Cache Coherence (Write-Through Protocol)

What is cache coherence?

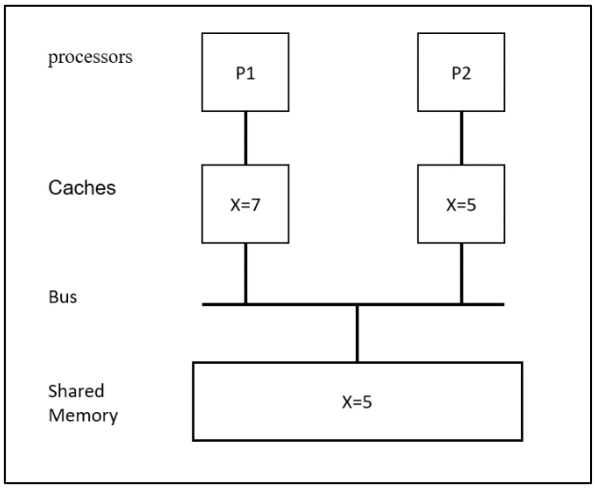
It is a concern in a multicore environment because of distributed Level 1 and Level 2 caches. Since each core has its cache, the copy of the data in that cache may not always be up to date.

Explaining throw the problem:

An issue with distributed L1 and L2 caches in multi-core systems is cache coherence. To always have the most recent copy of the data, each core's L1 and L2 caches must be in sync with one another.

PROBLEM EXAMPLE:

Imagine a multi-core processor where each core has its private cache.



1. Processor$_1$ reads X from memory
    - loads X=5 into its cache
2. Processor$_2$ reads X from memory
    - loads X=5 into its cache
3. Processor$_1$ writes X=7
    - stores X=7 in its cache
4. Processor$_2$ reads X from its cache
    - loads X=5 from its cache

Incoherent value for X on Processor$_2$

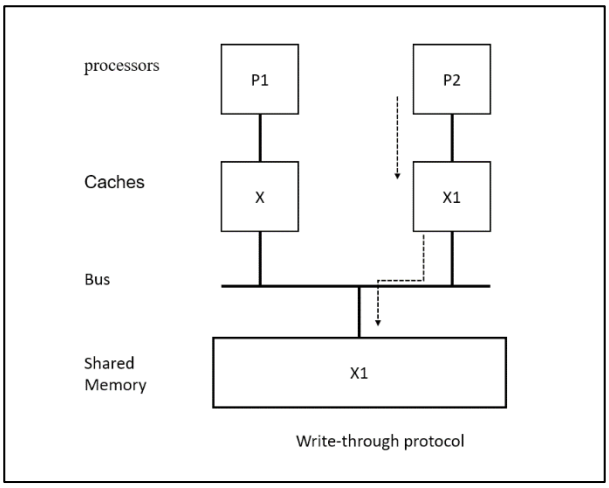
When the Processor$_2$ reads the value from its cache it will be an outdated version. That is cache coherence. If there is no cache coherence policy in place, the wrong data would be used, and invalid results would be produced.

Problem conclusion:

A challenge to keep multiple local caches synchronized is when one of the processors updates its local copy of data which is shared among multiple caches.

Write through protocol (Solution)

It is a cache write policy that ensures that all write operations are made to the cache as well as to the main memory, guaranteeing that the main memory is always valid (in sync).
Write through protocol has two fundamental implementations.



I. Write through with update protocol (Write- broadcasting):

The processor gives the cache a new value, which is also written into the memory module that holds the cache block being changed. Some copies of this block may exist in other caches. We update them by broadcasting the newly updated data to all processors modules in the system.

II. Write through with invalidation copies (Broadcast-invalidate):

When a processor updates a value in its cache, the updated value is also updated in the memory module that holds the cache block being, and all previous values are also invalidated by broadcasting the invalidation request through the system. This invalidation is received by all caches and the cache which contains the old data clears its cache line

Difference between Write invalidate protocol and Write Update Protocol:

Write Update Protocol	Write invalidate protocol
Multiple write broadcasts are required when multiple modifications to the same word is done with no read is done in between	One initial invalidation is required when multiple modifications to the same word is done with no read is done in between.
Less time is taken to read the data because the data modified by the processor is immediately updated in other caches.	Longer time is taken to read the data because any invalidation request requires the cache to get a new copy from the main memory.
The delay between writing a word and reading the written value in the processor is shorter.	Since the modified data is not instantly updated in cache, the delay between writing a word and reading it is higher.
Updated data is given to the processors who have the same cache block copy that was updated.	Updated data is gets to the processor who requires it.

The problem with Write Through cache is that it

1. Needs High bandwidth requirements
   - Every write from every processor goes to a shared bus and memory.
2. Writing data will experience latency
   - As the cash is going to write to two places every time and writing to the main memory is much slower than writing to the cache.

Solution Conclusion:

A more reliable and simpler process to get data updated in cache and memory

- The cache line is updated both in the cache and in Main Memory, so it minimizes the risk of data loss
- Simple to implementation

Resources:

https://nesoacademy.org/
https://www.geeksforgeeks.org/write-through-and-write-back-in-cache/
https://www.sciencedirect.com/topics/engineering/cache-coherence
http://meseec.ce.rit.edu/551-projects/fall2010/1-3.pdf