
Legends:

Case 1: Inclusive Case(I)
Case 2: Non-Inclusive Non-Exclusive Case(NINE)
Case 3: Exclusive Case(E)
L1M : L1 cache miss
L2H : L2 cache hit
L2M : L2 cache miss
L3H : L3 cache hit
L3M : L3 cache miss
Cache Hierarchy : L1 -> L2 -> L3

Policies:
---
1. Insertion Policy: Which level of cache hierarchy does the new block get inserted into on a miss and it's effect on other levels of cache hierarchy?
2. Replacement Policy: Which block should be replaced on a miss?
3. Eviction Policy: How replacing a block on a miss affects other levels of cache hierarchy?

****
Analysis 1: L2H(I) < L2H(NINE)
****
There are less hits in L2 inclusive cache as compared to L2 non-inclusive non-exclusive cache due to the eviction policy(L3). Insertion and Replacement poilicies are same for both the cases. When a block in L3 is to be replaced, the corresponding block in L2 is invalidated in case of inclusive hierarchy. As a result of this, block in L2 having the tag of "Most Recently Used" will be invalidated if the corresponding block in L3 is tagged as "Least Recenty Used" and a cache miss in L2 and L3 occurs. Next time, if the evicted block is searched in L2 and L3, it will be a miss in both and again, one more block will be victimized while filling the earlier evicted block. Misses may cascade drastically in L2 in this way. Hence, inclusive L2 will encounter less hits than non-inclusive non-exclusive L2.

****
Analysis 2: L3H(I) > L3H(NINE)
****
The hits in L3 inclusive cache is more than in L3 non-inclusive non-exclusive cache. Although, partial non-exclusivity can virtually increase the capacity of L3 non-inclusive non-exclusive cache as compared to L3 inclusive(where some blocks in L3 are same as those in L2) but the nature of application(whose trace is being processed) dominates the hit rate. In our case, as the applications might be using a particular block(present in L2) after some fixed time interval during which it might get evicted from L2 in both cases. In case of inclusive L3 cache, we will get a hit (for the second access to that block after it was evicted) after L2 miss while in case of non-inclusive non-exclusive L3 cache, it might be a miss with high probability. For this instance, partial exclusivity of L3 cache in non-inclusive non-exclusive hierarchy is a disadvantage. More of such accesses in the application can significantly increase L3 hits in inclusive hierarchy as compared to non-exclusive non-inclusive cache hierarchy.

****
Analysis 3: L2H(NINE) = L2H(E)

****
The hits in L2 non-inclusive non-exclusive cache is same as L2 exclusive even though the insertion and eviction policies of the two cases differ. The reason is that in both cases L2 cache sees the same number of misses from L1. In both cases, eviction policy does not have any effect on L2 unlike the case of inclusive hierarchy. Thus, no vital(most recently used) block is victimized and evicted from L2. Also, on a L2 & L3 miss, block is inserted in L2 in both cases. On a L2 miss and a L3 hit, the block is inserted from L3 to L2 in both cases. Hence, the effect on L2 due to inclusivity or exclusivity is not visible as insertion and eviction policy, as far as L2 is concerned, becomes same for both cases.

****
Analysis 4: L3H(NINE) > L3H(E)
****
The hits in L3 non-inclusive non-exclusive cache is greater than L3 exclusive due to the effect of different insertion(L2 & L3) and eviction policies(L2). As a L2 and L3 miss results in filling of the block only in L2 in exclusive case, the number of blocks in L3 disjoint of blocks in L2 increases and hence, should virtually increase capacity of L3 cache for exclusive case. But that does not happen due to two reasons. First, when a block in L2 is replaced in exclusive hierarchy, it is transferred to L3 cache where a "Most Recently Used" status block might get evicted. Second, on a L2 miss and a L3 hit, the block is brought into L2 and it is invalidated in L3 to maintain exclusivity, thus, effectively decreasing the virtual capacity of L3 cache and resulting in less hits in L3 cache when the evicted block is again looked for in L3 cache. The virtual increase in L3 capacity due to exclusive hierarchy insertion policy is overwhelmed by the above two factors. Therefore, L3 hits is less in exclusive case as compared to L3 non-inclusive non-exclusive case where the above mentioned effects are not seen. In our case, as the applications might be using a particular block(present in L2) after some fixed time interval during which it might get evicted from L2 in both cases. In case of non-inclusive non-exclusive L3 cache, we might sometimes get a hit (for the second access to that block after it was evicted) after L2 miss while in case of exclusive L3 cache, it will again be a miss. For this instance, complete exclusivity of L3 cache in exclusive hierarchy is a disadvantage. More of such accesses in the application can significantly increase L3 hits in non-inclusive non-exclusive hierarchy as compared to exclusive cache hierarchy.

****
Analysis 5: L2H(I) < L2H(E)
****
There are less hits in L2 inclusive cache as compared to L2 exclusive cache mainly due to the eviction policy. When a block in L3 is to be replaced, the corresponding block in L2 is invalidated in case of inclusive hierarchy. As a result of this, block in L2 having the tag of "Most Recently Used" will be invalidated if the corresponding block in L3 is tagged as "Least Recenty Used" and a cache miss in L2 and L3 occurs. Next time, if the evicted block is searched in L2 and L3, it will be a miss in both and again, one more block will be victimized while filling the earlier evicted block. Misses may cascade drastically in L2 in this way. Hence, inclusive L2 will encounter less hits than exclusive L2. In exclusive L2 cache, eviction policy(L3) does not have any effect on L2 unlike the case of inclusive hierarchy. Thus, no vital(most recently used) block is victimized and evicted from L2. Thus, L2 hits are more for exclusive L2 cache as compared to inclusive L2 cache.

****
Analysis 6: L3H(I) > L3H(E)
****
The hits in L3 inclusive is greater than L3 exclusive due to the effect of different insertion(L2 & L3) and eviction policies(L2). As a L2 and L3 miss results in filling of the block only in L2 in exclusive case, the number of blocks in L3 disjoint of blocks in L2 increases and hence, should virtually increase capacity of L3 cache for exclusive case as compared to inclusive L3 cache where most of the blocks are same as that in L2 cache(thus, reducing the virtual capacity of L2). But that does not happen due to two reasons. First, when a block in L2 is replaced in exclusive hierarchy, it is transferred to L3 cache where a "Most Recently Used" status block might get evicted. Second, on a L2 miss and a L3 hit, the block is brought into L2 and it is invalidated in L3 to maintain exclusivity, thus, effectively decreasing the virtual capacity of L3 cache and resulting in less hits in L3 cache when the evicted block is again looked for in L3 cache. The virtual increase in L3 capacity due to exclusive hierarchy insertion policy is so overwhelmed by the above two factors that the advantage imparted due to it is shadowed completely by L3 inclusive cache(even though the effective capacity of L3 was reduced in inclusive case due to insertions at both L2 and L3). Therefore, L3 hits is less in exclusive case as compared to L3 inclusive case where the two above mentioned effects are not seen. In our case, as the applications might be using a particular block(present in L2) after some fixed time interval during which it might get evicted from L2 in both cases. In case of inclusive L3 cache, we always get a hit (for the second access to that block after it was evicted) after L2 miss while in case of exclusive L3 cache, it will again be a miss. For this instance, complete exclusivity of L3 cache in exclusive hierarchy is a disadvantage. More of such accesses in the application can significantly increase L3 hits in inclusive hierarchy as compared to exclusive cache hierarchy.

NOTE:

Comparisons across traces are not possible unless we know how the load/store addresses are ordered which depends on the way the code is written in those trace applications.