Randomized Last-Level Caches Are Still Vulnerable to Cache Side-Channel Attacks! But We Can Fix It

Hyperthreads (**6**)
Sumon Nath
(21Q050007)

I. Summary

The authors show that state-of-the-art cache randomization techniques like CEASAR-S and ScatterCache make flawed assumptions which renders them vulnerable to conflict-based side channel attacks. They propose a few techniques as well as use some existing techniques to fix these vulnerabilities. The experimental results show that these new mitigation techniques does not have a large performance hit compared to the existing techniques. They also argue that randomized set-associative caches without any partitions are a reasonable choice for commercial systems as compared to their skewed counterparts by showing that they can be strengthened enough to mitigate the aforementioned attacks.

II. Details

A. Problem with existing techniques

- Small eviction sets with partially congruent addresses can be found (by algorithms like Conflict Testing) in a much lesser time window compared to the remap period of randomized caches.
- These small eviction sets have an eviction rate(probability of evicting the target address) of as low as 30%. But these evictions sets can lead to a high overall eviction rate (close to 90%) by using a prime-probe-flush mechanism repeatedly. This mechanism is proposed by the authors.
- Existing mitigation techniques ignore the facts that (1) eviction sets with partially congruent addresses can be used to launch effective attacks and (2) eviction sets with low eviction rate can be repeatedly flushed and re-accessed to get an overall high eviction rate.

B. Fixes proposed

- They propose to define the remap period of randomized caches in terms of LLC evictions rather than using LLC accesses. The reasons being (1) LLC accesses are filtered by higher-level cache accesses and (2) LLC evictions are much higher during an attack as compared to normal execution. Thus LLC evictions are a much better quantification for setting the remap period.
- They propose a mathematical model to estimate the number of evictions required to find a small eviction set(explained in section II.A) to set the remap period appropriately to avoid creation of eviction sets by attacker.
- They use ZCache a multi-step relocation technique to remap blocks in a randomized cache to reduce the number of LLC evictions in the remap process and hence reduces performance degradation.

C. No Skewed caches

- The problems with skewed caches are, (1) they complicate hardware design of LLC and (2) incurs high performance degradation in case of high skewing. So the authors propose to strengthen normal caches instead.
- They propose an attack detection mechanism for non-skewed caches which can stop algorithms like Prime, prune & test (PPT) and Group elimination(GE) to find eviction sets by remapping the randomized cache, whenever an attack is detected.
- They use a mathematical model based on Zscore for each cache set to detect an attack. Whenever the Zscore crosses some threshold (which is set carefully), an attack is detected and the cache is remapped.

III. Positives

- Assumptions regarding attackers are highly pessimistic in nature and thus the attack model is very strong. Mitigation against such an attack model will be hard to break.
- They validate the experiments for probability of finding eviction sets by attackers and compares them with a mathematical model. They find very close match between the experimental and mathematical results.
- The proposed idea to set remap period in terms of LLC evictions significantly reduces LLC MPKI by 69% and the number of remaps by 64% as compared to when LLC accesses are used to set remap period. This reduces overall performance degradation for randomized caches.
- They use the Spike simulator which increases the number of simulated instructions by 100 to 400 times compared to other simulators like Gem5 simulator. This results in better coverage of the benchmarks.
- They propose a hardware solution with minimal hardware overhead to implement the multi-step relocation technique to remap cache lines and to detect attacks(explained in the section II.C).

IV. Negatives

- The authors did not gave a proper explanation of how the small eviction sets with partially congruent addresses can yield a high eviction rate by repeatedly flushing and re-accessing the set (explained in Section II.A).

- For the above point they find the vulnerability but does not propose a mitigation for it.
- First they propose ideas to fix skewed caches. But in the later sections of original paper they say skewed caches should not be used as they incur high performance degradation, which makes the proposed fixes for skewed caches incoherent with the later argument of not using them at all.
- In section IV.B of the original paper, authors show that CT algorithm is faster than PPT and GE, but in section VI.B they find PPT and GE is faster than CT, but do not give a proper reasoning as to why this happens.
- In section VI.B of the original paper, the authors say that the remap period for non-skewed caches is short enough and instead of reducing them further they propose an attack detection mechanism to trigger remaps. But they do not give a proper reasoning of why not to reduce the remap period further.
- They find out appropriate remap period for different kinds of caches using a mathematical model to avoid attacks. But in section VI.B of the original paper, they say that the remap period is not long enough to mitigate attacks, which is contradictory to their previous findings.
- A minor issue, the strong attack model used may be a weakness in the sense that previously proposed mitigation techniques may not be secure enough against such an attack model, even though such a strong attack model is highly improbable in practical scenarios. For example, they assume there is no noise in the side channel, which is a bit superfluous.
- They do not include geomean/mean in the plots which makes them a bit difficult to comprehend.

V. Possible extensions

- The technique proposed to detect attacks only target few eviction set finding algorithms like CT, PPT and GE. There is scope of evading these mitigation techniques by introducing new algorithms to find eviction sets. So extending the mitigation techniques to encompass more attacks can be a possible extension.
- A possible Denial of service(DOS) attack can be made, where the attacker searches for eviction sets frequently enough to fool the attack detection technique proposed by the authors to frequently remap the randomized cache incurring huge performance penalty.

Review-2 : Randomized Last-level caches are still Vulnerable to side-channel attacks! But we can fix it.

1st Pass

The authors claim to show that cache randomization techniques likes CEASER-S & ScatterCache are not able to mitigate conflict-based side-channel attacks.

They also claim to fix the problems of the existing randomization techniques using new defense techniques, ~~and does not~~ which does not have a huge performance hit.

They also claim to show that randomized set-associative caches are a viable option for commercial processors.

Background

→ Conflict based-side channel attacks.
- preparation phase (creation of eviction sets).
- exploitation phase (prime-probe, ...).

→ Randomized caches target the preparation phase.
↳ attacker cannot simply build eviction sets by choosing appropriate addresses which goes to the same set.
- Instead attacker has to dynamically search for addresses that goes to the same set.
* With randomization this process is infeasibly long.

→ Double- let 2 partitions. — How to choose in which partition ~~address maps to partn~~ the block will go into?

→ Fast Algos for searching Eviction sets.
↳ Group elimination, conflict testing, prime, prune & then test.

All these can easily break ~~the~~ randomized caches that ~~are~~ ~~not~~ do not use ~~dynamic keys~~ dynamic remapping.

∴ Randomized caches need remapping.

→ very short remap rate — high performance penalty.

→ ~~solution - skewed~~ also to mitigate attacks a short remap rate is necessary.

→ solution - skewed caches (more than 1 partition).
which can tolerate a higher & ~~more precise~~ remap rate which mitigates attacks as well as doesn't incur high performance hit.

↳ problem: methods like conflict testing & prime, prune test can still compromise the defense of skewed caches with remapping.

Intro

1. Cache partitioning is one of the ways to mitigate side channel attacks, but...
   - secure, normal data are not always easy to separate.
   - decreases LLC utilization.
   - OS dependent.
2. Randomized caches - makes it difficult (slows down) for the attacker to create eviction sets.

Along with this
- Remap - makes it more difficult by reducing the time window for attack.
- Skewed cache further complicates searching for eviction sets.

Contributions.

- show that randomized caches with remap/skewed are still vulnerable to attacks, & attackers can create eviction sets.
- ~~tries to~~ fixes the problem that they ~~found~~ find about ~~about~~ randomized caches ~~by taking without~~ keeping performance degradation to the minimal.

2nd Pass & 3rd pass

Assumptions regarding attackers are highly ~~optimistic~~ e.g. pessimistic & the attack model is strong. which is both positive & negative.

- Positive in the sense, mitigation against such an attack model will be hard to break.
- Negative in the sense, previously proposed mitigations techniques may not be secure enough against such an attack model, even though such an attack model is highly improbable in practical scenarios.

Vulnerability of randomized skewed caches:

1 → Using prime-probe-flush, attacker can evict (attackers eviction set) the victim's target address, even with a partially congruent eviction set, with a fairly high eviction rate.

Idea:

- In case an attackers block is cached into the wrong partition, the target address will never get evicted. After this point the eviction set is deemed useless.

↳ Sol. After each probe, attackers flush the eviction set & accesses the eviction set again, in the hope that all the blocks will be placed in congruent sets.

↳ clflush can be used to do this.

(Authors also argue that clflush init will be here to stay!)

2 → ~~Using a small eviction set with high~~ Low
Find a small eviction set is faster.
So using a small eviction set with low eviction rate can be used to achieve an overall high eviction rate if the eviction set is accessed repeatedly.
(Flawed hypothesis of Scatter Cache).

3 → They show that using CT, it is possible to find a small eviction set (having ~30% eviction rate) in about 350 k LLC accesses. which is far less than the remap period.
And as mentioned in point 2, we can use this small partially congruent eviction set repeatedly to get high eviction rate using prime-probe-flush.

4 → Defining remap period by LLC accesses is a bad choice as most accesses to LLC are filtered by L1, L2 hits
But why is this problem in regards to attack mitigation

Main Crux (Problem):

~~→ Faster to find eviction~~
~~→ Small partially congruent~~
→ Small eviction set with partially congruent addresses can be found (by algos like CT) in a much lesser time compared to the remap period.
→ These small eviction sets with eviction rate as low as 30% can be repeatedly flushed & reaccessed to get an overall eviction rate of as high as 90%.

→ Partial reason → remap period in terms of LLC accesses.

FIXES

1. Define Remap period in terms of Evictions.

→ As LLC accesses are filtered by L1/L2 hits, use LLC evictions instead.

→ They find a mathematic equation to estimate the # of evictions required to find a small eviction set.
Using the equation they find appropriate remap periods for diff. # of partitions in caches.

(*) LLC evictions are much higher during an attack as compared to normal execution.
Thus Evictions is much better quantification technique for setting remap period.

2. Remap process hurts performance by 40% to 50%.
# of blocks are evicted in the process.
↓
this can also help DOS attacks.

→ They use ZCache a multi step relocation process to remap blocks. This reduces evictions to 10%

→ They propose a hardware design to implement the multi-step remap process

PROPOSAL - USE SKEWED NORMAL CACHES

Problems with skewed cache
↳ complicates design of LLC.
↳ performance hit for high skewedness.

∴ They try to make a case to strengthen randomized set-associative caches without skewing.

Mitigation for non-skewed caches.

→ With PPT & GT algorithm can find eviction sets within the remap period they propose earlier.

↳ They say that the remap period is already short enough & can't be reduced further.

→ They apply z-score on the distribution of LLC evictions to detect an attack..

→ They use a mathematical model for this.

→ They have a score for each cache set & whenever it crosses the threshold of 5, an attack is detected & the cache is remapped.

Performance analysis

→ MPKI with remapping by evictions reduces by 69% with attack & 87% without attack compared to remapping by accesses.

→ Reduction in remaps.

64% with attack

71% without

→ using # of evictions also reduces unnecessary evictions for each remap.

along with multi-step relocation.