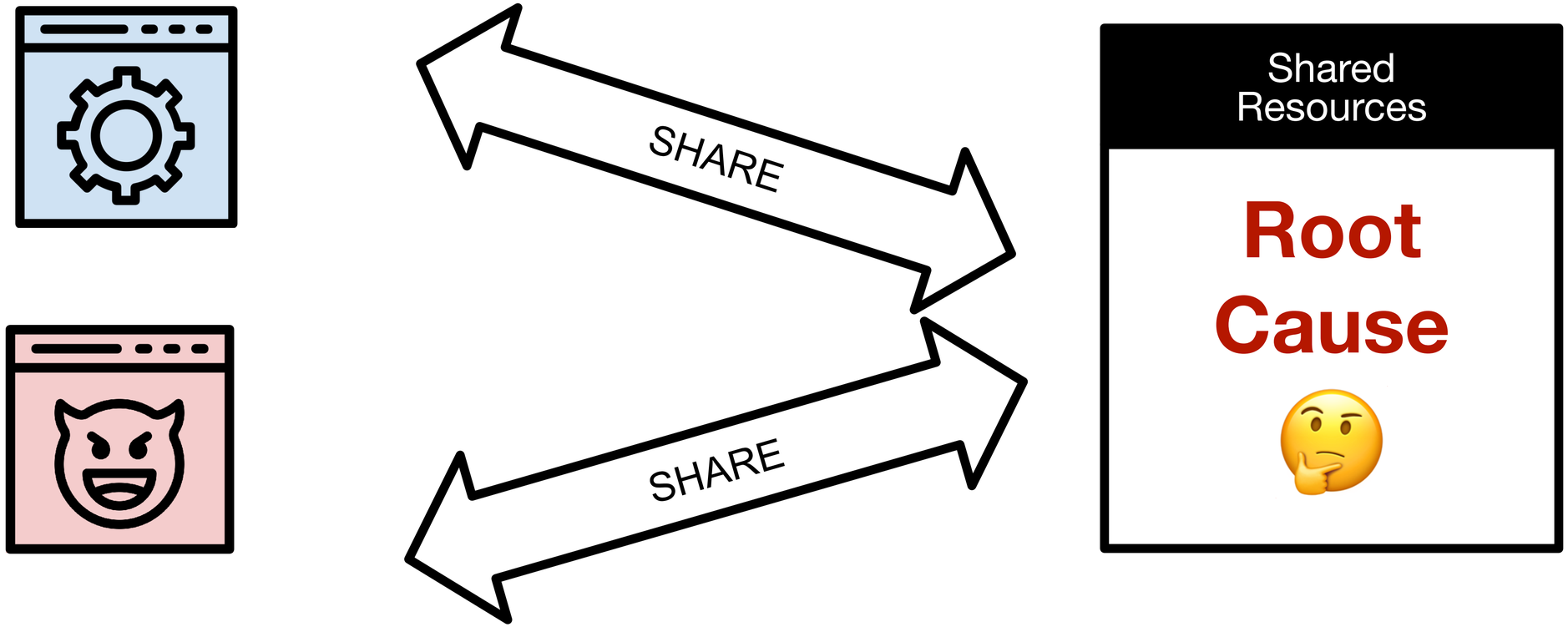
ATTACKER'S CODE

```
loop {
  start = time()
  counter = 0;
  while(time() - start < 5ms) {
    counter++;
  }
  Trace[start] = counter;
}
```

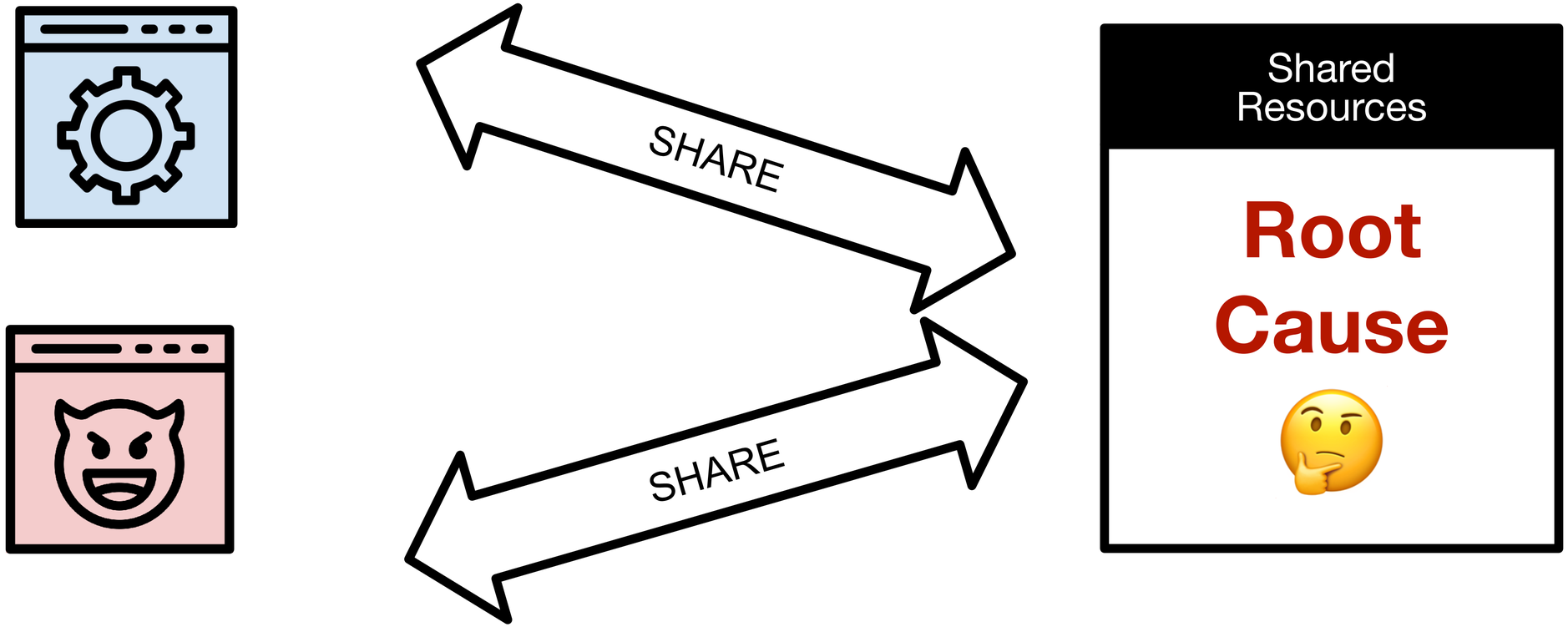
REMOVE MEMORY ACCESSES



What Is The Primary Side Channel?



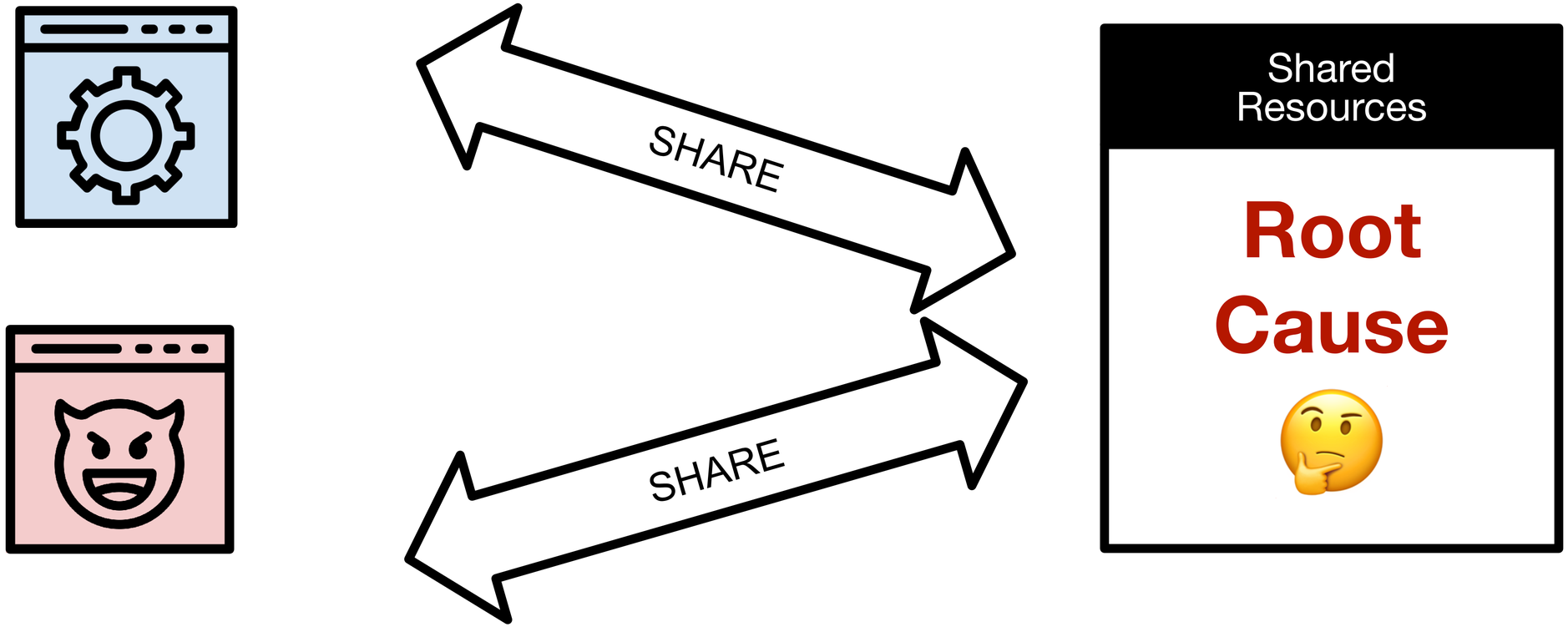
What Is The Primary Side Channel?



ML-assisted side-channel attacks work as a black box.

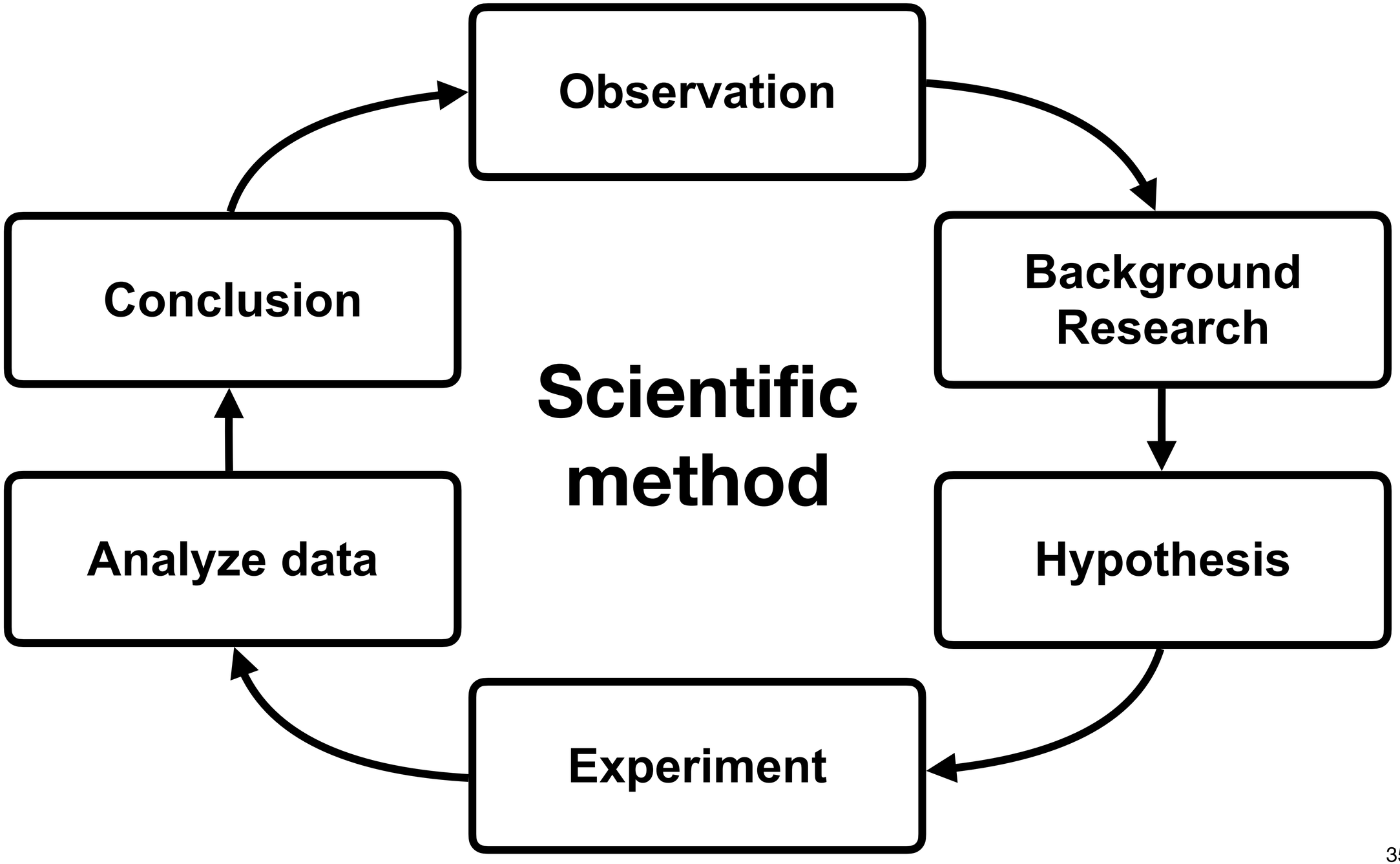
It is challenging to find the root cause.

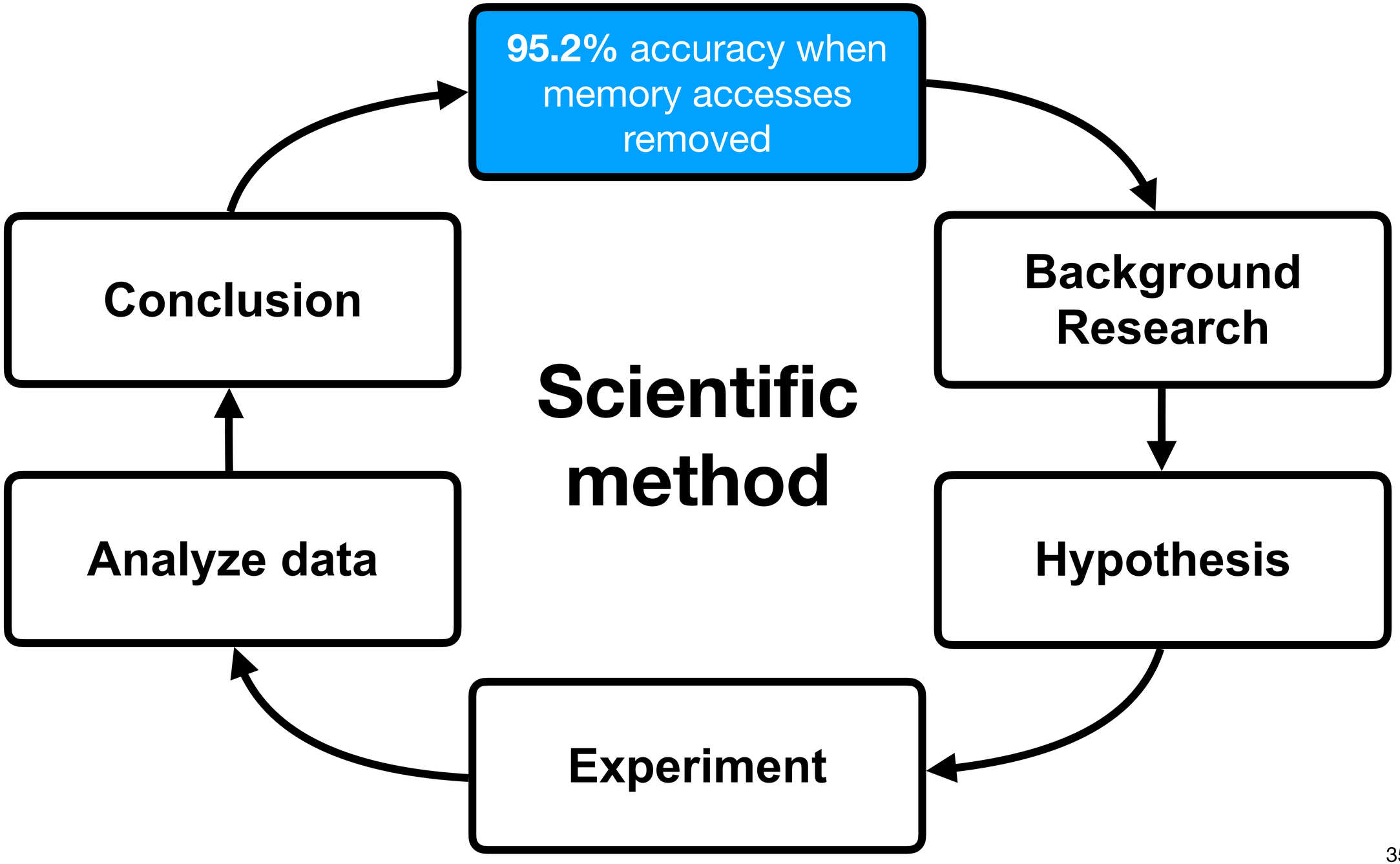
What Is The Primary Side Channel?

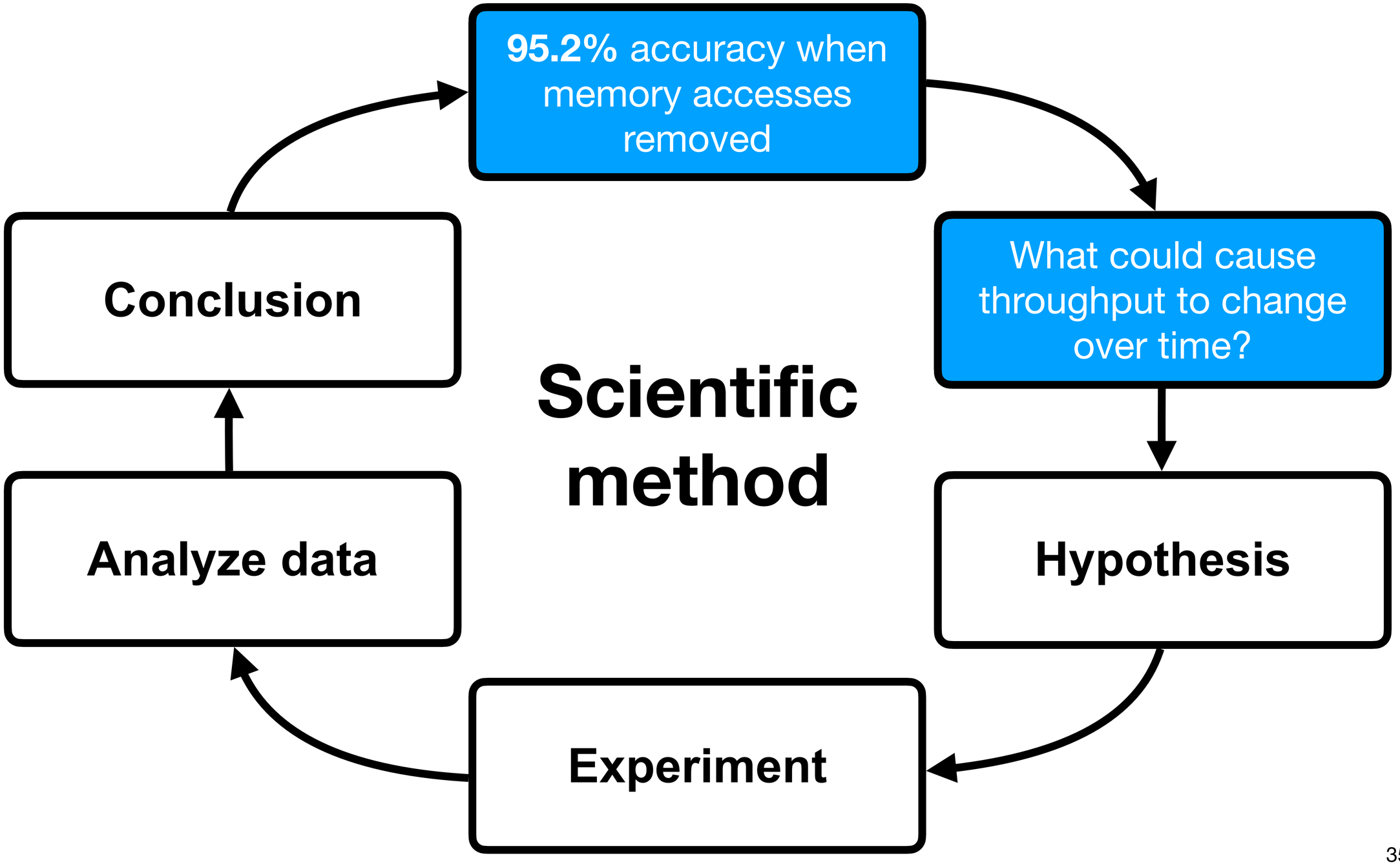


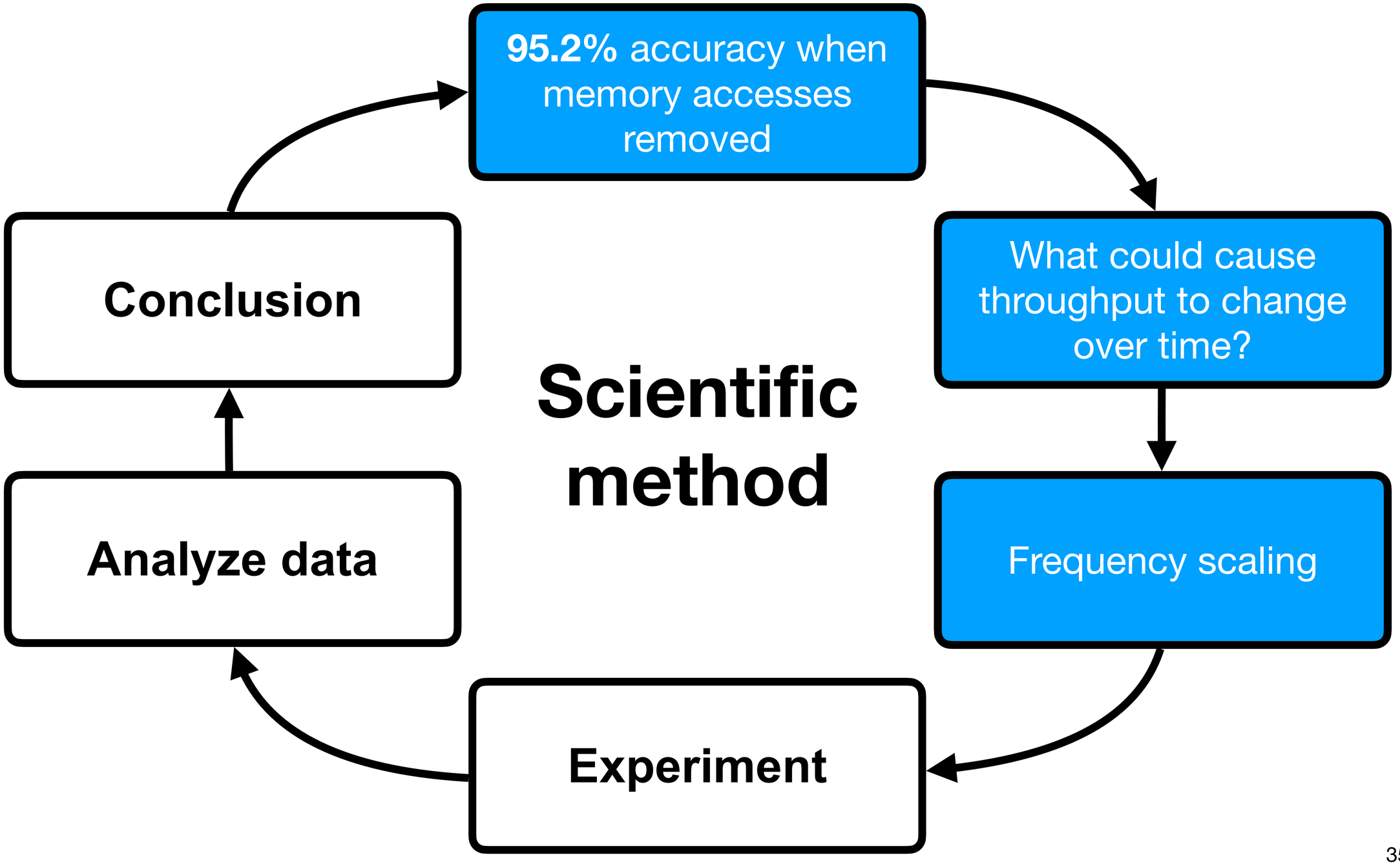
ML-assisted side-channel attacks work as a black box.

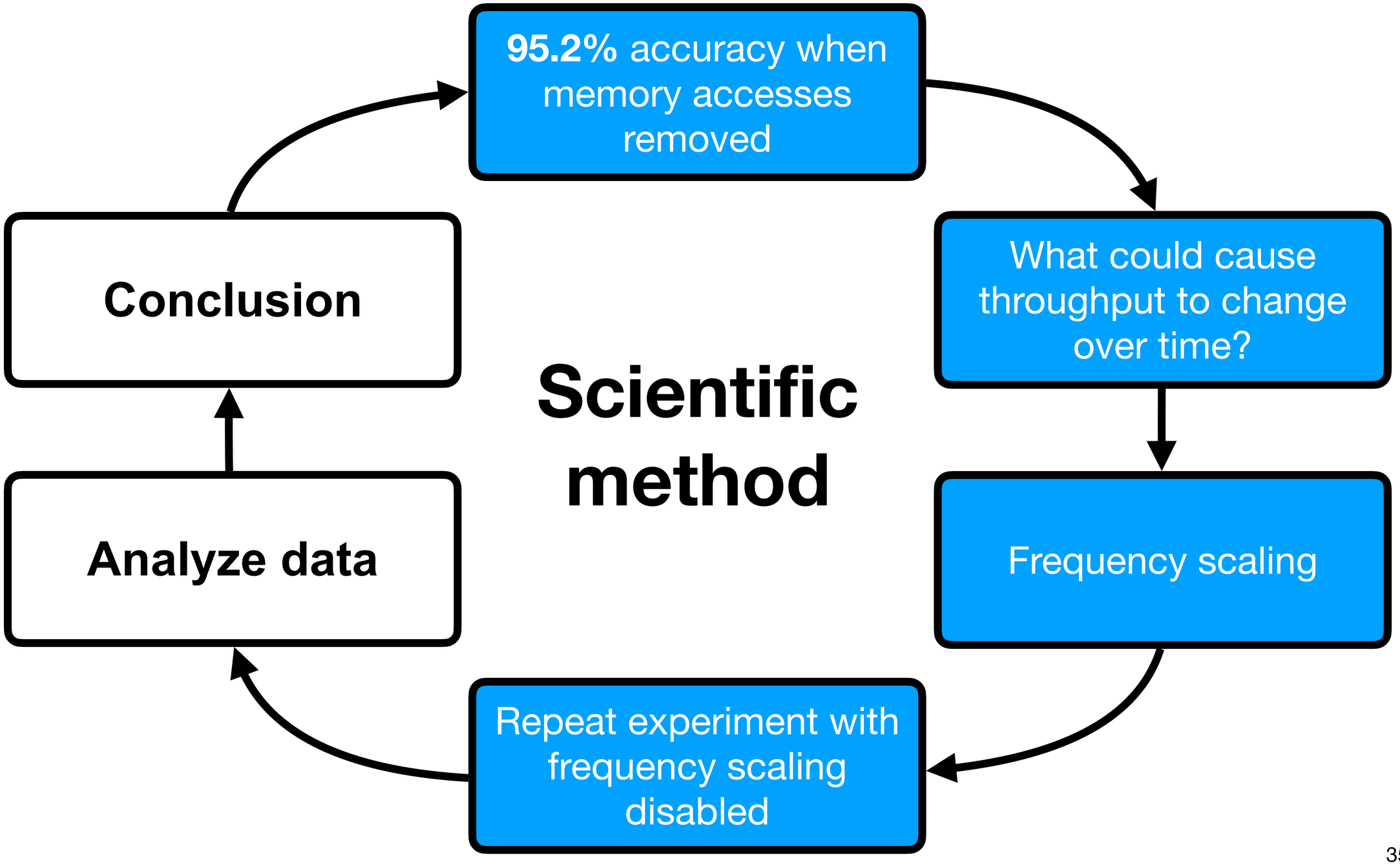
It is **challenging** to find the root cause**(s)**.

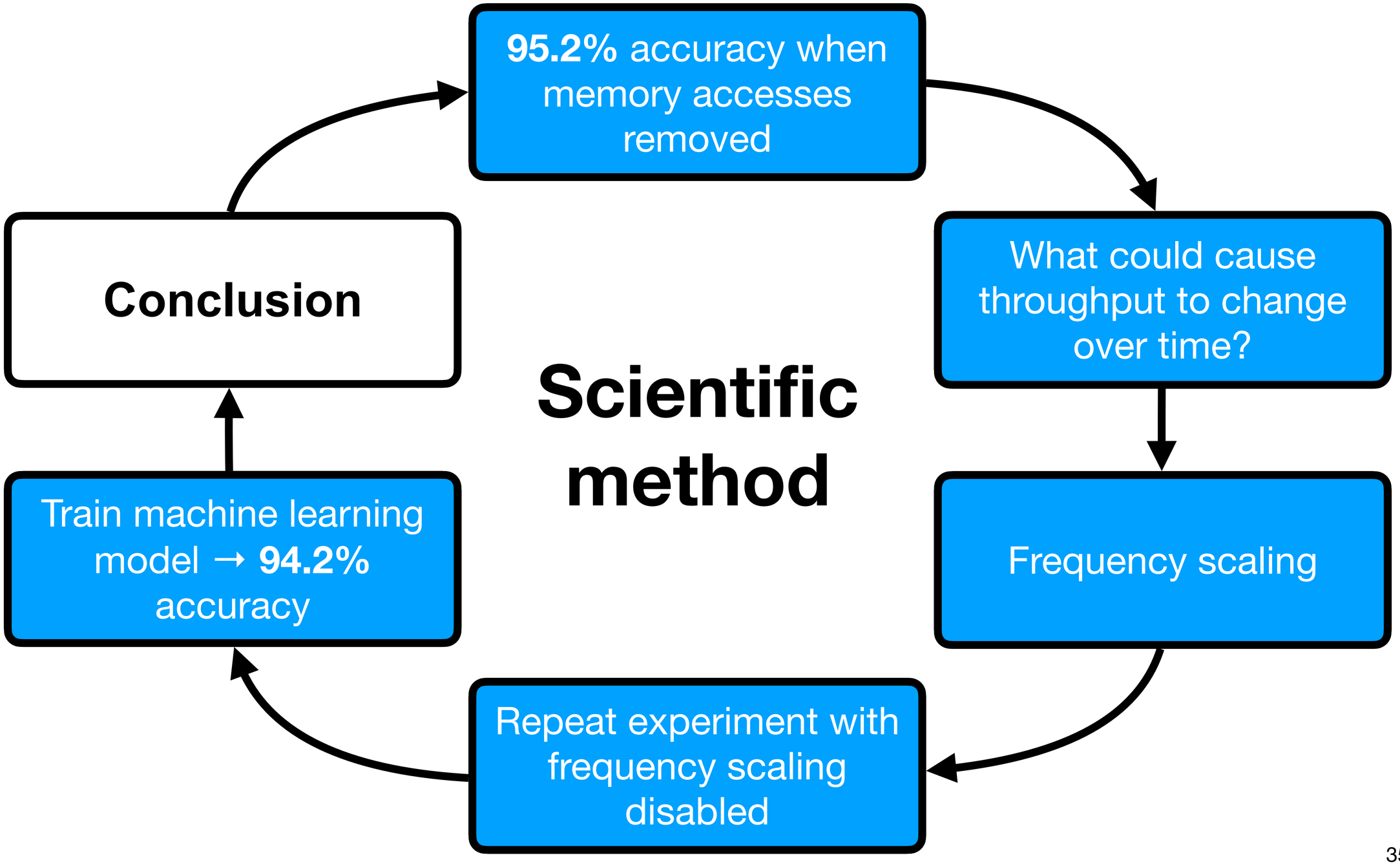

Observation
Background Research
Hypothesis
Experiment
Analyze data
Conclusion
Scientific method

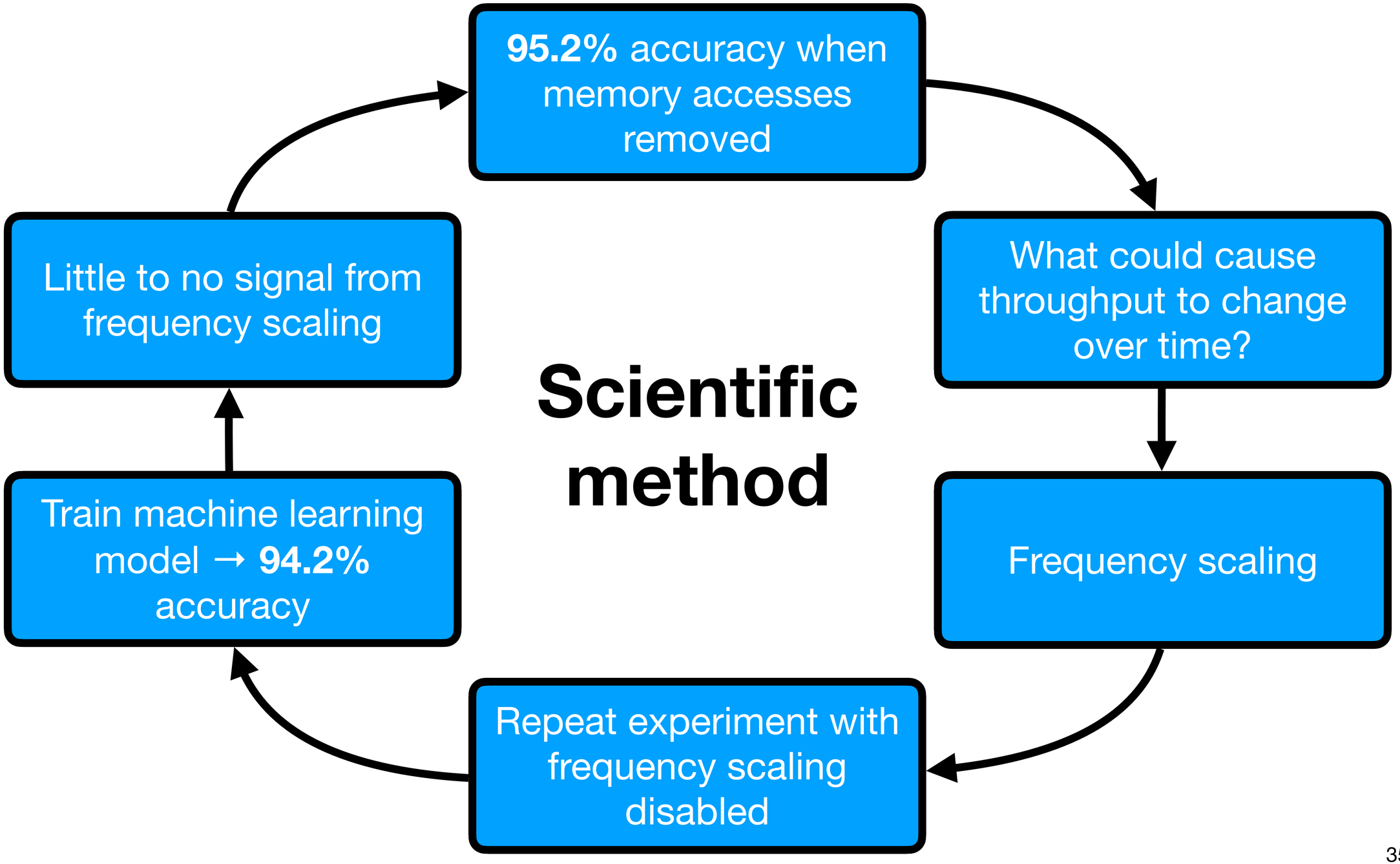

95.2% accuracy when memory accesses removed
Background Research
Hypothesis
Experiment
Analyze data
Conclusion
Scientific method

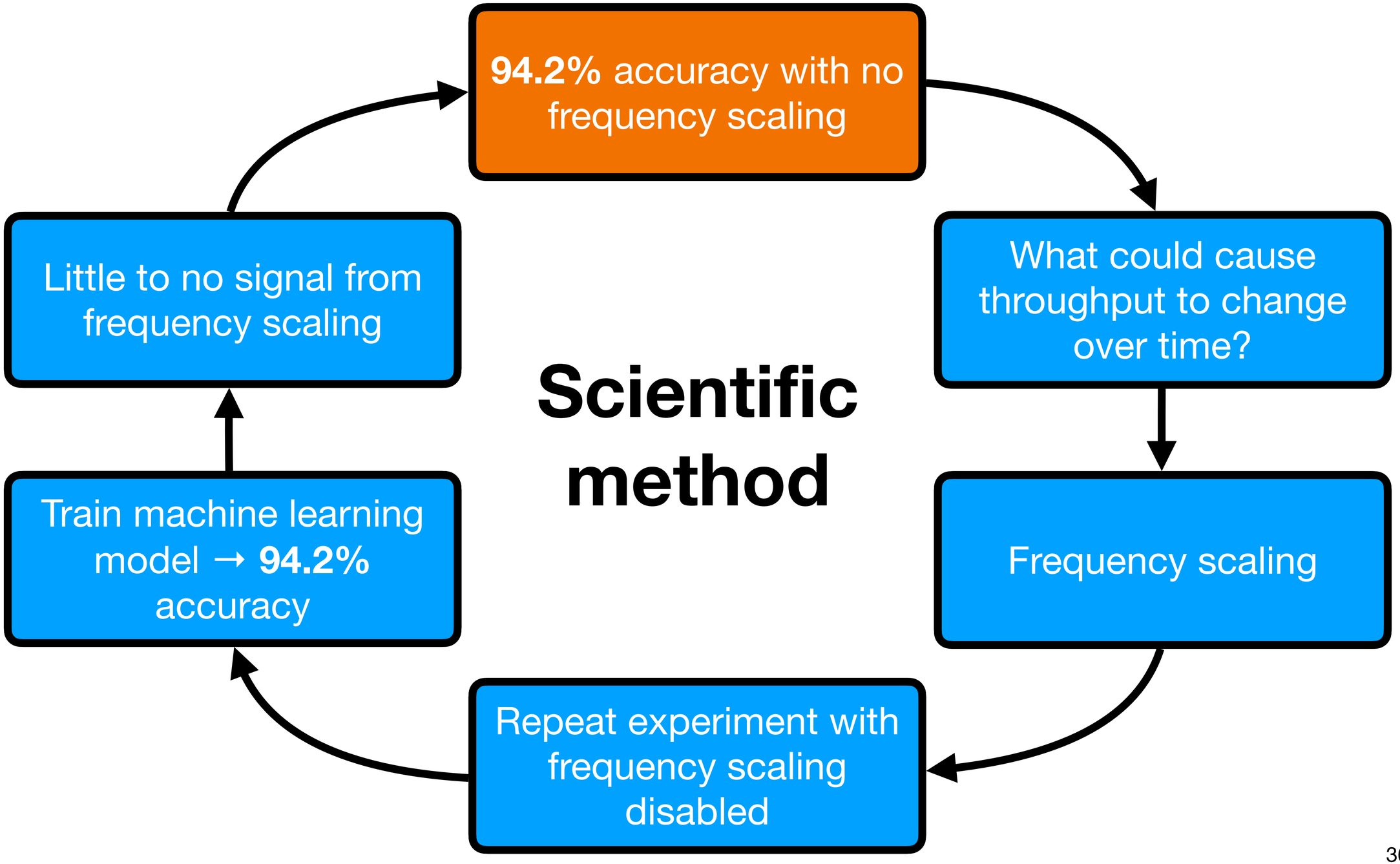

95.2% accuracy when memory accesses removed
What could cause throughput to change over time?
Hypothesis
Experiment
Analyze data
Conclusion
Scientific method

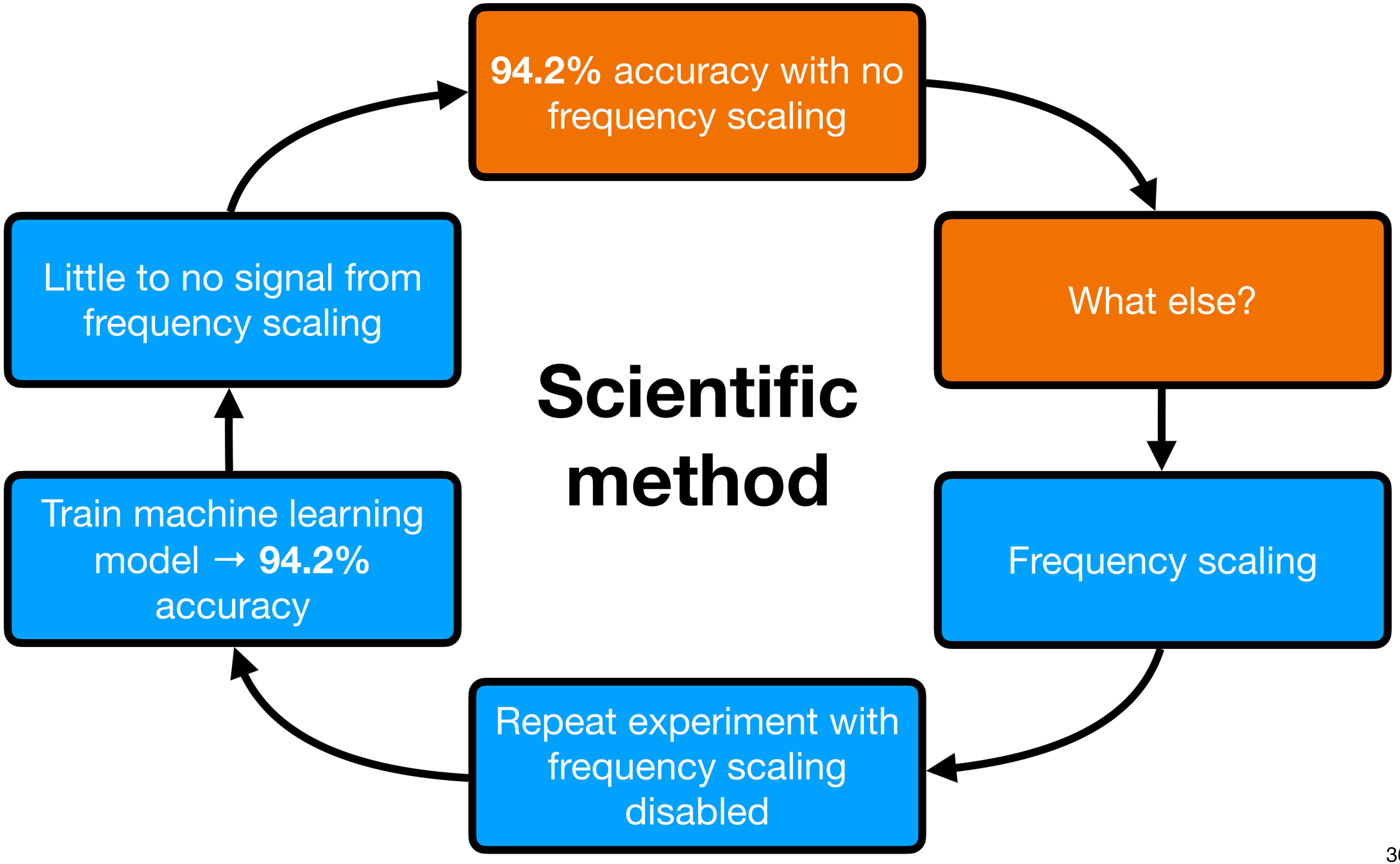

95.2% accuracy when memory accesses removed
What could cause throughput to change over time?
Frequency scaling
Experiment
Analyze data
Conclusion
Scientific method

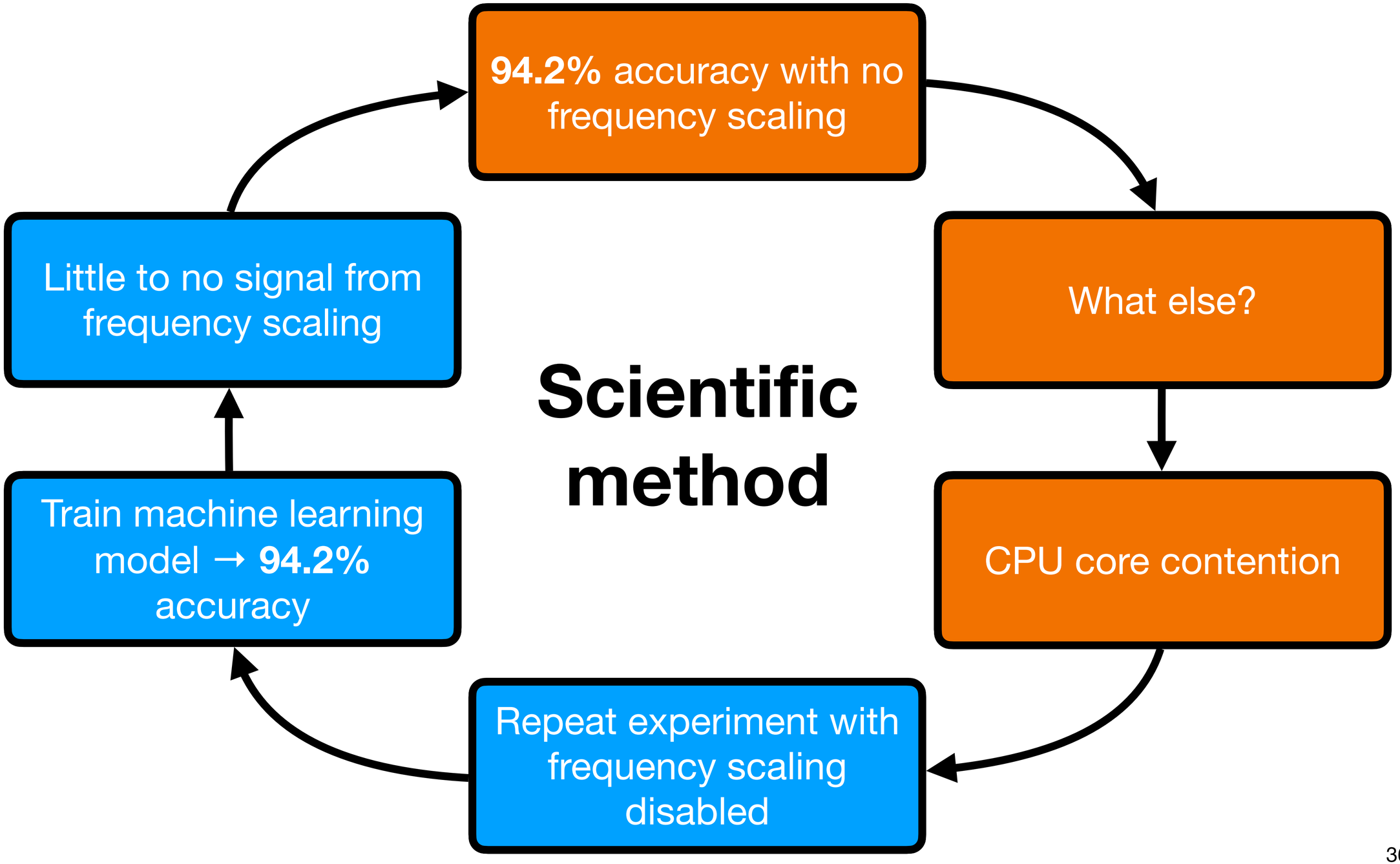

95.2% accuracy when memory accesses removed
What could cause throughput to change over time?
Frequency scaling
Repeat experiment with frequency scaling disabled
Analyze data
Conclusion
Scientific method

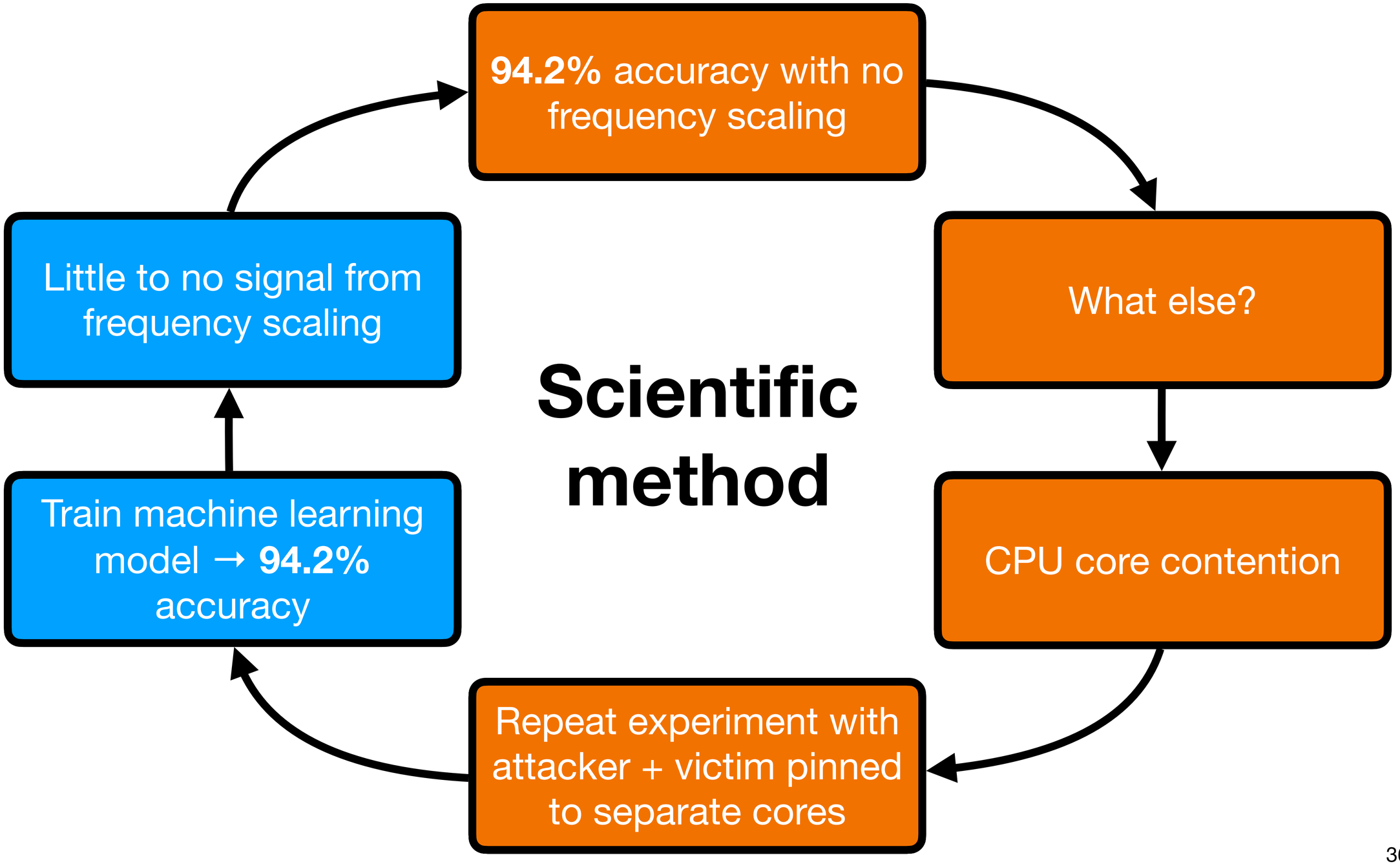

95.2% accuracy when memory accesses removed
What could cause throughput to change over time?
Frequency scaling
Repeat experiment with frequency scaling disabled
Train machine learning model → 94.2% accuracy
Conclusion
Scientific method

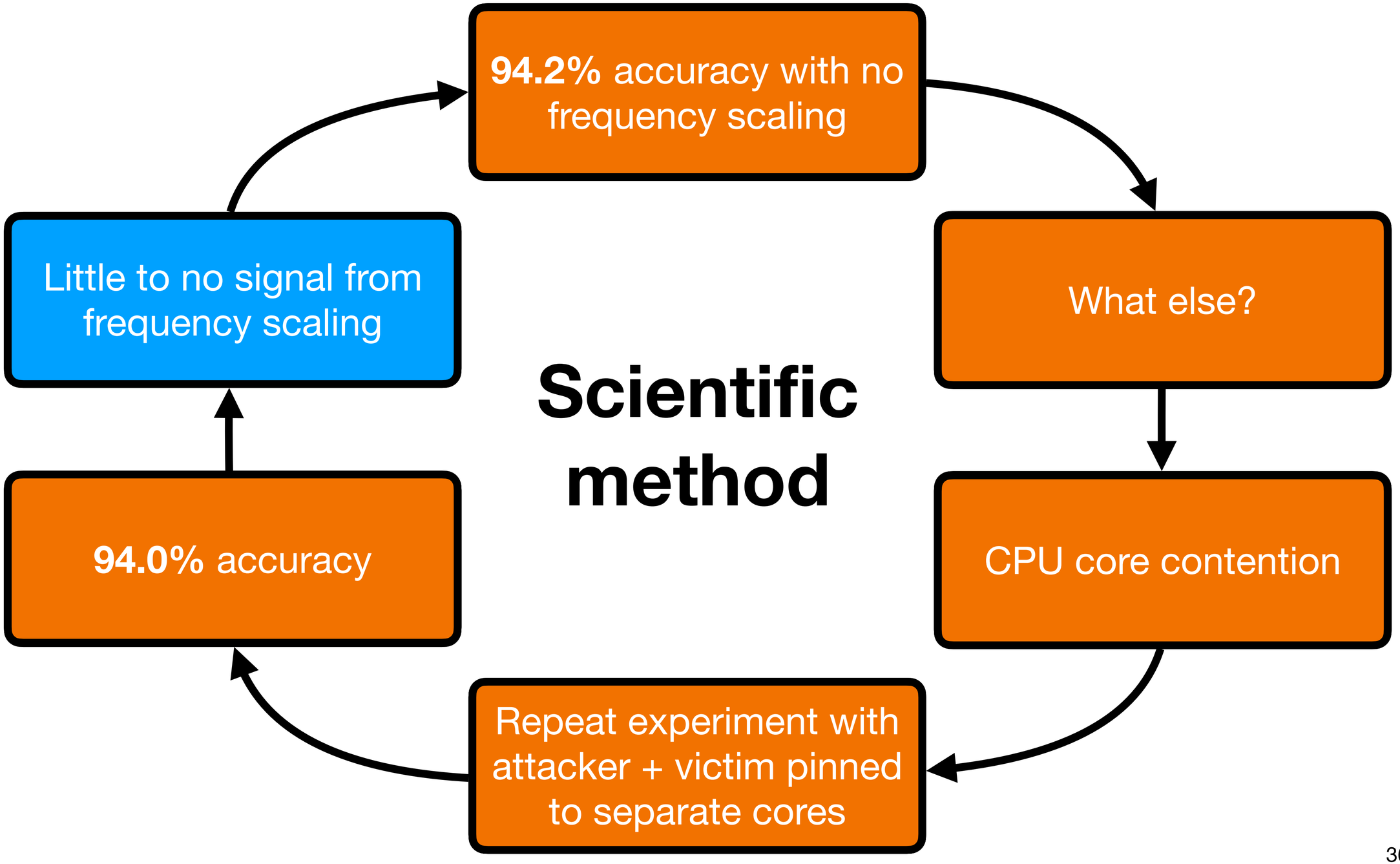

95.2% accuracy when memory accesses removed
What could cause throughput to change over time?
Frequency scaling
Repeat experiment with frequency scaling disabled
Train machine learning model → 94.2% accuracy
Little to no signal from frequency scaling
Scientific method

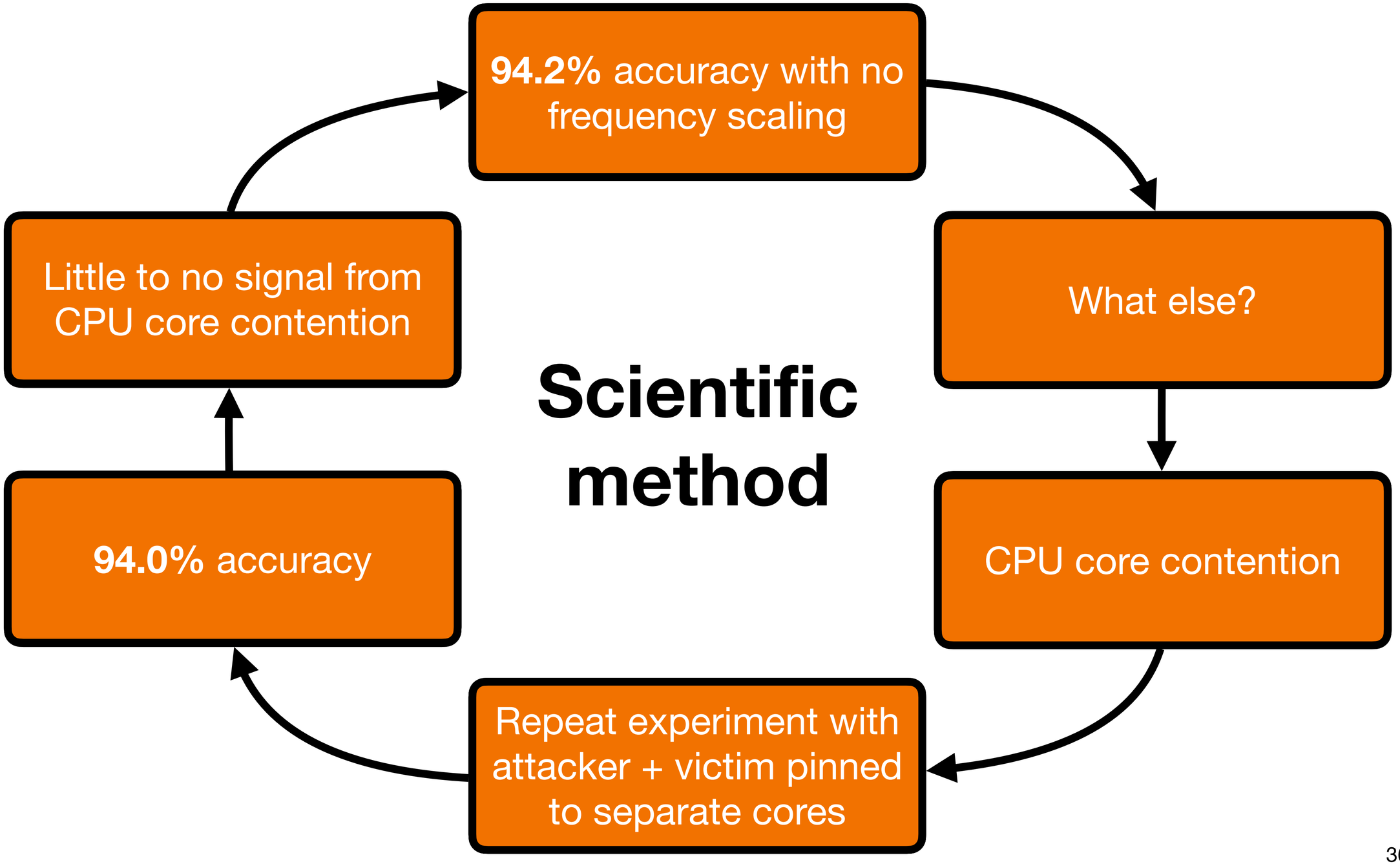

94.2% accuracy with no frequency scaling
What could cause throughput to change over time?
Frequency scaling
Repeat experiment with frequency scaling disabled
Train machine learning model → 94.2% accuracy
Little to no signal from frequency scaling
Scientific method

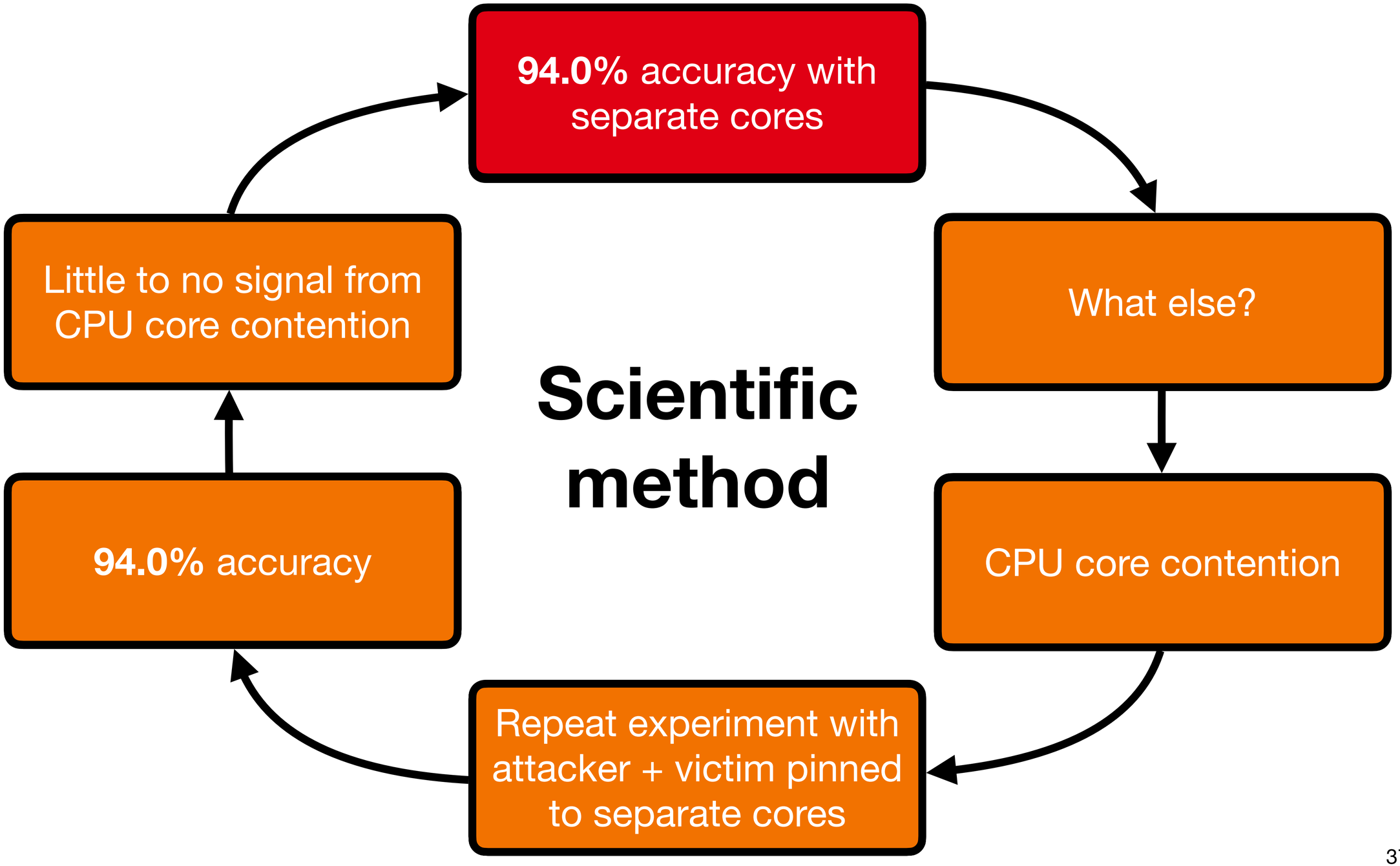

94.2% accuracy with no frequency scaling
What else?
Frequency scaling
Repeat experiment with frequency scaling disabled
Train machine learning model → 94.2% accuracy
Little to no signal from frequency scaling
Scientific method

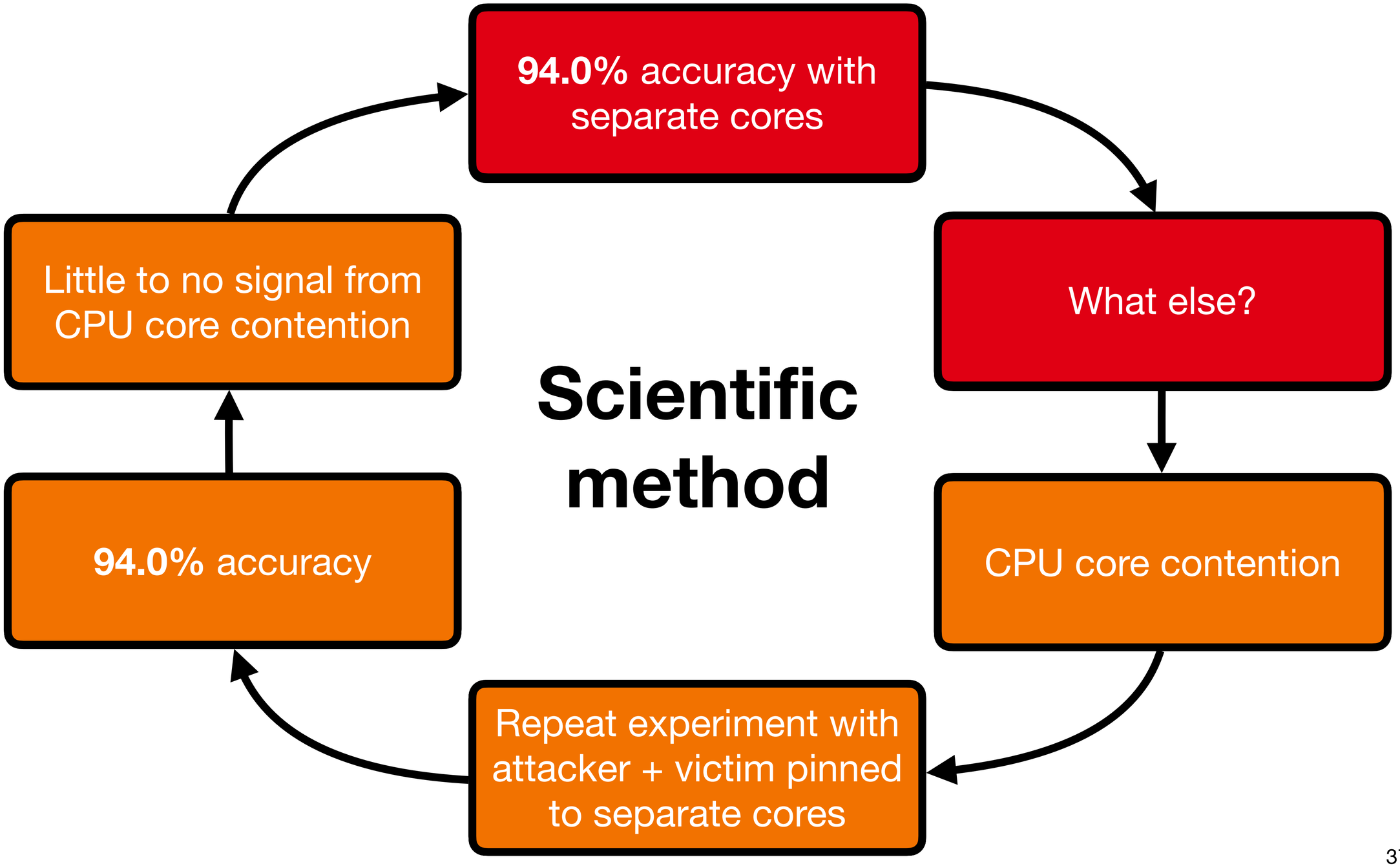

94.2% accuracy with no frequency scaling
What else?
CPU core contention
Repeat experiment with frequency scaling disabled
Train machine learning model → 94.2% accuracy
Little to no signal from frequency scaling
Scientific method

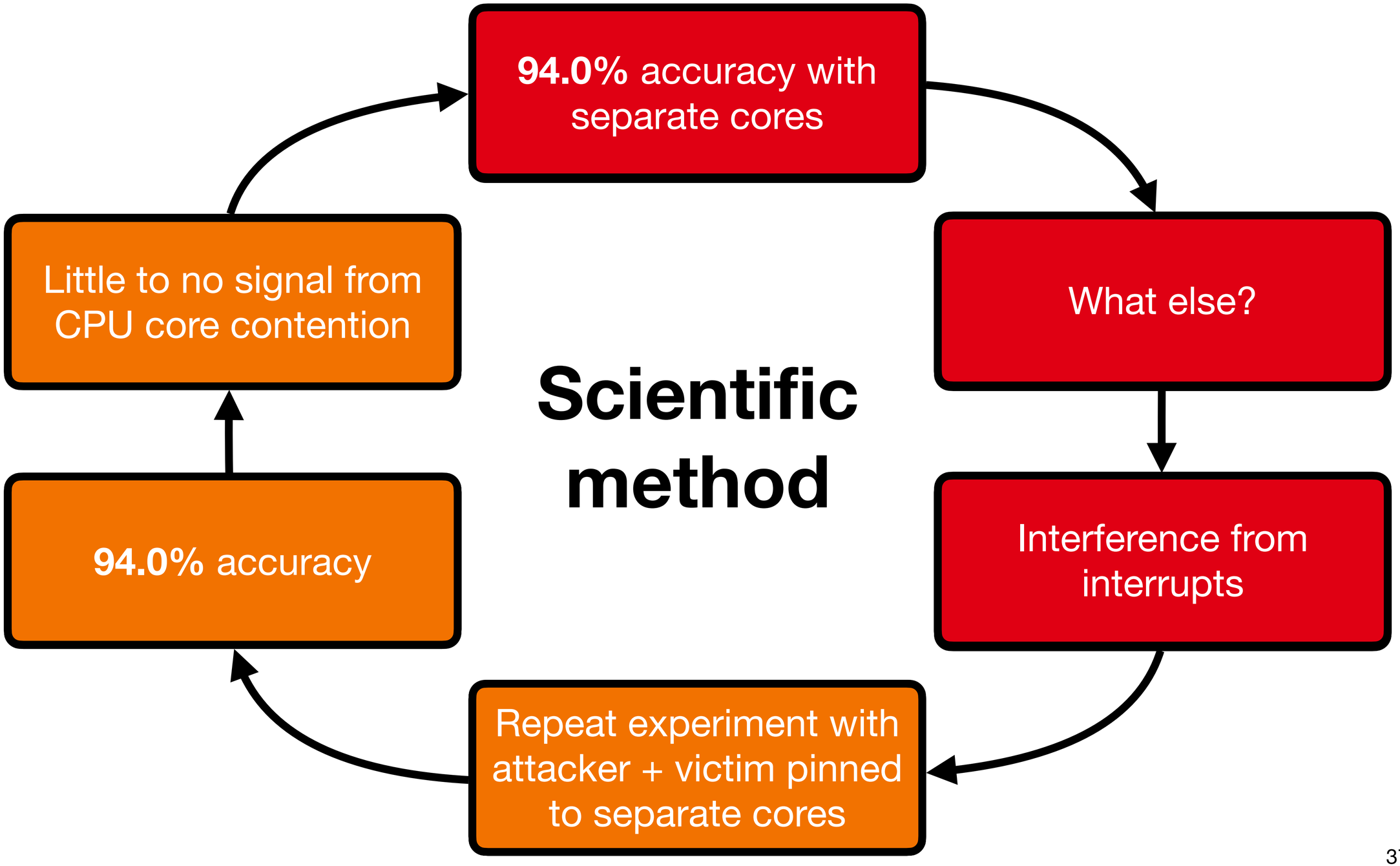

94.2% accuracy with no frequency scaling
What else?
CPU core contention
Repeat experiment with attacker + victim pinned to separate cores
Train machine learning model → 94.2% accuracy
Little to no signal from frequency scaling
Scientific method


94.2% accuracy with no frequency scaling
What else?
CPU core contention
Repeat experiment with attacker + victim pinned to separate cores
94.0% accuracy
Little to no signal from frequency scaling
Scientific method


94.2% accuracy with no frequency scaling
What else?
CPU core contention
Repeat experiment with attacker + victim pinned to separate cores
94.0% accuracy
Little to no signal from CPU core contention
Scientific method

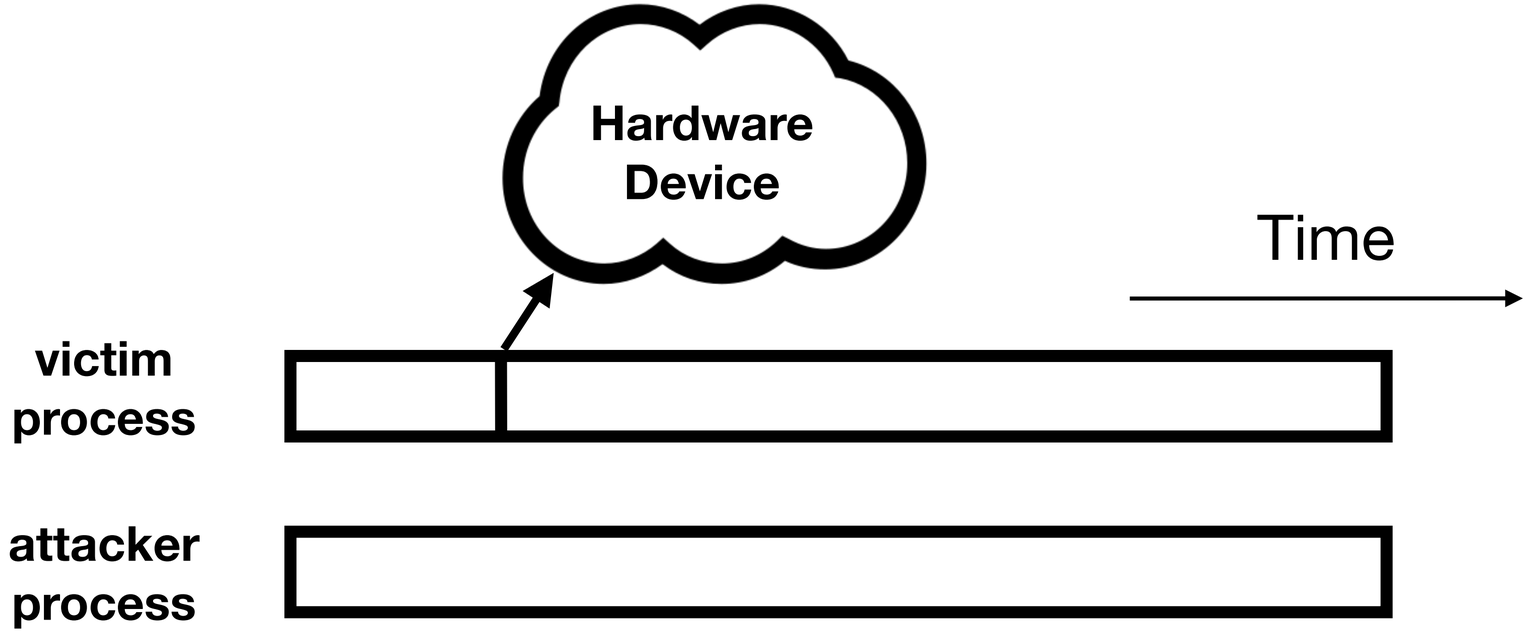

94.0% accuracy with separate cores
What else?
CPU core contention
Repeat experiment with attacker + victim pinned to separate cores
94.0% accuracy
Little to no signal from CPU core contention
Scientific method


94.0% accuracy with separate cores
What else?
CPU core contention
Repeat experiment with attacker + victim pinned to separate cores
94.0% accuracy
Little to no signal from CPU core contention
Scientific method


94.0% accuracy with separate cores
What else?
Interference from interrupts
Repeat experiment with attacker + victim pinned to separate cores
94.0% accuracy
Little to no signal from CPU core contention
Scientific method

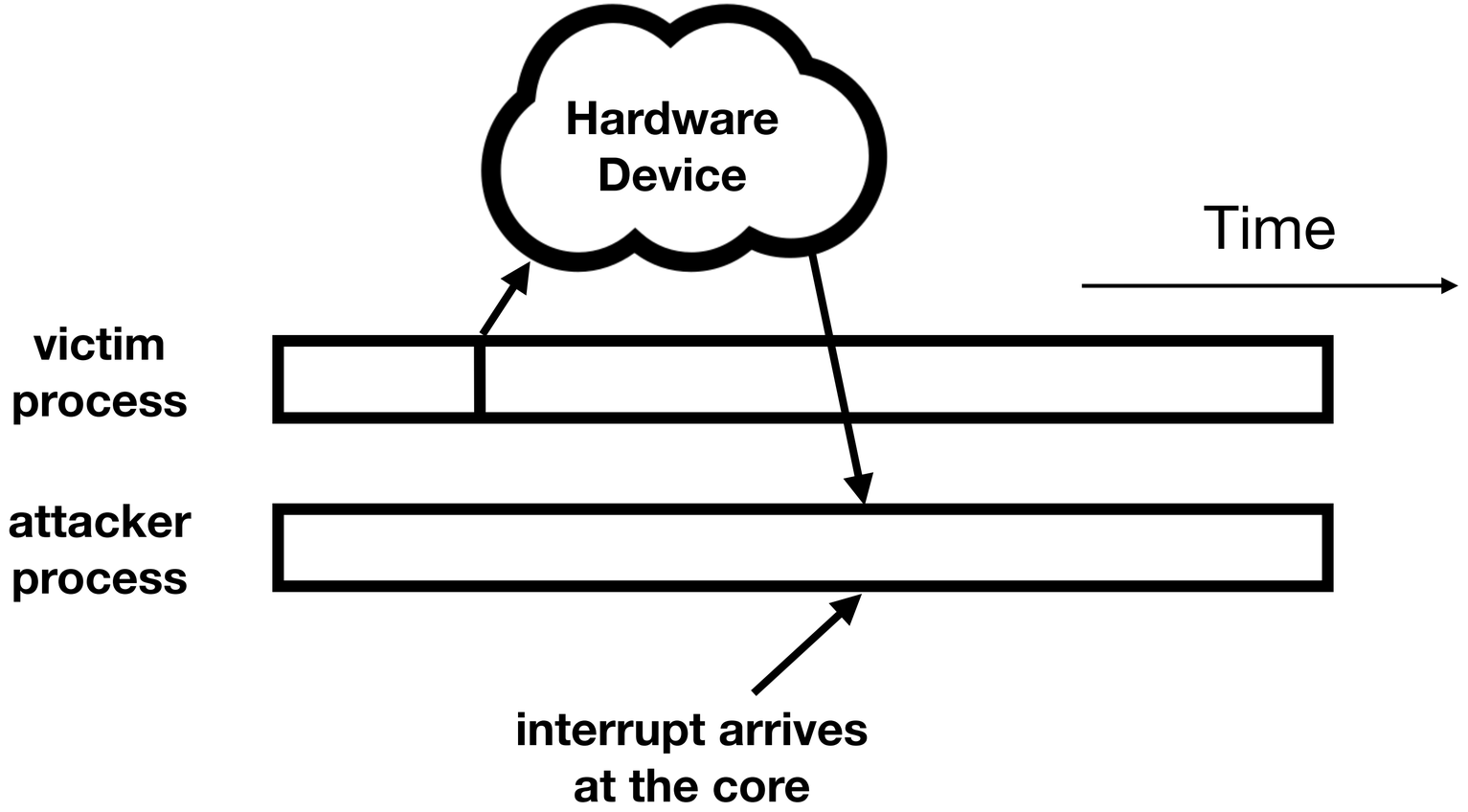
System Interrupts

- Used to deal with asynchronous events
  - e.g. Graphics interrupts render content on a display



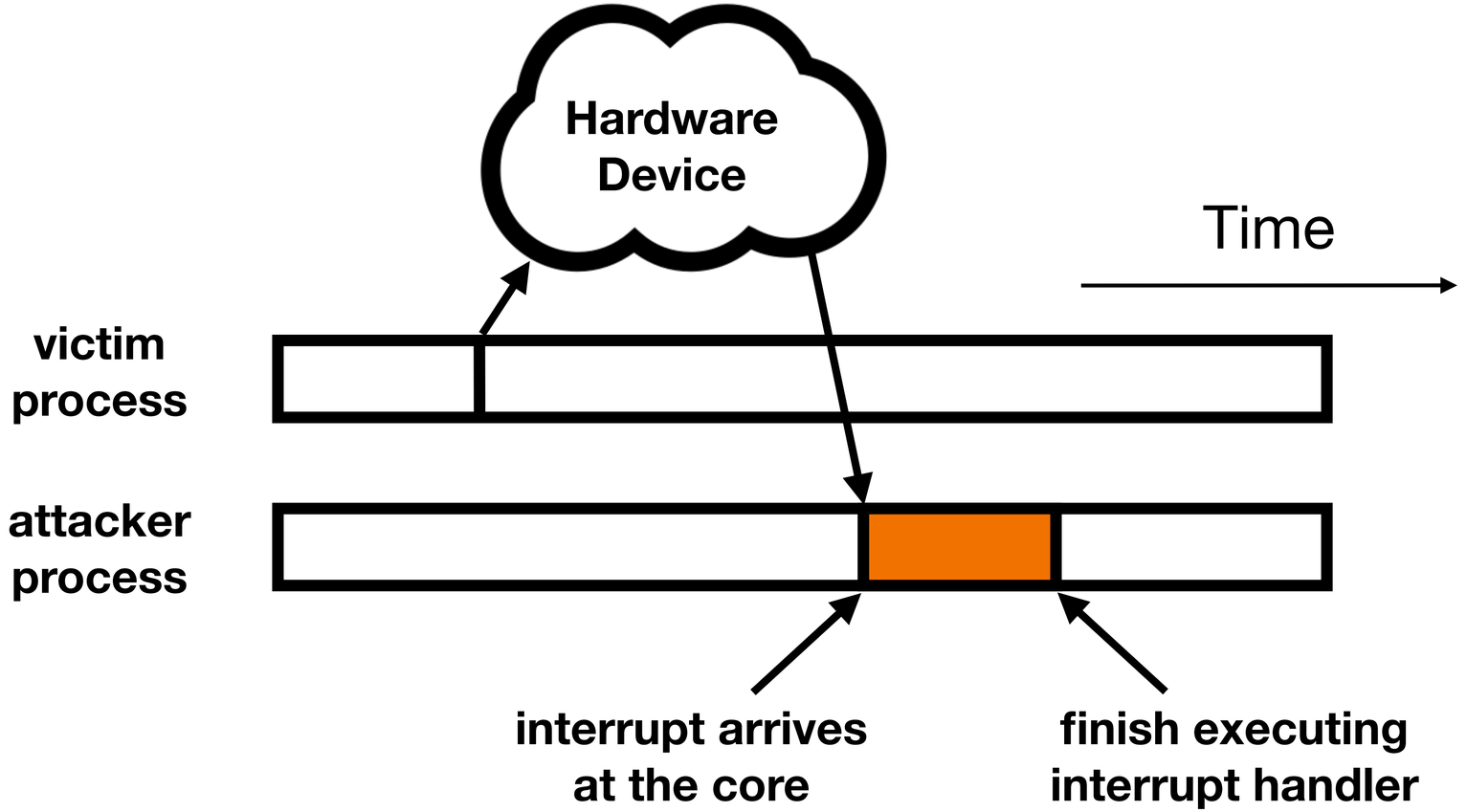
System Interrupts

- Used to deal with asynchronous events
  - e.g. Graphics interrupts render content on a display



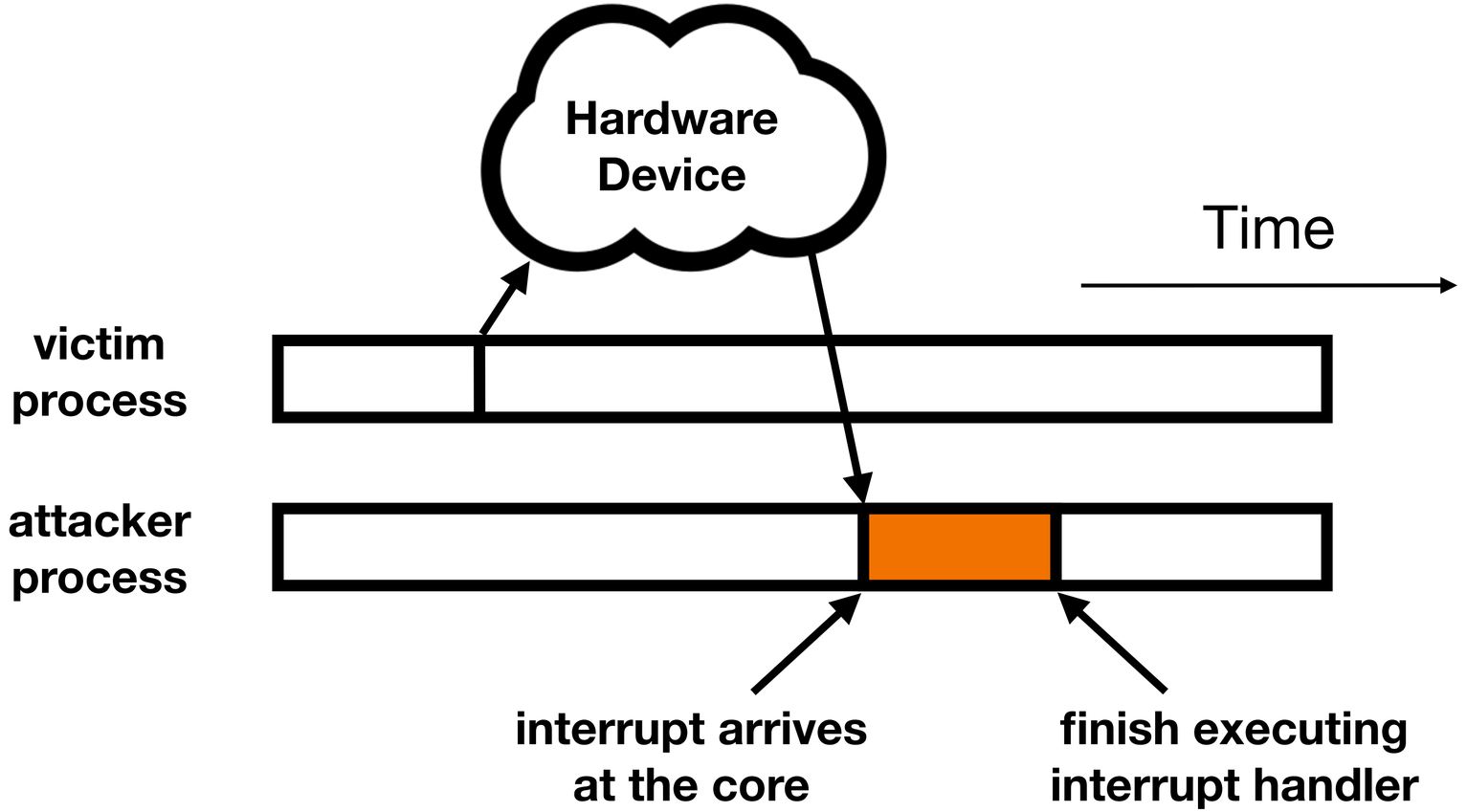
System Interrupts

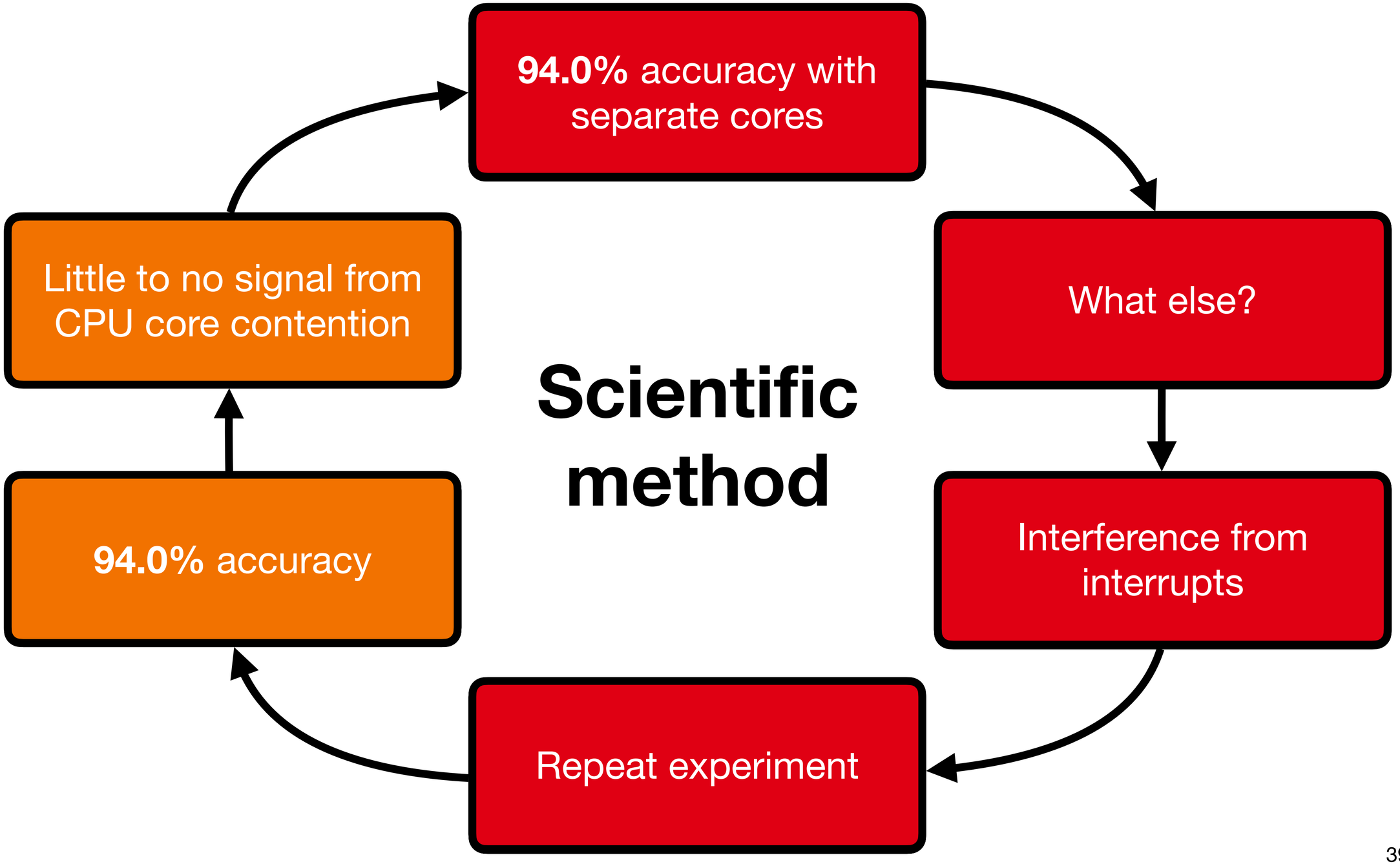
- Used to deal with asynchronous events
  - e.g. Graphics interrupts render content on a display

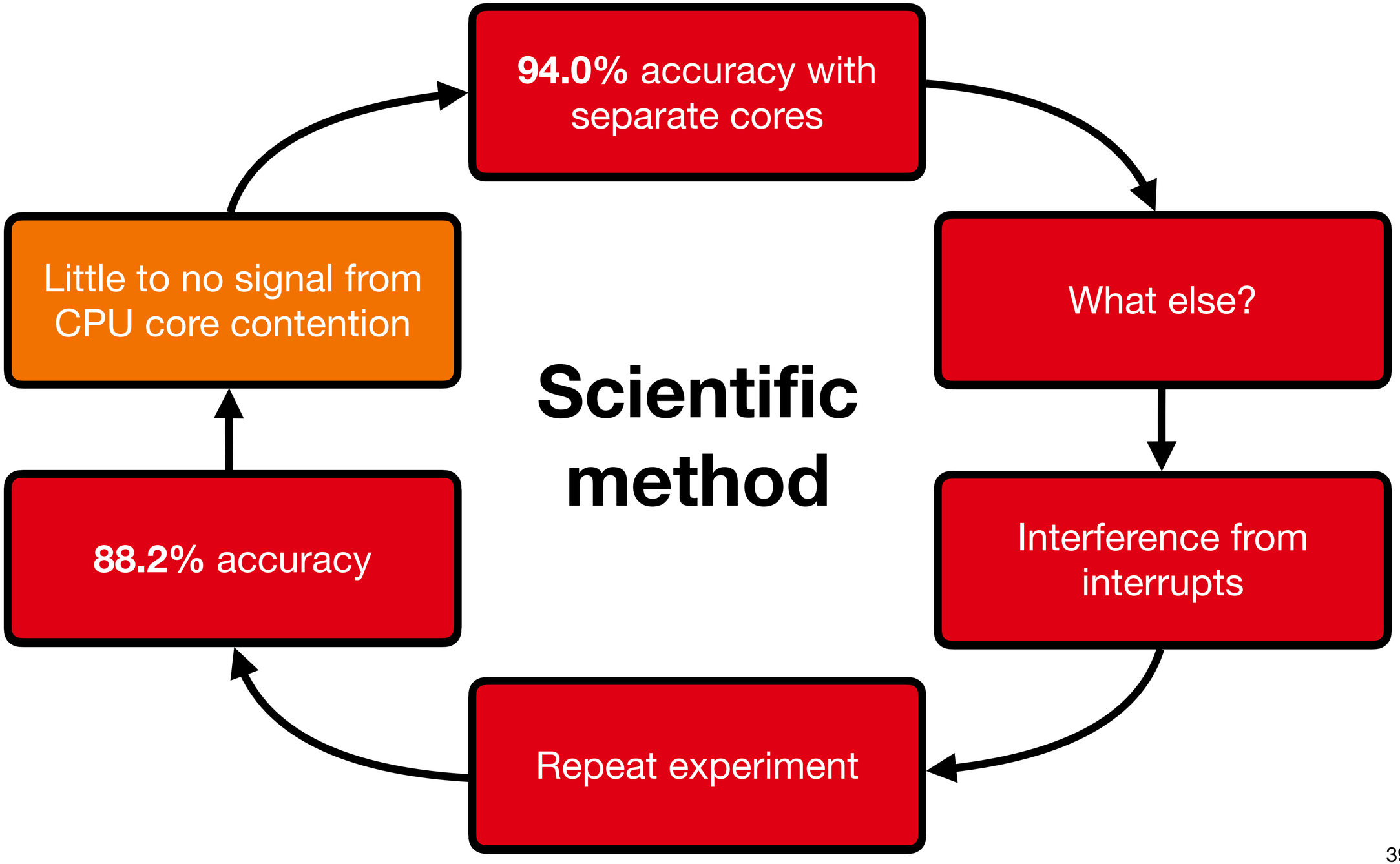


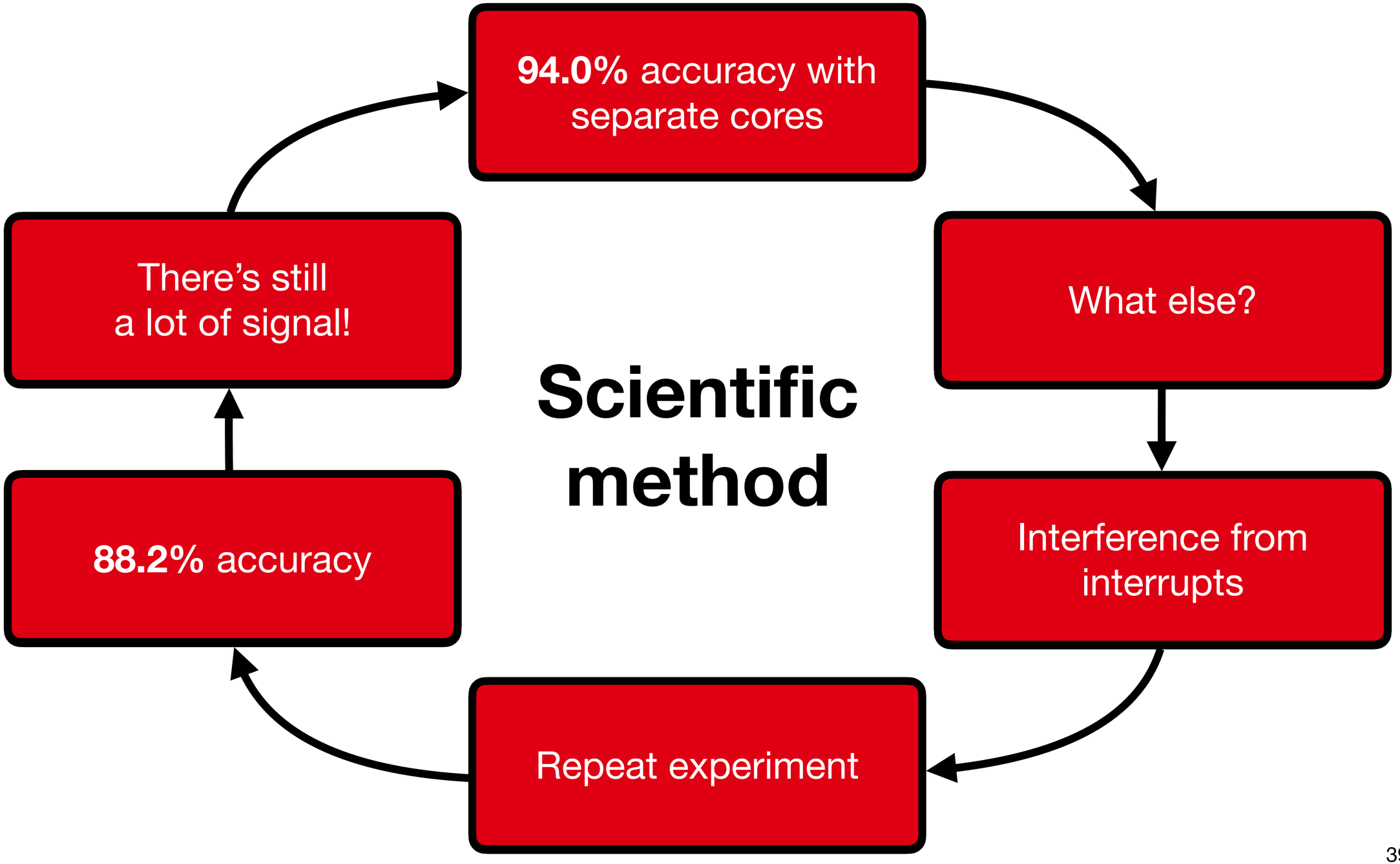
System Interrupts

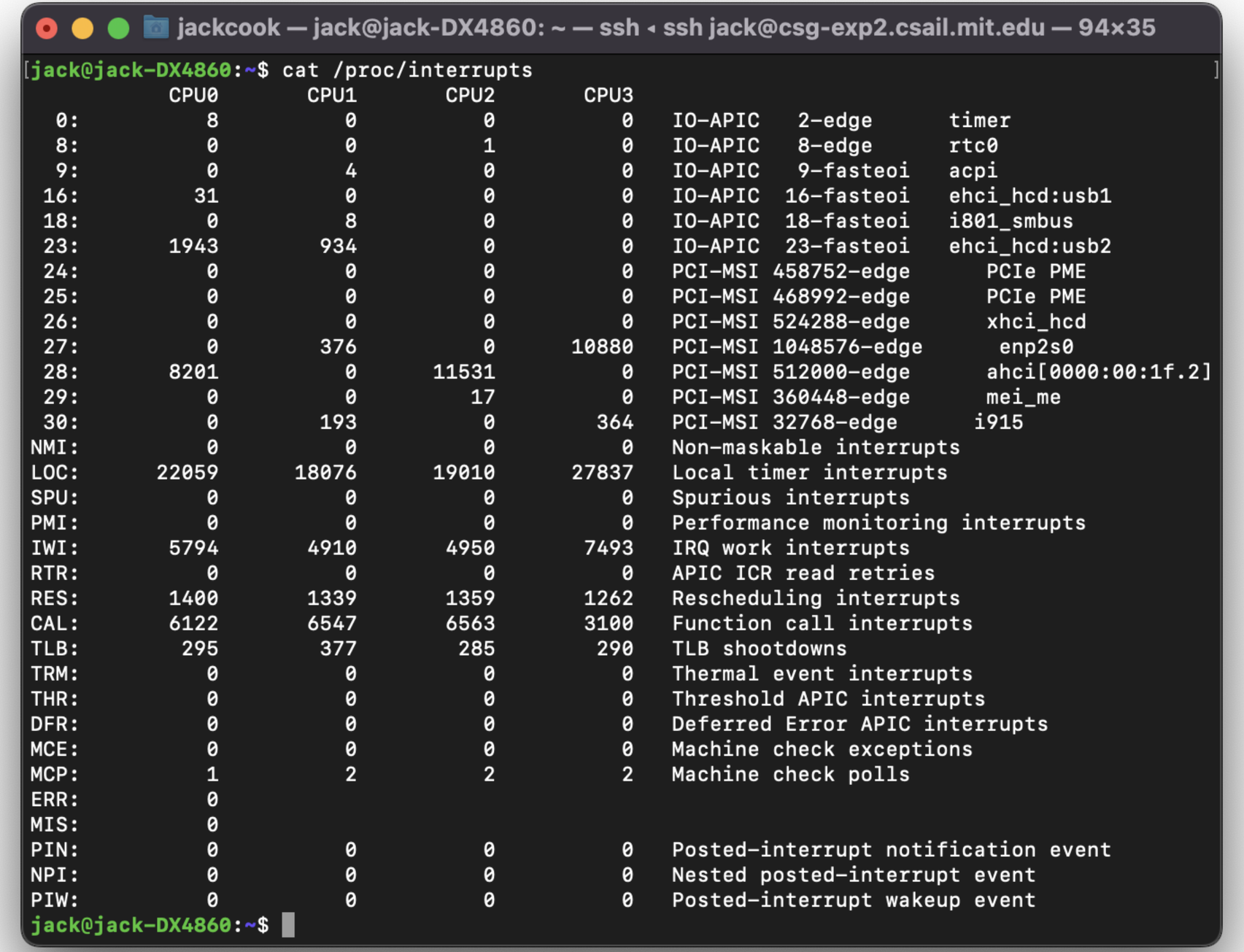
- Used to deal with asynchronous events
  - e.g. Graphics interrupts render content on a display
- Some can be “pinned” to specific cores, some can’t




94.0% accuracy with separate cores
What else?
Interference from interrupts
Repeat experiment
94.0% accuracy
Little to no signal from CPU core contention
Scientific method


94.0% accuracy with separate cores
What else?
Interference from interrupts
Repeat experiment
88.2% accuracy
Little to no signal from CPU core contention
Scientific method


94.0% accuracy with separate cores
What else?
Interference from interrupts
Repeat experiment
88.2% accuracy
There's still a lot of signal!
Scientific method


jackcook — jack@jack-DX4860: ~ — ssh ◂ ssh jack@csg-exp2.csail.mit.edu — 94×35
jack@jack-DX4860:~$ cat /proc/interrupts
CPU0 CPU1 CPU2 CPU3
0: 8 0 0 0 IO-APIC 2-edge timer
8: 0 0 1 0 IO-APIC 8-edge rtc0
9: 0 4 0 0 IO-APIC 9-fasteoi acpi
16: 31 0 0 0 IO-APIC 16-fasteoi ehci_hcd:usb1
18: 0 8 0 0 IO-APIC 18-fasteoi i801_smbus
23: 1943 934 0 0 IO-APIC 23-fasteoi ehci_hcd:usb2
24: 0 0 0 0 PCI-MSI 458752-edge PCIe PME
25: 0 0 0 0 PCI-MSI 468992-edge PCIe PME
26: 0 0 0 0 PCI-MSI 524288-edge xhci_hcd
27: 0 376 0 10880 PCI-MSI 1048576-edge enp2s0
28: 8201 0 11531 0 PCI-MSI 512000-edge ahci[0000:00:1f.2]
29: 0 0 17 0 PCI-MSI 360448-edge mei_me
30: 0 193 0 364 PCI-MSI 32768-edge i915
NMI: 0 0 0 0 Non-maskable interrupts
LOC: 22059 18076 19010 27837 Local timer interrupts
SPU: 0 0 0 0 Spurious interrupts
PMI: 0 0 0 0 Performance monitoring interrupts
IWI: 5794 4910 4950 7493 IRQ work interrupts
RTR: 0 0 0 0 APIC ICR read retries
RES: 1400 1339 1359 1262 Rescheduling interrupts
CAL: 6122 6547 6563 3100 Function call interrupts
TLB: 295 377 285 290 TLB shootdowns
TRM: 0 0 0 0 Thermal event interrupts
THR: 0 0 0 0 Threshold APIC interrupts
DFR: 0 0 0 0 Deferred Error APIC interrupts
MCE: 0 0 0 0 Machine check exceptions
MCP: 1 2 2 2 Machine check polls
ERR: 0
MIS: 0
PIN: 0 0 0 0 Posted-interrupt notification event
NPI: 0 0 0 0 Nested posted-interrupt event
PIW: 0 0 0 0 Posted-interrupt wakeup event
jack@jack-DX4860:~$


jackcook — jack@jack-DX4860: ~ — ssh ◂ ssh jack@csg-exp2.csail.mit.edu — 94×35
jack@jack-DX4860:~$ cat /proc/interrupts
CPU0 CPU1 CPU2 CPU3
0: 8 0 0 0 IO-APIC 2-edge timer
8: 0 0 1 0 IO-APIC 8-edge rtc0
9: 0 4 0 0 IO-APIC 9-fasteoi acpi
16: 31 0 0 0 IO-APIC 16-fasteoi ehci_hcd:usb1
18: 0 8 0 0 IO-APIC 18-fasteoi i801_smbus
23: 1943 934 0 0 IO-APIC 23-fasteoi ehci_hcd:usb2
24: 0 0 0 0 PCI-MSI 458752-edge PCIe PME
25: 0 0 0 0 PCI-MSI 468992-edge PCIe PME
26: 0 0 0 0 PCI-MSI 524288-edge xhci_hcd
27: 0 376 0 10880 PCI-MSI 1048576-edge enp2s0
28: 8201 0 11531 0 PCI-MSI 512000-edge ahci[0000:00:1f.2]
29: 0 0 17 0 PCI-MSI 360448-edge mei_me
30: 0 193 0 364 PCI-MSI 32768-edge i915
LOC: 22059 18076 19010 27837 Local timer interrupts
SPU: 0 0 0 0 Spurious interrupts
IWI: 5794
RTR: 0
RES: 1400
CAL: 6122
TLB: 295
TRM: 0 0 0 0 Thermal event interrupts
THR: 0 0 0 0 Threshold APIC interrupts
DFR: 0 0 0 0 Deferred Error APIC interrupts
MCE: 0 0 0 0 Machine check exceptions
MCP: 1 2 2 2 Machine check polls
ERR: 0
MIS: 0
PIN: 0 0 0 0 Posted-interrupt notification event
NPI: 0 0 0 0 Nested posted-interrupt event
PIW: 0 0 0 0 Posted-interrupt wakeup event
jack@jack-DX4860:~$
Movable interrupts

⬅ 16: Mouse

Movable interrupts

⬅ 23: Keyboard

Movable interrupts

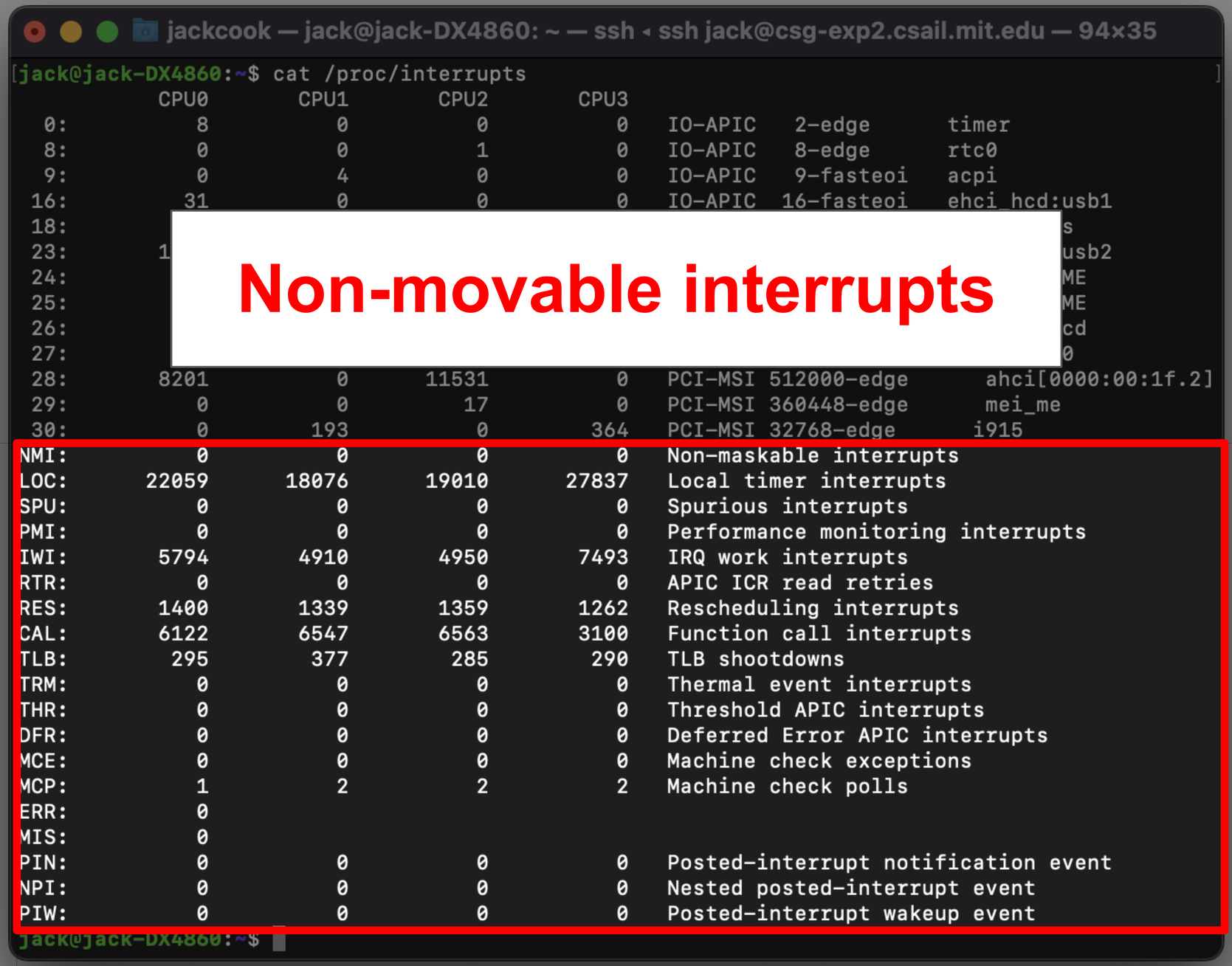
⬅ 27: Network card

Movable interrupts

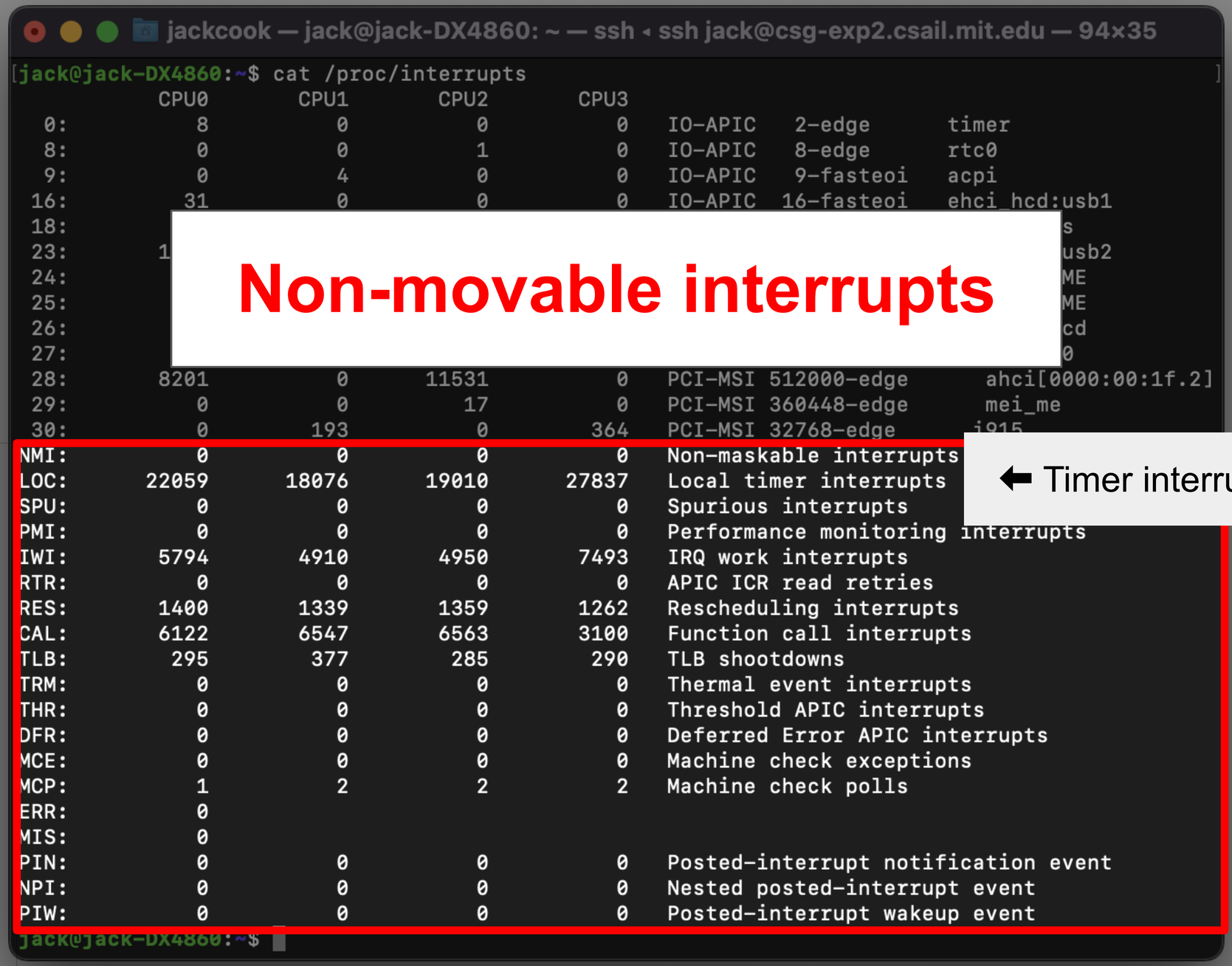
Movable interrupts

⬅ 30: Graphics card


jackcook — jack@jack-DX4860: ~ — ssh ◂ ssh jack@csg-exp2.csail.mit.edu — 94×35
jack@jack-DX4860:~$ cat /proc/interrupts
CPU0 CPU1 CPU2 CPU3
0: 8 0 0 0 IO-APIC 2-edge timer
8: 0 0 1 0 IO-APIC 8-edge rtc0
9: 0 4 0 0 IO-APIC 9-fasteoi acpi
16: 31 0 0 0 IO-APIC 16-fasteoi ehci_hcd:usb1
18: 0 8 0 0 IO-APIC 18-fasteoi i801_smbus
23: 1943 934 0 0 IO-APIC 23-fasteoi ehci_hcd:usb2
24: 0 0 0 0 PCI-MSI 458752-edge PCIe PME
25: 0 0 0 0 PCI-MSI 468992-edge PCIe PME
26: 0 0 0 0 PCI-MSI 524288-edge xhci_hcd
27: 0 376 0 10880 PCI-MSI 1048576-edge enp2s0
28: 8201 0 11531 0 PCI-MSI 512000-edge ahci[0000:00:1f.2]
29: 0 0 17 0 PCI-MSI 360448-edge mei_me
30: 0 193 0 364 PCI-MSI 32768-edge i915
LOC: 22059 18076 19010 27837 Local timer interrupts
SPU: 0 0 0 0 Spurious interrupts
IWI: 5794
RTR: 0
RES: 1400
CAL: 6122
TLB: 295
TRM: 0 0 0 0 Thermal event interrupts
THR: 0 0 0 0 Threshold APIC interrupts
DFR: 0 0 0 0 Deferred Error APIC interrupts
MCE: 0 0 0 0 Machine check exceptions
MCP: 1 2 2 2 Machine check polls
ERR: 0
MIS: 0
PIN: 0 0 0 0 Posted-interrupt notification event
NPI: 0 0 0 0 Nested posted-interrupt event
PIW: 0 0 0 0 Posted-interrupt wakeup event
jack@jack-DX4860:~$
Movable interrupts

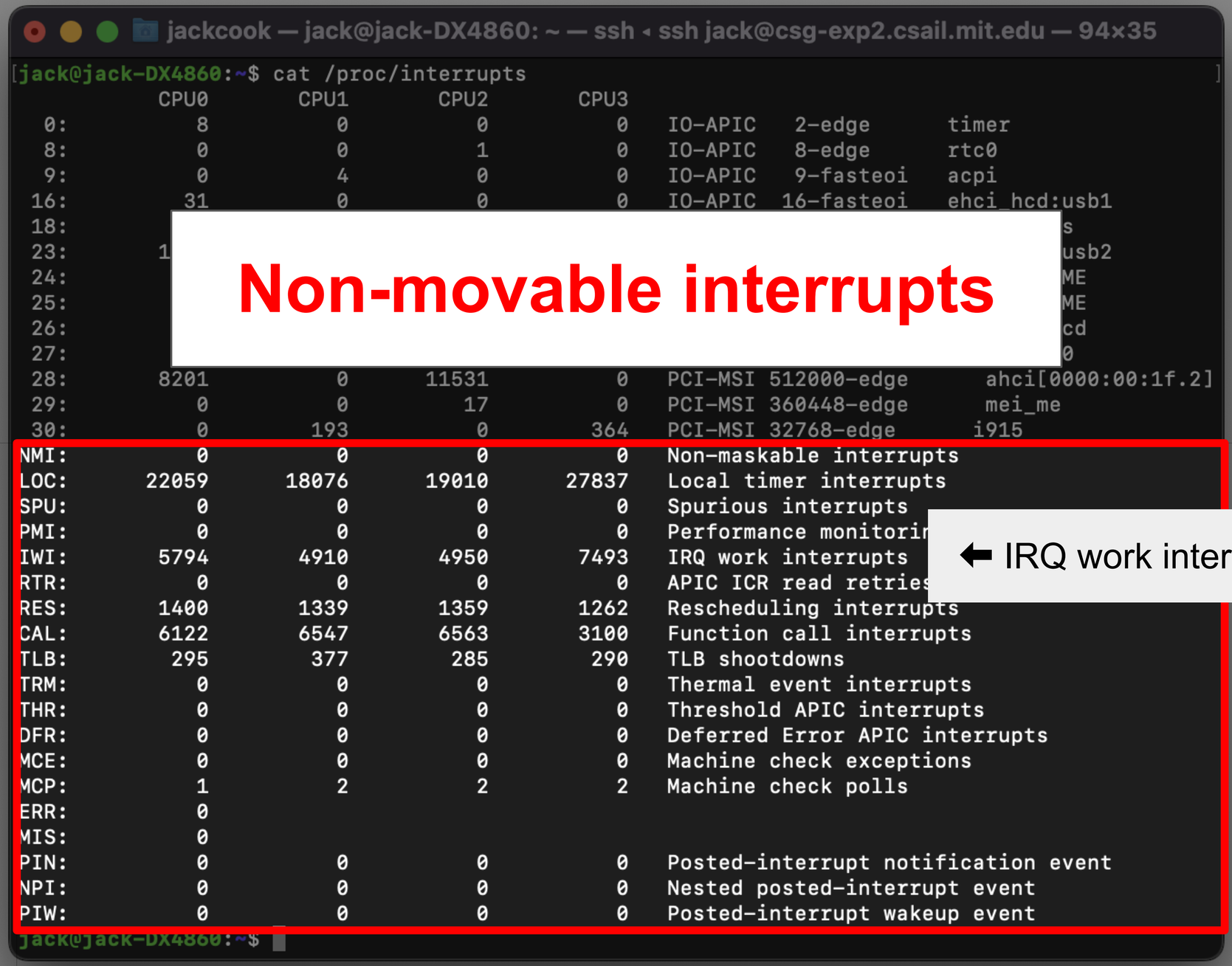


Non-movable interrupts



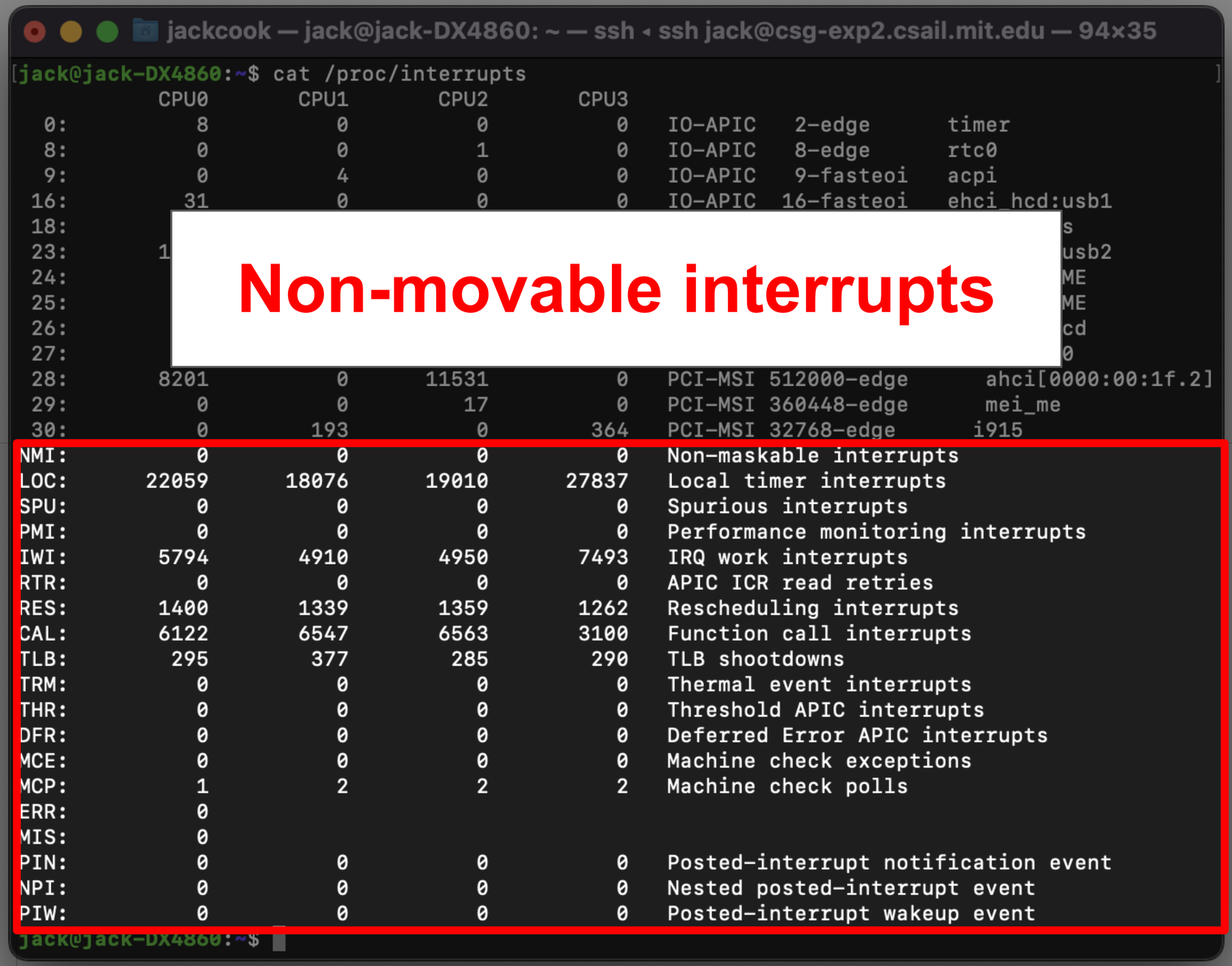
Non-movable interrupts

⬅ Timer interrupts



Non-movable interrupts

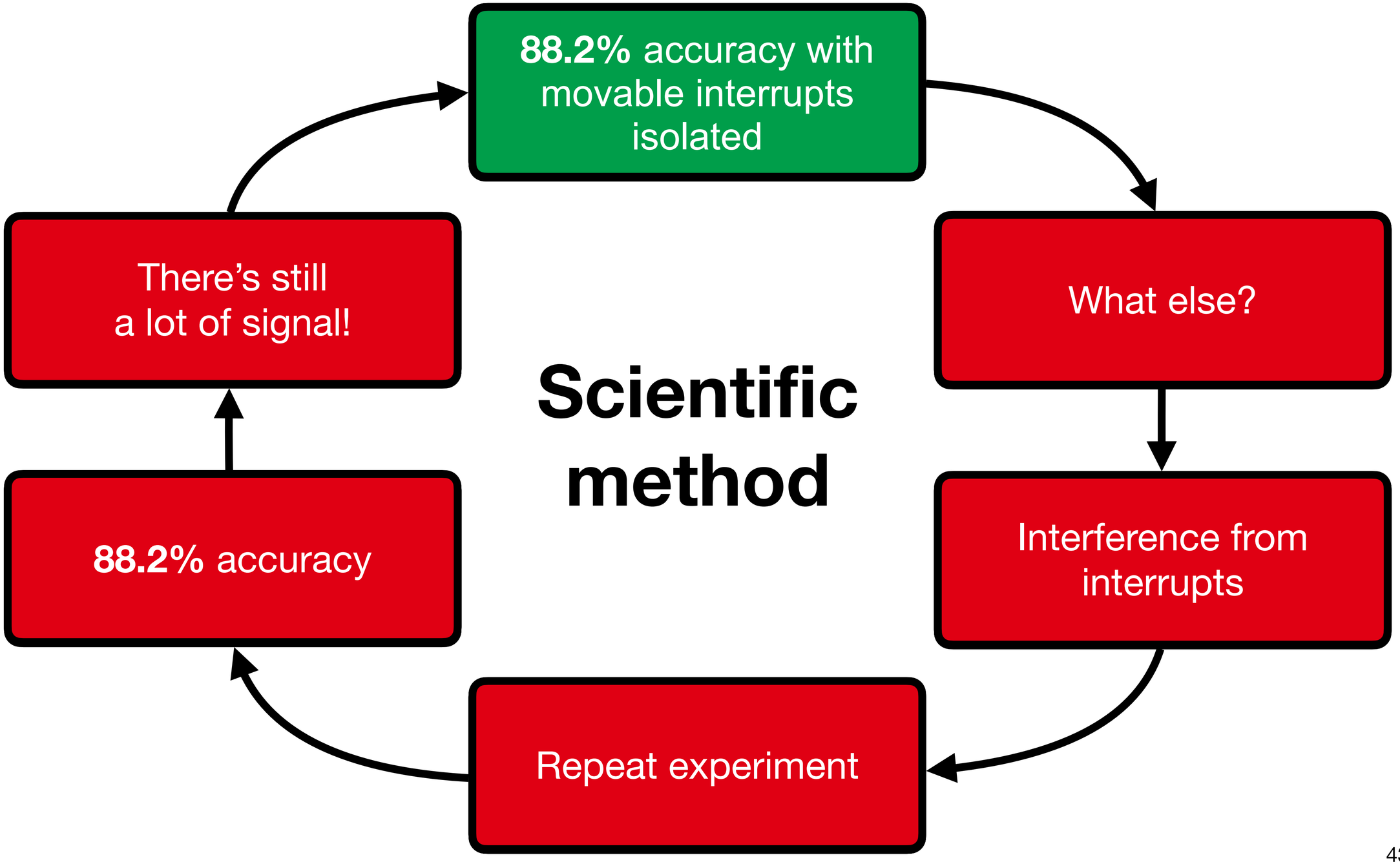
⬅ IRQ work interrupts

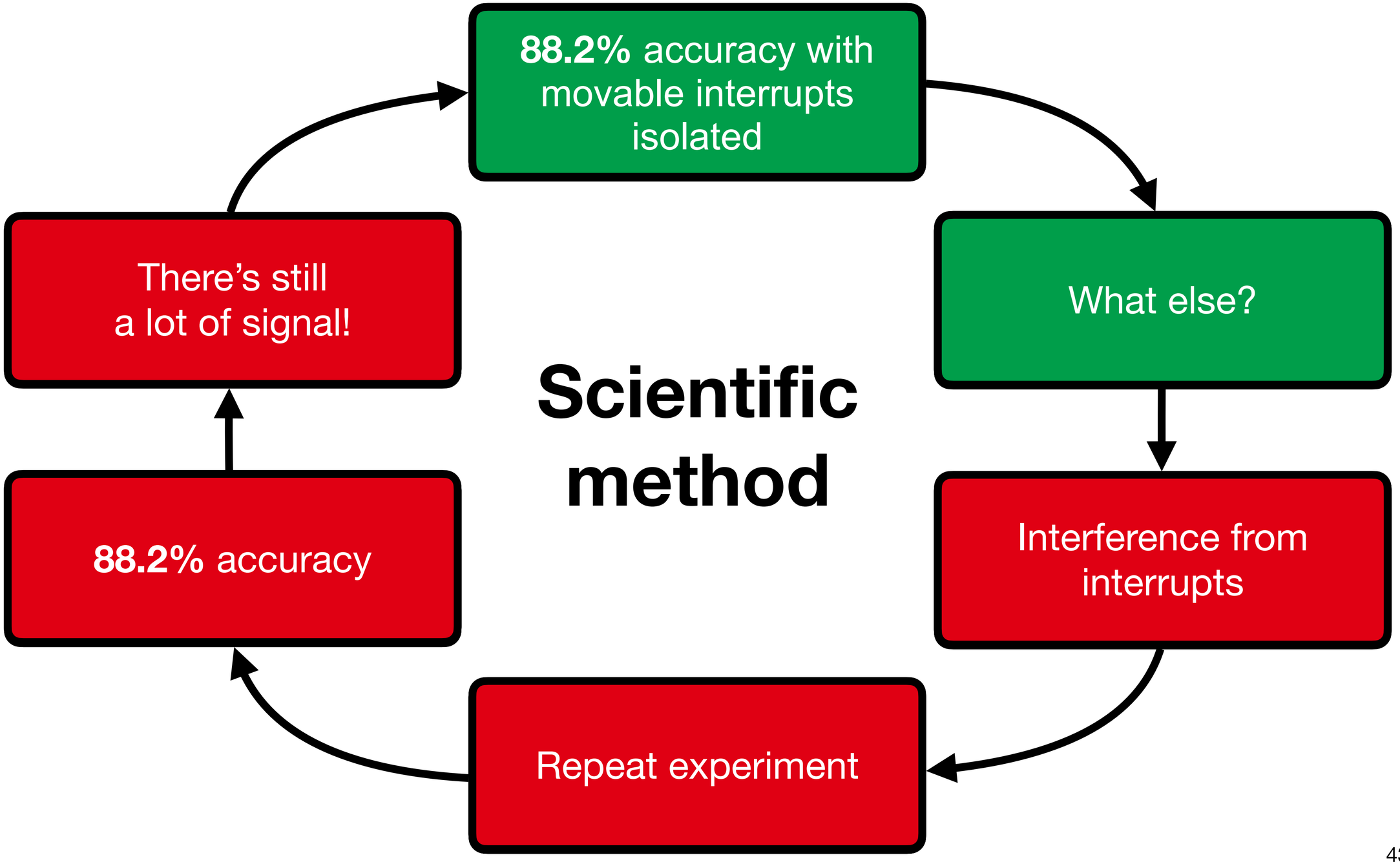


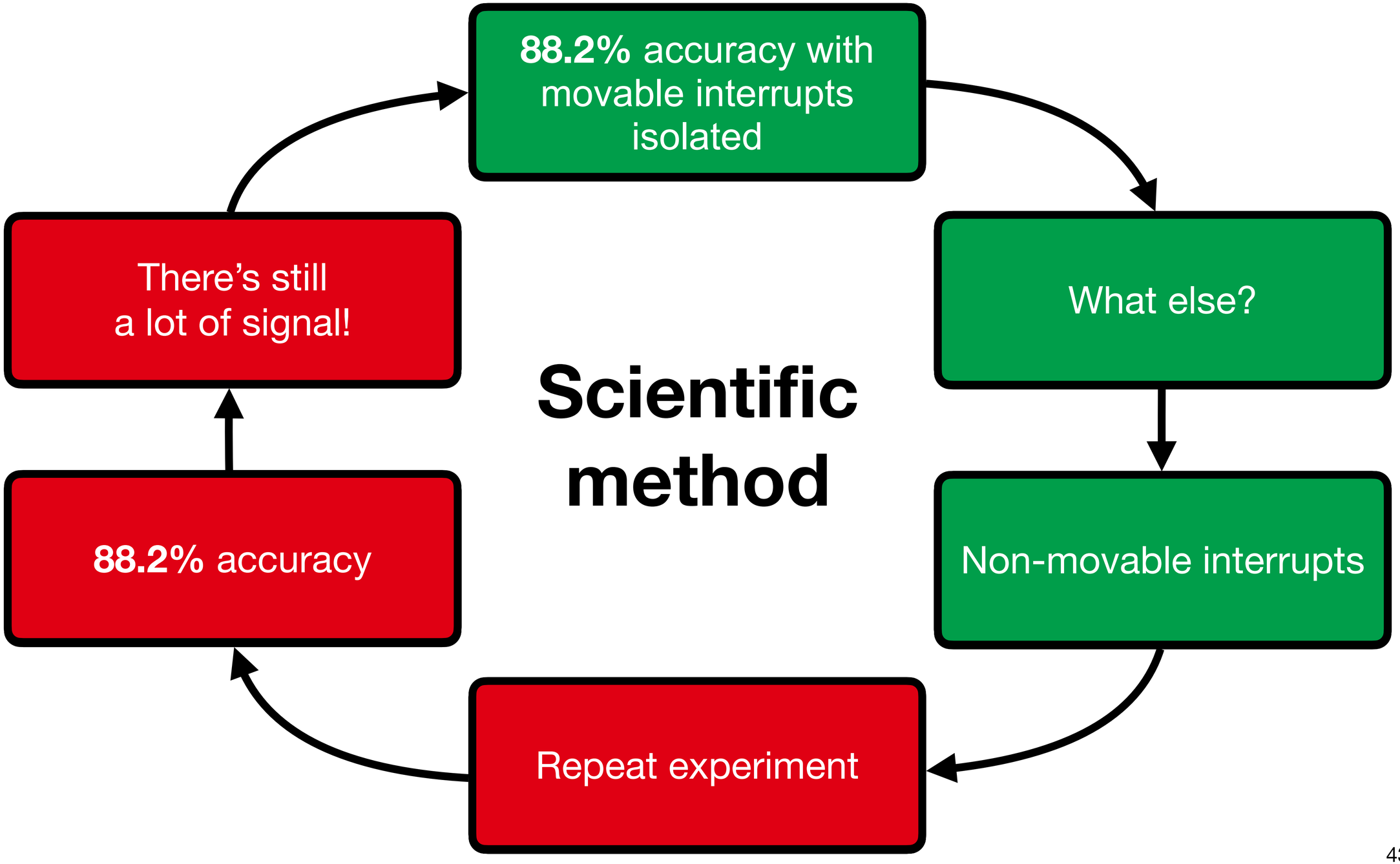
Non-movable interrupts

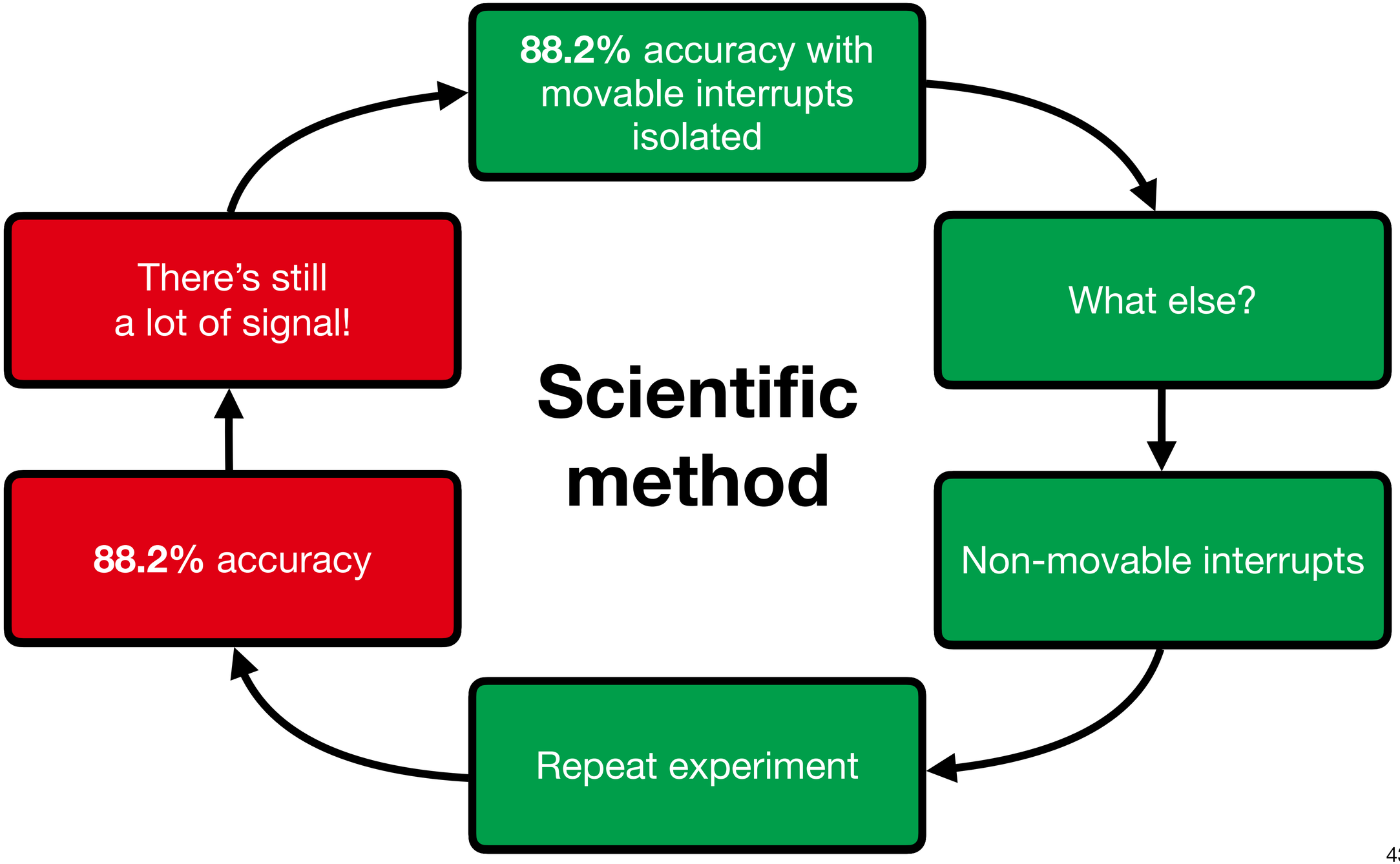
Non-Movable Interrupts

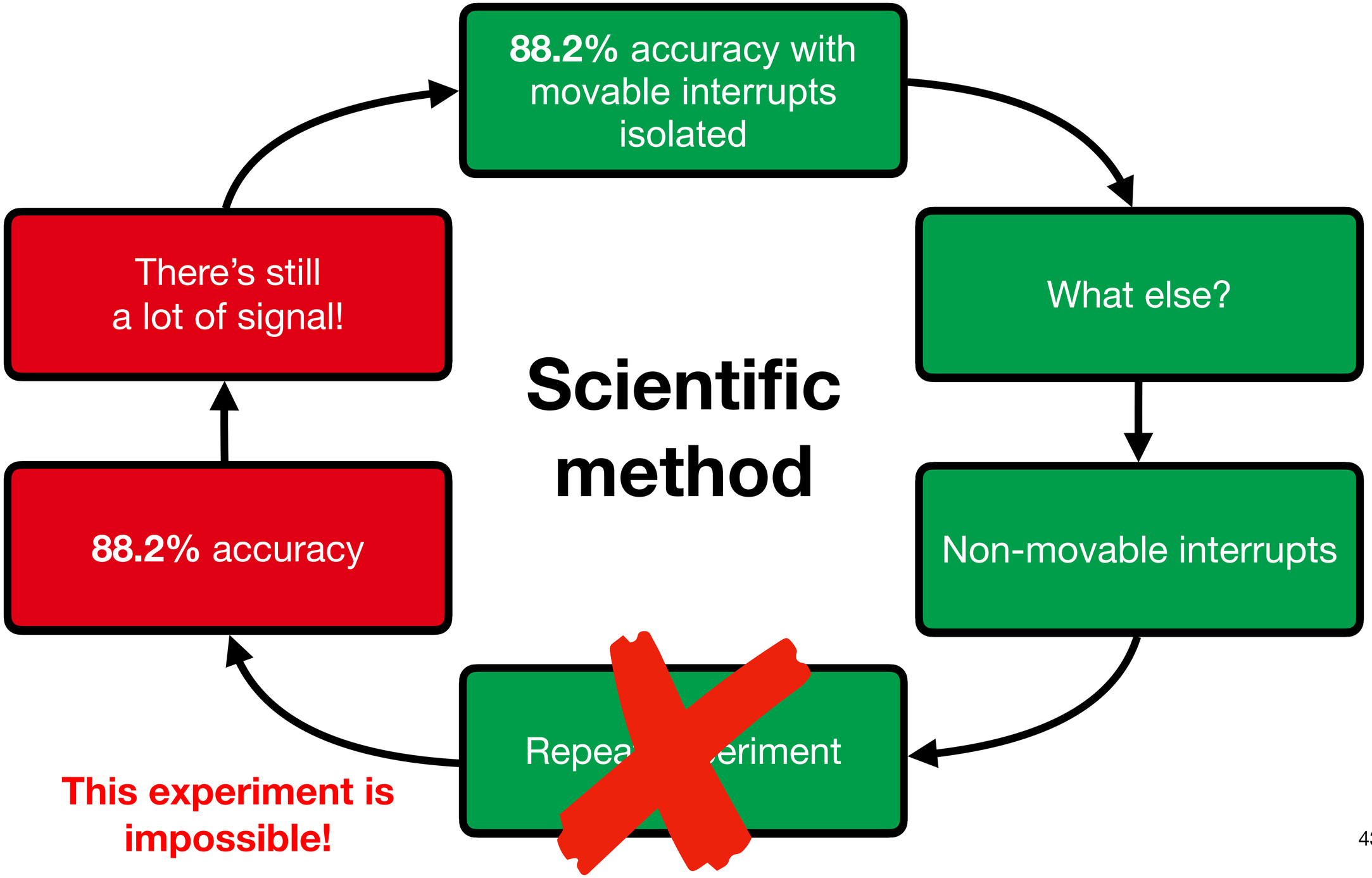
- Can’t be isolated from any cores
- Are necessary for the operating system to function
- Have not been studied in detail for side channels


88.2% accuracy with movable interrupts isolated
What else?
Interference from interrupts
Repeat experiment
88.2% accuracy
There's still a lot of signal!
Scientific method


88.2% accuracy with movable interrupts isolated
What else?
Interference from interrupts
Repeat experiment
88.2% accuracy
There's still a lot of signal!
Scientific method


88.2% accuracy with movable interrupts isolated
What else?
Non-movable interrupts
Repeat experiment
88.2% accuracy
There's still a lot of signal!
Scientific method


88.2% accuracy with movable interrupts isolated
What else?
Non-movable interrupts
Repeat experiment
88.2% accuracy
There's still a lot of signal!
Scientific method


88.2% accuracy with movable interrupts isolated
What else?
Non-movable interrupts
Repea eriment
88.2% accuracy
There's still a lot of signal!
Scientific method
This experiment is impossible!

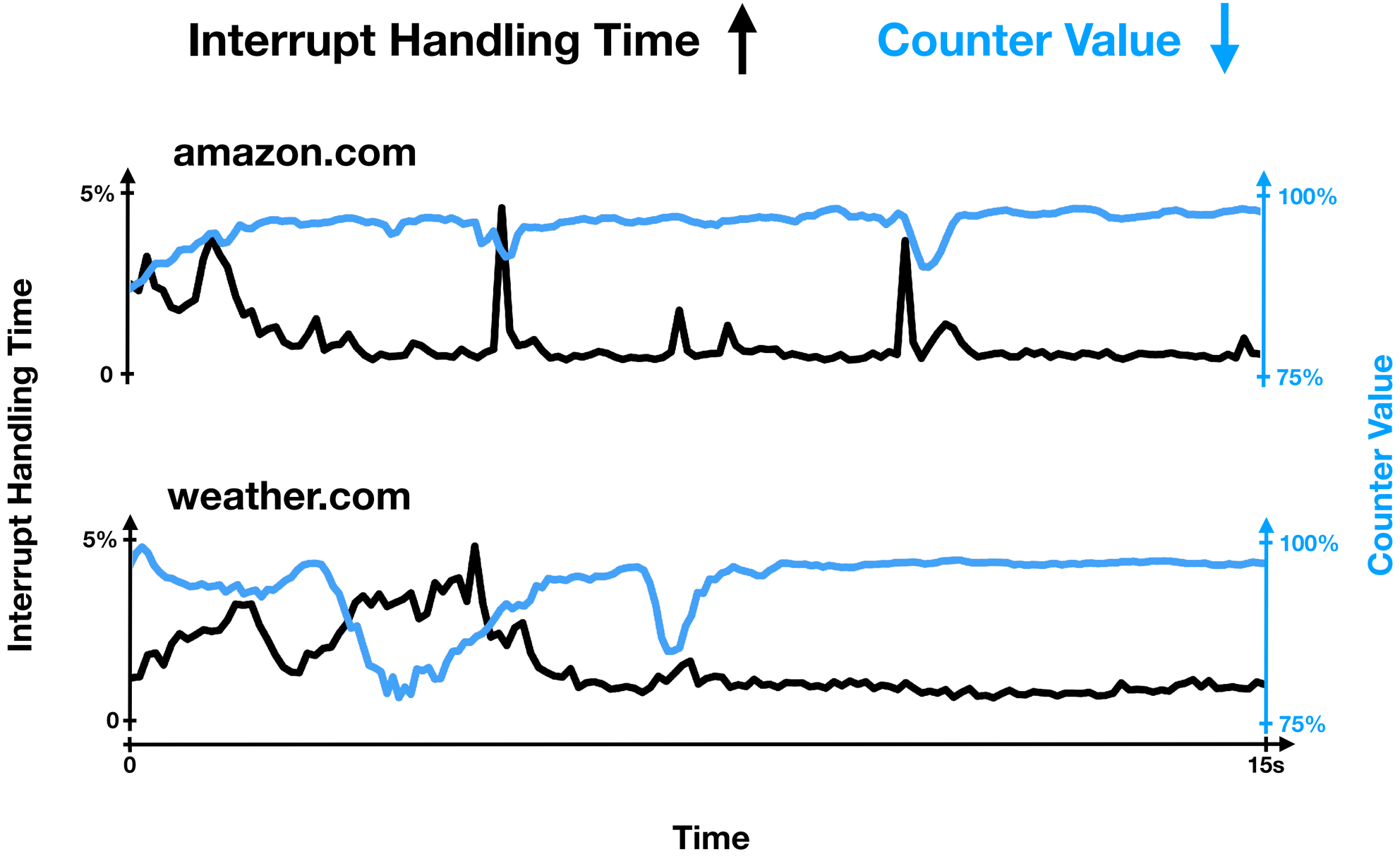
System Instrumentation

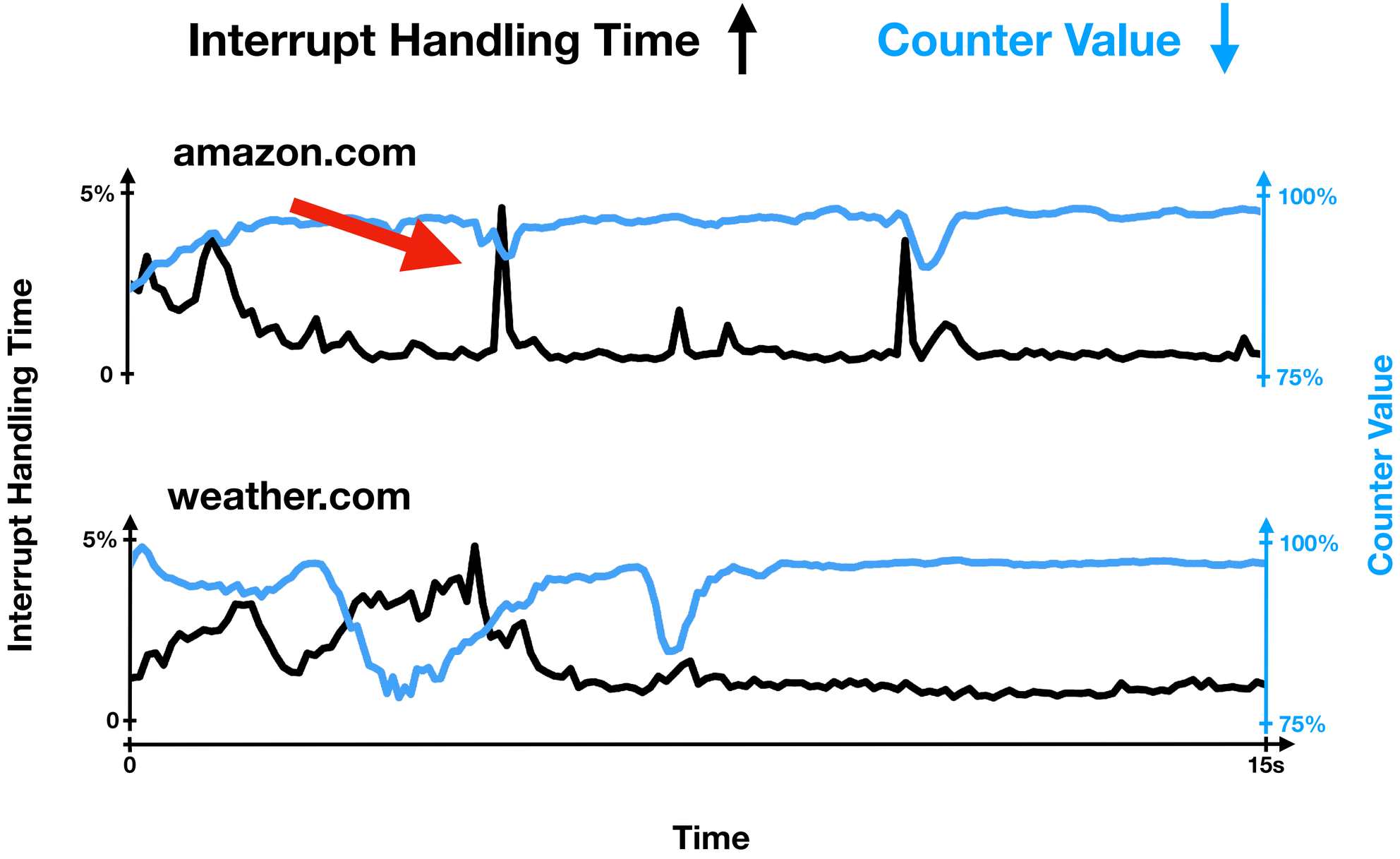
In the kernel space: use eBPF

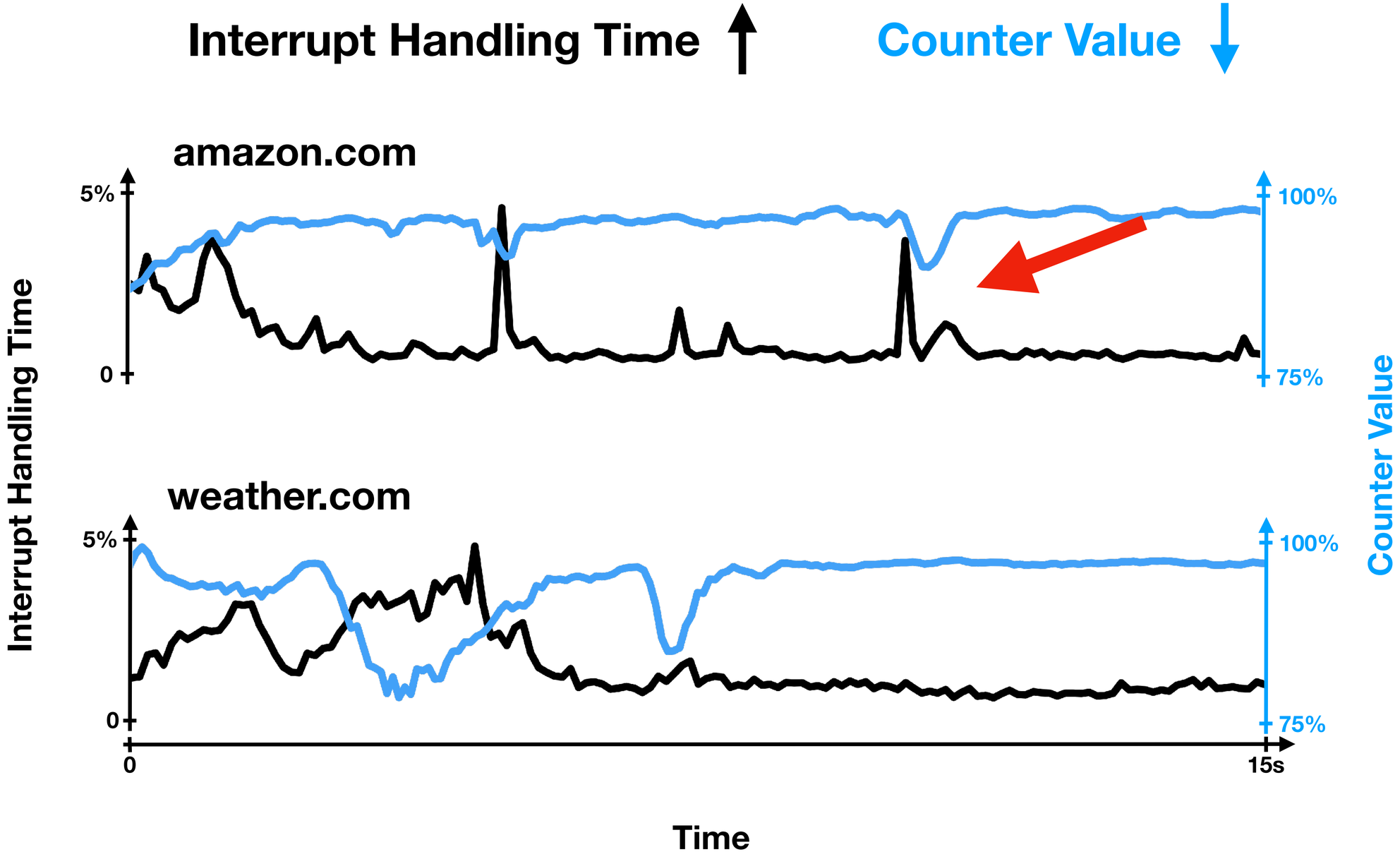
- Allows instrumentation of the Linux kernel at runtime
- We developed a tool to monitor interrupt characteristics by recording time at beginning and end of interrupt handlers

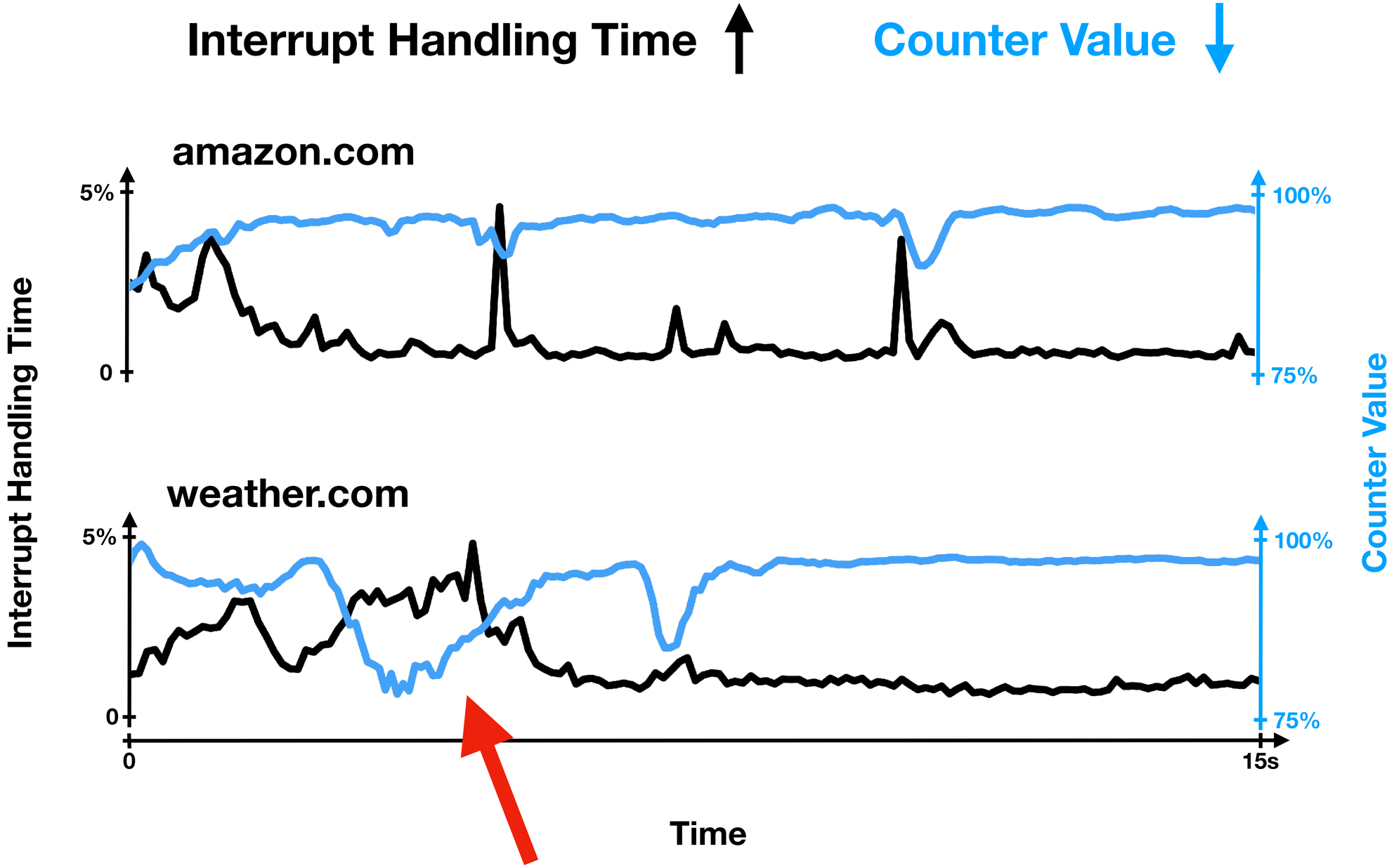
In the user space: attacker code in Rust

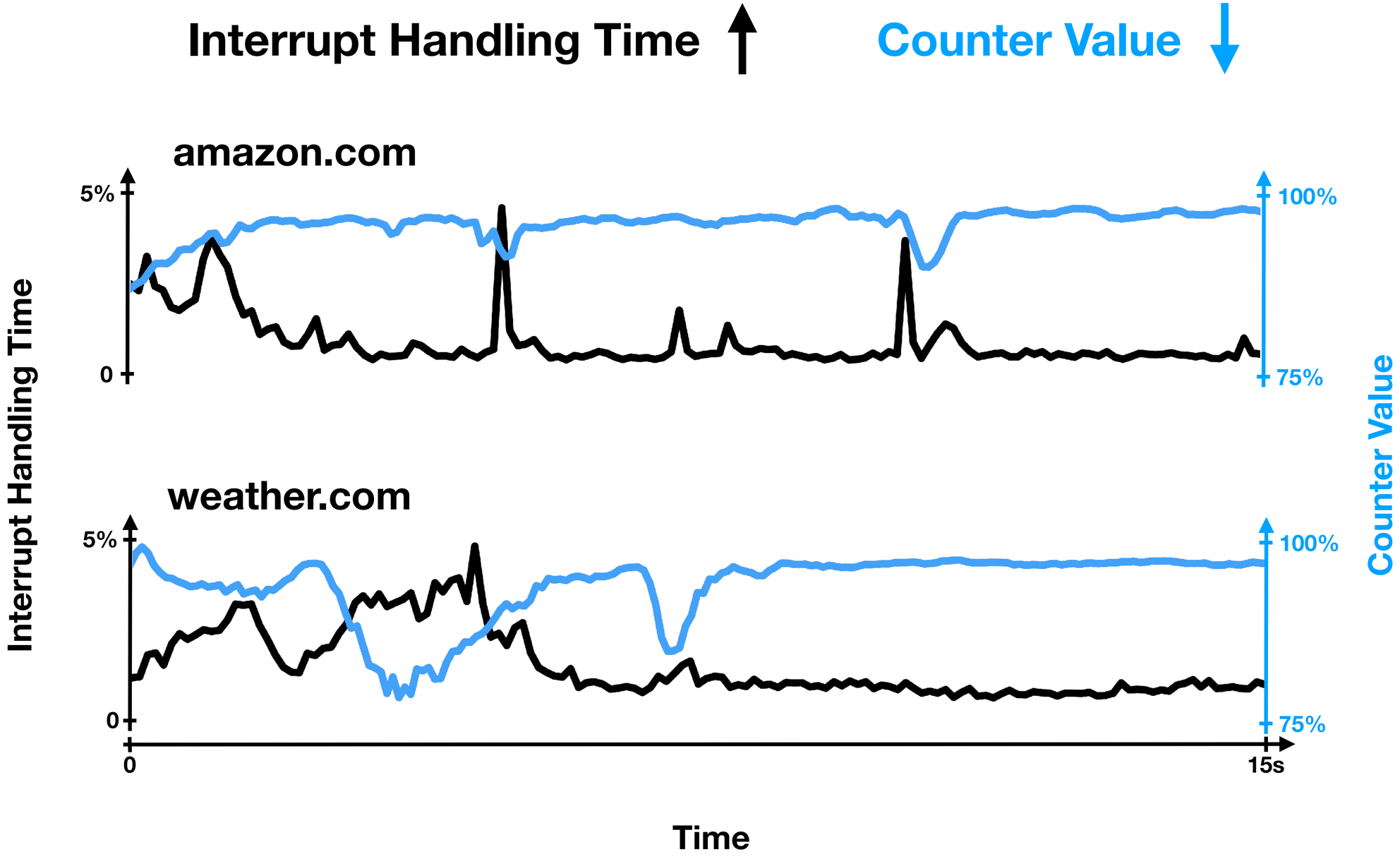
- Records time leaves and re-enters the user space

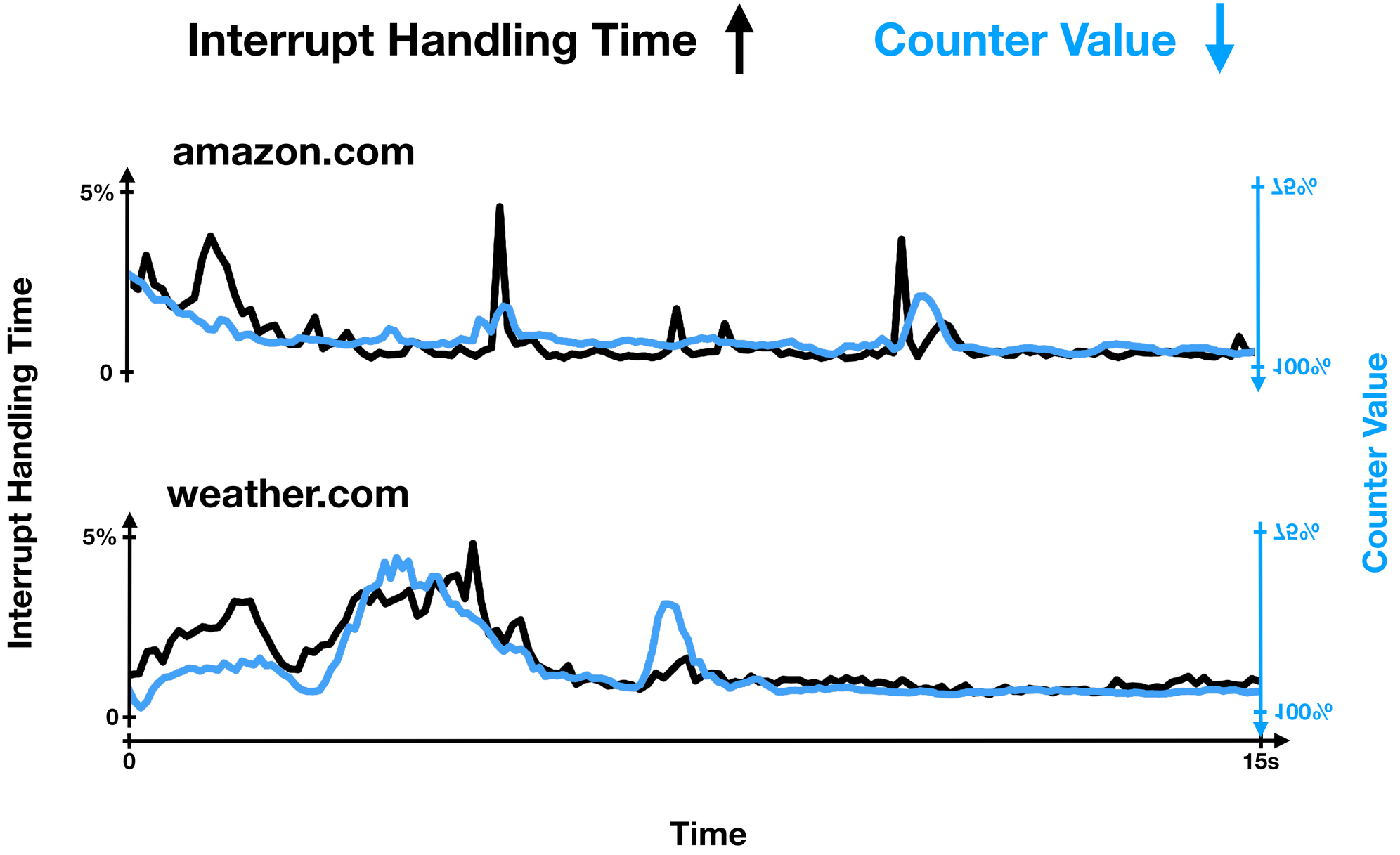

Interrupt Handling Time
Counter Value
amazon.com
5%
0
100%
75%
weather.com
5%
0
100%
75%
0
15s
Interrupt Handling Time
Counter Value
Time

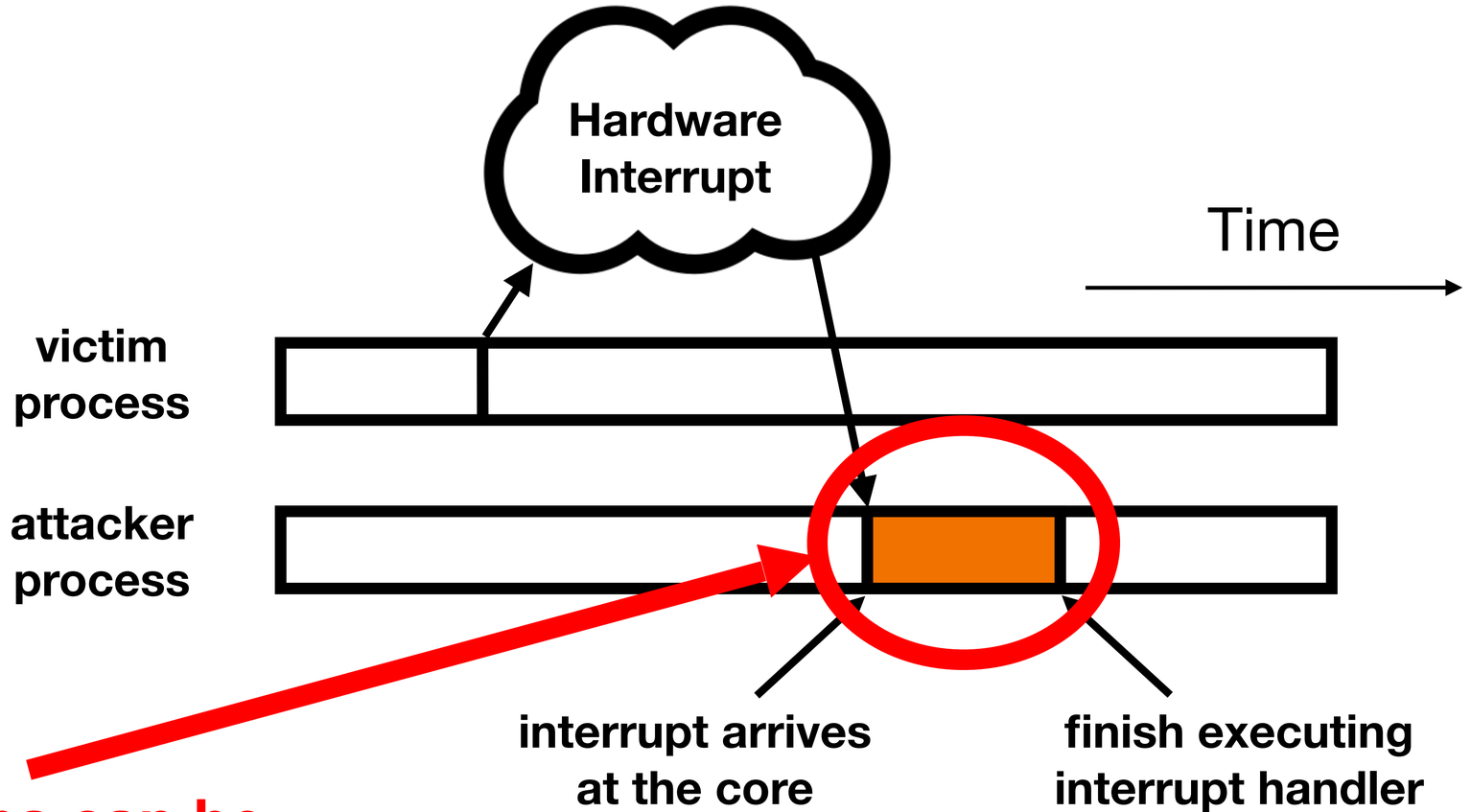

Interrupt Handling Time
Counter Value
amazon.com
5%
0
100%
75%
weather.com
5%
0
100%
75%
0
15s
Interrupt Handling Time
Counter Value
Time


Interrupt Handling Time
Counter Value
amazon.com
5%
0
100%
75%
weather.com
5%
0
100%
75%
0
15s
Interrupt Handling Time
Counter Value
Time


Interrupt Handling Time
Counter Value
amazon.com
5%
0
100%
75%
weather.com
5%
0
100%
75%
0
15s
Time
Interrupt Handling Time
Counter Value


Interrupt Handling Time
Counter Value
amazon.com
5%
0
100%
75%
weather.com
5%
0
100%
75%
0
15s
Interrupt Handling Time
Counter Value
Time


Interrupt Handling Time
Counter Value
amazon.com
5%
0
weather.com
5%
0
0
15s
Time
Interrupt Handling Time
Counter Value



99% of gaps can be explained by the presence of interrupts

Jack Cook, Jules Drean, Jonathan Behrens, Mengjia Yan

Published at the 49th International Symposium on Computer Architecture (ISCA)

Paper Code

Interactive Demo

Use this tool to collect your own traces! For best results, close all other programs and keep your mouse still while collecting traces.

Trace length

○ 1 second ◉ 5 seconds ○ 15 seconds

Website

○ None ○ nytimes.com ◉ amazon.com ○ weather.com ○ Custom

Collect trace Download traces

Key Takeaways

- Machine learning can be used to identify activity on your computer from traces recorded in JavaScript that measure CPU instruction throughput over time
- We found this type of attack exploits signals from system interrupts, which operating systems use to interact with hardware devices
  - When a core processes interrupts, it pauses the execution of an attacker, creating a signal that can be exploited
- Our *loop-counting attack* can correctly identify one of 100 websites being opened 96.6% of the time in Chrome on Linux

Jack Cook, Jules Drean, Jonathan Behrens, Mengjia Yan

Published at the 49th International Symposium on Computer Architecture (ISCA)

Paper Code

Interactive Demo

Use this tool to collect your own traces! For best results, close all other programs and keep your mouse still while collecting traces.

Trace length

○ 1 second ◉ 5 seconds ○ 15 seconds

Website

○ None ○ nytimes.com ◉ amazon.com ○ weather.com ○ Custom

Collect trace Download traces

Key Takeaways

- Machine learning can be used to identify activity on your computer from traces recorded in JavaScript that measure CPU instruction throughput over time
- We found this type of attack exploits signals from system interrupts, which operating systems use to interact with hardware devices
  - When a core processes interrupts, it pauses the execution of an attacker, creating a signal that can be exploited
- Our *loop-counting attack* can correctly identify one of 100 websites being opened 96.6% of the time in Chrome on Linux

Jack Cook, Jules Drean, Jonathan Behrens, Mengjia Yan

Published at the 49th International Symposium on Computer Architecture (ISCA)

Paper Code

Interactive Demo

Use this tool to collect your own traces! For best results, close all other programs and keep your mouse still while collecting traces.

Trace length

○ 1 second ◉ 5 seconds ○ 15 seconds

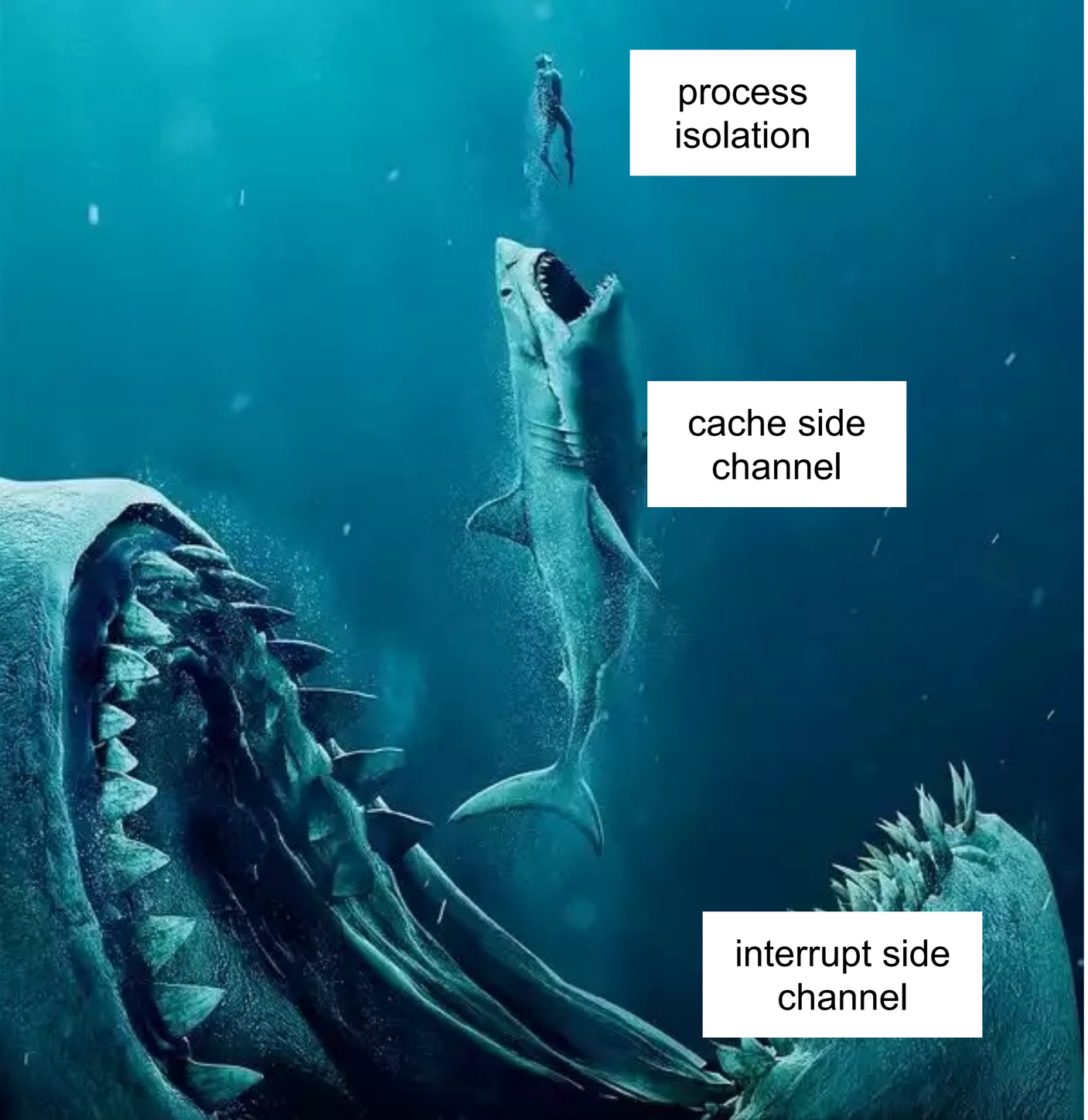
Website

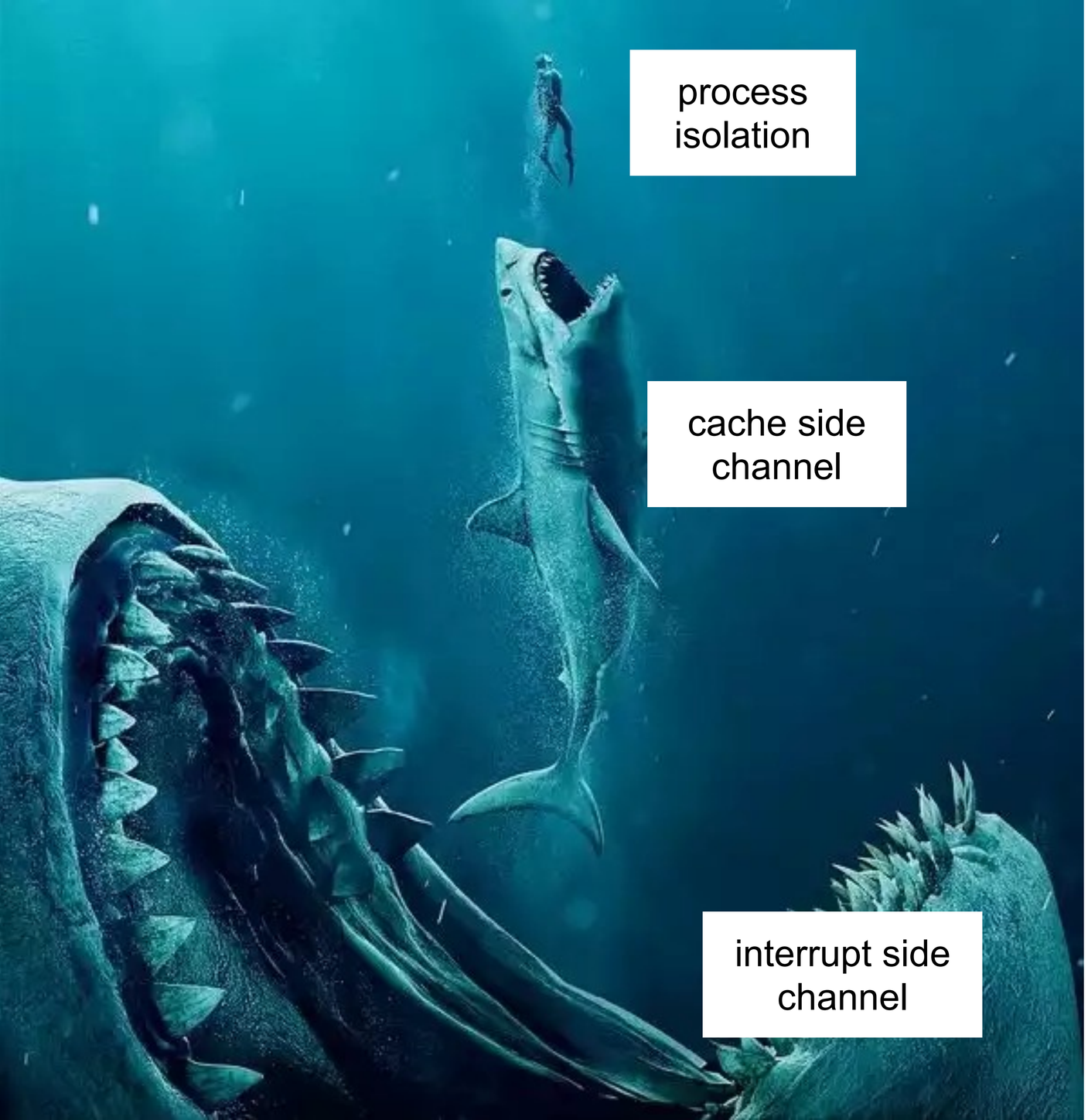
○ None ○ nytimes.com ◉ amazon.com ○ weather.com ○ Custom

Collect trace Download traces

Key Takeaways

- Machine learning can be used to identify activity on your computer from traces recorded in JavaScript that measure CPU instruction throughput over time
- We found this type of attack exploits signals from system interrupts, which operating systems use to interact with hardware devices
  - When a core processes interrupts, it pauses the execution of an attacker, creating a signal that can be exploited
- Our *loop-counting attack* can correctly identify one of 100 websites being opened 96.6% of the time in Chrome on Linux


process isolation
cache side channel
interrupt side channel



Takeaway 2: There's always a bigger fish!

Need comprehensive security analysis in complex SW-HW systems

End of The Story?

End of The Story?

- Run in separate VMs with interrupts pinned properly

End of The Story?

- Run in separate VMs with interrupts pinned properly
  - 88.2% → 91.6%

End of The Story?

- Run in separate VMs with interrupts pinned properly
  - 88.2% → 91.6%

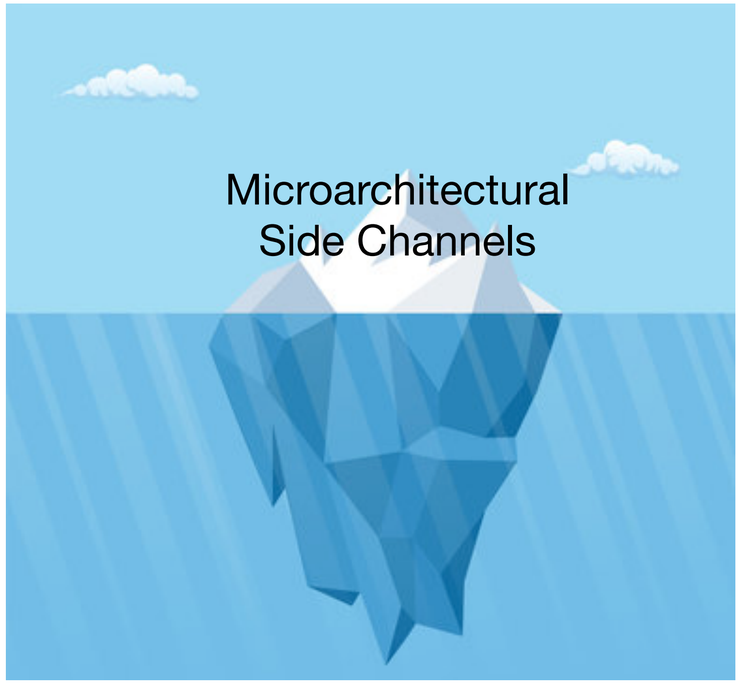
🤔

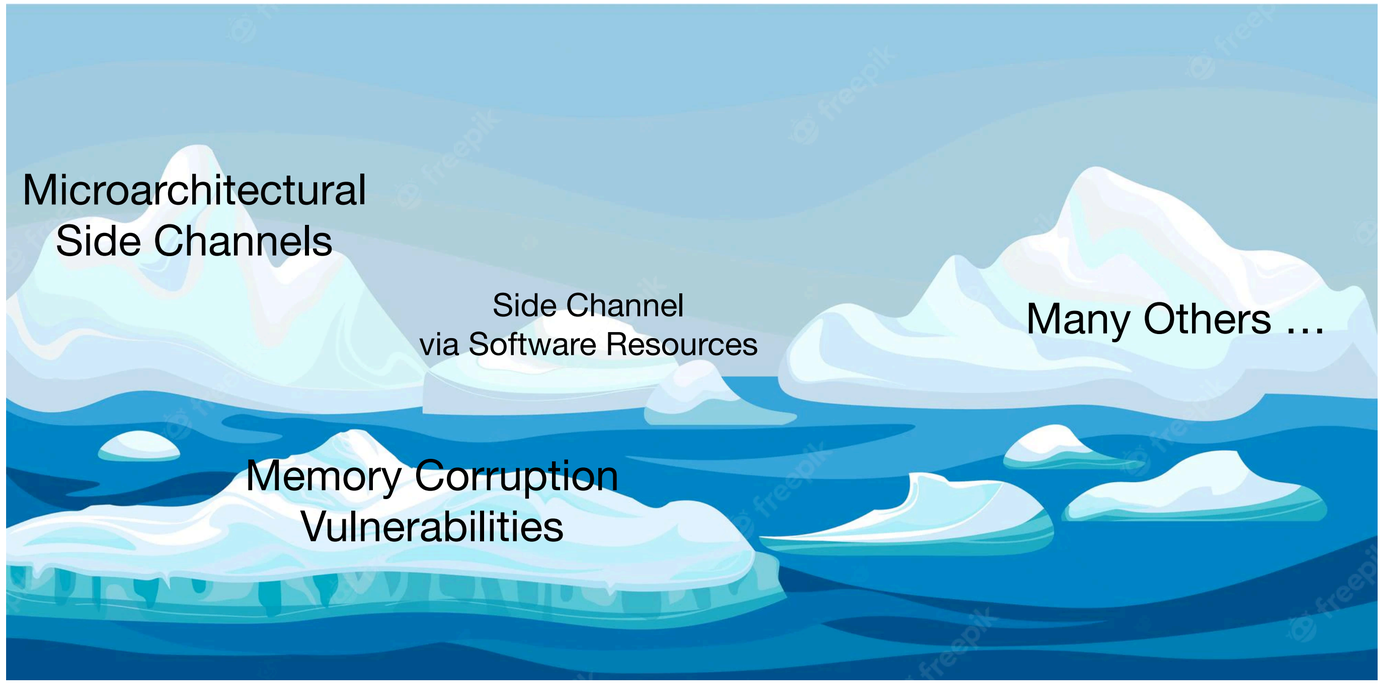
- How to decipher signals from the ML model output?

End of The Story?

- Run in separate VMs with interrupts pinned properly
  - 88.2% → 91.6% 🤔
- How to decipher signals from the ML model output?
- A “bigger bigger” fish?

Looking Beyond Microarchitectural-Only Side Channels

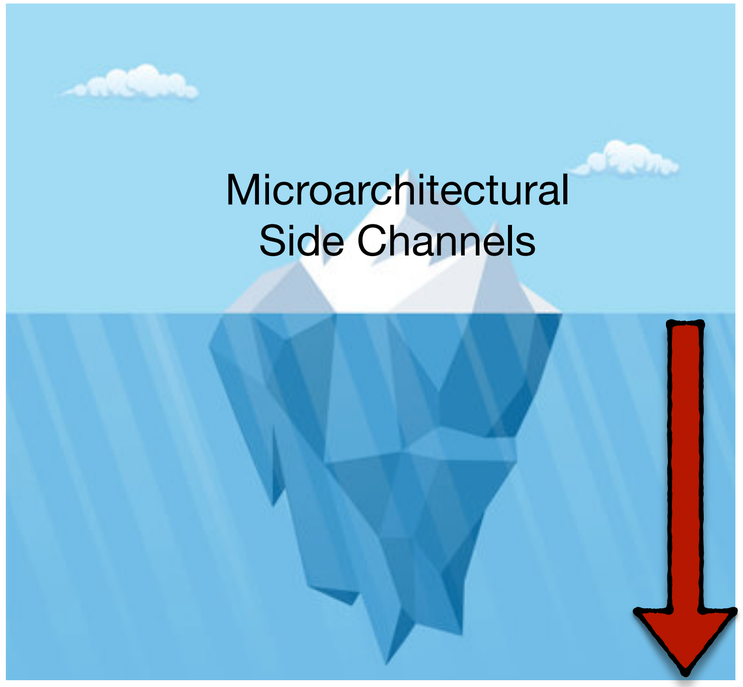


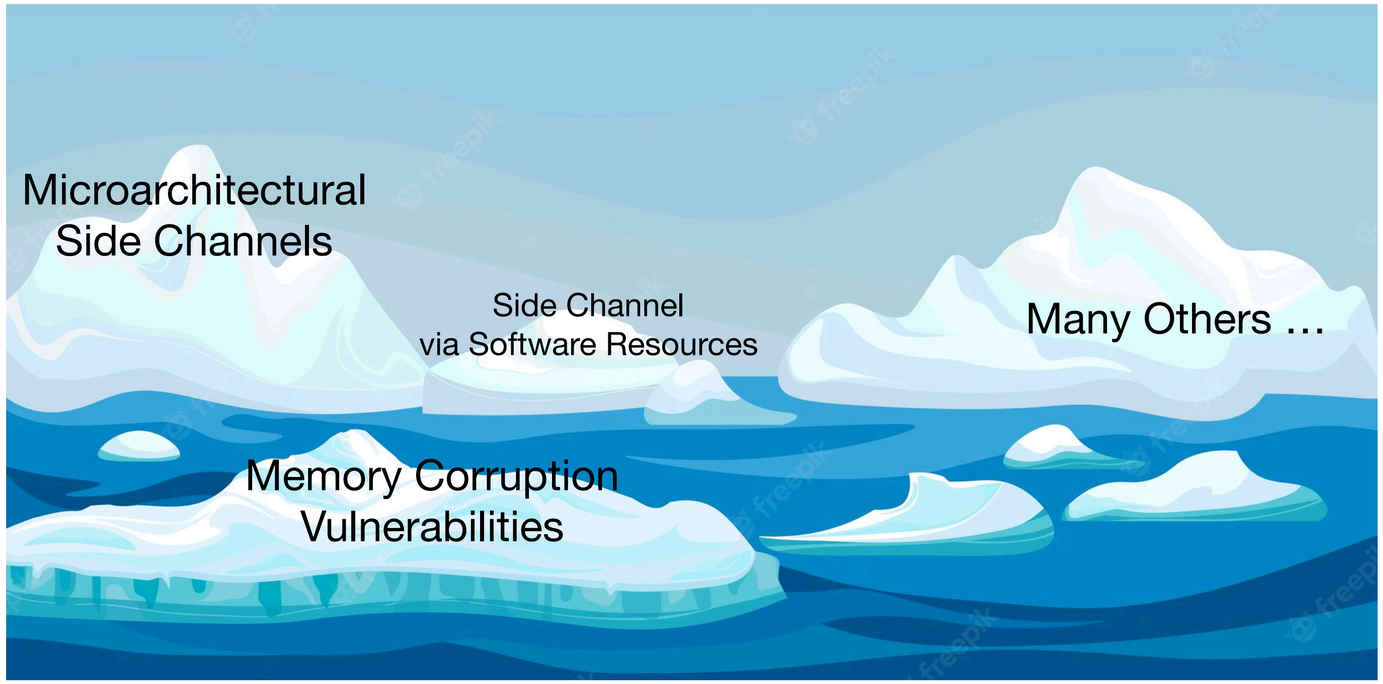


Takeaway 1: New threats arising from compound threat models

Takeaway 2: Need comprehensive security analysis for complex SW-HW systems

Looking Beyond Microarchitectural-Only Side Channels

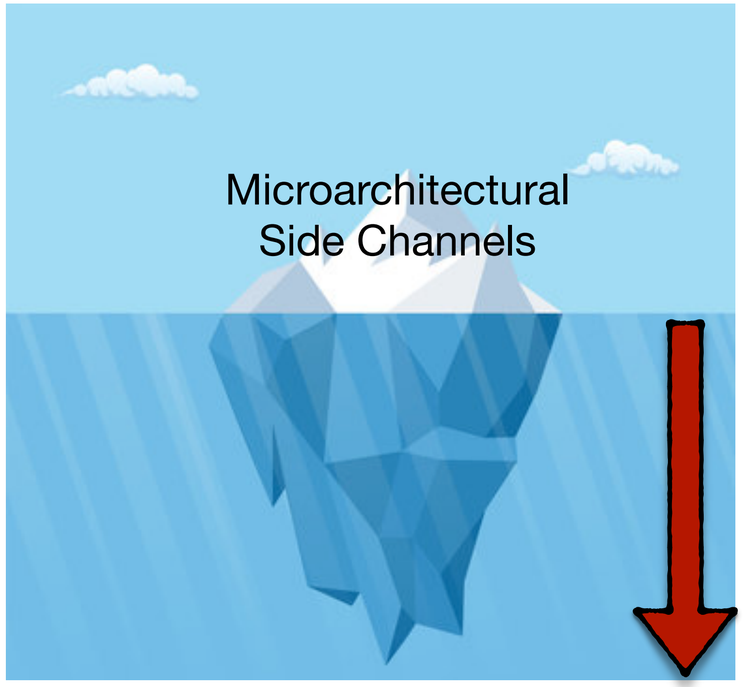


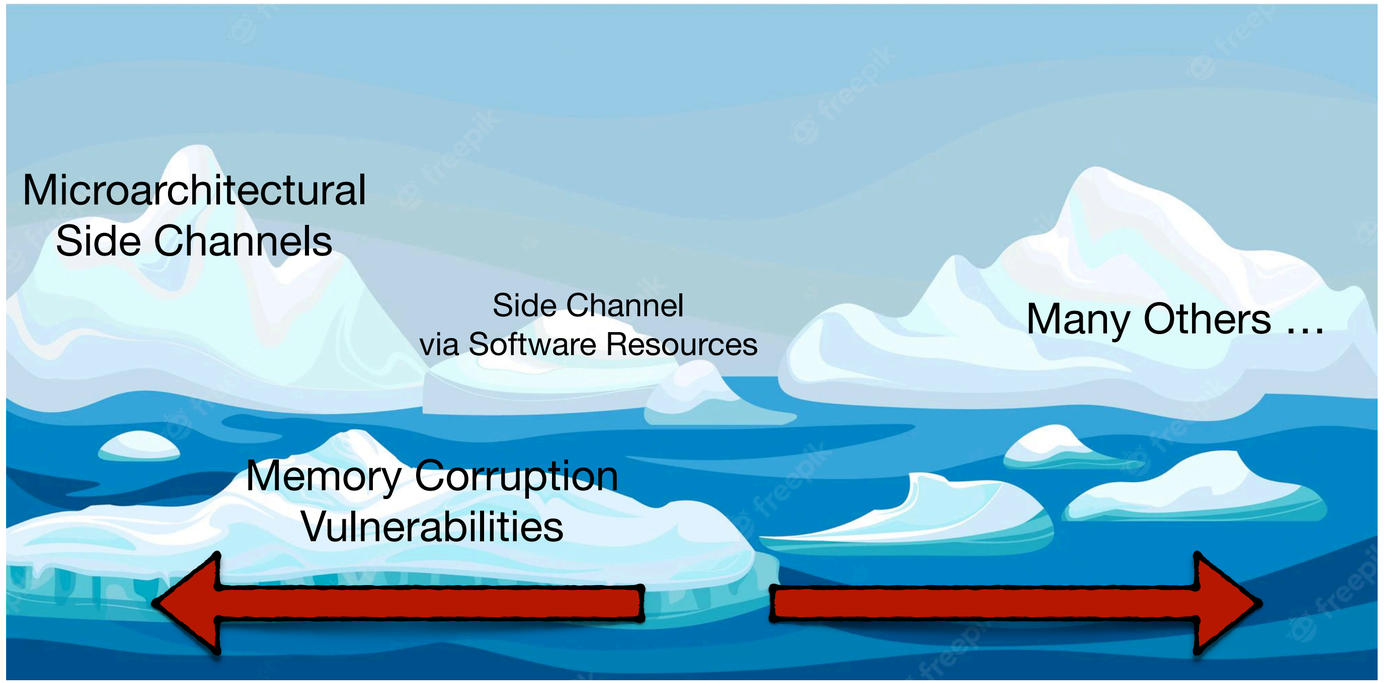


Takeaway 1: New threats arising from compound threat models

Takeaway 2: Need comprehensive security analysis for complex SW-HW systems

Looking Beyond Microarchitectural-Only Side Channels





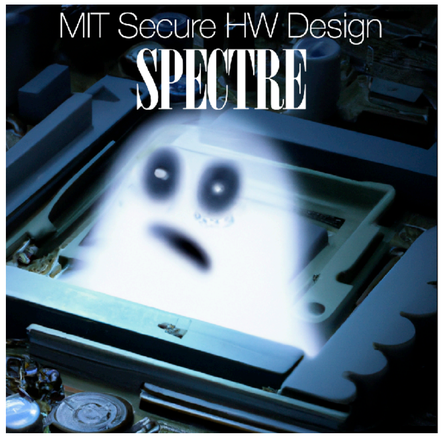
Takeaway 1: New threats arising from compound threat models

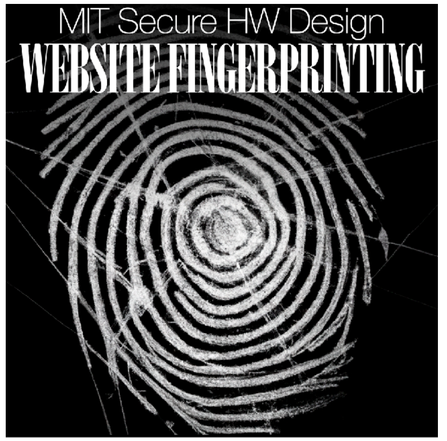
Takeaway 2: Need comprehensive security analysis for complex SW-HW systems

Learning Computer Architecture Security For Fun — 5 Lab Assignments

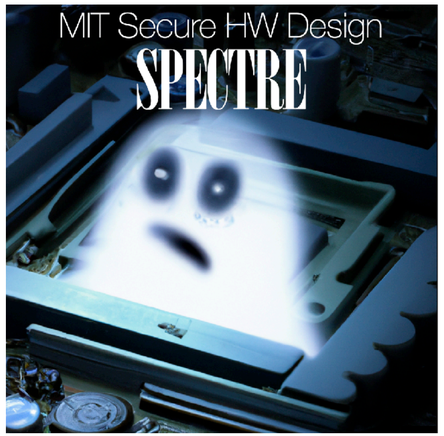
Learning Computer Architecture Security For Fun — 5 Lab Assignments

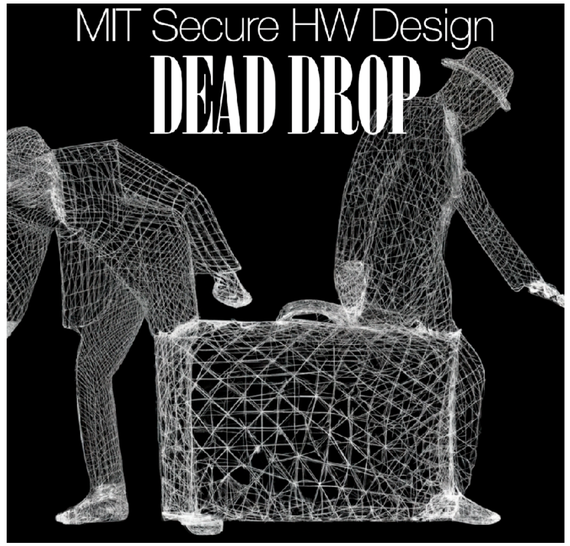
Learning Computer Architecture Security
For Fun — 5 Lab Assignments





Learning Computer Architecture Security
For Fun — 5 Lab Assignments

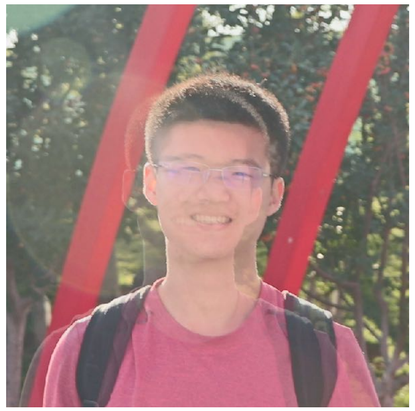




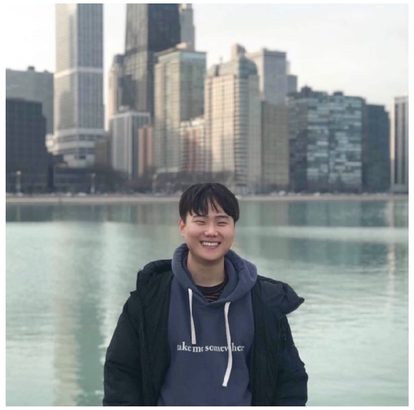
http://csg.csail.mit.edu/6.888Yan/for_instructors/

The Team

Peter Deutsch



Yuheng Yang



Weon Taek Na

Joseph Ravichandran

Mengyuan Li

Jules Drean

Shixin Song

Jack Cook

Miguel Gomez-Garcia

Looking Beyond Microarchitectural-Only Side Channels