Parallel Computer Architecture

Prof. Hemangee K. Kapoor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Week - 05
Lecture - 28

Lec 28: Coherence misses

Hello everyone. We are doing module 3 on cache coherence. This is lecture number 10. In this lecture, we are going to take a look at the coherence related misses. If you recollect, we have done the types of cache misses in our previous module on the recap of caches. Right. So, if you recollect, we were looking at three C's model of the cache and then we also added the fourth C during the cache miss discussion. Okay.

So, the first miss is normally called the cold miss that is this is a compulsory miss which always occurs because we are accessing the block for the first time. It is going to come from the main memory to the cache and obviously on the first access it is not there in the cache. Right. So, this is a compulsory miss. It is going to happen.

Then the second type of miss is capacitive miss because the cache is small and we cannot accommodate all the accessed blocks within this cache. Hence, some blocks will have to be removed before you bring in new blocks into the cache. The third type of miss was conflict miss. So, this was occurring because our cache had a limited associativity that is if we have a four-way set associative cache and if another fifth address maps to the same set, then one of these four have to be moved out. So, these moved out block may again be accessed in future and you will again miss on this particular address. Okay. Because we can only accommodate four in this particular set.

So, that is the conflict miss. And the fourth C was the coherence miss. Okay. So, this is the first miss called the cold miss. So, the first type of miss is the cold miss that is the compulsory misses. Second is the capacity misses because our cache is small and we cannot accommodate all the reference blocks.

The third is the conflict miss. This occurs in blocks which are less than fully associative size because if you have a fully associative cache and certain blocks cannot fit in that, then such type of misses will move to capacity misses and not conflict miss. Okay. And this is also called collision miss because some blocks collide with each other kind of in the set and then they remove each other. And the fourth C is the coherence miss. So, this is the new miss we are going to discuss today. Okay.

So, now this miss occurs because the blocks of data are shared among multiple processes. Right. So, if I have a block B shared in a processor and another say P1, P1 has this block and P2 wants to write to this block. So, what will happen? It will send invalidation to this block from P1 and P1 will have to relinquish B for P2 to write to it. Okay. So, in future when P1, when P1 wants to access B again, it will incur a cache miss. Alright. So because of this sharing between P1 and P2, so I will say P1 has to give the block to P2 because P2 wants to write and hence in future P1 will incur an additional cache miss which it need not have incurred if P2 was not using the block. Alright.

So, because of sharing of the data items across multiple processors, we incur the coherence miss. Now, we further categorize it into two types. One is the true sharing miss and the second is the false sharing miss. True sharing is the word says that this miss which P1 incurs was required for the correct execution of the program. In the sense, the blocks which P2 accessed and the words which it changed, P1 will use them in future that is the changes of P2 are required for P1's correctness and if this is not the case, it is the false sharing miss. Okay.

We will take an example in the next slide. So right now let us read the definition. So true sharing miss happens when a word in the cache block is produced by one processor and used by the second processor. Right. If P2 writes to a particular word and P1 uses this word in future, such misses are called true sharing miss and in case P2 modify certain words which P1 does not require, but still P1 incurs a miss, it is called a false sharing miss. When different words are accessed by different processors and they do not happen to use each other's words, but these are placed in the same block. So happen to be placed in this, this is the same block.

Okay, so let us take an example. Here I have got processor P1, P2 and P3. It is the same block. The color represents the processor and the data or the block is the same. So we are discussing about block B1 being shared in all three processors.

Okay, and as you can see the value is same because they are all reading the cache block. The first event which occurs is, P1 wants to write X equal to 4 in this block. So what will it do? In an invalidation based protocol, it will invalidate this, so this block gets removed and even it gets removed from P3. So both of them remove the cache block. So P1 successfully makes X equal to 4.

All right. Then P2 wants to read X in the second event. So P2 wants to read X and you remember that here in the beginning, if I say event 0, it is the block was valid in P2. But now this will incur a cache miss because the block was invalidated due to event 1. This particular miss I would say was a forced miss on P2 because P2 already had the block and because P1 wanted to write, P2 had to relinquish the block. Hence, this kind of a miss is a coherence miss.

This miss would not occur if there was no sharing of the blocks. All right. So when P2 wants to read the block, it incurs an additional miss. In this case, P1 has to give the data to P2. So P2 takes the data and when it gets the data, if you see what is the value of X it is going to read, it is correctly going to read the new value of X equal to 4. So in this case, this miss which P2 incurred was required to get the newest value of X.

If this miss had not occurred, it would still read the stale value of X. Third event is P3 reads Y. Even here if you see, this is going to incur a cache miss because we have invalidated the block in event number 1. In this case also, P1 provides the data. When P1 provides the data, what value do you think P3 is going to read? P3 is going to read Y equal to 3 and it is the same value it had in the beginning. Right.

So no new value has changed because P1 did not write to the variable Y. Okay. But unnecessarily P3 incurred a cache miss and had to again reload the block. Okay. So if we categorize these two events, I will say that the event number 2 when P2 incurs a cache miss, it is called the true sharing miss because the variable X it uses was really modified whereas I call it a false sharing miss because the variable Y which P3 uses was not changed at all. Okay. So that is the meaning of a true sharing versus a false sharing miss. Okay. So this slide essentially gives the summary of what a sharing miss is.

So true sharing miss is when one processor accesses a particular block and changes some words. And because it is writing to some words, other processors have to invalidate this block. Later these processors will incur an additional cache miss, they will reload the block again from the previous processor and in case they happen to read the words modified by that processor, they will incur a true sharing miss and this is required for the correctness because otherwise the coherence of the data items won't be maintained because we always want the most recently written value for a given variable. So this miss which is incurred is mandatory so that the fresh value is fetched from the recently writing processor. So these are true sharing misses.

False sharing misses again same scenario, processors end up invalidating their blocks because some other processor is writing but the words which those processors wrote, this processor doesn't need it. So this way the miss is unnecessary. So it just spends time deleting the block and fetching the new block with the same content it had already of its own interest. Right. So this is the description and we have seen the example in the previous slide. Okay. So we call it a true miss because there is a true communication of the newly defined data items.

We want the newest value to be shared between processors and this is essential for program correctness that is why it is called a true miss. And the false sharing misses, there are some words but this processor doesn't want them and so we call it a false sharing. So it reads a different word in the same cache block. Okay. So the next question is how to reduce the sharing misses. So true sharing miss if you try to concentrate on.So true sharing miss occurs because there are shared data items being written by different processors and the processor which is changing the shared data item will have the most recent value.

Newer readers will need this particular newest written value and the new processors will incur this miss. So overall the list of actions is, P1 wants to write so it invalidates other blocks. Later P2 wants to write it invalidates other blocks. So whenever a shared data item is written we end up invalidating others and in future these other processors will incur a cache miss and reload these blocks. Okay. So this is the scenario which is there and how do I reduce this so called traffic or lot of protocol exchanges which is happening.

So the idea behind this is if all these so called shared data items are packed into one same block then when a processor wants to change them one invalidation will remove this big block from every processor. It will be modified and then later misses will again bring all these new variables to all the processors. So for every word or every different shared variable I will not incur invalidations and future miss sequence. So this sequence can be broken if I can pack all the shared variables in one block and this is only possible if my block size is big. So having bigger cache blocks will help us to reduce the true sharing misses.

However if you recollect now that if I have a big cache block there will be other words also into this and there is a possibility that other processors will be using those words and not the ones which are modified and this will indirectly have a negative impact on the false sharing miss. Okay. So true sharing misses are inherent because we are going to assign variables or share variables across different processes and if I increase the block size then due to spatial locality I can reduce the number of invalidations happening to and fro and the amount of communicated data can be maximized. So I can maximize the communicated data if I have a larger block size. Okay. So more share items will fit in the same block and hence in one invalidation we can handle many shared variables if the block size is big. In a false sharing miss, if the block size is bigger then definitely there will be more words which are not used across all processors incurring or increasing the false sharing misses. Okay.

So I will say that false sharing miss increases with the block size. Now when will we be able to reduce false sharing misses? False sharing miss will reduce if the cache block is of the size one word. Okay. So if the cache block is so tiny that I just fit a single word into that cache block then you will never incur a false sharing miss because the block only has a single word and if a processor is not going to change that word it will not invalidate it from this particular processor. Okay. So here I have multiple words and it is possible that I use two of them and I don't use two of them. So these two will incur a false sharing miss if I increase the block size whereas in this case where my cache block only has a single word.

So this big block I am going to make into small boxes and if this is shared this will be a true sharing miss and if this is not used it will never be invalidated here. So this will not be invalidated, it will stay with the processor and hence not incur the false sharing miss as we saw in the bigger block case. But again because of the gap between the processor and the memory, the current trend is still on enlarging the cache block size. So this is potentially going to increase the false sharing misses. Having cache block size of one word is still not a practical design decision. So understanding those two types of misses, we are now going to go into detailed classification of these misses.

Now you would say we have already seen four names and what is more to do in this classification because we have seen cold miss, capacity miss, conflict miss and and, coherence misses. The four names are already there so what new classification should be done. So in this particular discussion I am going to treat the capacity and conflict misses as same. Because if you imagine the misses between capacity and conflict will change the category depending on the associativity of the cache. Hence, we will just call them capacity misses because indirectly capacity miss can also imply a conflict miss. Definitely cold misses are there but capacity and conflict can be treated as a single category and to further understand the details of a type of miss I am going to define lifetime of a cache block.

Now lifetime of a cache block is the duration of time it resides in the cache, from being loaded into the cache, it gets used into the cache and then after getting used what are the three ways in which it gets removed. One is it gets evicted because of replacement that is there is a conflict miss or a capacity miss the block gets removed. Even if the processor wants to use it but there is no space in the cache. So this is one reason the block will go out of the cache. The second reason of going out is it gets invalidated because of the coherence traffic and the third reason is definitely the program finishes. So the lifetime of a cache block is from the cache miss, access till it gets removed from the cache. Right. So during this lifetime we access certain words and if I ask you what was the reason for the cache miss. We cannot tell the reason when the miss occurs because when the miss occurs we only know that the block is not present in the cache. But when we bring the block and we access it we will really understand why the block was removed.

Was it removed because somebody wanted to invalidate the block and modify the words or was it removed just because we had less capacity? Was it removed because of this invalidation and I ended up using the freshly written words? So all these reasons will help me to give a correct definition of why the miss occurred. Okay. So here in this slide, we just decided that capacity and conflict will not be distinguished. Then we discussed the definition of a lifetime of a cache block, that is the time interval during which the block is valid in the cache and the validity starts from when it misses up to being invalidated, being replaced or if the program finishes. Then we cannot classify misses when they actually occur but when this block gets removed from the cache, at that time we will be able to give a label to this miss. It is like how successful the stay was in this cache would decide the category of the miss it incurred. Okay.

So whether the modified words were accessed, whether the modified words or the words we accessed were changed by others, all these things will decide the category of the cache miss. So we are going to look at the different categories and do a detailed example. Okay. So this is a complete chart. I have tried to fit it in one slide but we will see it piece by piece in the next few slides. Okay. So we will start with the first four cases and I have cropped the picture from this slide only the top left corner. Okay.

So you can see the flow chart on the right hand side and the categories on the left hand side. So we will concentrate on the flow chart. So I will start from here. When a cache block is accessed and there is a cache miss. So we need to identify the reason for the miss that is when we go it is not a cache hit but a miss then what is that time.

So the first path I will take is here. So this is the first reference to this block by this particular processor. Right. So this processor has never accessed this block it is the first time I am accessing this block so I will take this left hand side path. Then I have another decision box which says, is it a first access to this block system wide, that is there is no other processor in the system which has accessed? So when I say system wide you have all these say 3, 4 processors and that is the block B from the cache. Right. So this was sitting in the memory and suppose P1 accesses B and nobody has accessed B till now in system.

So that is the meaning of first access system wide. If yes, this is the first access system wide then I say that this is a cold miss or a compulsory miss. Okay. So this is the reason I label this miss as a cold miss because P1 is accessing B for the first time and P2 and P3 have never accessed B at all. Okay. First category done.

So case 1. Now let us see case 2. Coming back to the flowchart, if it, is it a first access system wide? Now I will take the path of no. Now what does this mean? This means that P3 had accessed B at some time t1 and at time t3 suppose P1 starts accessing. That is P3 had already used B and now P1 is wanting to use B.

But for P1 this is the first access for B. Alright. So it was not the first access system wide because P3 had used it and now we further categorize it. Did P3 write to this block? Was this block written before? So if B was modified by P3 or B was not modified. If it was not modified then it is called a cold miss. Okay. So it is a cold miss for P1 because it is accessing it for the first time and nobody has changed this block ever.

So that is the explanation for case 1 and case 2. These are cold misses occurring on previously unwritten blocks. Then case 3. Coming back to the flowchart, we will move on to this third decision box.

Suppose P3 had changed B. Okay. From here I am going this direction. Written before, yes that is P3 had loaded this block B. It has changed this block B. And when P1 incurs a miss, P1 it is a cold miss.

You will say yes, it is a cold miss. Why new names? But P1's cold miss also comes with more adjectives which say that because P3 had changed it, P1 is kind of going in the coherence miss path and getting the freshly written data from P3. So P3's data will be provided to P1 on its cold miss. Okay. So modified data, this is modified data or words are accessed during the lifetime. That is if P3 changed certain words in this block, P1 would get the new data items.

Now if P1 uses those data items, I will say it was a true sharing miss. If P1 does not use those data items, I will call it a false sharing miss. Okay. So here again I will explain once more. If you look from the top, from the flow chart, what is the reason for the miss? Is this a first access by processor P1? Yes it is the first access.

But is it a first access system wide? No, because P3 had accessed the block. Now did P3 change the block? Yes P3 changed the block. Right. So we are here. So P3 changed the block, but did the words which P3 changed was accessed by P1? No, then it was a false sharing miss.

If the words changed by P3 were accessed by P1, then it is a true sharing miss. But at the same time, the word cold remains with it. So case 3 and case 4 is a cold miss because the processor had never accessed this block. But addition to this cold, I am going to add two more words which are either true or false because other processor had written to this block.

Okay. So that is the meaning of case 3 and 4. So next we will start case 7 and 8. Again part of the flow chart is pasted here. We will start from the top. Reason for the miss. Now look, we did not go in the left direction because the left side was first accessed by this processor.

This other means, this is not the first accessed by this processor. That is I had this block initially, but that block got removed for some reason. Okay. So if I go back here, you will see on the left hand side, the first reference to the memory block by the processor is the left arc and here I am taking this saying that this is not the first reference. Okay. But even if it is not the first reference, why was this block absent? It was absent because either it was replaced or, or it was invalidated.

So these are the two reasons. I will take the invalidation reason. This block was invalidated from this processor because some other processor wanted to write to this block. So P1 and say P3 or P2. So B had this block and because P2 wanted to write, it was invalidated here. Right. So B was removed from P1 because P2 wanted to write. Now when P1 wants to access B again, it will incur a cache miss and it comes to the cache and checks.

Is the old copy still present in the cache? That is, this was the cache of P1 and B was there, but I will just change its valid bit to invalid. I just changed the state of the cache block because P1 started using it. It was invalidated, but the data, the stale data is still there in P1. If the stale data of B is still there in P1, it is not valid, but it is still there, then we will take this path.

Okay. Old copy with state invalid is still there. If it is still there, then we will check whether the modified words accessed during the lifetime are used by P1. That is the words which P2 changed, were they used by P1 or not used by P1. So accordingly, it will be either a false miss or a true miss. If P1 uses the words newly written by P2, I will call it a true miss.

If P1 does not use the words written by P2, I will call it a false miss. So this is, these are the two adjectives. Now this miss had to occur because we had to replace the block or it was invalidated. Right. So the block was invalidated. P1 still wanted B. P1 had the capacity to keep B, but we had to remove that because P2 wanted to write.

Hence we will not call it a capacity or a conflict miss, but we will just simply say it is a pure miss. It is not a cold miss. It is not a capacity miss.

It is not a conflict miss. Okay. It is a pure coherence miss. So case 7 and 8. It is either false or true depending on the word's access and it was previously invalidated in this cache, but yet to be replaced. Right. So we did not remove B from this. Hence we will use pure. Why are we using pure? Because we did not remove the cache.

We still have space for B and B is still sitting there. So it is not a capacity miss and then either it is a false or a true sharing miss. Okay.

So that is case 7 and 8. Moving on. Case 9-10. Okay, case 9-10. Take the right hand path from the top. The reason for elimination of the last copy was replacement. Previous slide said that the block was invalidated. This path is the block was replaced.

That is if I use the example, P1 had the B, but it was evicted by P1. Nobody else removed it, but P1 had to remove this block. Because P1 had to remove this block, why will it remove? Because it had less capacity. So the first thing we are going to classify it as capacity. So I am going to say capacity is the first reason for this miss to occur.

Because we had to remove B. Okay. Replacement of the cache block results into the capacity issue because of capacity I replaced it. Now coming to this decision box. Has the block been modified since replacement? Okay. So we replaced the block, but did it change after that? Since, so this is coming back to here if you read case 9. Capacity or conflict means since the block was previously replaced from this cache.

And the words in the cache block not modified since the last access. So coming here were the words changed? They were not changed. So this is not related to any coherence problem. It is pure capacity miss.

Because nobody else changed this block, nobody else wanted B. They did not touch B at all. We removed B and then we have to reload B. Okay. So if this B is not shared, it is not changed by anybody, then I will call it a pure capacity miss. But in case some P2 in the meanwhile accessed B, when we removed B, but P2 acquired this B, it wrote to this B and later when we incur a miss, we get the newest value of this B items and hence we can say that good we removed the block because anyway somebody would have removed this block and that is why we call it a true sharing miss. Okay. So case 10, it is a capacity miss because we removed the block, but some other processor has already changed some words which we access. Right.

So it is like we get these updated words for free. We had already removed B, P2 changed B and we use those new words. Hence it is called a true sharing miss. So it is a true sharing concatenated with capacity.

So these are some pure categories. Now we look at mixed categories. So mixed categories have multiple causes of misses. That is, if you see a scenario, block replaced in A's cache, so A removes the block, then B writes to that block and then A again reads the block. Right. So A had the block, processor A, then processor B takes it and then A takes it again. So overall it is a capacity miss if A removes it. It could be an invalidation miss because it could be related to coherence and then true and false sharing misses.

Okay. So the remaining categories are in the mixed category classifications. Case 11 and 12 on this slide. Okay. So we start from the top. The block was replaced. Not invalidated but replaced. So when you say replaced it is related to capacity.

So when we replace it comes with this capacity keyword. If you see here. The cap word is written here. Okay. Has the block been modified since replacement? That is P1 had the block B, it was removed, evicted by P1, later P2 got the block B, it has written to this block. Okay. Modified word, so it has written to this block and did we use those words which P2 has changed? If yes, true, if no, false. Okay. So you would say this is similar to case 9 and 10 because in 9, 10, 9, 10 was again capacity and capacity but here there is a new word added called invalidation. Now why is this word here because if P2 had wanted to write to this even if P1 would have not evicted this block, P2 would have invalidated P1's block. Getting it? So P1 evicted it in our case but if P1 had retained the block and P2 wanted to write, P2 would have invalidated this block indirectly in future in case it was not evicted.

Hence we add this phrase inval to this. So case 11 and 12, block is replaced. It is either a true or false sharing miss. So always sharing gets the priority. So when we write a label for a miss, we first write the true sharing or the false sharing and then we write information related to replacement or invalidation. So capacity, inval, all these words come at the end.

So when you phrase the classification, it is false sharing cap inval. So that is 11 and 12. Okay. So case 5 and 6, reason for elimination, it is not replacement but invalidation. Okay invalidation. Now this box, is the old copy present in the cache? No. Right. So we evicted the block because somebody else asked us to remove it. So invalidation message came, we deleted the block and after getting the block deleted, so that particular location in my cache got replaced with a new block.

So this, the block B was sitting here, it was invalidated. So after, although it was invalidated, after a while the same position of B, I filled with a new block C. Because B was anyway not there, I replaced it with a new block. So old copy is not present in my cache block. So B was removed and was replaced by another block C.

Later I want to again access B. So we accessed B, then we accessed C, which was sitting in the same location. Again I want to access B which sits in the same location. But in the meanwhile through this path if you see, B would have been changed by another processor. So here were the modified words, that is when the B was not in this cache and it was reloaded into the cache, during that were the words changed? and did I use them?, will decide the true or false sharing. So here again we have either false sharing or true sharing.

It is happening because of capacity if you recollect because we had to remove B and B got replaced by C. Okay. So how do I name this? It is first a sharing miss, because sharing takes priority. Then I need to add some more information. One more information is, is it a capacity miss? A capacity miss would occur if we evict the block. And in my current example, the B got replaced by C, that is that location was filled by another one but not immediately but little later.

So capacity problem would have come later but before capacity problem came, the block got invalidated. Hence we first tried the word share, true or false and then what happened first? Inval happened and even if inval would have happened or not, we had a capacity issue. Why capacity issue? Because the old copy was not present in my cache. Okay. So this is how we name the cache miss. Invalidation happened before replacement. Okay.

So with this we have covered all the 12 cases and also understood how to give names to them. Right. So we have seen all the misclassifications in these slides, all the cases have been done in detailed manner. And so in the next lecture, we will see a detailed example of this. So with the classification, we finish this lecture. Thank you.