BCE-R: LLC Clusters re-ordering

Harsh Raj
Department of CSE
IIT Ropar
Ropar, India
2022csm1004@iitrpr.ac.in

Kumar Mangalam
Department of CSE
IIT Ropar
Ropar, India
2022aim1002@iitrpr.ac.in

T.V. Kalyan
Department of CSE
IIT Ropar
Ropar, India
kalyantv@iitrpr.ac.in

I. Problem Statement

Last level cache are cache memory shared among multiple cores. This means that processes from different cores have to share the memory at LLC which increases the opportunity for an Cache Side-Channel attacks intended to leak sensitive data from a victim process which is a serious security challenge for Last level Cache. Cache Side-Channel attacks extracts information from a process through unintended channels. Explicitly in timing based cache side channel attacks, the attacker can observe latency of cache accesses to infer which data was accessed by the victim process without having direct access to the data.

There are various countermeasures such as cache partitioning, cache way prediction cache randomization technique available to mitigate the LLC side-channel attacks but they also incur reduced performance and increased complexity. Bespoke cache enclave has implemented a flexible set based cache partitioning technique which provides fine grained and scalable isolation from cache side channels. It assigns cache to the domains as group of clusters but as soon as the execution of domain completes the clusters assigned to it are not reclaimed by the LLC so that other domain's demand for more clusters can be fulfilled. Therefore there is a need for set based cache partitioning technique which can flexibly allocate cache as clusters to respective domain such that address are uniformly mapped to clusters of domain with minimum set conflicts and access latencies remain uniform to eliminate security challenge from cache side channel attacks.

II. Motivation for solution

The Cache-side channel attacks have the potential to leak sensitive information, compromise system integrity and violate the user privacy. These attacks exploit the information leakage through cache behaviour to infer the sensitive data. In timing based cache side channel attacks, the variation in latencies can be exploited to determine which cache blocks have been accessed by the process and based on that the attacker can leak the sensitive data. Also, the existing cache side-channel mitigation techniques often provide coarse-grained isolation from cache side-channel attacks which limit their effectiveness. The coarse grained isolation mechanisms often cannot provide sufficient protection against cache side-channel attacks as they fail to prevent the leakage effectively. For example, way based set-partitioning flexibility is bounded by the number of ways available with the Last level cache. Also, irrespective of how much space needed for a process in the cache the process will be allocated space in cache in the multiple of ways in cache. There is possibility of processes which are memory intensive for cache but not for main memory on the other hand processes which implement graph algorithms are memory intensive for main memory but only need fewer cache blocks which must be taken into account for efficient utilization of cache memory. The existing mitigation techniques often add significant performance overhead which degrade system performance and hinder the overall efficiency of the system. The mitigation technique should provide uniform faster lookup for every memory access to cache. Thus, there is a need for flexible set-partitioning based fine-grained and scalable isolation from Cache side channel attacks on Last level cache.

III. Proposed Idea

A. Implementation and Working

The Bespoke cache enclave consists of two modules named Load Balancing hash(LBH) and Cluster Indirection module(CIM). A new process is assigned an existing or a new domain. I f the new process desires security OS allocates an isolated partition in LLC and assigns domain ID to the process. Thus all the read/write accesses to the LLC in BCE is accompanied with Domain ID. The Load Balancing hash(LBH) hashes the line address into the logical cluster ID(LCID) and then the Cluster indirection module using the LCID obtained from LBH and the LCID0 obtained from Domain base table(DBT) gives the Physical cluster ID(PCID). This PCID along with the cluster offset bits determines the set index for the LLC. Then, the set is checked for the tag match for accessing the required cache block. If there is a hit the replacement state update remains unchanged otherwise, on LLC miss eviction candidate is chosen from entire set. In case of dirty eviction, the cluster offset is appended to the tag bits to generate the write-back line address. At the end, when the domain execution completes the LLC partitions allocated to that domain are reclaimed.

The Load balancing hash(LBH) is responsible for uniformly mapping line addresses to logical clusters of the domain(LCID). The Load balancing hash is implemented using multiple randomizing hashes. For mapping line address to LCID, the bottom n bits of line-address are checked to see if it is less than Number of clusters. If not, the line address

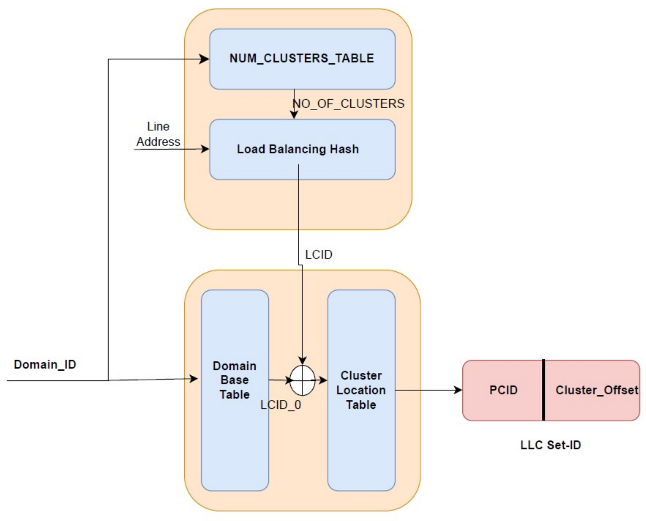

Fig. 1. State-of-the-art BCE implementation

is transformed using a low latency randomizing hash function $H_1(x)$ and the bottom n bits are checked to see if they can be used as LCID. If the LCID is still not obtained the inverted value (x[n-1:0]) is used as the LCID. The hash function $H_1(x)$ to $H_i(x)$ is constructed using Random binary matrix for faster hardware implementation where, each $H_i(x)$ consists of Bxn bits sized matrix (24x9 by default) where bit values are statically populated from uniform random distribution. Bitwise AND is computed for input vector and corresponding row of the RBM followed by XOR reduction to get bits of the output hash. Any two hash function generated as mentioned above will have very low probability of generating same hash for any two functions. The Cluster indirection module(CIM) is responsible for locating the LLC clusters allocated to domain by translating the logical clusters of a domain that is Logical cluster ID(LCID) must be translated to its allocated LLC cluster that is Physical cluster ID(PCID). The Cluster location table(CLT) keeps an ordered list of valid LCID to PCID mappings for each domain with at least 1 valid cluster. The mapping for each domain are kept contiguous in the CLT, starting with the base entry (LCID-0) for each domain. The Domain base table (DBT) tracks the location of the base entry (LCID-0) in the CLT for each domain. The DBT is accessed with the DomainID to obtain a pointer to the base entry (BasePtr) of that domain in the CLT. Adding the LCID to the BasePtr provides the CLT location for the LCID of the domain. The values stored in the CLT location provides the reqiured PCID, which is concatenated with the cluster-offset bit to retrieve the set-index. The CIM module provides the cache isolation security as only the LLC clusters allocated to a domain that is clusters that are addressable by the CLT entries of the domain are accessible to cache lookups from the domain. The LCID used to calculate the CLT index is always less than the number of clusters allocated to that domain thus, CLT entries of a particular domain are accessible on cache lookups from that domain. The CLT needs 512 valid entries to support 512 PCIDs similarly, DBT must have atmost 512 entries for 512 domains.

The BCE_alloc instruction is used to create new LLC partition for a trust domain. The input taken is domain ID and number of cluster and if the number of clusters requested are available then it returns success else failure. On success, contiguous list of CLT entries with the PCIDs allocated for the domain. The DBT contains entry for the first CLT entry of the domain and the NumClusterTable is updated with the allocated clusters.BCE_dealloc instruction is used to deallocates the existing LLC partition. It takes a Domain ID as an input and invalidates the associated DBT and CLT entries. Thus the lines in the physical LLC clusters of the domain are flushed and PCIDs of the domain becomes free to be allocated to other domains. The invalidated entries are also compacted and moved to the end of the CLT to ensure that a subsequent cche alloction can obtain a contiguous cluster of free CLT entries.

For the implementation of BCE-R fowllowing are some of the additional support from the system is needed. Processes that trust each other are alloted the same DomainID thus, they have to share cache-sets with each other like conventional caches and have read-write sharing of pages. Therefore, page sharing with malicious processes among other processes must be regulated by the OS to prevent any direct information leakage through no read-write sharing. Also OS must flush out read-only pages (whose shared cache lines get duplicated in each domain) from each of the LLC domain.

IV. Experimental Setup

A. Simulator details

For performance evaluations, we used the ChampSim for running program execution traces (of length 10 million instructions). ChampSim is a trace-based simulator for a microarchitecture study.

B. Configuration

Core	2-cores
L1, L2-Cache	L1-32KB, L2-256KB
LLC (shared)	32MB, 16-way

TABLE I
Baseline System Configuration

The system modeled in our study is shown in Table I. We use a 2-core system with a 32MB 16-way shared L3 cache. For performance evaluations, we use a trace-driven simulator (ChampSim) running program execution traces (of length 10 million instructions). Our baseline is a non-secure shared LLC with LRU page-replacement policy.

C. State-of-the-art papers

Utility-Based Cache Partitioning [2] discusses the problems faced in partitioning a shared cache or last level cache. As the last level cache is shared by multiple core, the pressure to sustain the meory requirements of concurrently executing

applications increases such that the DRAM access is less. Traditional LRU replacement policy partitions a shared cache on demand basis but it has flaws in it. For example, a streaming/thrashing application there is a large number of unique cache blocks access but those cache blocks are of no help though the application has high demand. It was tested on two SPEC benchmarks, vpr and equake, with the variation in cache size by changing the number of ways and keeping the number of sets constant. For vpr, it was observed that the number of misses decreases with the increase in number of ways in the cache size but for equake the number of misses remain constant. The equake have no benefit from increasing the number of ways while for vpr it is beneficial. So proposing a new Utility-Based Cache Partitioning, where a partitioning algorithm is proposed to decide the amount of cache to be allocated for each competing application.
High-Resolution Side-Channel attack on Last-Level Cache [1] introduces a new high-resolution LLC side channel attack on the real system. Today the systems and applications use cryptography based security. Although ciphers are cryptographically secure, side channels are prone to leak sensitive information compromising security. Some recent targets for the attacks are Shared LLC and other conventional microprocessor. One such attack is Cache based side channel attack in which the spy process shares the cache with the victim process and monitors the cache accesses performed by the victim process. The cache set accessed by the victim correlates with the indices in the cryptographic tables which are used by many ciphers. This is often sufficient to clone the secret key in sufficient time. The LLC is more vulnerable to these attacks because it is shared among all the processor cores and also it is easier for the spy process to stay in the LLC along with the victim process.

FLUSH+RELOAD is one of the L1 cache attacks which is also used for LLC attacks where the cryptographic lookup tables are located in a memory region shared by both victim and spy processes and visible to both processes. The spy process can flush the cache lines containing the cryptographic tables from all cache levels using cache flush instruction. The attacker later determines if the victim process accessed the data by reaccessing the memory line and timing the accesses.

PRIME+PROBE is an alternative to FLUSH+RELOAD when the critical data and cryptographic data is not shared. This LLC attacks the victim data and is evicted from the cache by filling all cache ways. On access by spy process if there is a miss then the victim process must have accessed the cache line.

In this paper, a new PRIME+PROBE style high resolution LLC attack is proposed that rules out the dependency on cryptographic data between victim and attacker and can work on any page size. This new attack is effectively demonstrated on AES cipher on Intel SandyBridge processor with 8MB shared LLC by the authors. The authors, with this attack, were able to construct a secret key in a few minutes.
Limiting Cache-based Side-Channel in Multi-tenant Cloud using Dynamic Page Colouring [4] discusses the Page coloring technique. Page-colouring is a technique that directs how memory pages are mapped to cache lines. It is also used to improve fairness and utilisation of cache in multicore. Essentially, it ensures that a group of pages with the same colour will be mapped to the same set of cache lines.

V. Results and Analysis

We used SPEC-22 traces for the analysis of our experiment.

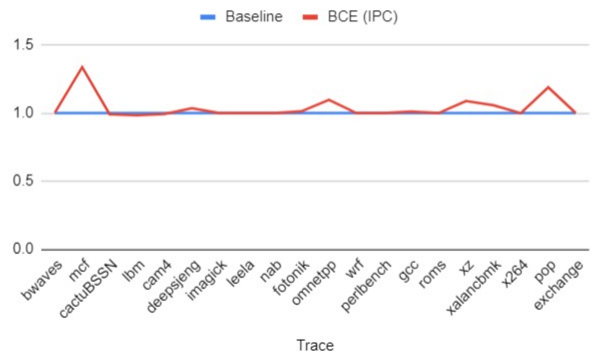

Fig. 2. Baseline vs BCE-R IPC comparison

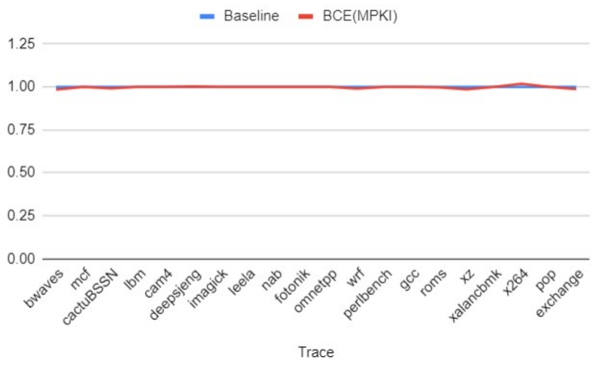

Fig. 3. BCE Slowdown compared to Baseline

Figure 2 & 3 is a comparison of SPEC traces on BCE-R and Baseline over IPC and MPKI. It was observed that the average slowdown w.r.t MPKI of BCE-R was $< 1\%$.

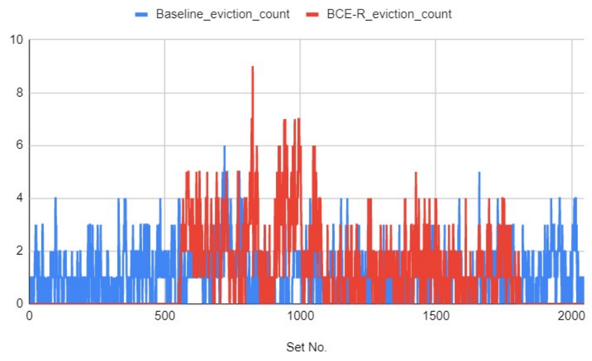

Fig. 4. 1-core: LLC block eviction per set comparison

Figure 4 depicts the LLC block utilization per set over perlbench trace for single core and it was observed that the block evictions per set over LLC were not uniformly distributed for the BCE-R.

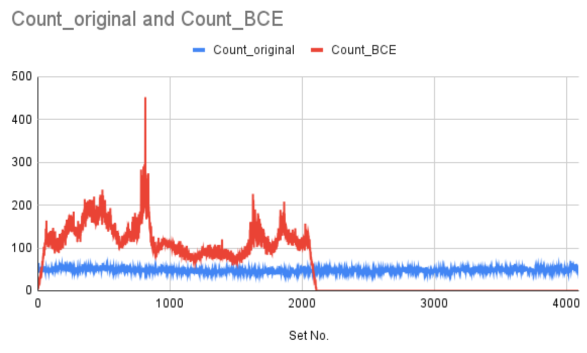

Fig. 5. 2-core: LLC block eviction per set comparison

Figure 5 depicts the LLC block utilization per set over bwaves and mcf traces for two cores and it was observed that the block evictions per set over LLC were not uniformly distributed for the BCE-R. To resolve that, our idea was to re-order the cluster based on the requirement.

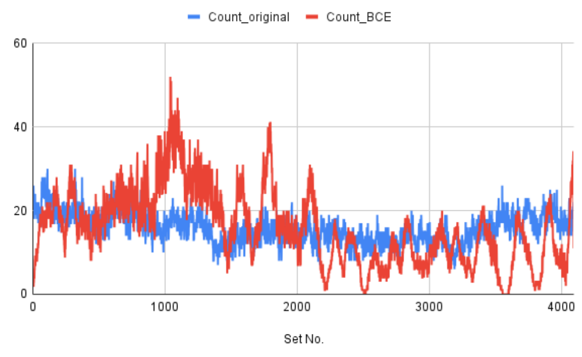

Fig. 6. 2-core: LLC block eviction per set comparison

A few assumptions were taken while implementing BCE, the two basic requirements to implement BCE were DomainID and 32-bit line address. While implementing over ChampSim, we have no control over DomainID as it is provided by the Operating System, so we are assuming cpu-id as the DomainID. ChampSim provides a 64-bit line address instead of the 32-bit line address mentioned in the original paper [2], to solve this problem, we ignore the initial 32-bit, and only consider the last 32-bits. Which causes a few sets to be missed.

VI. Conclusion

Bespoke Cache enclave [3] is capable of providing reliable defence against Last-level cache side channel attacks, provide fine-grained LLC partitions to domains and manage hundreds of security domains simultaneously. Although, there was slight slowdown($< 1\%$) compared to baseline with less memory($< 2\%$) overhead was observed in the implementation of BCE. BCE is successfully implemented on trace based simulator-ChampSim and all the objectives of BCE were satisfactorily realized.

References

[1] Mehmet Kayaalp, Dmitry Ponomarev, Nael Abu-Ghazaleh, and Aamer Jaleel. A high-resolution side-channel attack on last-level cache. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[2] Moinuddin K. Qureshi and Yale N. Patt. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pages 423–432, 2006.

[3] Gururaj Saileshwar, Sanjay Kariyappa, and Moinuddin Qureshi. Bespoke cache enclaves: Fine-grained and scalable isolation from cache side-channels via flexible set-partitioning. In *2021 International Symposium on Secure and Private Execution Environment Design (SEED)*, pages 37–49, 2021.

[4] Jicheng Shi, Xiang Song, Haibo Chen, and Binyu Zang. Limiting cache-based side-channel in multi-tenant cloud using dynamic page coloring. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 194–199, 2011.