Prefetch et Reload

Théo Mainguet
Anass Belarbi

March 16, 2023

1 Introduction

1.1 Cache

La plupart des caches des processeurs modernes x86 modernes sont divisés en L1, L2 et L3. Les caches L1 et L2 sont très rapides (moins de 50 cycles pour la majorité d'entre eux) mais leur taille est limitée, généralement réservés à chaque cœur de processeur, ils sont appelés "Cache privé".
En comparaison, la mémoire cache L3, aussi appelée "Cache partagé" (ou en anglais "Last-Level Cache") est un cache plus large mais plus lent par rapport au cache L1 et L2, elle est utilisée comme mémoire partagée entre les différents cœurs des processeurs.

De plus les caches sont pour la plupart set-associative, ce qui signifie que la mémoire cache est divisée en ensembles de N lignes de cache (l'unité de base pour la transition de cache).

Quand le CPU fait une requête mémoire, il va d'abord vérifier si la ligne de cache recherchée est présente dans les mémoires cache L1 et L2, si elle n'est pas présente, il va vérifier dans le cache LLC et si encore une fois elle n'est pas présente ici il finira par rechercher cette ligne dans la DRAM.

1.2 Cohérence de Cache

Dans un système multi-coeur, une ligne de cache pourrait être présente dans plusieurs caches privés, dû au partage de données entre les processus. C'est pourquoi il est important d'avoir un protocole pour maintenir la cohérence de cache, la plupart des processeurs modernes en x86 utilisent le protocole MESI qui va attribuer à une ligne de cache l'un de ces 4 attributs :

- Modified (M) : La ligne de cache n'est présente que dans une seule mémoire privée, elle est considérée comme "sale" car elle n'est pas cohérente avec la mémoire centrale, le coeur qui la possède a les droits d'écriture/lecture dessus, par contre la mémoire centrale doit être mise à jour avec la nouvelle version local avant de pouvoir être lu.
- Exclusive (E) : La ligne de cache n'est présente que dans une seule mémoire privée, elle est "propre", c'est à dire qu'elle est cohérente avec le contenu de la mémoire centrale, le coeur qui la possède a les droits d'écriture/lecture dessus, par contre écrire dessus la fera passer en Modified (M).
- Shared (S) : La ligne de cache est présente dans une ou plusieurs mémoire(s) privée(s), elle est propre, c'est à dire qu'elle est cohérente avec les données contenues dans le LLC, le coeur qui la possède a juste des droits de lecture dessus.
- Invalid (I) : La ligne de cache est invalide (pas à jour), le coeur qui la possède n'a aucun droit dessus.

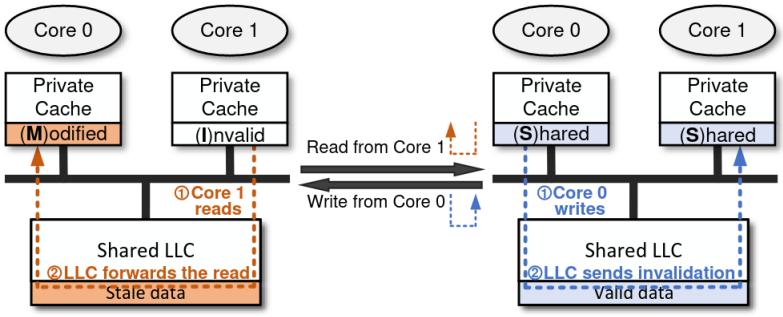
Avec le protocole MESI nous allons observer deux choses qui peuvent arriver lors d'une requête mémoire provenant du CPU, un changement d'attribut de cohérence pour la ligne de cache demandée et une différence de temps au niveau de la récupération de données suivant l'attribut de cohérence.

1.2.1 Changement d'attribut de cohérence

Un changement d'attribut peut arriver lors d'une requête d'écriture ou de lecture suivant les attributs que possède la ligne de cache concernée, voilà deux exemples.

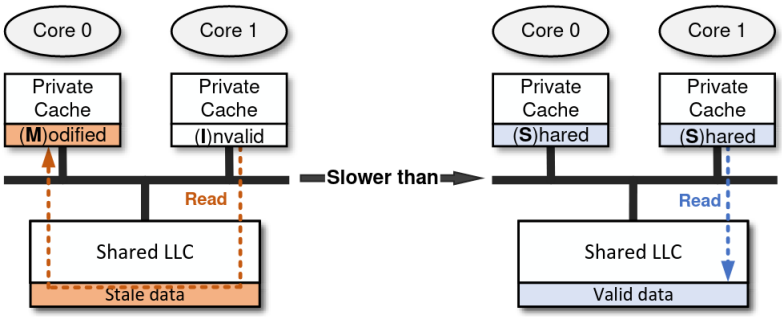
Nous allons ici prendre le cas d'un coeur de CPU (core 1) qui voudrait lire une ligne de cache qui est à la fois présente dans le LLC et dans le cache privé d'un autre coeur (core 0) dont l'attribut est Modified (M), comme le cache privé du core 0 a été modifié, nous devons d'abord mettre à jour le LLC qui est devenu "périmé" avant de pouvoir lire son contenu, pour ce faire le LLC va récupérer la ligne de cache contenue dans le cache privé du core 1, changer l'attribut de cohérence de cette ligne de ce cache privé pour passer de Modified à Shared (S) et remplacer son contenu "périmé" par la nouvelle ligne qu'il vient de récupérer. Il pourra ensuite renvoyer son nouveau contenu au core 0 qui aura aussi l'attribut Shared.

Un autre cas serait si un coeur de CPU (core 0) voulait écrire une nouvelle ligne de cache dans son cache privé mais que cette ligne possède l'attribut Shared, alors ce cache privé doit d'abord envoyer une requête au LLC pour avoir un droit d'écriture, cela aura pour effet que le LLC va envoyer un signal à tous les caches privés ayant la même ligne de cache afin de les rendre Invalide (I), puis l'attribut de la ligne de cache du core 0 va passer de Shared à Modified afin que ce dernier puisse être modifié, il ne sera pas recopié automatiquement dans les autres caches privés.



1.2.2 Différence de temps suivant l'attribut de cohérence

Comme nous pouvons l'imaginer, une requête de lecture d'une ligne de cache ayant un attribut Modified prendra plus de temps que si il était Shared, car le temps d'aller récupérer des données dans un autre cache privé ainsi que de modifier les attributs de cohérence des caches privés concernés rend le tout plus lent que si la ligne de notre cache privé était Shared depuis le début.



1.3 Prefetch

Le prefetch est le principe de précharger les données dans le cache avant qu'elles ne soient utilisées afin d'améliorer les performances du processeur. Les processeurs x86 proposent de nombreuses instructions de préchargement logiciel : PREFETCHT0, PREFETCHT1, PREFETCHT2 et PREFETCHNTA... Elles sont utilisées pour indiquer au processeur qu'un emplacement mémoire est le plus enclin à être consulté ou modifié prochainement, le processeur précharge alors les données correspondantes, ce qui permet d'accélérer les futurs accès à ces données.

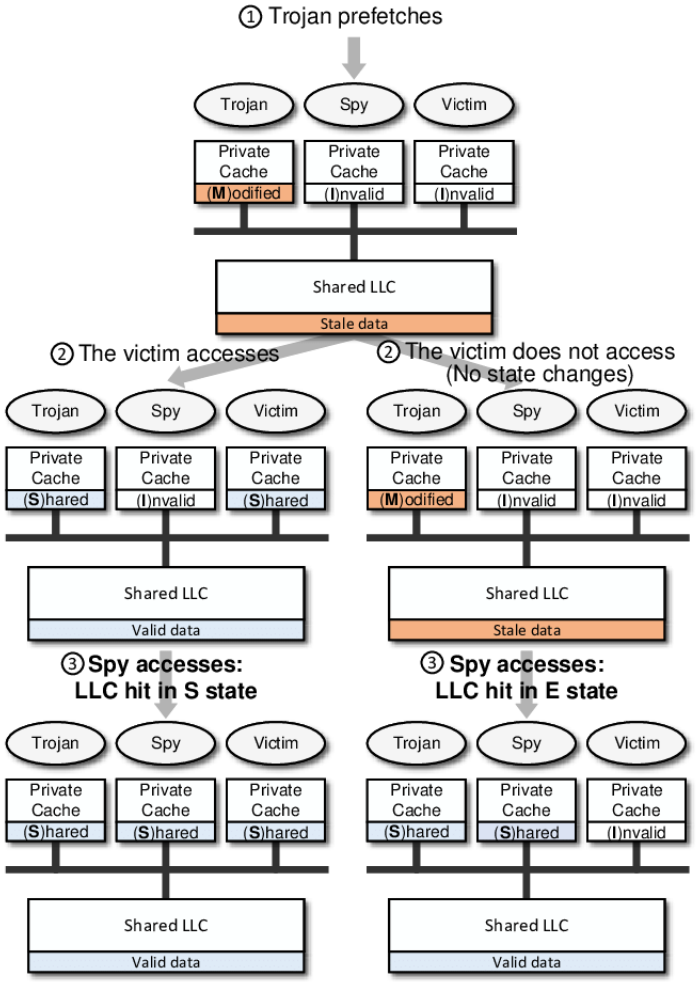
2 Prefetch et Reload

L'attaque Prefetch et Reload est une attaque qui permet de dévoiler le schéma d'accès au cache partagé d'une victime et d'en déduire certains secrets. L'attaquant contrôle deux threads nommés Trojan et Spy, ces threads ainsi que la victime doivent tous être localisés sur des coeurs différents les uns des autres. L'attaque se déroule en 3 étapes comme suit:

1. Le Trojan execute *PREFETCHW* sur la ligne de cache ciblée, l'amène dans son cache privé et cela change l'état de cohérence de la ligne de cache en *Modified* en même temps que d'invalider (*Invalid*) les copies de cette ligne de cache dans les caches privés du Spy et de la victime.

2. L'attaquant attend que la victime effectue une action. Si la victime accède à la ligne de cache alors l'état changera de *Modified* en *Shared* et cette ligne de cache sera copiée dans le cache privé de la victime. Le changement d'état cohérence causé par l'accès de la victime ne peut pas être observé par le Trojan: si le Trojan accède ultérieurement à cette ligne de cache (reload), il verra un hit privé du cache peu importe si la victime a effectué un accès ou non car cet accès n'invalide pas la copie de la ligne de cache présente dans le cache privé du Trojan.

3. Le Spy quant à lui possède déjà une copie invalidée grâce à l'étape 1, ainsi quand il accède à la ligne de cache il peut mesurer le temps d'accès pour déterminer si l'état est *Modified* ou bien *Shared*. Si l'état est *Modified* alors la victime n'a pas accédé à la ligne de cache, si c'est *Shared* alors elle y a accédé.



[1]

References

[1] Y. Guo, A. Zigerelli, Y. Zhang, and J. Yang, "Adversarial prefetch: New cross-core cache side channel attacks," oct 2021.