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

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

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)"

Motivation

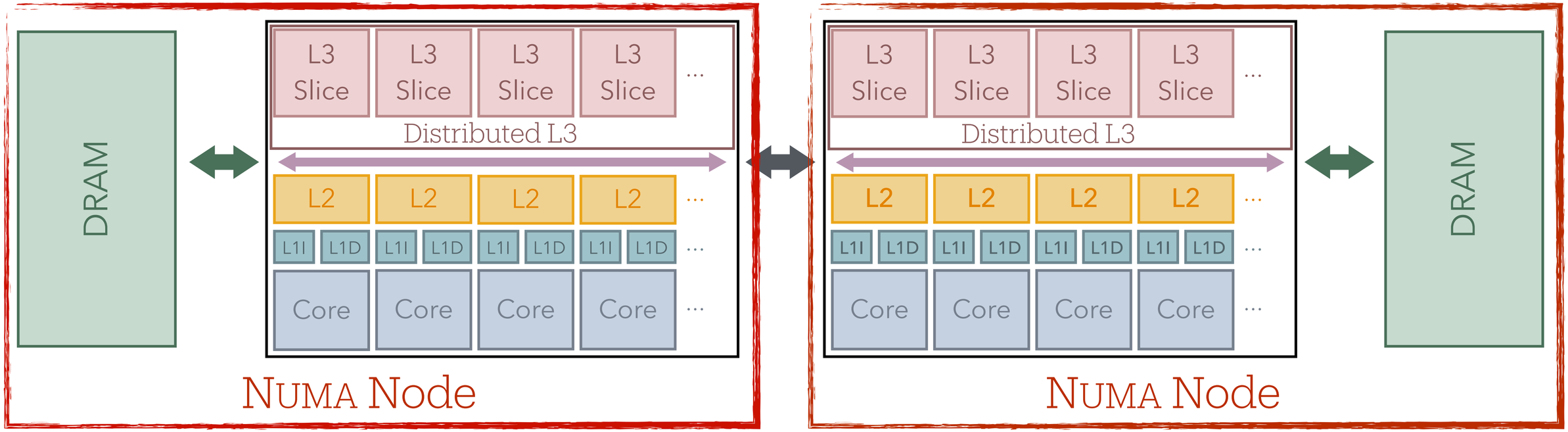
The limits of the state of the art

- 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

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: `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

The *Flush+Reload* attack

- *Flush+Reload* reaches all sockets (coherence domain)
- Requires *shared* memory? *Read-only* shared memory is *realistic*:
  (shared library .TEXT / .RODATA, hypervisor memory deduplication, transient execution attacks)

Bonus: *Flush+Flush*

- `clflush`'s execution time depends on the *cache coherence state*
  (at least on Intel CPUs)
- We can remove the load and time the *flush* instead.

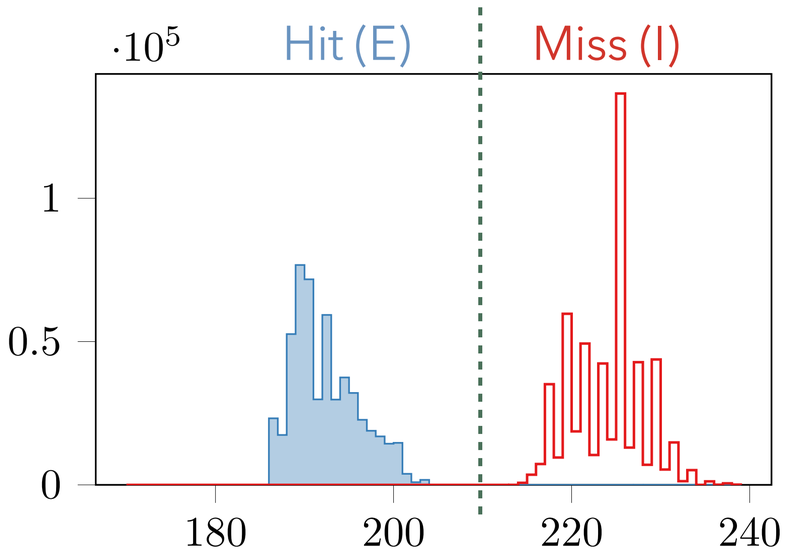
Background: Calibration

Finding the best threshold(s)

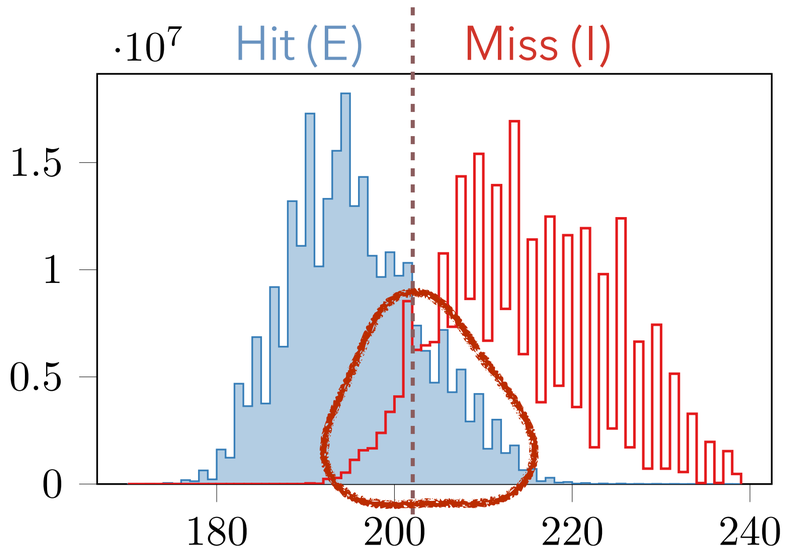
1. Measure execution times for all topology configs *(attacker, victim, target)*

2. Build histograms & Find Threshold(s)

Good



Bad

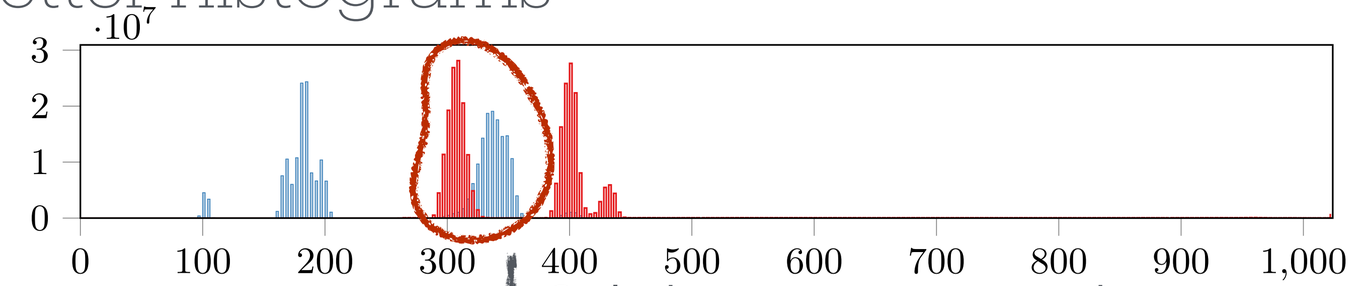


Guillaume Didier, Clémentine Maurice: *Calibration Done Right: Noiseless Flush+Flush Attacks*. DIMVA 2021

Topology-awareness

How to get better histograms

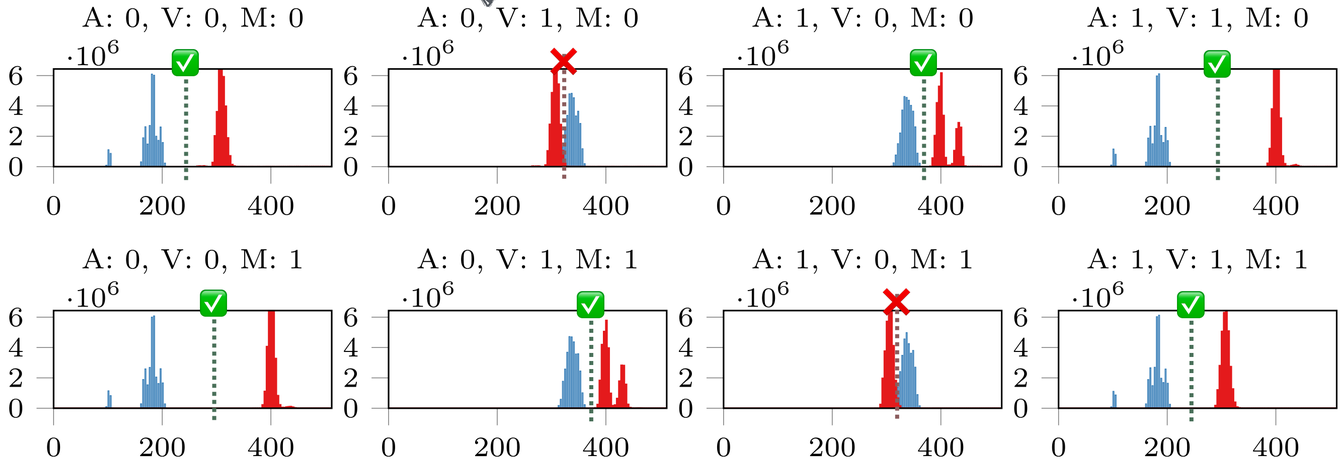
Topology-unaware



25% average error rate

Split histograms according to topology

NUMA-aware



2,8% average error rate

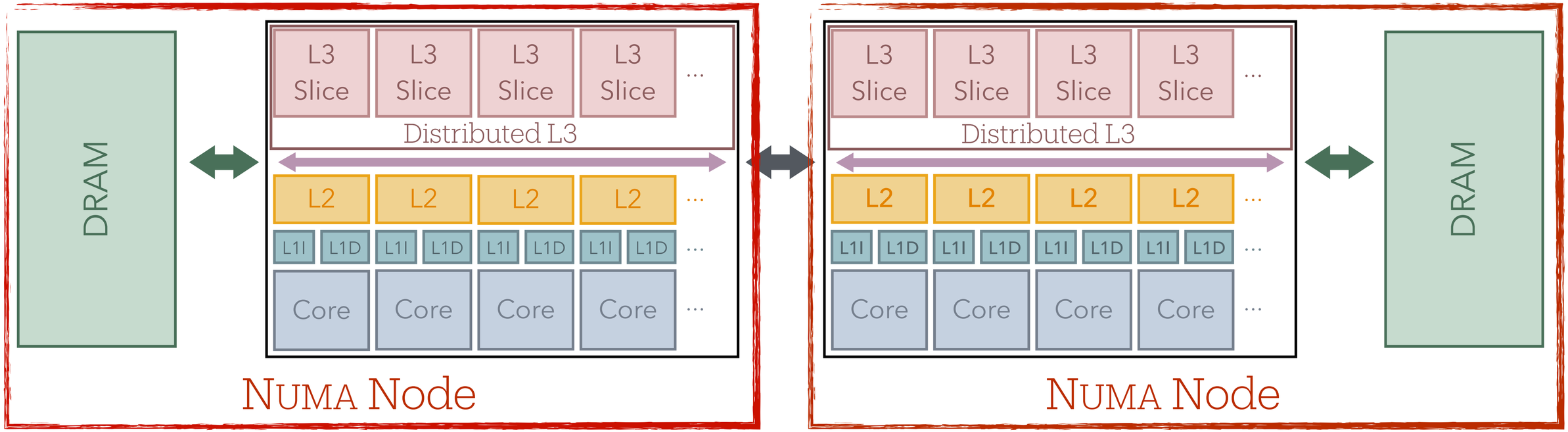
Fig. 2: Flush+Reload Histograms on a 2×Haswell EP system (2015). Hits are in blue, filled; misses are in thick outline red.

Know the configuration, use the right threshold.

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

Experiments

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)

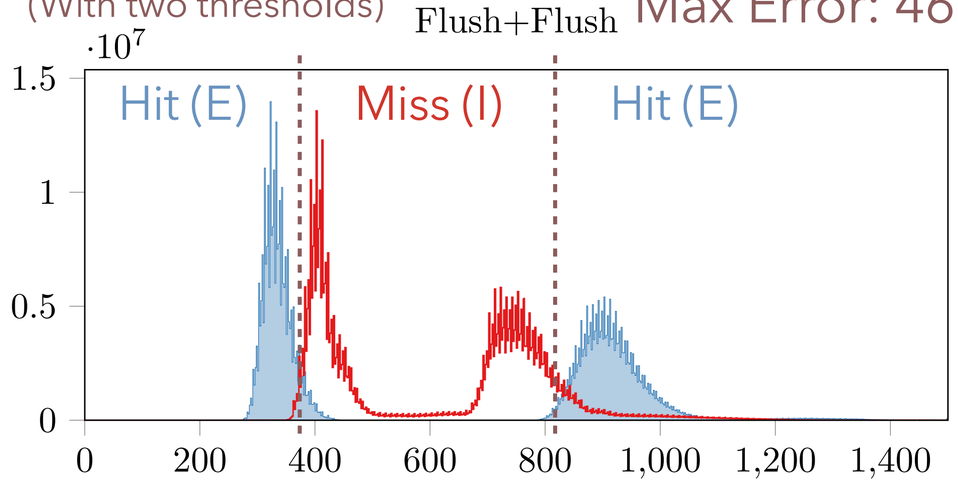
Preliminary Analysis

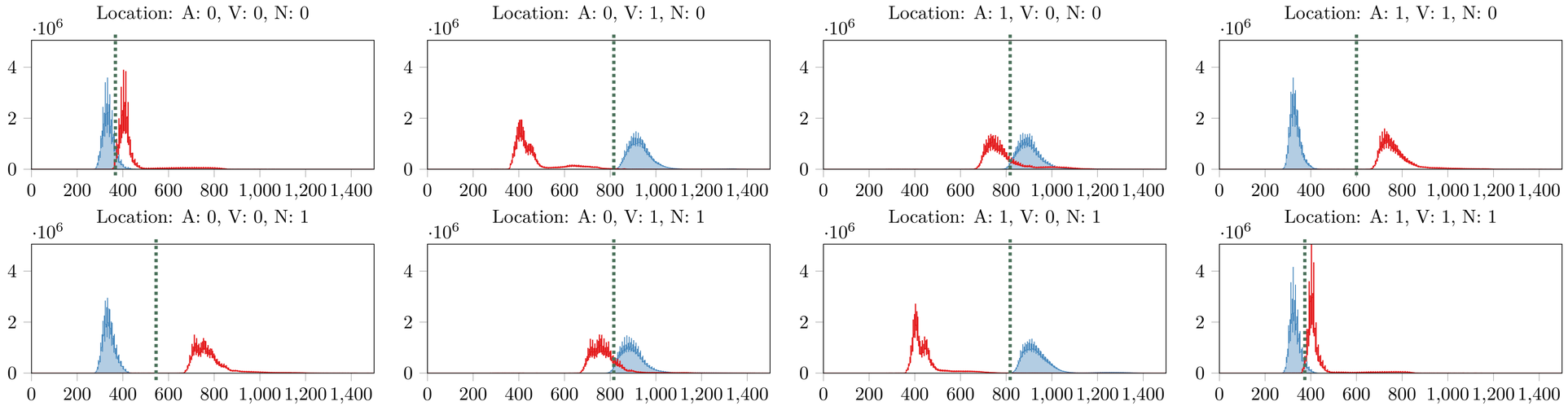
NUMA-awareness only

1. Find *global threshold* (also try a pair of thresholds)
2. Split histogram on NUMA (usually 2×2×2 histograms)
3. Evaluate the *global threshold*
4. Find and evaluate *local thresholds*

Average Error: 34.50%
(With two thresholds)

Min Error: 19.34%
Max Error: 46.79%





Average Error: 4.75%
(With 8× one threshold)

Min Error: <0.01%
Max Error: 14.36%

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

Preliminary results

Intel — Single-socket (topology-unaware)

Error Rates (%)	Arrow Lake (2024, client)	Emerald Rapids (2024, server)
Flush+Reload	2.45	Error: 33.85 (With two thresholds)
Flush+Flush	0.02	0.04

Preliminary results

Intel — Multi-socket: 2×Sapphire Rapids (2023)

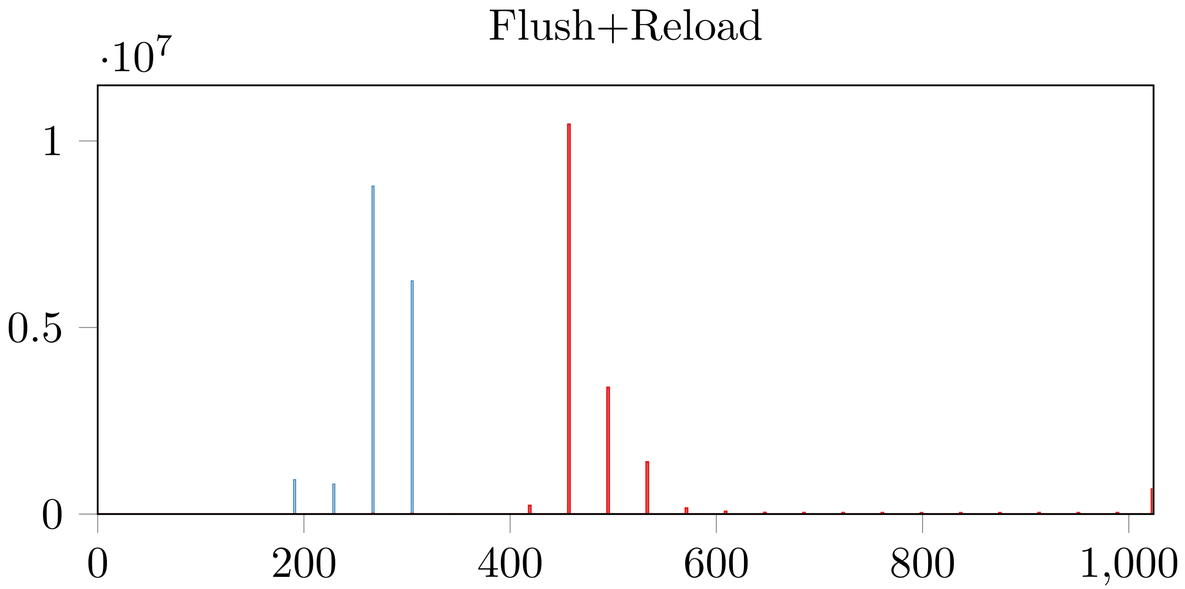
Error rates (%)		Topology-unaware Single-threshold	Topology-unaware Dual-threshold	NUMA-aware (Single-threshol
Flush+Reload	Min	6.82	19.34	1.56
	Average	**37.67**	**34.50**	**19.34**
	Max	63.85	46.79	44.40
Flush+Flush	Min	2.62	1.50	<0.01
	Average	**27.41**	**8.12**	**4.75**
	Max	51.23	14.85	14.36

See our appendix for numbers on 20 other Intel machines of different generations

Preliminary results

AMD — Single socket (topology-unaware)

Error rates (%)	Zen+ (2018)	Zen2 (2019)	Zen3 (2020)	Zen4 (2022)	Zen5 (2024)
Flush+Reload	28.82	26.56	0.01	<0.01	3.05
Flush+Flush	9.27	20.78	10.92	10.11	16.24



On AMD, `rdtsc`, seems to have a lower precision (time stamp counter)

Preliminary results

AMD — Multi-socket: 2× Zen4 (2022)

Error rates (%)		Topology-unaware Single-threshold	Topology-unaware Dual-threshold	NUMA-aware (Single-threshold)
Flush+Reload	Min	4.53	4.53	4.13
	Average	**22.99**	**16.74**	**14.73**
	Max	55.14	42.64	22.26
Flush+Flush	Min	0.35	0.35	0.33
	Average	**13.17**	**13.17**	**0.83**
	Max	49.99	49.99	1.62

See our appendix for numbers 3 other AMD machines of different generations

Conclusions

We can already say a few things

	Intel Single-Socket		AMD	Intel	AMD
	Client	Server	Single-Socket	Multi-socket	Multi-Socket
Flush+Reload	Works	Works poorly	Works Hit-or-miss	Numa-aware works	Numa-aware works
Flush+Flush	Works well	Works well	Works Hit-or-miss	Numa-aware works well	Numa-aware works well

Planned work

Why is this a work-in-progress ?

- Analyse **beyond socket/NUMA** node: **core**, **target** line **address**.
- Several **attacker models**
- How **Fast** is it ? (Covert channel benchmark)

Thank you for your attention

Questions ?

I can show you the plots

	Intel Single-Socket		AMD	Intel	AMD
	Client	Server	Single-Socket	Multi-socket	Multi-Socket
Flush+Reload	Works	Works poorly	Works Hit-or-miss	Numa-aware works	Numa-aware works
Flush+Flush	Works well	Works well	Works Hit-or-miss	Numa-aware works well	Numa-aware works well

- Flush+Flush works on AMD, not as good as Flush+Reload
- Flush+Flush better than Flush+Reload on recent Intel, and all multi-socket systems
- NUMA-awareness improves attack a lot.
- Stay tuned for the fine-grained topology-awareness