`clflush`-based Attacks on Modern Cloud Servers

ESORICS 2026
14/09/2026

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)* 🇫🇷

Jasper QUIRK, Ruhr-Universität Bochum 🇩🇪

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

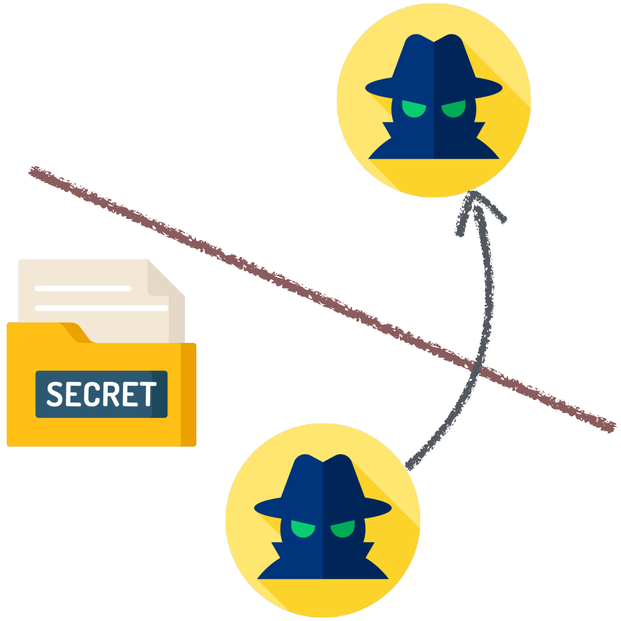
This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101020415) and the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972

Background: Use cases for attacks

What sort of evil can we do

1. Side-channel

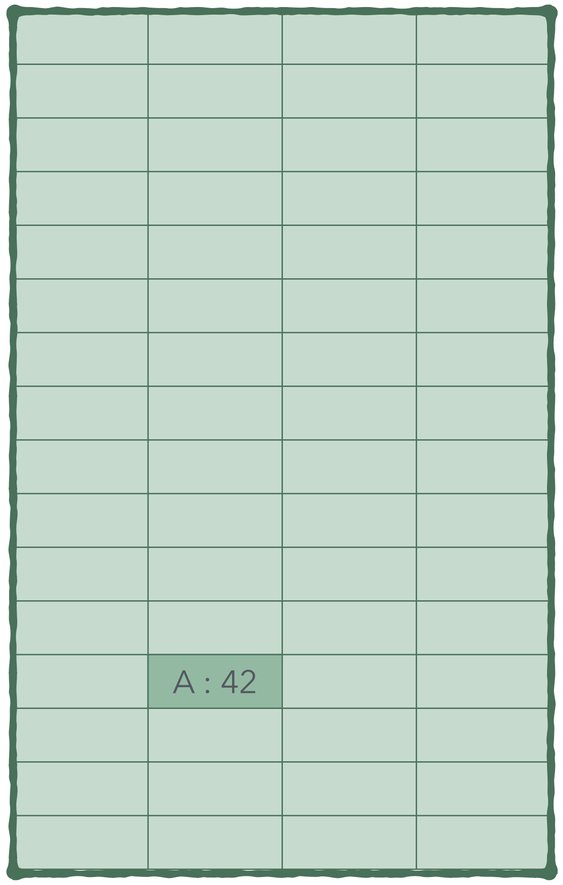
2. Covert-channel



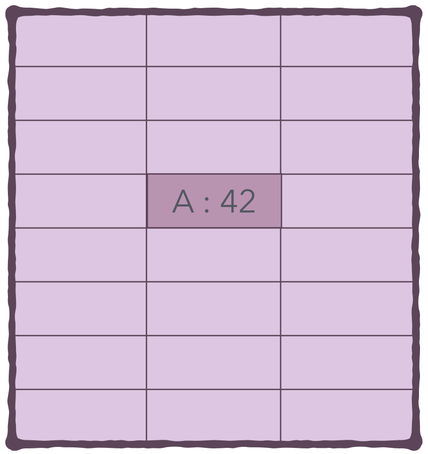
Background: Flush+Reload intuition

The basic idea

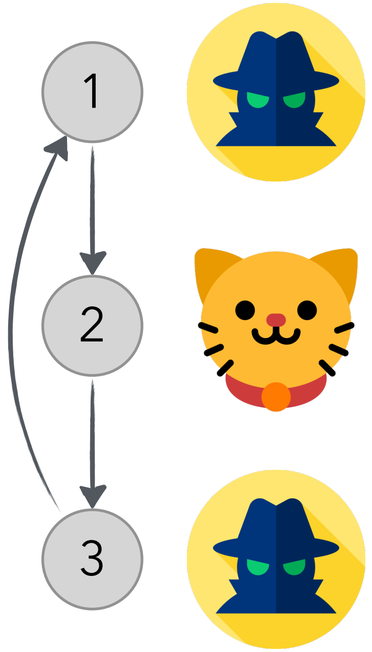
Requirement: Shared (RO) Memory



Main Memory



Cache



Set-up: Attacker Flush
`clflush` is unprivileged

Maybe load

Timed-reload

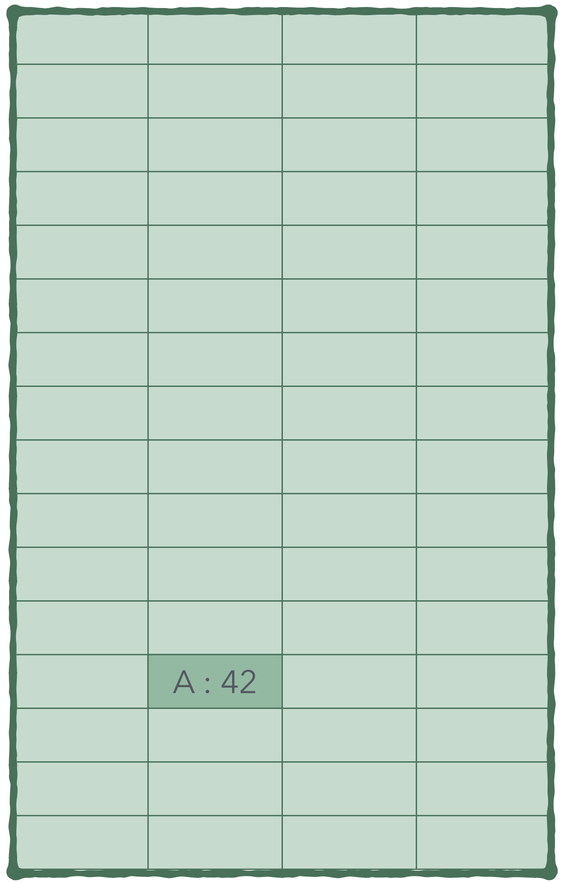
A : 42 Load DRAM: Slow

A : 42 Load Cache: Fast

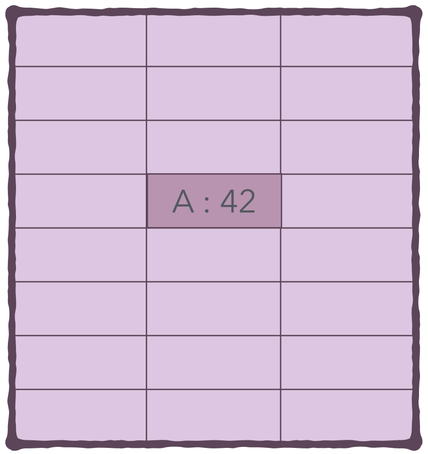
Background: Flush+~~Reload~~ Flush intuition

The basic idea

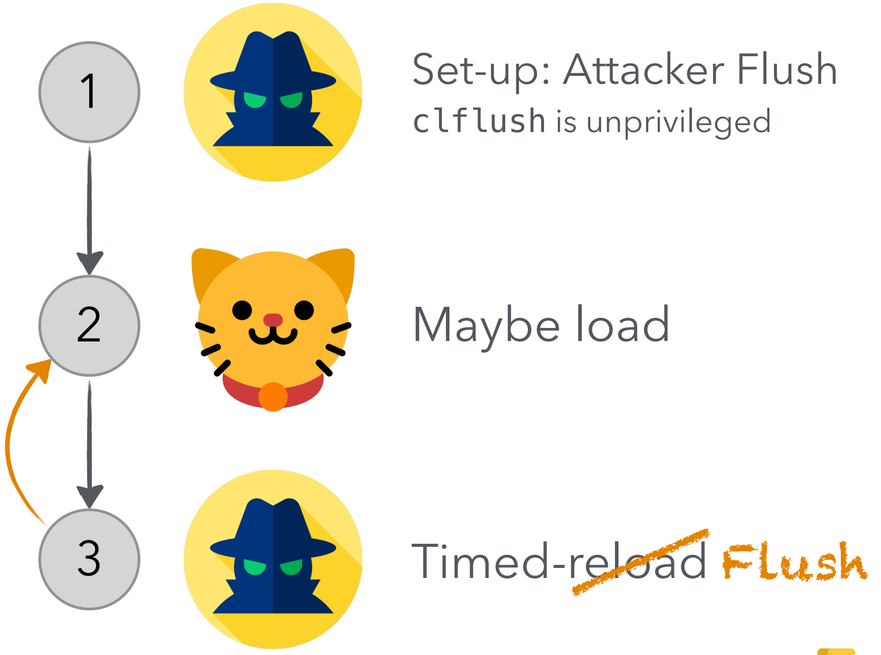
Requirement: Shared (RO) Memory



Main Memory



Cache



Set-up: Attacker Flush
`clflush` is unprivileged

Maybe load

Timed-~~reload~~ Flush

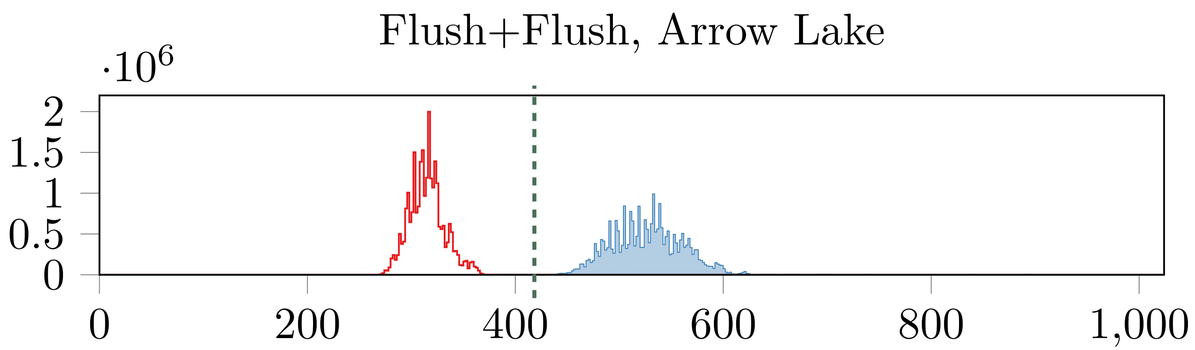
A : 42 ~~Load~~ Flush DRAM: ~~Slow~~ Fast

A : 42 ~~Load~~ Flush Cache: ~~Fast~~ Slow

Motivation

- Flush+Reload and Flush+Flush are great attacks on x86
- But they are a decade old
- Modern systems are more complex

Hits are in blue, filled; misses are in thick outline red.

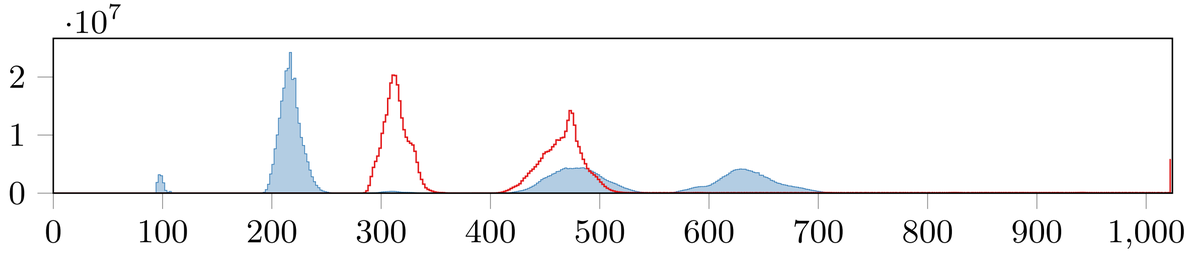


Histogram of `clflush` execution time

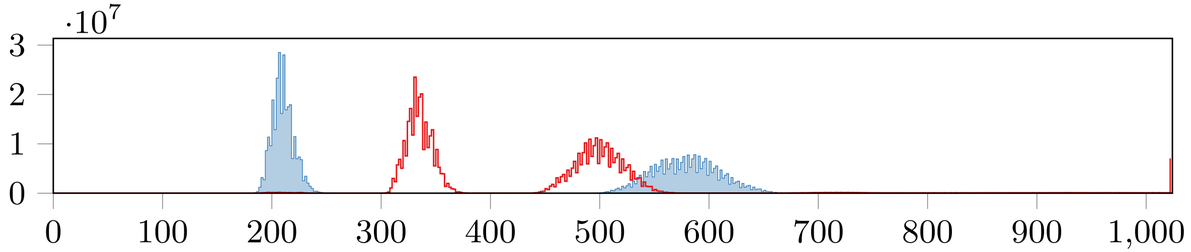
Motivation

- Flush+Reload and Flush+Flush are great attacks on x86
- But they are a decade old
- Modern systems are more complex
- Do they work cross-socket?

Hits are in blue, filled; misses are in thick outline red.



(a) Histogram of reload execution time (in rdtsc unit)



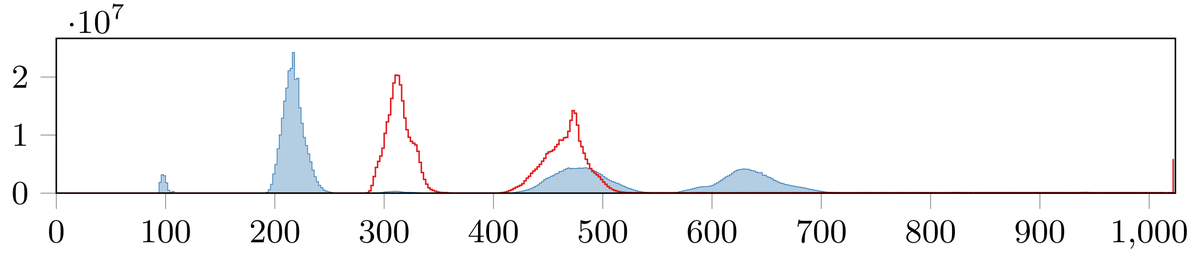
(b) Histogram of `clflush` execution time (in rdtsc unit)

Histograms on a 2×Cascade Lake X system.

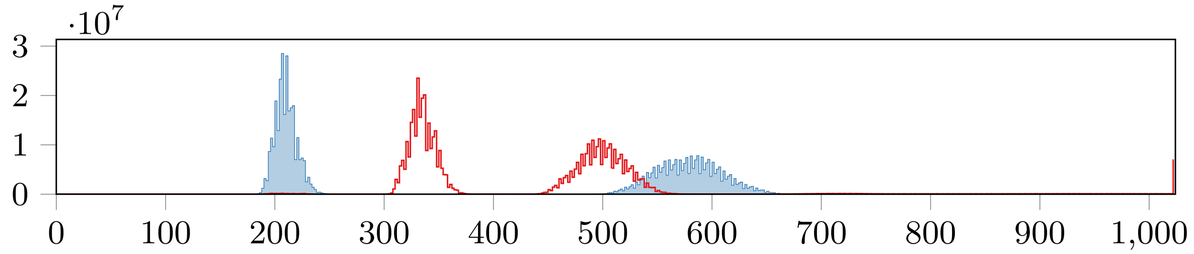
Motivation

- Flush+Reload and Flush+Flush are great attacks on x86
- But they are a decade old
- Modern systems are more complex
- Do they work cross-socket?

Hits are in blue, filled; misses are in thick outline red.



(a) Histogram of reload execution time (in rdtsc unit)

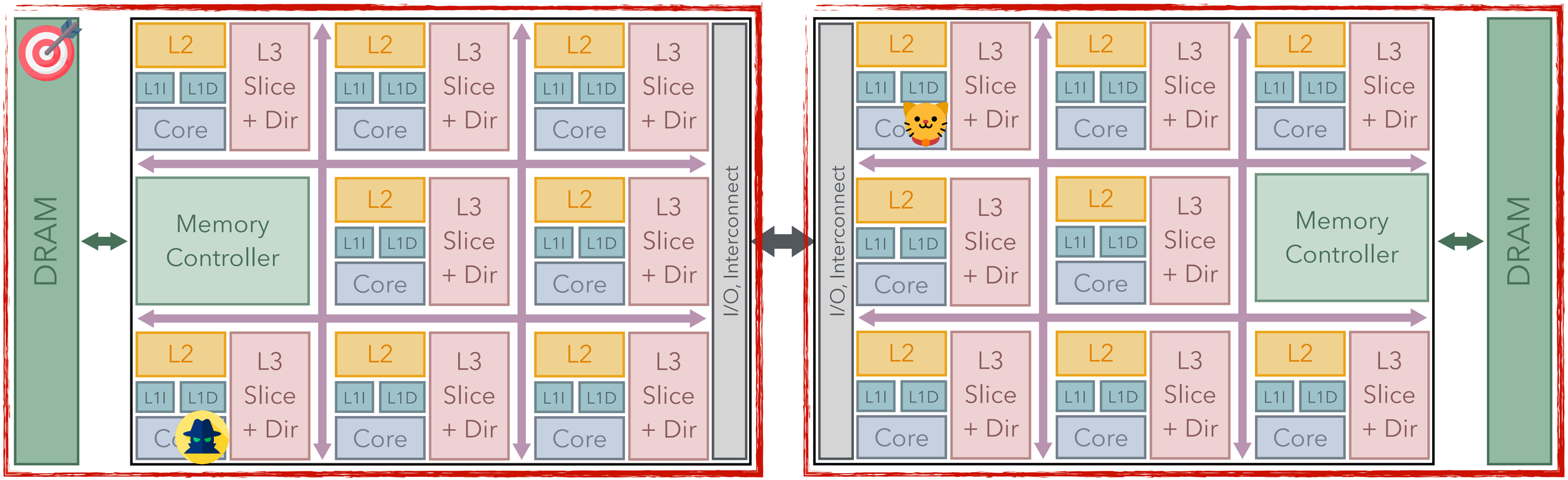


(b) Histogram of `clflush` execution time (in rdtsc unit)

Histograms on a 2×Cascade Lake X system.

	Intel Single-Socket	AMD Single-Socket	Intel Multi-Socket	AMD Multi-Socket
Flush+Reload	works	works	unknown	unknown
Flush+Flush	works until Skylake (2018) unknown otherwise	**works (Zen 2)**	unknown	unknown

Background: Modern NUMA x86 systems

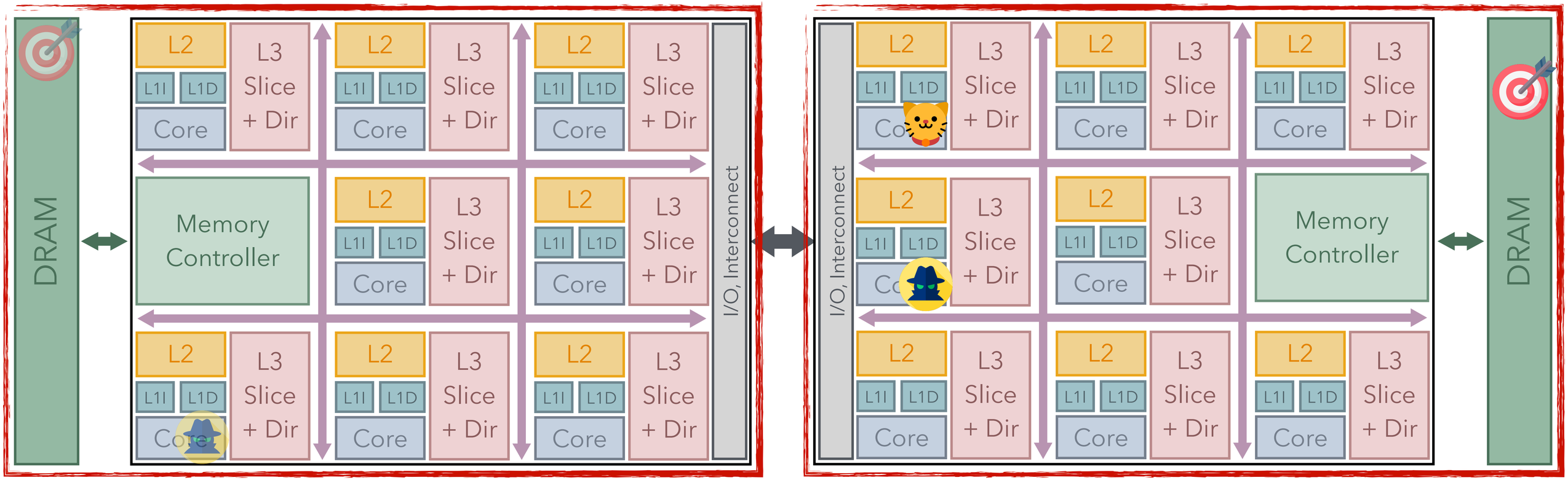


NUMA Node (Non-Uniform Memory Access) NUMA Node

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

Per-socket directories, and cross-socket coherence protocols

Background: Modern NUMA x86 systems

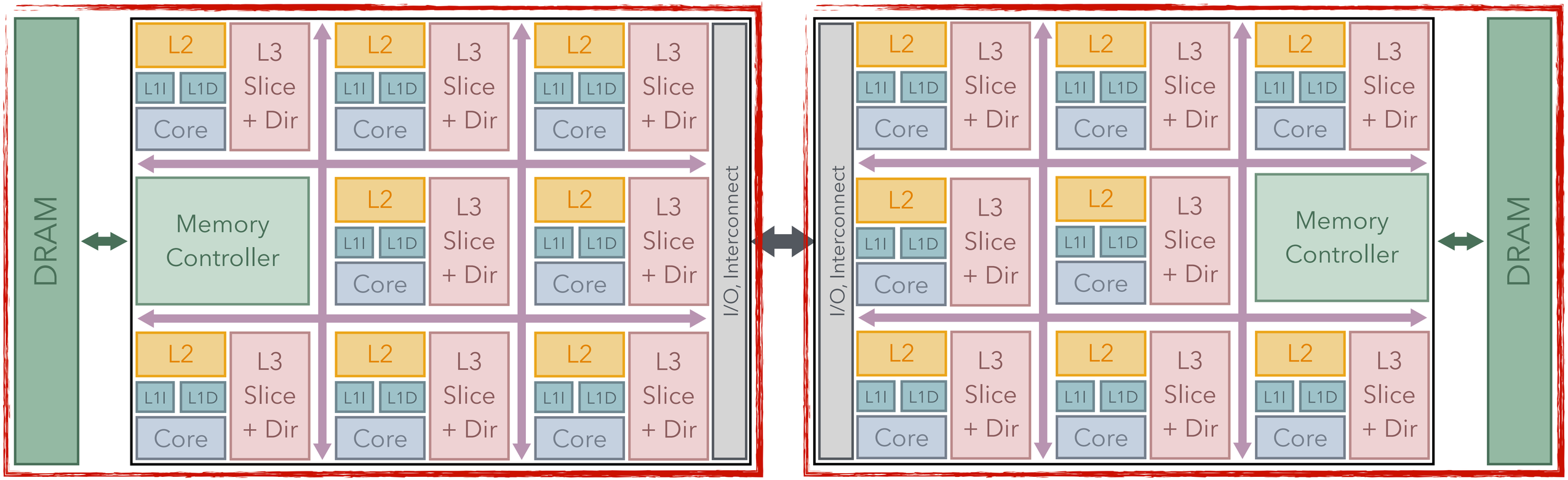


NUMA Node (Non-Uniform Memory Access) NUMA Node

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

Per-socket directories, and cross-socket coherence protocols

Cross-socket attacks



NUMA Node

NUMA Node

`clflush` can reach across socket (eviction sets cannot)

Contributions

A broad survey

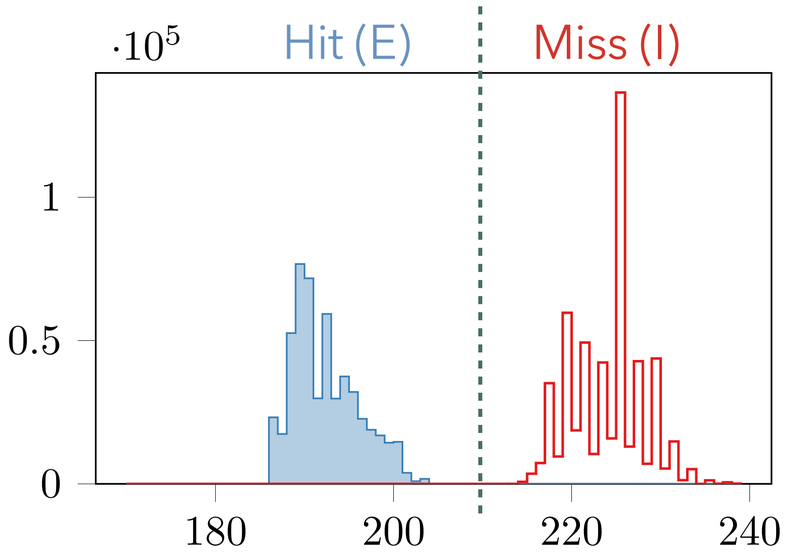
- Current state of Flush+Reload and Flush+Flush ?
- Error predictions and calibration data
- Covert Channel Performance
- A new Covert Channel, Load/Flush+Reload

Background: Calibration

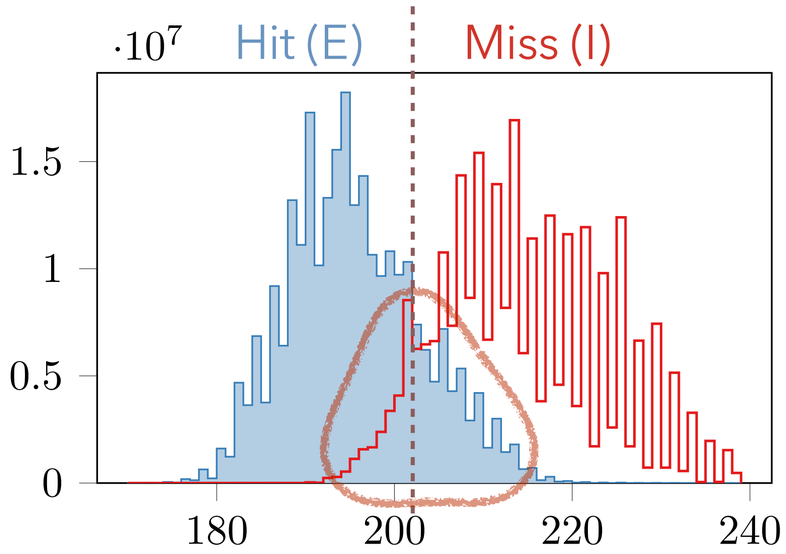
Finding the best threshold to distinguish between cache states

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

Good



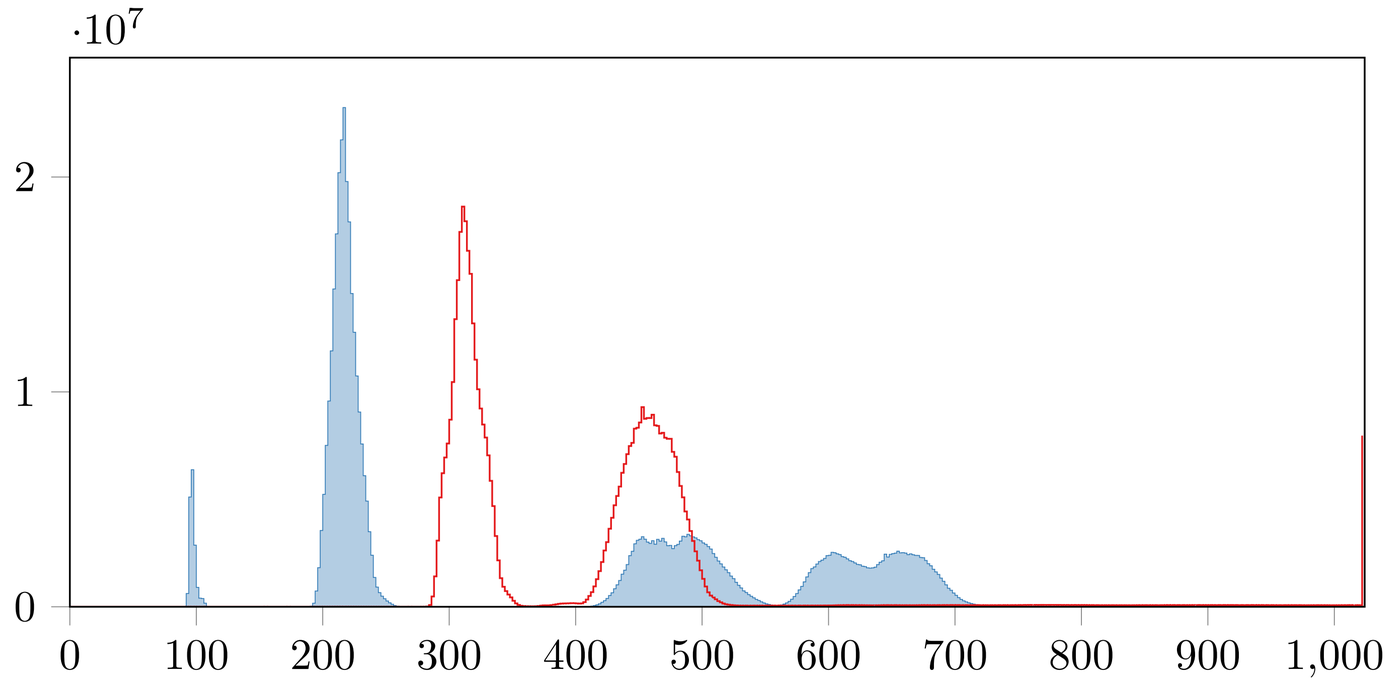
Bad



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

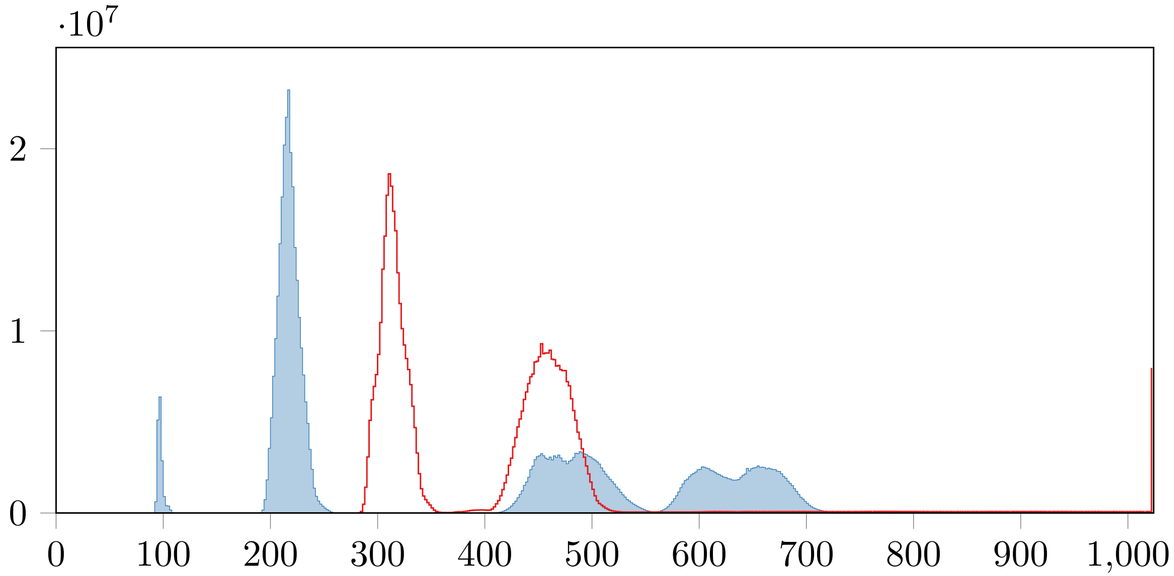
Topology-unaware attack

Weigh equally all Attacker × Victim × Target Locations



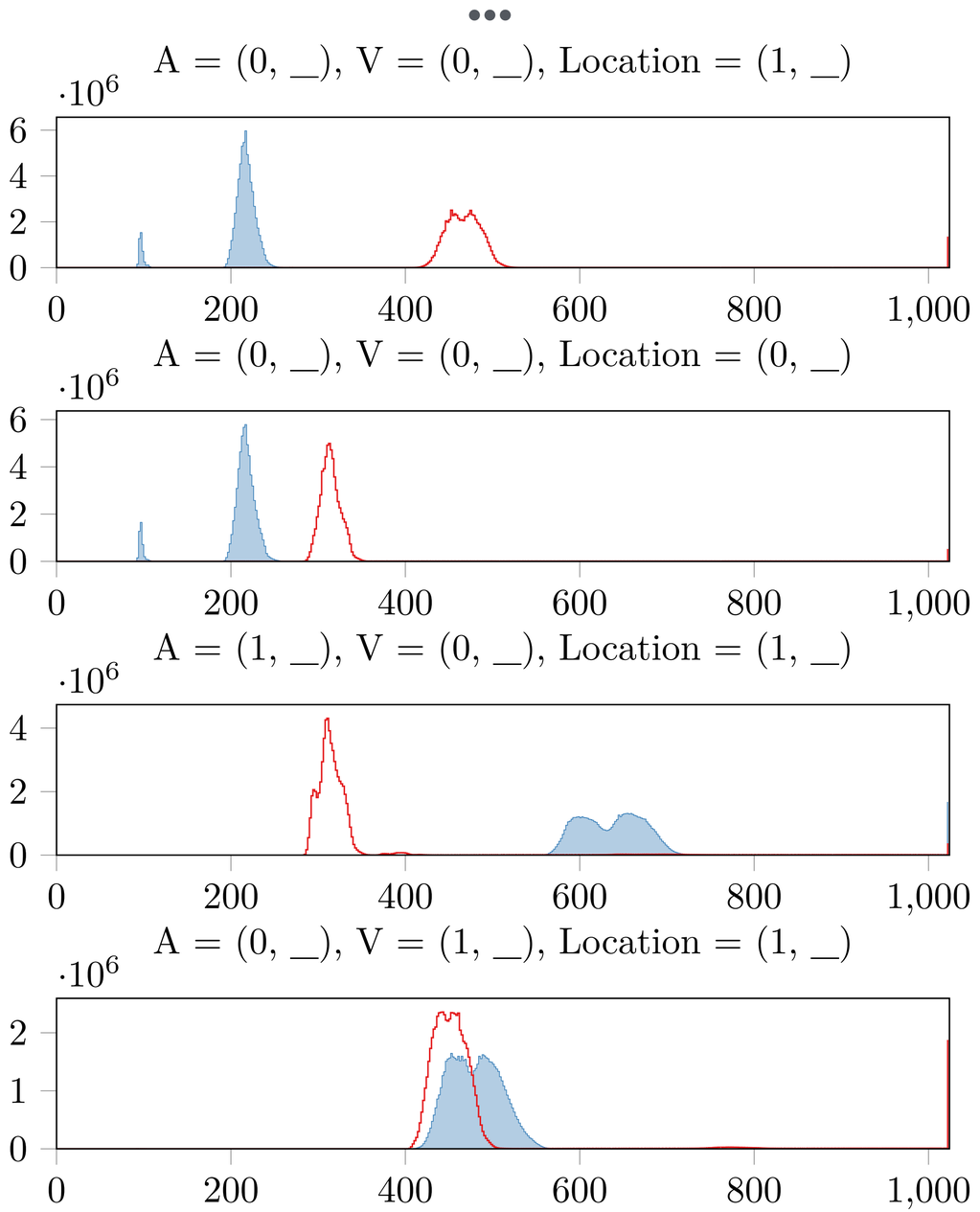
Global Reload Histograms on a 2×Cascade Lake X system. Hits are in blue, filled; misses are in thick outline red.

NUMA-aware attacker



Global histogram (all A×V×L)

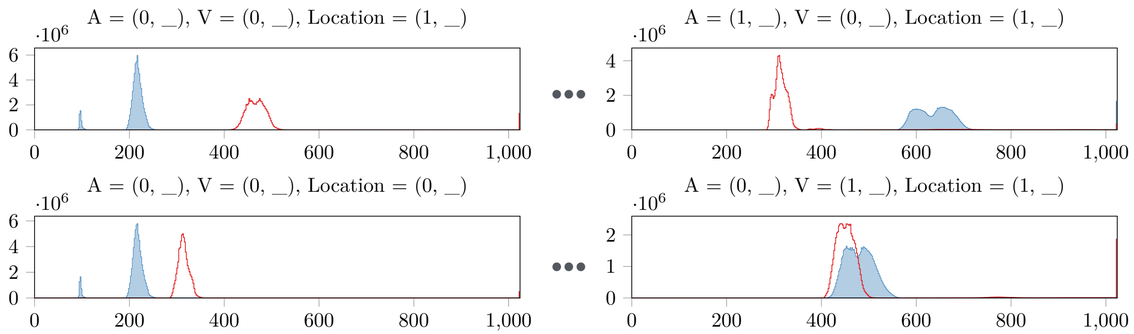
Split by NUMA nodes



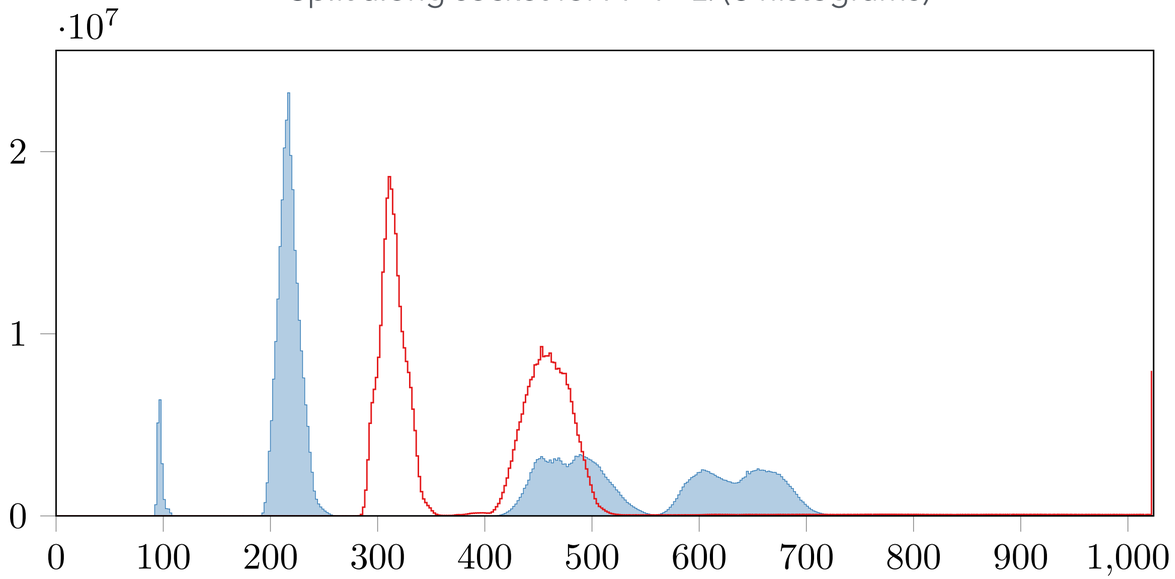
Split along socket for A×V×L. (8 histograms)

Reload Histograms on a 2×Cascade Lake X system. Hits are in blue, filled; misses are in thick outline red.

Topology-aware Attacker

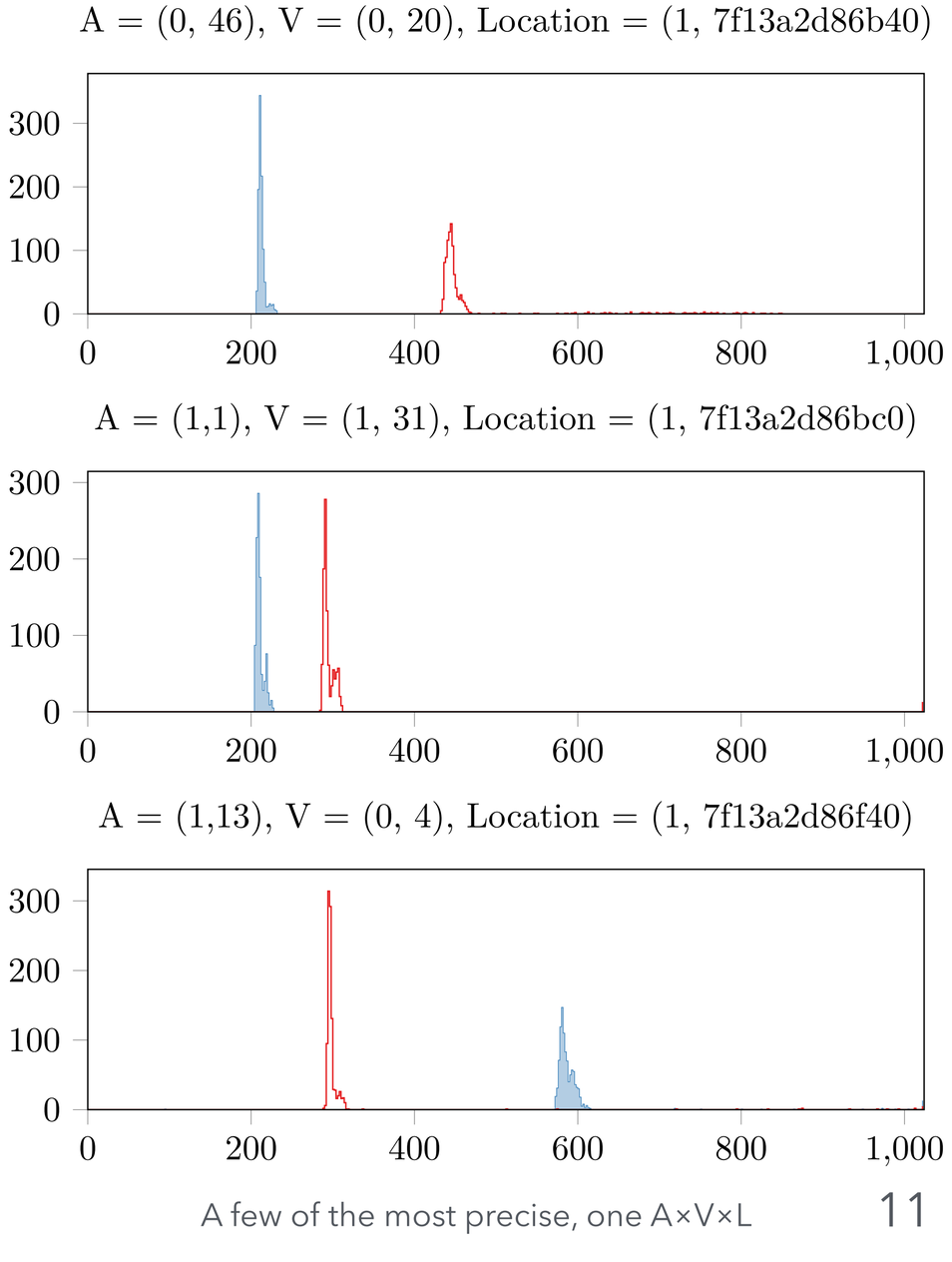


Split along socket for A×V×L. (8 histograms)



Global histogram (all A×V×L)

Further split by A, V cores and M addresses

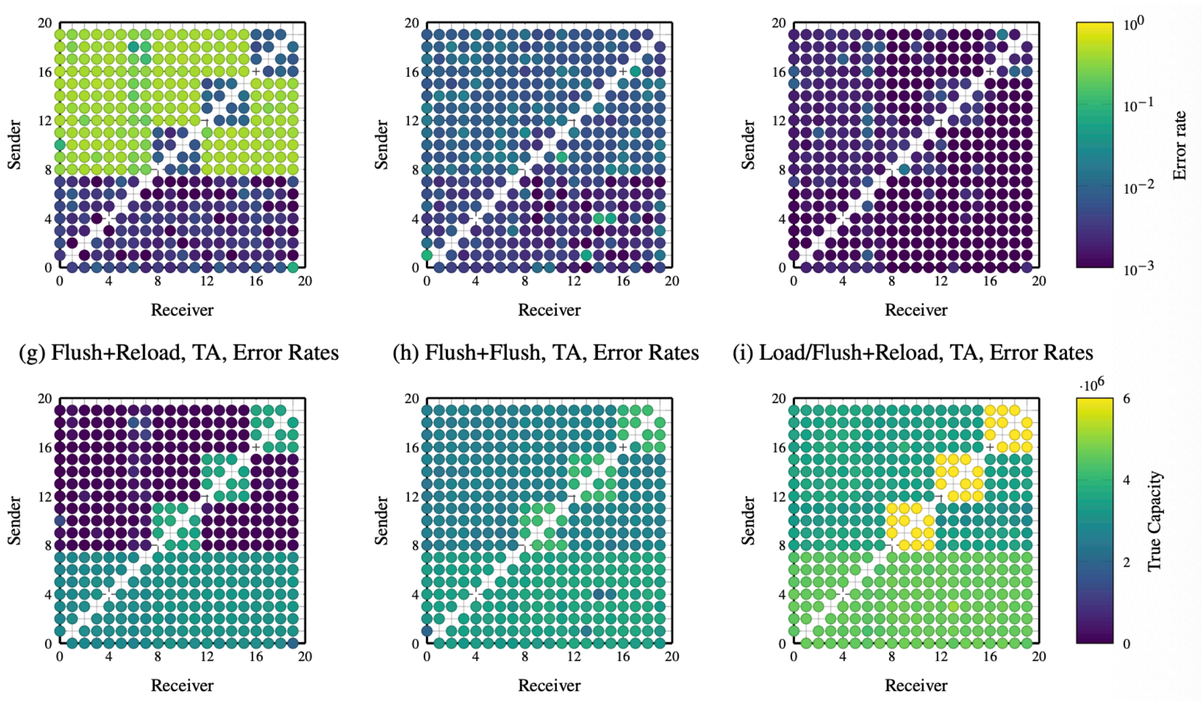


A few of the most precise, one A×V×L

Reload Histograms on a 2×Cascade Lake X system. Hits are in blue, filled; misses are in thick outline red.

Covert Channel Benchmark Results

A few examples



(g) Flush+Reload, TA, Error Rates (h) Flush+Flush, TA, Error Rates (i) Load/Flush+Reload, TA, Error Rates

(j) Flush+Reload, TA, True Cap. (k) Flush+Flush, TA, True Cap. (l) Load/Flush+Reload, TA, True Cap.

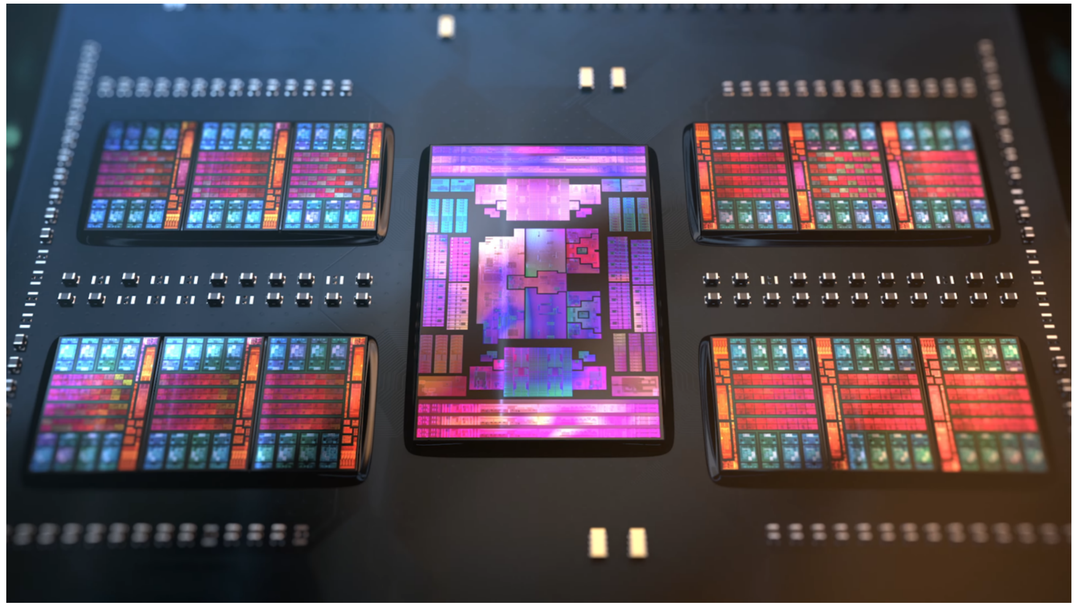
Figure 4: Core to core covert channel error rates (log. scale) and true capacity for ARL, variable frequency

Die shot from https://www.techpowerup.com/img/P3NjCzBEYcfZk5Fe.jpg

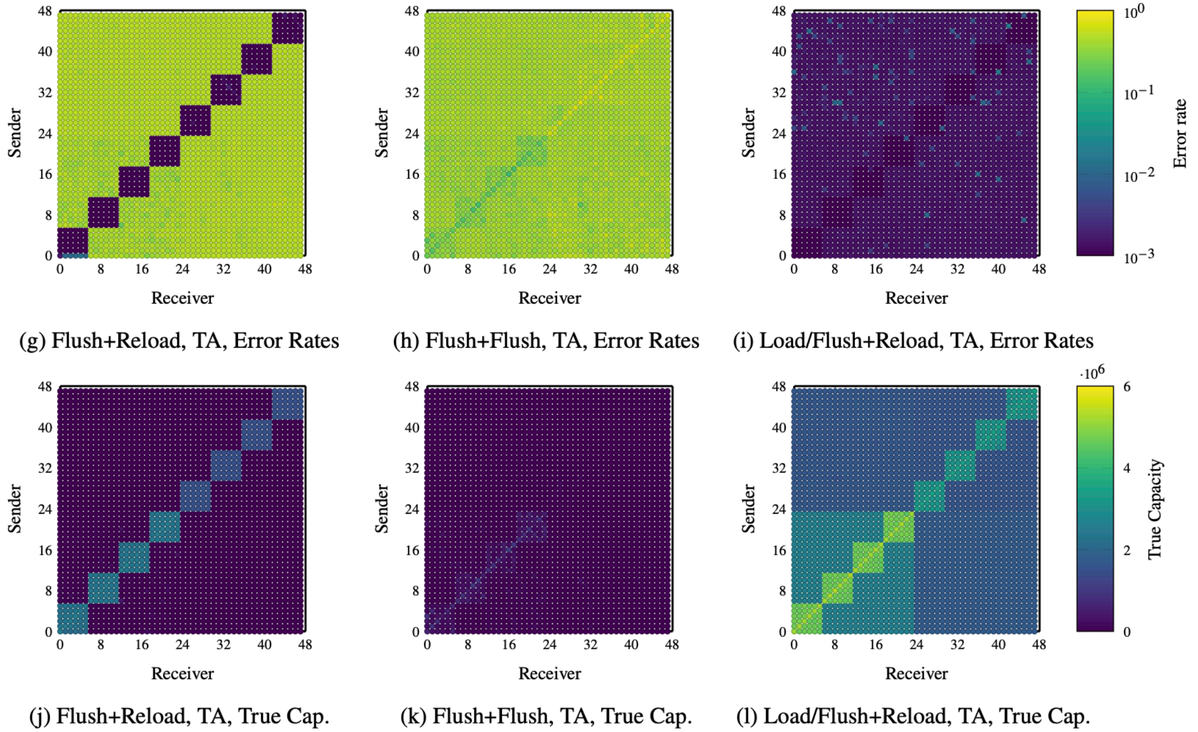
1×Arrow Lake (8P + 12E)

Covert Channel Benchmark Results

A few examples



Die shot from https://wccftech.com/amd-epyc-genoa-zen-4-cpu-pictured-in-all-its-glory-12-ccds-featuring-up-to-96-cores-massive-iod/



(g) Flush+Reload, TA, Error Rates (h) Flush+Flush, TA, Error Rates (i) Load/Flush+Reload, TA, Error Rates

(j) Flush+Reload, TA, True Cap. (k) Flush+Flush, TA, True Cap. (l) Load/Flush+Reload, TA, True Cap.

Figure 103: Core to core covert channel error rates (log. scale) and true capacity for `musa`, variable frequency

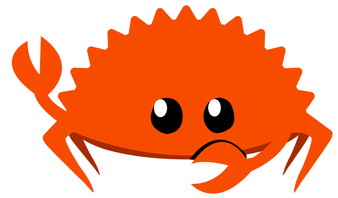
2×Zen4 EPYC

Results

A large data set

- 36 machines
  - 25 two-socket servers
  - 11 single-socket machines
- Artifact containing:
  - Appendix with each machine's:
    - Calibration histograms
    - Error prediction for all attacker model
    - Covert channel error rate and capacity heat maps.
  - Code
  - Raw data File

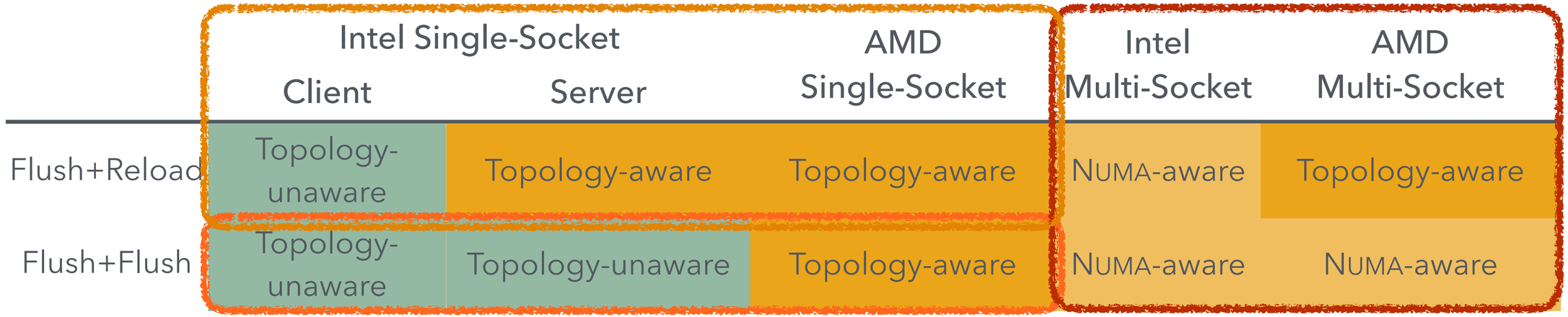
https://doi.org/10.5281/zenodo.17485252



Summary

	Intel Single-Socket		AMD Single-Socket	Intel Multi-Socket	AMD Multi-Socket
	Client	Server			
Flush+Reload	Works	Works	Works	unknown	unknown
Flush+Flush	works until Skylake (2018)	unknown	unknown	unknown	unknown

Summary



- Topology has a large impact nowadays
- Flush+Flush often better than Flush+Reload
- A new optimised covert channel: Load/Flush+Reload (LF+R)
- Don't let the kernel NUMA rebalance you

Future work

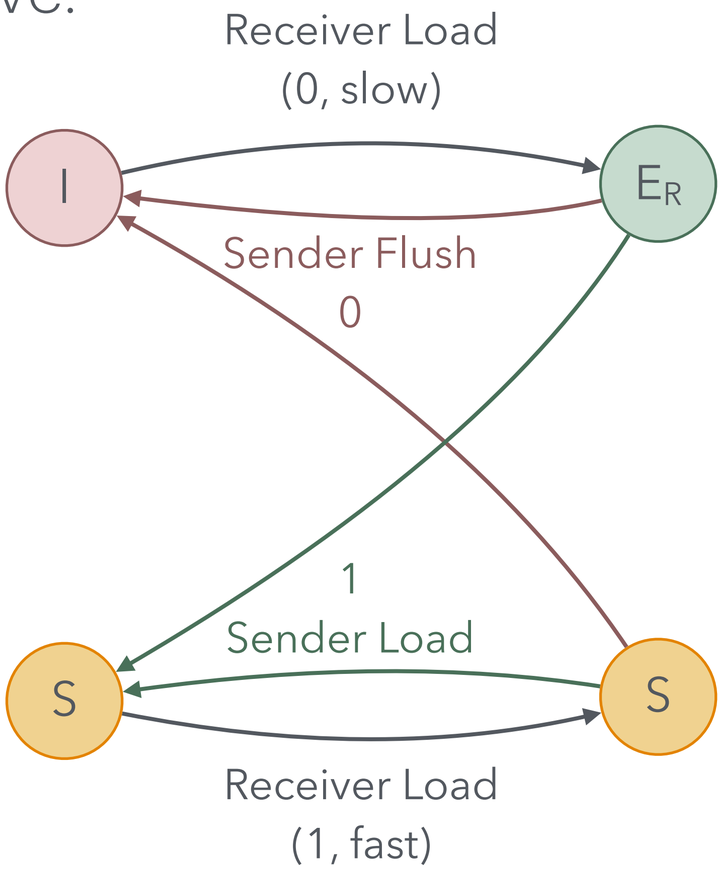
- Use the data to reverse-engineer?
- Prefetch instruction based attacks?
- End to end, DVFS robust attack?

Questions?

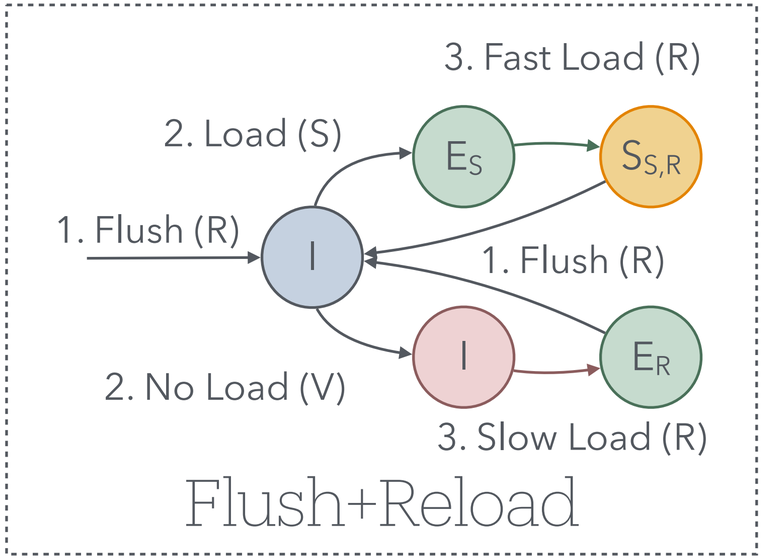
Various images and icons by flaticon, freepik et al. https://www.flaticon.com/free-icons/

Load/Flush+Reload (LF+R)

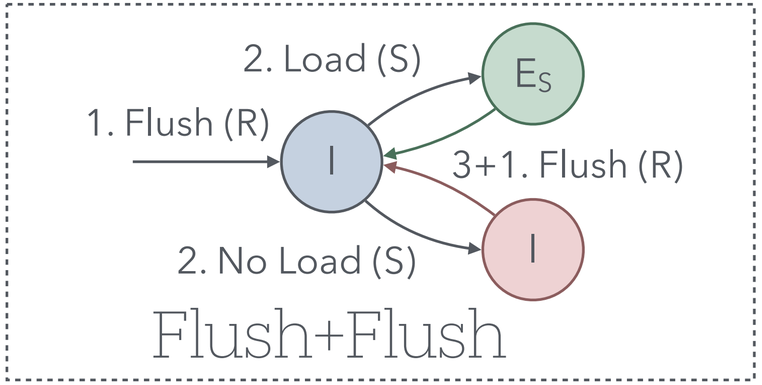
- New ***Covert Channel*** primitive:
- Compare:
  - *Shared Load*
  - *Invalid Load*
- Much better than Flush+Reload / Flush+Flush



Load/Flush+Reload (New)



Flush+Reload



Flush+Flush