Flush-based cache Attacks in Modern/Multi-Socket x86 Systems

Work-in-progress
HS3 2025 Workshop
Co-located with ESORICS 2025

To be submitted at uASC
(Microarchitecture Security Conference)

25/09/2025

Guillaume Didier, Universität des Saarlandes 🇩🇪; *work started at IRISA (Univ. Rennes, Inria, DGA)* 🇫🇷

Augustin Lucas, Département d'Informatique, ENS de Lyon; *internship work at IRISA (Univ. Rennes, Inria)* 🇫🇷

Thomas Rokicki, CentraleSupélec, Inria, CNRS, IRISA, Université de Rennes 🇫🇷

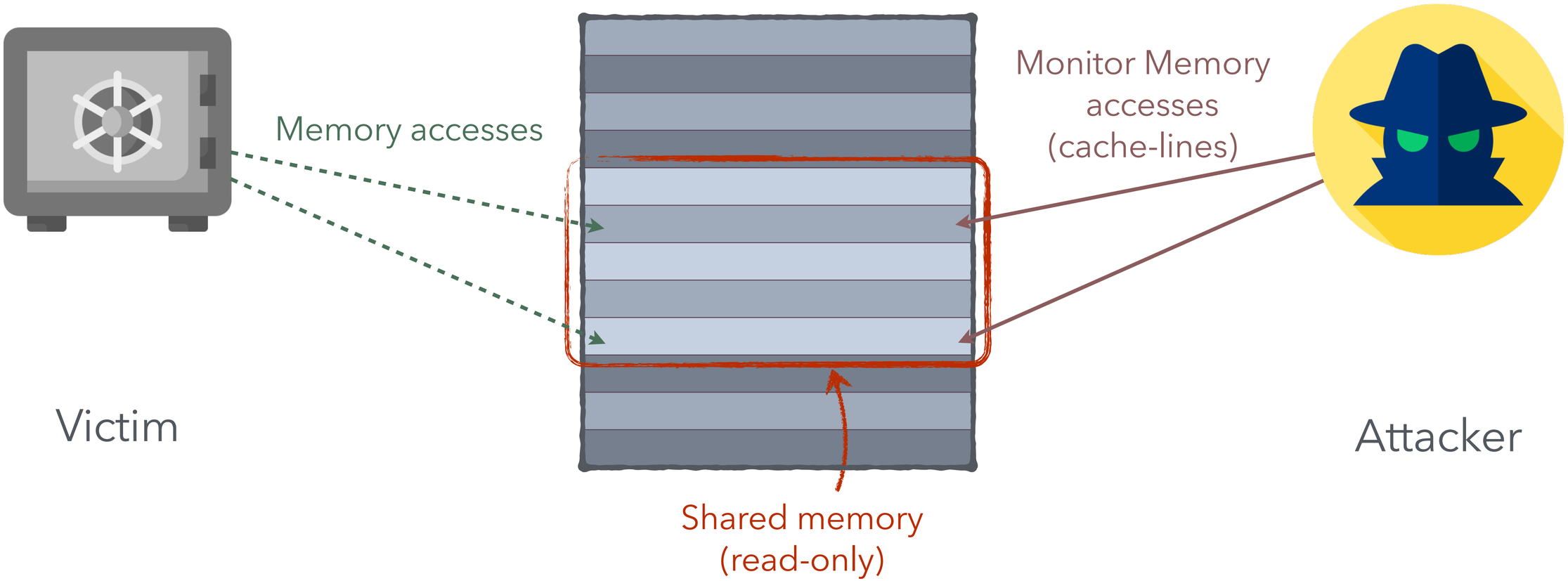
This work has received funding from an Intel Strategic Research Alliance (ISRA) and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101020415)"

Timeline

- 2 socket systems were a loose end of my PhD (2022)
- Augustin Lucas internship in Taran to investigate Summer 2024
- Identify automatic NUMA-rebalancing as the cause.
- Updated the framework in 2025, brought Thomas in to get access to Grid 5k
- Work in progress presented at HS3
- Deadline 04/11 for uASC

Flush-based attack ?

What the attacker gets

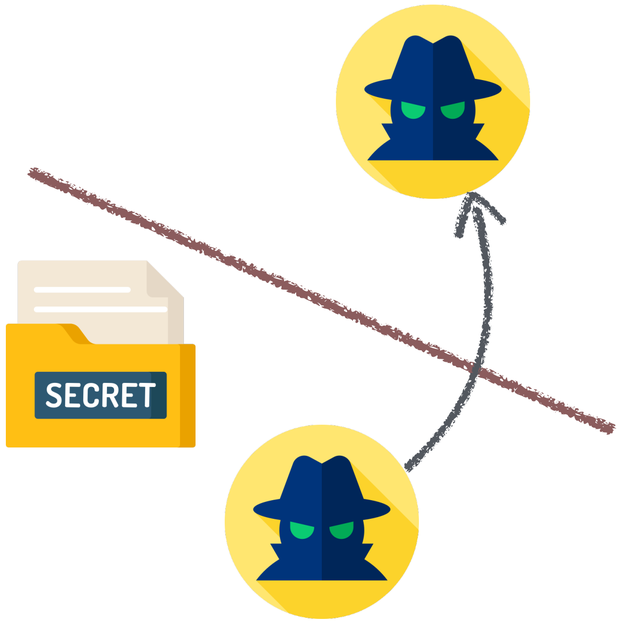


Use cases for attacks

What sort of evil can we do

1. Side-channel

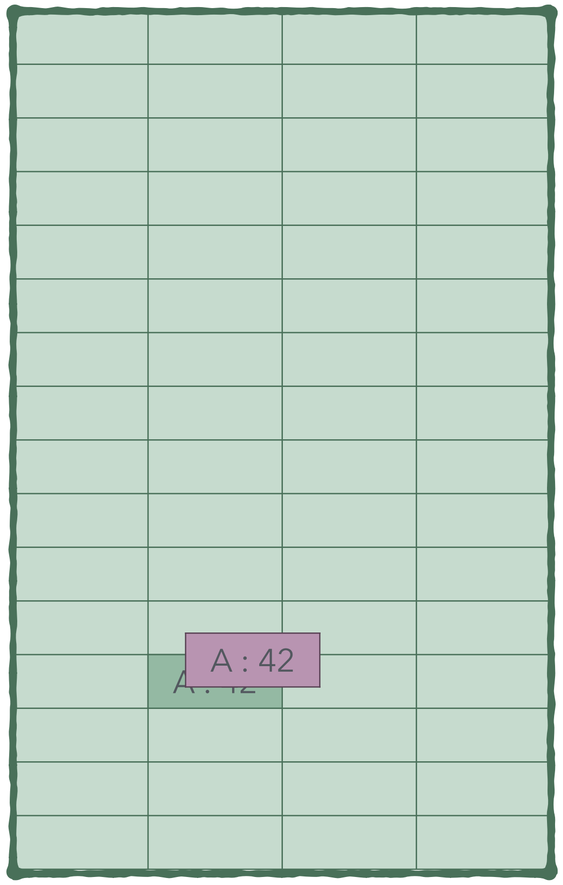
2. Covert-channel



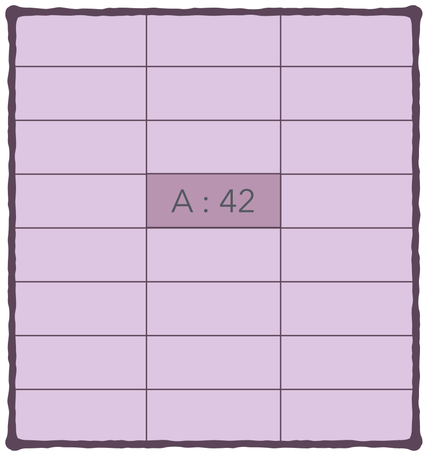
3. Transient-execution attacks

Flush+Reload intuition

The basic idea



Main Memory



Cache

1  Set-up: Attacker Flush

2  Maybe load

3  Timed-reload

A : 42 DRAM: Slow 

A : 42 Cache: Fast 

Flush+Flush

An amusing variant

Flush-time also depends on the cache state

Noisier, but faster

Motivation

The limits of the state of the art

- Flush+Reload and Flush+Flush are great attacks on x86
- But they are a decade old.
- We don't know what happens on quite a few machines : ***Let's fix this !***

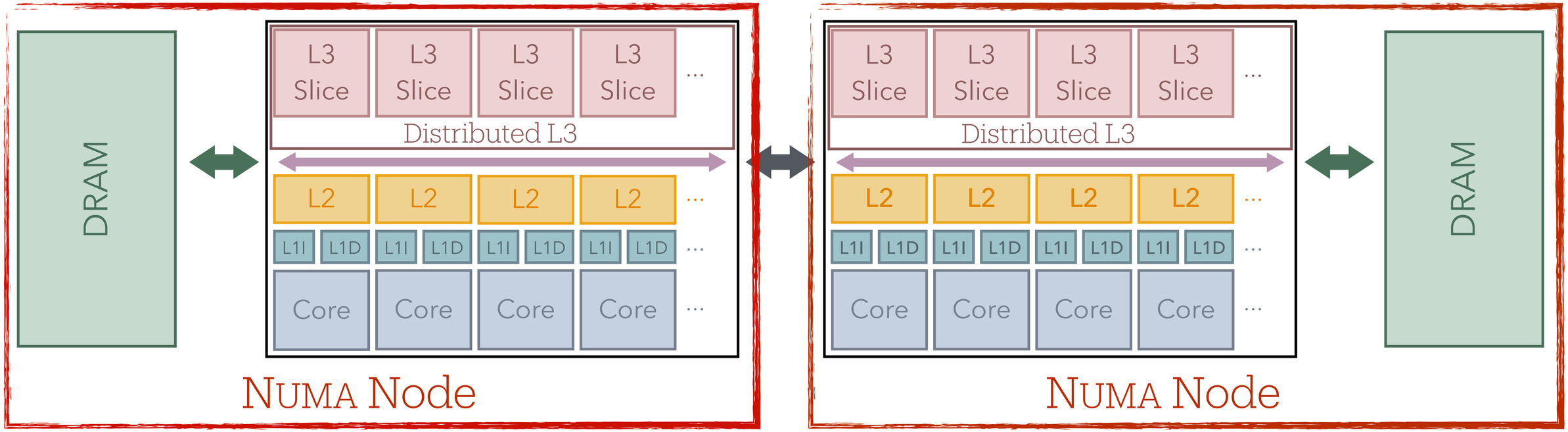
	Intel Single-Socket	AMD Single-Socket	Intel Multi-socket	AMD Multi-Socket
Flush+Reload	works	works	**assumed** to work	**assumed** to work
Flush+Flush	works until Skylake (2018) unknown otherwise	unknown	unknown	unknown

Background

CPUs are complicated,
so are cache-attacks on them

Modern x86 systems

NUMA — Non-Uniform Memory Access



Intel old CPU structure, details differ on recent servers and AMD

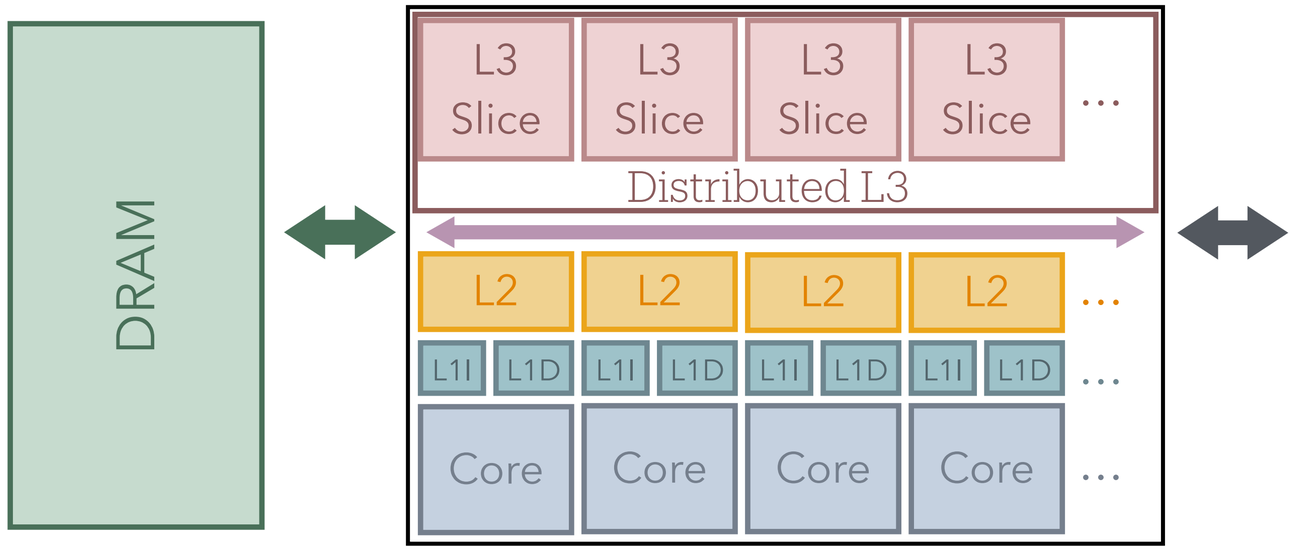
All those cache have to be kept ***coherent*** !

Per-socket directories, and cross-socket coherence protocols

Background: Cache Coherence

Thinking properly about the state transitions

- Make all core agree on the sequence of value taken by a location.
- States for each cache line
  - Modified
  - Exclusive
  - Shared
  - Invalid
- Centralised / Distributed approach ?
- You must **think** of the **coherence state** in cache attacks, not just Hit/Miss



Background: `clflush`

This instruction should never have been unprivileged

Description

Invalidates from every level of the cache hierarchy in the cache coherence domain the cache line that contains the linear address specified with the memory operand. […]

From the Intel Manual

- **`clflush`** reaches all sockets (coherence domain)
- Requires *shared* memory

Bonus: **`clflush`**'s execution time depends on the *cache coherence state*
(at least on Intel CPUs)

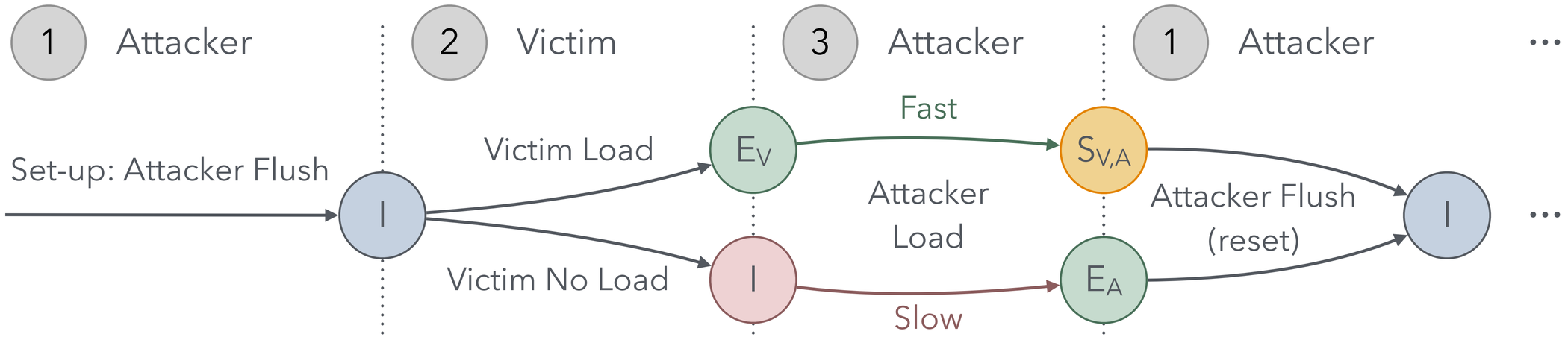
Background: Flush+Reload in details

Assumptions, cache coherence transitions

- ***Shared, Read-only*** memory sections are realistic:
  (shared library .TEXT / .RODATA, hypervisor memory deduplication, transient execution attacks)

- Leaks ***cache-line*** addresses

Cache coherence transitions

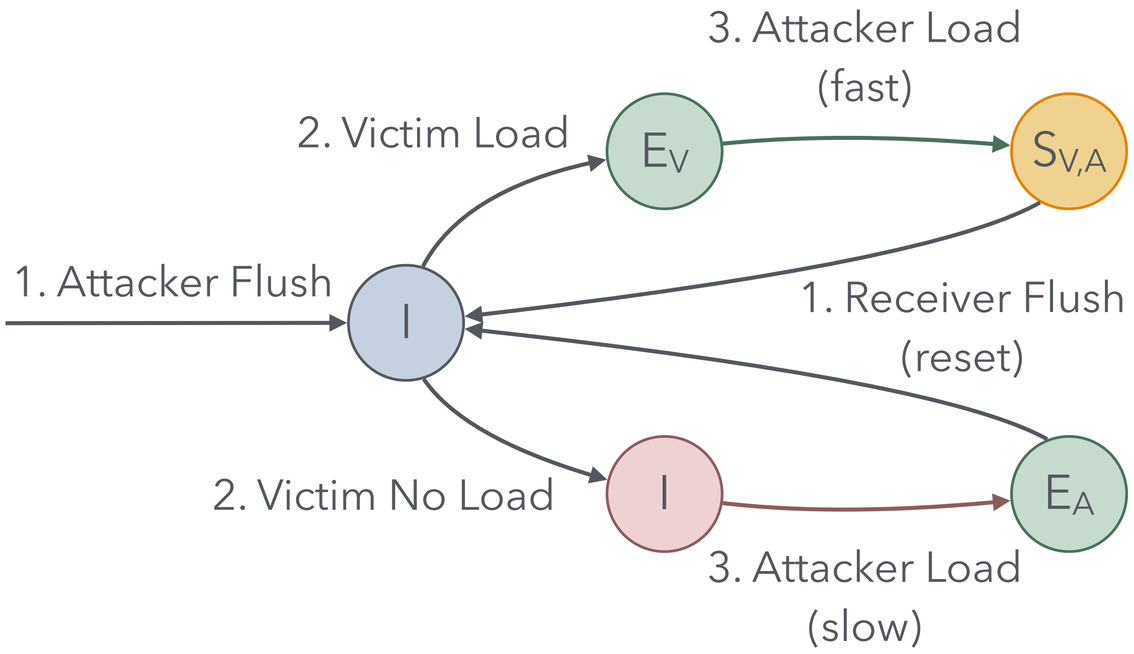


Yuval Yarom, Katrina Falkner: *FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack*. USENIX Security 2014

Background: Cache transitions

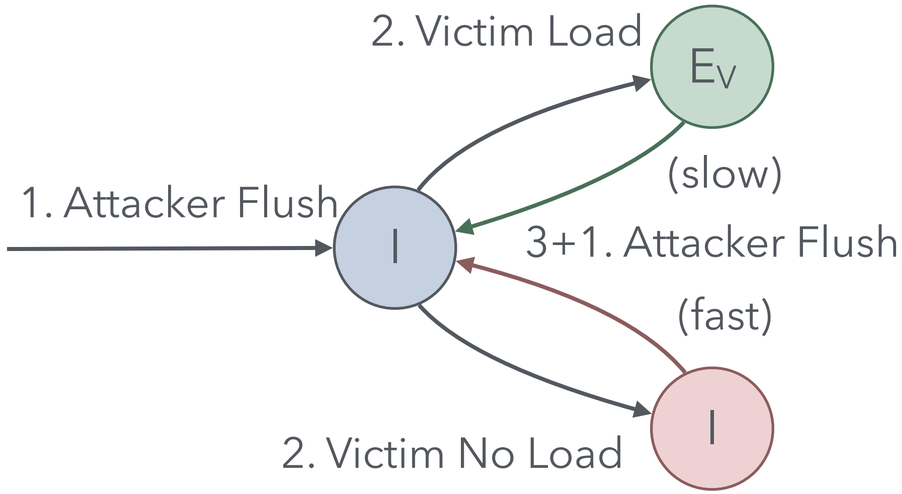
Flush+Reload — Flush+Flush

Flush+Reload



Yuval Yarom, Katrina Falkner:
FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack.
USENIX Security 2014

Flush+Flush



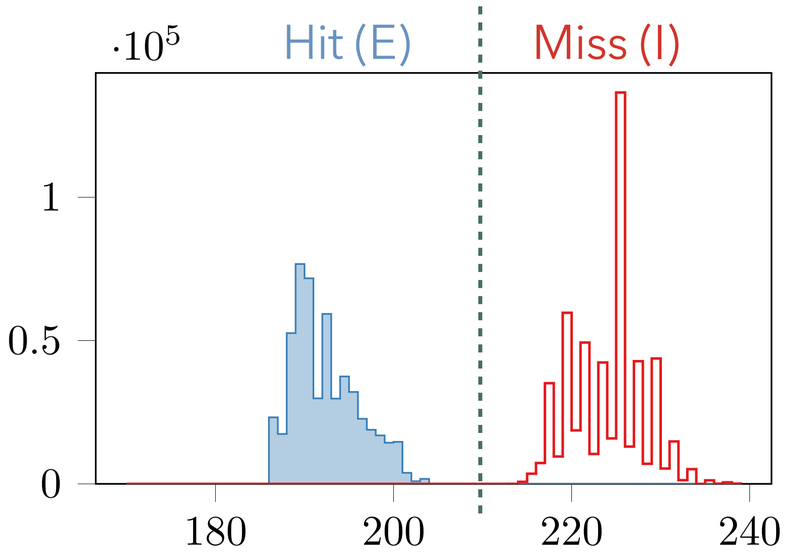
Daniel Gruss, Clémentine Maurice, Klaus Wagner, Stefan Mangard:
Flush+Flush: A Fast and Stealthy Cache Attack.
DIMVA 2016

Background: Calibration

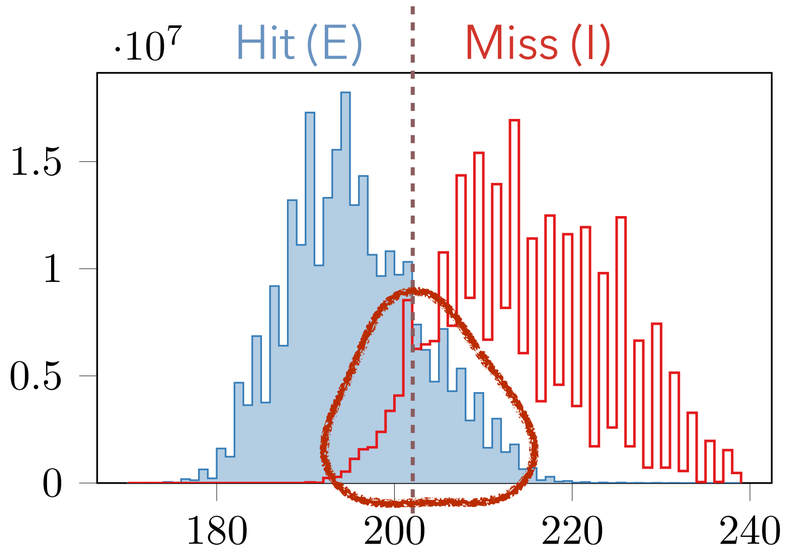
Finding the best threshold to distinguish between cache states

1. Measure execution times for all topology configs *(attacker, victim, target)*
2. Build histograms & Find Threshold(s)

Good



Bad

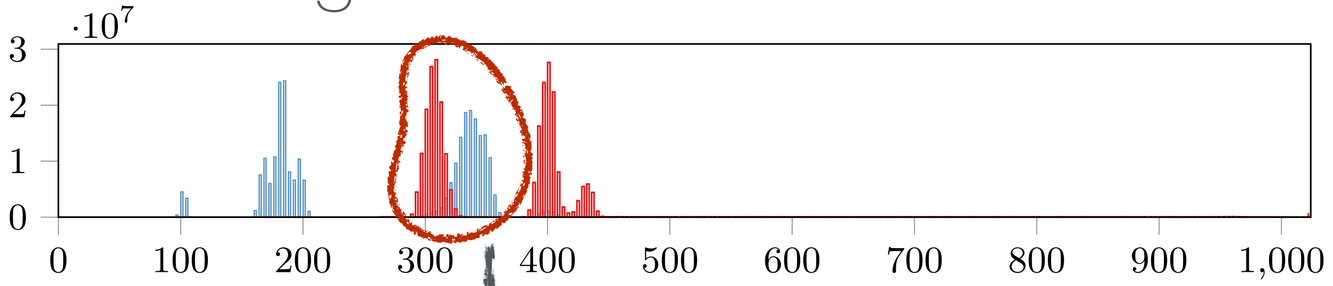


Guillaume Didier, Clémentine Maurice: *Calibration Done Right: Noiseless Flush+Flush Attacks*. DIMVA 2021

Topology-awareness

How to get better histograms

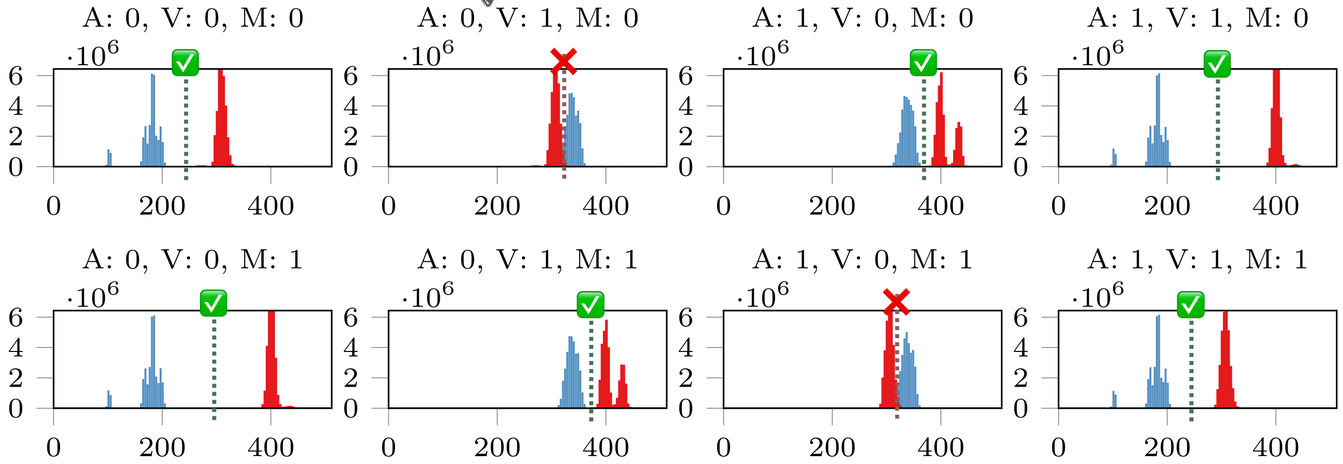
Topology-unaware



25% average error rate

Split histograms according to topology

NUMA-aware



2,8% average error rate

Fig. 2: Flush+Reload Histograms on a 2×Haswell EP system (2015). Hits are in blue, filled; misses are in thick outline red.

Know the configuration, use the right threshold.

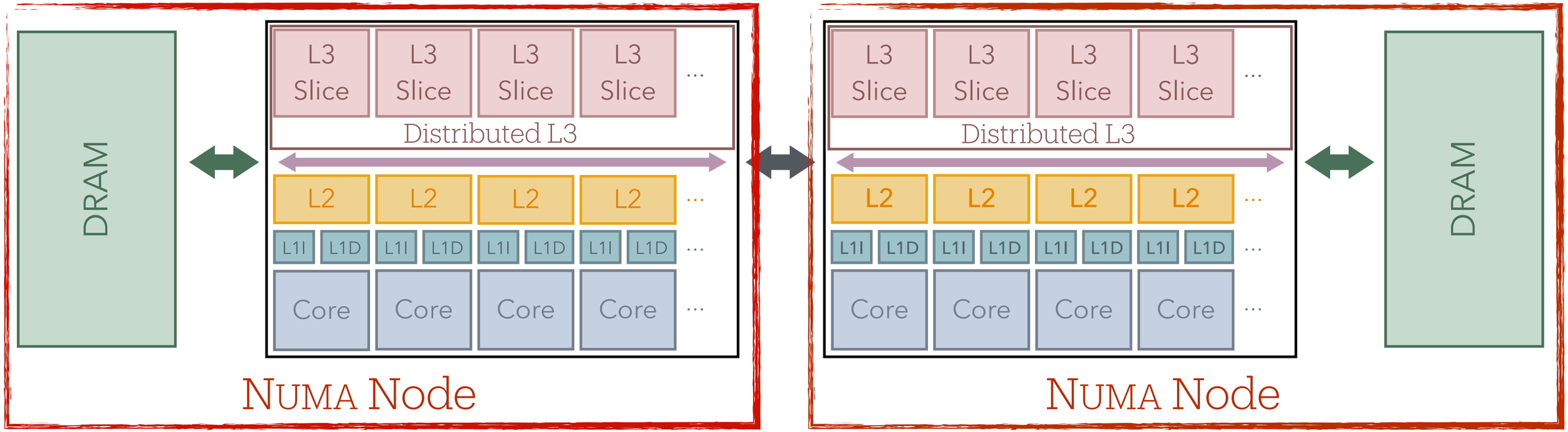
Choose the configuration, get the best threshold. 0.01% best error rate

Experiments

What have we done ?

Experiments

Adapting *Calibration Done Right* (DIMVA 2021) to NUMA



Iterate on all possible locations for ***Attacker***, ***Victim*** and ***Target*** (Memory and L3 Slice)

Challenges

- *Quadratic* execution time (worst case: over a ***day***)
- *Massive* data set
  (worst case: ***80 GiB*** in RAM, 1.5 GiB compressed on disk)

Covert Channel Benchmark

How fast are the attacks ?

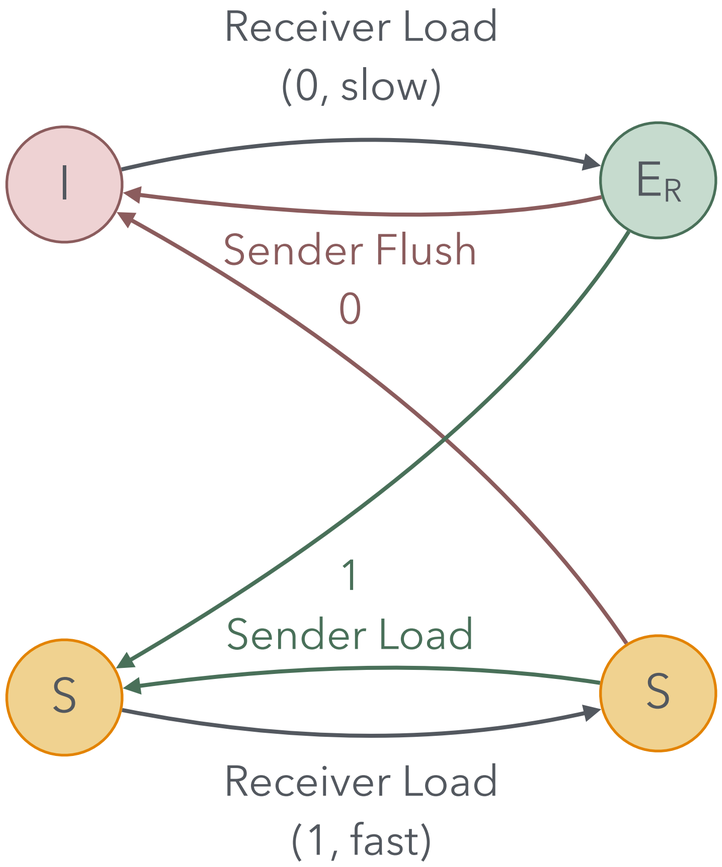
We want to leak data quickly, marginal accuracy improvement can be bad.

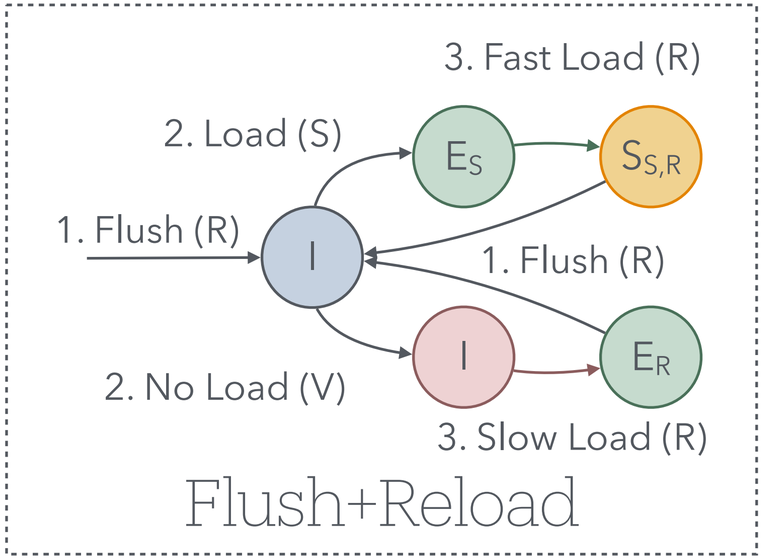
- Transmit 128 bytes (1024 bits), measuring time, repeat 16 times.
- Do this for all pairs of cores and memory, using 1 to 10 different cache-lines (in a different page each).
- Measure number of errors
- Compute bandwidth, error rate and deduce true capacity

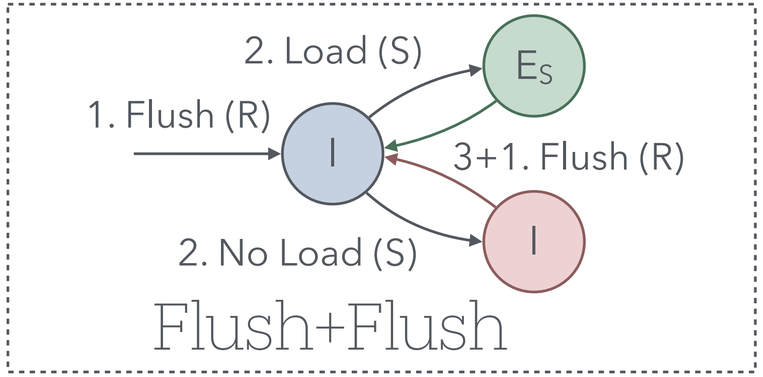
Load/Flush+Reload (LF+R)

This is not Flush+Reload

- Flush+Reload covert channel ***unexpectedly much better*** than predicted ?
- Subtle bug, implements a ***different*** set of ***cache transitions***.
- Only works as a ***covert channel***, but much better than Flush+Reload and Flush+Flush
- Compares ***Shared Load*** to ***Invalid Load***.







Load/Flush+Reload (New)

Surveying Intel and AMD Machines

Many machines !

- More than 30 machines surveyed
  - > 25 multi-socket systems, from Grid 5k (thanks to Irisa), 2006 to 2024
  - 9 single socket machines, from the uops.info farm
    (Zen+ to Zen5, and 2025 Intel single-socket CPUs)
- Large Data set
  - 45 GiB of compressed experimental data
  - Complexity issues (time and space) in analysis.
  - 1.5 GiB can expand into 80 GiB and the analysis consume 300 GiB
    (that's Grenoble's abominable 4-socket snowman, `yeti`)

Calibration Analysis: Attacker Models

What attackers do we consider — are they realistic ?

Known for Side-channels
Covert-channel may Chose
Kernel / libnuma API

	Memory		Attacker		Victim	
	NUMA Node	Addr (proxy for slice)	Node	Core	Node	Core
Topology Unaware (TU)	U ?	U ?	U ?	U ?	U ?	U ?
NUMA-AVM	K 📝	U ?	K 📝	U ?	K 📝	U ?
NUMA-AVM-Best	C ✅	U ?	C ✅	U ?	C ✅	U ?
NUMA-AVM-Addr	K 📝	K 📝	K 📝	U ?	K 📝	U ?
NUMA-AVM-Addr-S-Best-AVM	C ✅	K 📝	C ✅	U ?	C ✅	U ?
NUMA-AVM-Addr-Best-AVM-Addr	C ✅	C ✅	C ✅	U ?	C ✅	U ?
NUMA-M-Core-AV	K 📝	U ?	K 📝	K 📝	K 📝	K 📝
NUMA-M-Core-AV-Best	C ✅	U ?	C ✅	C ✅	C ✅	C ✅
NUMA-M-Core-AV-Addr	K 📝	K 📝	K 📝	K 📝	K 📝	K 📝
NUMA-M-Core-AV-Addr-Best-Addr	K 📝	C ✅	K 📝	K 📝	K 📝	K 📝
NUMA-M-Core-AV-Addr-Best-No-Addr	C ✅	K 📝	C ✅	C ✅	C ✅	C ✅
NUMA-M-Core-AV-Addr-Best	C ✅	C ✅	C ✅	C ✅	C ✅	C ✅

Kernel API for core affinity
(or libnuma for node pinning)

Our Attacker Models are
Realistic

Best offline-calibrated model

Online-calibrated needed

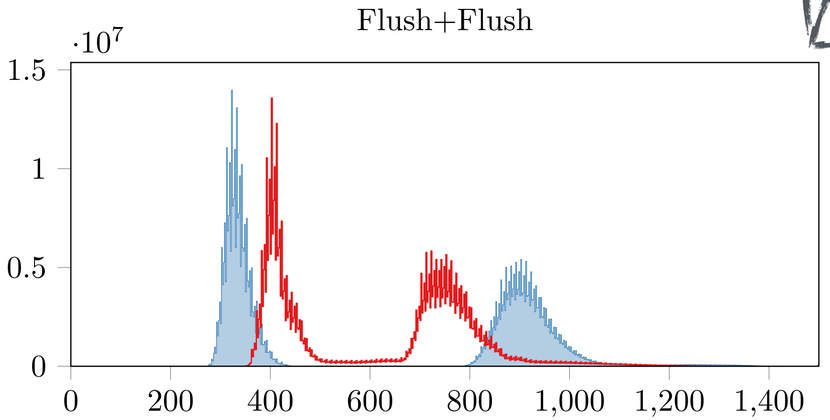
Best side-channel model

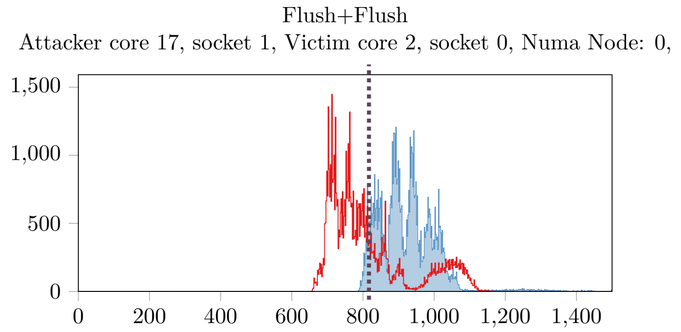
Best covert-channel model

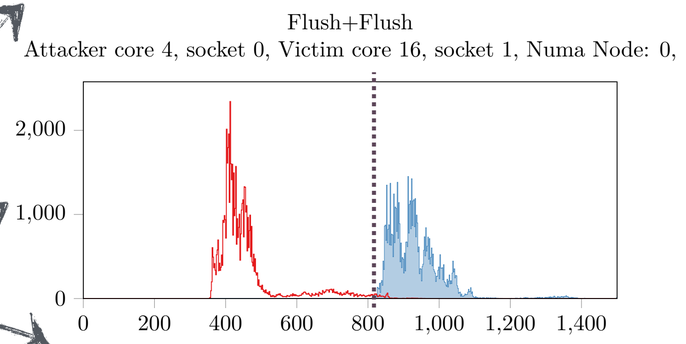
Calibration Analysis: Example

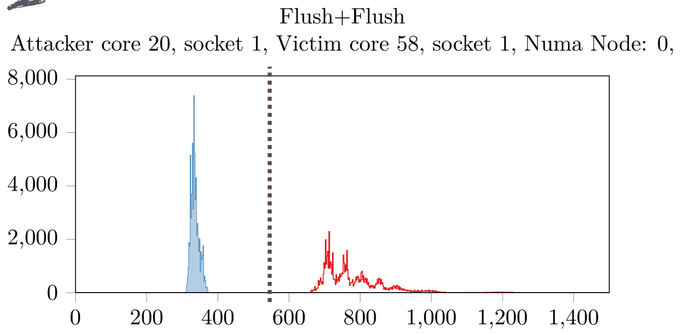
Let's look at the Numa-M-Core-AV model

1. Split according to model
2. Compute Thresholds and Average Error
3. Evaluate threshold against NUMA-M-Core-AV-Addr series
4. Compute statistics
   (Min, Q1, Med, Q3, Max)

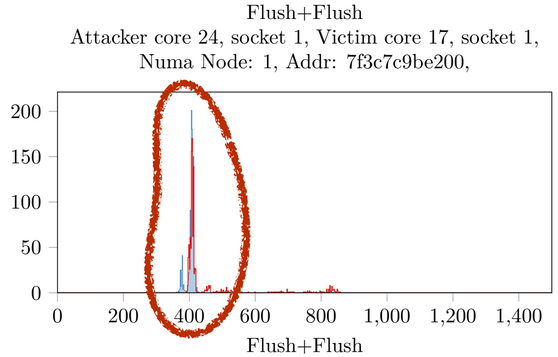


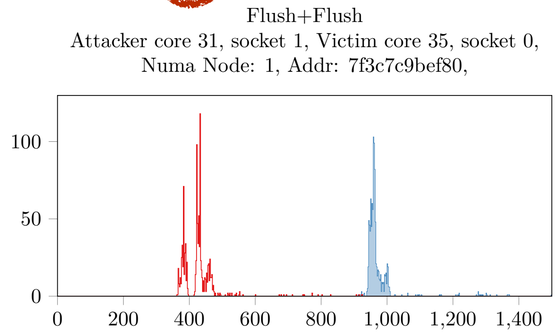


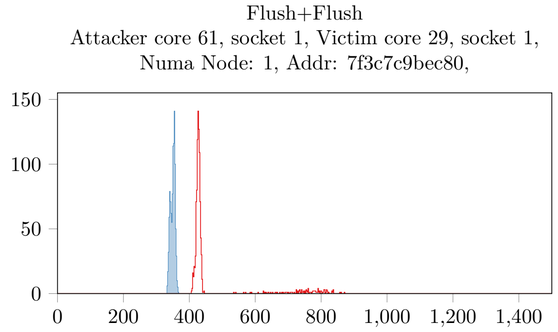




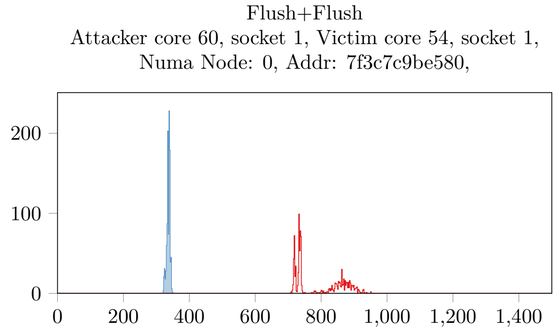
Stat	Error rate (%)
Avg	4.02
Max	56.64
Q3	4.10
Med	0.05
Q1	<0.01
Min	<0.01







...



Flush+Flush histograms on a 2×Sapphire Rapids system (2023). Hits are in blue, filled; misses are in thick outline red.

Calibration Results

A lot of data,
And that’s only a subset.

Calibration Analysis: Example

Statistics for 2×Sapphire Rapids (Intel Server, 2023)

Attacker Model	Primitive	Average	Min	Q1	Med	Q3	Max
Topology-Unaware	FF (D)	7.55	<0.01	0.20	2.39	10.89	100.00
	FR (D)	32.65	<0.01	9.23	33.74	50.00	100.00
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*26.37*
NUMA-M-Core-AV (offline-trainable)	FF (S)	4.02	<0.01	<0.01	0.05	4.10	56.64
	FR (S)	17.77	<0.01	<0.01	4.25	36.28	100.00
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*1.61*
NUMA-M-Core-AV-Best (offline-trainable)	FF (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	FR (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*
NUMA-M-Core-AV-Addr	FF (S)	1.95	<0.01	<0.01	<0.01	1.27	39.99
	FR (S)	7.77	<0.01	<0.01	1.76	12.55	48.10

Results on a 2×Sapphire Rapids system (2023). (S) uses simple thresholds, (D) uses pairs of thresholds as the classifier

Calibration Analysis: Example

Statistics for 2× Zen4 (AMD Server, 2022) — using `rdpru` not `rdtsc`

Attacker Model	Primitive	Average	Min	Q1	Med	Q3	Max
Topology-Unaware	FF	12.66	<0.01	<0.01	<0.01	49.85	51.51
	FR (D)	18.01	<0.01	6.25	8.89	22.31	90.48
	L/F+R	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*0.34*
NUMA-M-Core-AV (offline-trainable)	FF (S)	0.12	<0.01	<0.01	<0.01	<0.01	50.05
	FR (S)	7.61	<0.01	4.64	6.79	9.18	87.55
	FR (D)	2.10	<0.01	0.20	0.68	1.95	68.80
NUMA-M-Core-AV-Best (offline-trainable)	FF	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	FR	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	L/F+R (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
NUMA-M-Core-AV-Addr	FF (S)	0.03	<0.01	<0.01	<0.01	<0.01	49.56
	FR (S)	6.55	<0.01	3.86	6.54	8.94	48.39
	FR (D)	0.93	<0.01	0.05	0.39	0.93	47.36

Results on a 2× Zen4 system (2022). (S) uses simple thresholds, (D) uses pairs of thresholds as the classifier, if unspecified both classifier give the same result.

Calibration Analysis: Example

Statistics for 1× Zen5 (AMD Client, 2024) — using `rdpru` not `rdtsc`

Attacker Model	Primitive	Average	Min	Q1	Med	Q3	Max
Topology-Unaware	FF	8.67	<0.01	<0.01	0.05	0.63	50.34
	FR	3.19	<0.01	<0.01	0.10	0.20	50.10
	L/F+R	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*0.15*
*(NUMA-M-)*Core-AV (offline-trainable)	FF	5.54	<0.01	<0.01	0.05	0.20	50.24
	FR	3.14	<0.01	<0.01	0.10	0.20	61.04
(NUMA-AVM)-Addr	FF	0.98	<0.01	<0.01	<0.01	0.05	43.21
	FR	3.19	<0.01	<0.01	0.10	0.20	50.10
(NUMA-M)-Core-AV-Best (offline-trainable)	FF (S)	0.01	<0.01	<0.01	<0.01	<0.01	0.15
	FR (S)	<0.01	<0.01	<0.01	<0.01	<0.01	0.10
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*
(NUMA-M)-Core-AV-Addr	FF (S)	0.38	<0.01	<0.01	<0.01	0.05	27.59
	FR (S)	3.01	<0.01	<0.01	0.10	0.20	49.76
(NUMA-M)-Core-AV-Addr-Best (covert channel)	FF (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	FR (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

Results on a 1× Zen5 system (2024). (S) uses simple thresholds, (D) uses pairs of thresholds as the classifier, if unspecified both classifier give the same result.

Calibration Analysis: Example

Statistics for 1× Zen2 (AMD Client, 2019) — using `rdpru` not `rdtsc`

Attacker Model	Primitive	Average	Min	Q1	Med	Q3	Max
Topology-Unaware	FF	0.74	<0.01	<0.01	<0.01	1.12	12.55
	FR	16.97	<0.01	0.05	2.44	38.28	84.33
	L/F+R	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*0.05*
*(NUMA-M-)*Core-AV (offline-trainable)	FF	0.52	<0.01	<0.01	<0.01	0.24	9.47
	FR	12.79	<0.01	<0.01	0.05	21.04	84.72
(NUMA-AVM)-Addr	FF	0.61	<0.01	<0.01	<0.01	0.83	11.18
	FR	10.81	<0.01	<0.01	0.10	3.52	52.64
(NUMA-M)-Core-AV-Best (offline-trainable)	FF	<0.01	<0.01	<0.01	<0.01	<0.01	0.15
	FR (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*
(NUMA-M)-Core-AV-Addr	FF (D)	0.09	<0.01	<0.01	<0.01	<0.01	4.10
	FR (D)	5.07	<0.01	<0.01	<0.01	1.12	48.44
(NUMA-M)-Core-AV-Addr-Best (covert channel)	FF (S)	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

Results on a 1× Zen5 system (2024). (S) uses simple thresholds, (D) uses pairs of thresholds as the classifier, if unspecified both classifier give the same result.

Calibration Analysis: Example

Statistics for 1× Arrow Lake (Intel Client, 2024)

Attacker Model	Primitive	Average	Min	Q1	Med	Q3	Max
Topology-Unaware	FF	<0.01	<0.01	<0.01	<0.01	<0.01	0.29
	FR	0.01	<0.01	<0.01	<0.01	<0.01	0.59
	L/F+R	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*0.10*
(NUMA-M)-Core-AV (offline-trainable)	FF	<0.01	<0.01	<0.01	<0.01	<0.01	0.24
	FR	<0.01	<0.01	<0.01	<0.01	<0.01	0.59
(NUMA-M)-Core-AV-Best (offline-trainable)	FF	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	FR	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*
(NUMA-M)-Core-AV-Addr	FF (D)	<0.01	<0.01	<0.01	<0.01	<0.01	0.20
	FR (S)	<0.01	<0.01	<0.01	<0.01	<0.01	0.54

Results on a 1× Arrow Lake system (2024). (S) uses simple thresholds, (D) uses pairs of thresholds as the classifier, if unspecified both classifier give the same result.

Calibration Analysis: Example

Statistics for 1× Emerald Rapids (Intel Server, 2024)

Attacker Model	Primitive	Average	Min	Q1	Med	Q3	Max
Topology-Unaware	FF	0.05	<0.01	<0.01	<0.01	<0.01	50.05
	FR (S)	41.34	9.13	25.54	38.82	50.00	100.00
	FR (D)	33.67	1.27	18.80	29.88	42.24	99.80
	L/F+R	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*2.44*
*(NUMA-M-)*Core-AV (offline-trainable)	FF	0.02	<0.01	<0.01	<0.01	<0.01	45.90
	FR (D)	19.74	<0.01	6.93	16.89	28.12	90.09
*(NUMA-M)-*Core-AV-Best (offline-trainable)	FF	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	FR	<0.01	<0.01	<0.01	<0.01	<0.01	0.05
	L/F+R (S)	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*	*<0.01*
*(NUMA-M)-*Core-AV-Addr	FF (S)	0.01	<0.01	<0.01	<0.01	<0.01	23.78
	FF (D)	<0.01	<0.01	<0.01	<0.01	<0.01	3.12
	FR (D)	7.79	<0.01	1.66	5.86	11.91	43.41
*(NUMA-M)-*Core-AV-Addr-Best (covert channel)	FR	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

Results on a 1× Emerald Rapids system (2024). (S) uses simple thresholds, (D) uses pairs of thresholds as the classifier, if unspecified both classifier give the same result.

Flush+Reload is way harder to make accurate than Flush+Flush

Calibration Analysis: Summary

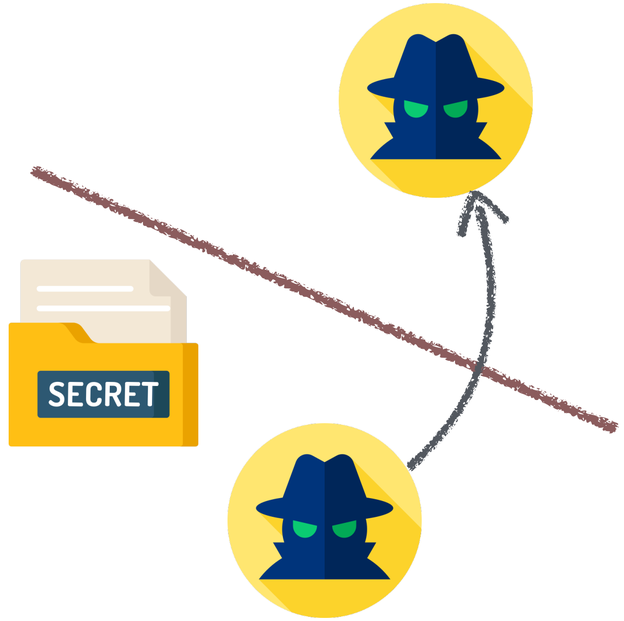
What can we take away from all this data ?

	Intel Single-Socket Client	Intel Single-Socket Server	AMD Single-Socket	Intel Multi-socket	AMD Multi-Socket
Flush+Reload	Works	Works only with topology awareness	Works only with topology awareness + `rdpru`	Numa-aware works	Numa-aware works
Flush+Flush	Works well	Works well	Works well with topology awareness + `rdpru`	Numa-aware works well	Numa-aware works well

- Flush+Flush works on AMD, `rdpru` needed, and often, topology-awareness.
- Flush+Flush better than Flush+Reload on recent Intel, and all multi-socket systems.
- NUMA-awareness improves attack a lot, Topology-awareness improves it further.
- New, accurate covert-channel variant, need a better name than Load/Flush+Reload ?

Covert Channel

But first, any *questions* ?



Covert channel benchmark

Implementation notes

- The goal is to estimate the theoretical limit
- We use a generic implementation, parametrised with :
  - Thresholding strategy (i.e. attacker model)
  - Attack primitive
- We use two threads, and a synchronise each cache-line independently.
- Longer experiments, we optimised by exploiting symmetries.
- Only Topology-Unaware, NUMA-M-Core-AV-Addr and NUMA-M-Core-AV-Addr-Best-Addr

Covert Channel Benchmark

How *fast* are the attacks ? Preliminary findings

- LF+R is usually twice as good, with average true capacity in the 2-4 Mbps range
- We only need 3-4 pages to get the optimal bandwidth
- The sibling hyper thread case is an outlier, significant impact on the averages.
- Huge difference between the topology-unaware and the topology-aware channel (NUMA-M-Core-AV-Addr-Best-Addr).

Covert channel Results

Statistics for 1× Zen5 (Amd, 2024)

Attacker Model	Primitive	Error rate	Bandwidth (Mbps)	True Capacity (Mbps)
Topology-Unaware	FR	0.56	2.30	2.23
	FF	24.58	2.22	0.45
	L/F+R	0.04	4.89	4.86
NUMA-M-Core-AV-Addr-Best-Addr	FR	1.12	2.29	2.13
	FF	19.73	2.21	0.64
	L/F+R	0.05	4.89	4.07
NUMA-M-Core-AV-Addr-Best	FR	0	2.27	2.27
	FF	0.67	2.07	1.95
	L/F+R	0	4.98	4.98

Further work

What can we do from here

- Improve the thresholding algorithm (maximise the error margin around it)
- Rauscher et al., 2025, introduce metrics to evaluate single threshold channels. These would need to be adapted for topology aware channels.
- Topology-awareness on other ISA
  (e.g. Evict+Reload, or Flush-based attack on RISC-V, extension Zicbom)
- AMD address-dependent result suggest a non-trivial cache structure, which has yet to be reverse-engineered.

Thank you for your attention

	Intel Single-Socket Client	Intel Single-Socket Server	AMD Single-Socket	Intel Multi-socket	AMD Multi-Socket
Flush+Reload	Works	Works only with topology awareness	Works only with topology awareness + `rdpru`	Numa-aware works	Numa-aware works
Flush+Flush	Works well	Works well	Works well with topology awareness + `rdpru`	Numa-aware works well	Numa-aware works well

- Topology-awareness is pretty much needed on modern systems
- Don't let the kernel Numa rebalance you
- Flush+Flush is more than an odd variant
- A new optimised covert channel in need of a catchy name.

Questions ?

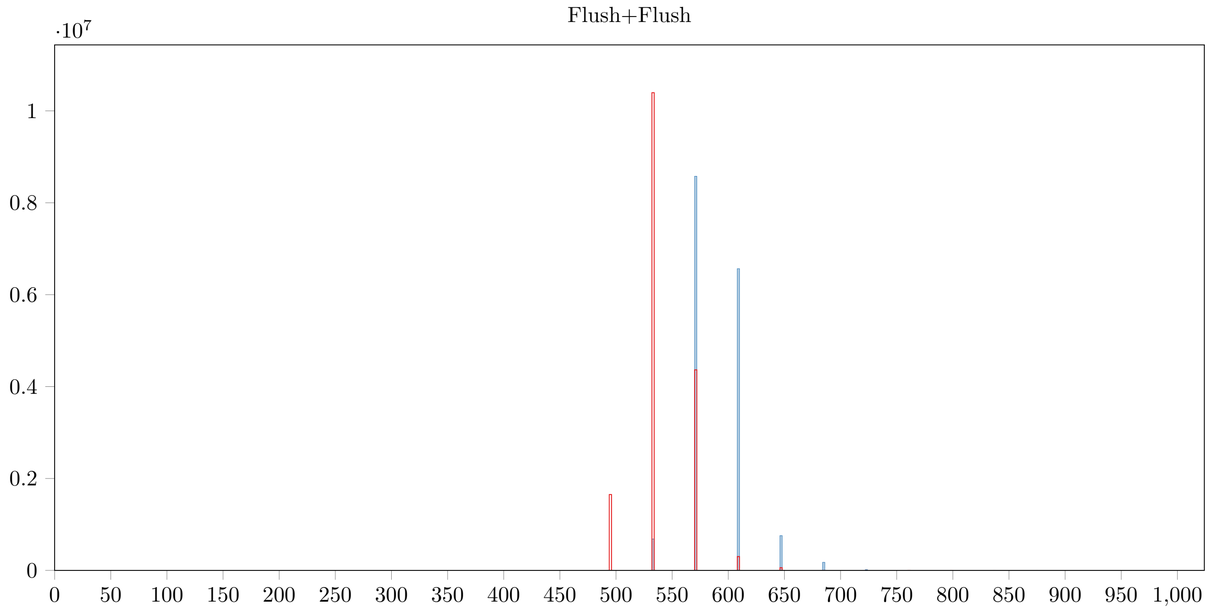
I can show you the plots

Various images and icons by flaticon, freepik et al. https://www.flaticon.com/free-icons/

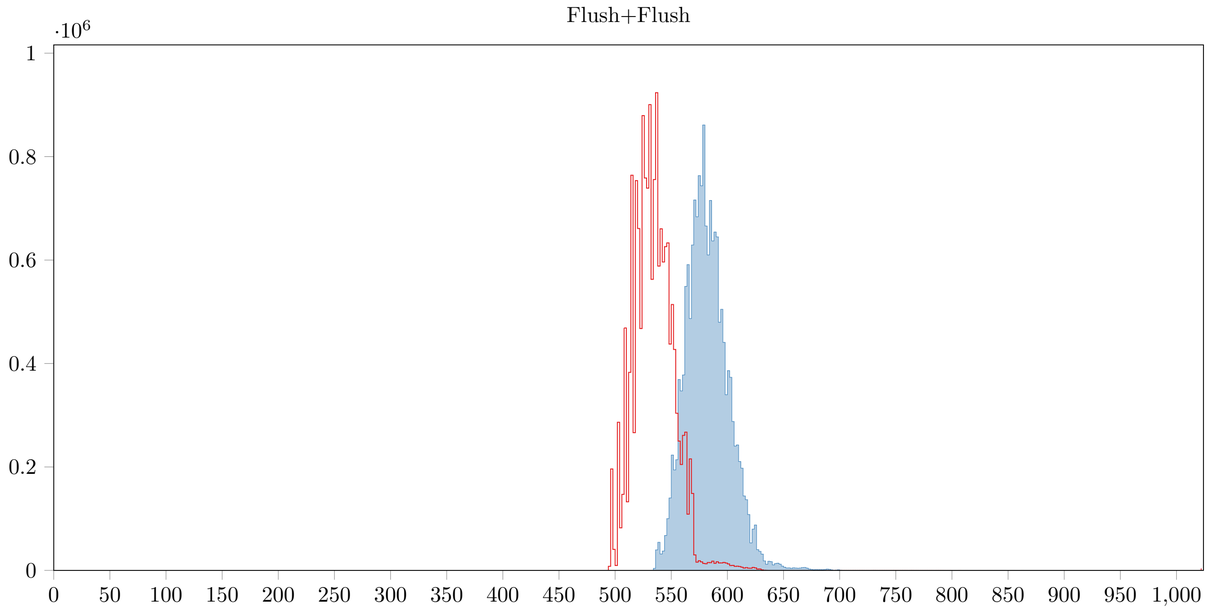
rdtsc vs rdpru

What's rdpru, how does it make a difference

- AMD custom instruction, gives access to some MSR that can be used to compute frequency
- One of them is an accurate cycle counter
- Used by the SQUIP paper, amongst others



rdtsc



rdpru