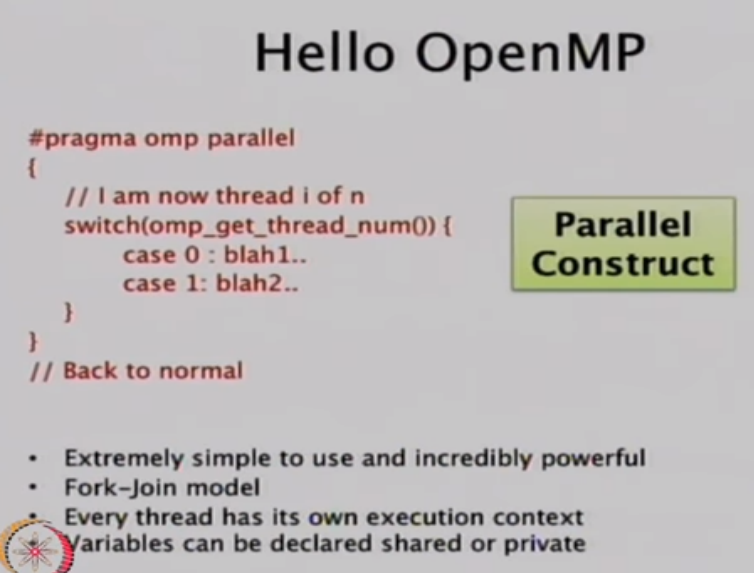


Parallel Computing
Prof. Subodh Kumar
Department of Computer Science & Engineering
Indian Institute of Technology – Delhi

Module No # 01
Lecture No # 05
Open MP

Okay let us begin we have not that long left so we are taking about open MP and what I was saying is that I am going to give that high level design and stuff to get you started with open MP and do not expect to learn entire open MP in class everything that you are going to you need for this assignment it is not necessarily going to be in the class okay.

(Refer Slide Time: 00:57)



Hello OpenMP

```
#pragma omp parallel
{
    // I am now thread i of n
    switch(omp_get_thread_num()) {
        case 0 : blah1..
        case 1: blah2..
    }
}
// Back to normal
```

Parallel Construct

- Extremely simple to use and incredibly powerful
- Fork-Join model
- Every thread has its own execution context
- Variables can be declared shared or private

But enough to for you to figure how to figure out the rest and the design of open MP is extremely simple and powerful actually in terms how you parallelize score. So if you have done for example a thread programming here probably here you would not get 6 or 7 here is an example of how would do open MP based parallel programming this is compiler supporting.

Of course you put in your court instruction for the compiler for it to be able to parallelize and the main way or one way which is give instructions to compiler at its ash tag month instructions and all open MP pragma's being with ash pragma OMP and then there are lots of different types of OMP instructions, OMP guidance that you get to the compiler okay here is one simple one it says

pragma OMP parallel and then there is open space and close space. It is that region is to be run in parallel.

And then you write some code it may be function call it may be at somewhere else some other variable whatever is your desire. We will talk in more detail they are all available but there are features that we need to understand it is also hidden here it is not explicit in this thread how many different times to run this common thing okay. That is driven typically by environment variable it can also control it expressively for example in this parallel call you can say pragma OMP parallel ten.

So that mean you run this ten times okay everybody is running concurrently if you have ten CPU they are running exactly at the same time. If you do not have 10 CPU's then their 10 threads which are going to text which with each other okay. And so the notion of thread is still in valid so in principle we are saying I want 10 threads each thread should run at the piece of core okay when I have not searched anything then default that getting used.

And most compiler default is the number of course so if I run it on my laptop it is dual core CPU so it is going to generate two third. There is a dynamic version also you can create these also at run time okay but in this case the number of threads it is not specified yet is predetermined. Yeah so if you know you are hyper threading you can say do not run two threads run 4 right but this an in fact if you have turned hyper threading on when in response to your instructions compiler is executing in library function right it has its Concorde.

And it is going to query the systems asking how many course you have if you have any hyper threading with dual core machine then it will get the answer you have 4 okay. And so it will think that you will have four course and it will run forth. And so inside you have do something like switch a variable a function called that it may tells you are one of so many threads right it says my thread at its 0 your thread as it is 1 and thread at it is and so depending on your thread ID is 3

And so depending on your thread ID you can do different things if you want exactly it is not a piece of code that for which parallelism is to be figured out by the compiler it is a piece of code by the compiler is to generate an threads to replicate the execution of okay. And once now you got suddenly N number of threads running in piece of code right you got to figure out how the tables values reflected with each other and all that we will talk about it.

Once they are done with their execution we are going to come to the end of the construct and once all of the reached the end of that construct or end of their pieces of code once thread to continue on from their okay. So there is an implicit barrier which is at the end of that code at which all of the thread come and stop and after all has reached there they join into a single codes four join okay.

So in open GL terminology there will be master thread that is going to reach so there is a execution before this is an piece of code somewhere in your program and the execution has reach this point. Whoever was executing here might have been one and the only one or might have been one of many already there is some of the parallel construct that poured of many okay that who reached there is called the master thread.

And it is going to generate N -1 new threads which all will die at the end of the barrier when they join and the master will continue. So each thread may or may not reach this OMP pragma they have their piece of code for some condition they may be a jump for some condition you reach this. One of them reach then everybody will replicate ten times right nesting is allowed is actually turned off by default.

Because of that buggy in the beginning and also people have their trouble trapping mind around hot to nest parallelism but it is not buggy any more at least it is still may had a problem tapping their minds around it you can turn it on you can nest it okay. Nesting means you may have one OMP parallel construct inside another bigger OMP parallel construct. No there is each there is contact switches so because do not know how many processors there are we do not know how many thread we are going to run.

So context switching is implicit and then there is this thread mechanism that the OS understands, implements support and so context switching is allowed among threads. And so and two threads get forked off and each unit executes encounters another OMP pragma parallel. So each will fork off another so now got 4 I may not have 4 at the same time.

Because this may be running at its own space this may be running at its own space may not have reached the pragma yet. This may have ended the pragma okay there is no notion or understanding that these are running at the same speed. But yes it is possible that at any given time for threads may be run it is possible that there is two threads are allowed because this guy forks and this guy up 3 forked into 2 this guy as forked and this guy not forked yet.

This has not joined and this guy not forked it has much more baggage than fork okay the sharing is either implicitly written or explicitly provided but you have essentially set of variables that are shared meaning that they are shared across this group that has been created at the parallel construct meaning that anybody who writes to that variable is communication with threads in that variable what the value of that variable is?

So what it says that not talking about types of variables yet those I have got a shared variable this code does not have much shared variables okay. Inside this 0 case one etc., when I say $N = 3$ am talking about the exact same memory address that some other side is talking about okay that is all it means the mapping of the name to the address that is controlled by using either explicitly or implicitly there will be confusion yes it is your job to fix that confusion but when you are sharing then yes you have to somehow figure out how to make those updates with systems each other.

(Refer Slide Time: 12:25)

Execution Model

- **Encountering** thread creates a **team**:
 - Itself (**master**) + zero or more additional threads.
- Applies to **structured block** immediately following
 - Each thread executes separately the code in {}
 - But, also see: Work-sharing constructs
- There's an implicit **barrier** at the end of block
- Only master continues beyond the barrier
- May be nested
 - Sometimes disabled by default

I mean very quickly talk about what memory consistency model is assumed by open MPI which means that every implementation of open MP. So open MP is not a software right it is something that is going to be built into a compiler open MP do not get a library called open MP just like through might get a thread or windows thread. So different implementation software of open MP may have slightly different properties.

It open MP says that these are the things are undefined these are the things are that have to be guaranteed so there some consistency does memory that will have to be guaranteed on limitation software open MP. It is just a model it is a specification GCC implements it visual studio implements it how compilers implemented right there may be a different versions. So the latest version of the open MP right now is 3.0 GCC 4.3 also implemented 3.0 came out in 2008.

Visual studio is that I think 2.0 there was also 2.5 so GCC 4.1 will be at 2.5 or 4.2 will be at 2.5 and similarly Intel's compiler Intel actually implements 3 Intel's compiler will also 3.0 let us compile. Okay here is the execution model some thread is running through its code okay and it is going to encounter pragma from the source code point of view and at that point either through the pragma instructions or through some other library function that we may called earlier or through an environment variable they had set earlier.

Or through the default that open MP sets some number will be known that so many thread needs to be created and that number -1 threads will be created because one of them is master thread who encountered this they are all run their versus they are all get own PC's right is honesty goodness OS thread and they will run this piece of code and at the end of the Piece of code they will wait for everybody else to get the same piece of code okay.

Once everybody gets there the master will proceed and at that time there is now trace of $N - 1$ threads. Because this is built upon process threads or windows threads or some other threads so the compiler actually assuming that there is a thread implementation okay. And when you for convenience when you built your code compiler have quick lag to enable open MP okay for example if you say GCC thread dash F open MP.

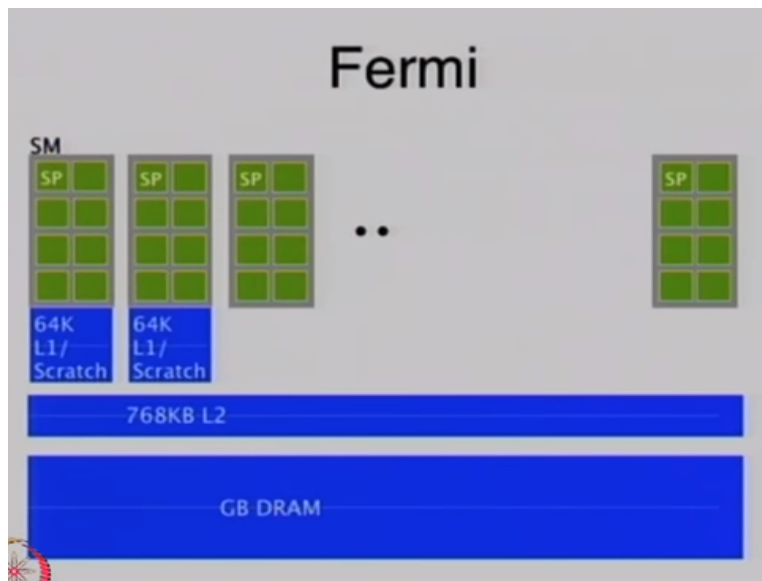
If you use visual studio then there is a pull down thing that says enable of open MP okay internally when you say -1 open MP it is saying dash L thread P thread okay. And it is expanding all those things it is expecting all those things are be read it provides that I include load pass wherever you may have installed the stuff at the compilation time of the compiler but you simply hide all of them behind N bash open MP and all those internal mechanisms for the threading thread creations thread fork and join is going to be call by the compiler too.

Alright the memory model we probably if I am going start it going to take 10 minutes time to explain it so my 10 minutes will stop here it is six thirty we will continue with the memory model on Saturday. Because there is a class on Saturday and hopefully finish as much as open MP we are going to talk about on Saturday okay any questions? Locking is on top of the model right memory model says when two people share what is underlying implementation of open MP provide what must it its support okay.

And we talked about basic notion of consistency right if there is variable that says 5 somebody must have written it that is basic expectation right. It should be 5 that just appeared that from the wire. We will talk about more details we will actually talk about memory consistency models in more broader sense later on the course. We will talk about what it means of open MP there are

couple of things I want to add or revise from the NVidia architecture discussion that we were having.

(Refer Slide Time: 18:16)



One is that we were having copy pasted from G80 architecture to semi architecture to semi architecture that one had 128 kilobytes of L2 which was not getting used. But by the time it went to for me between G80 had gone up a couple times also but by for me it had become 768 kilobytes which is the current size the other discussion we had about these size the number of threads expectant get on and that limit is 48 times 32.

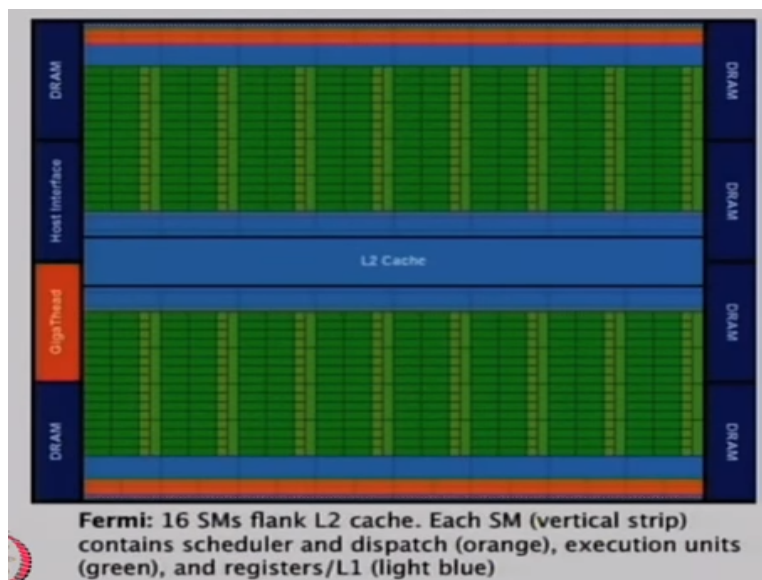
So there is something called a war which is SIMD with 32 threads have one instruction another 32 may have another instructions and so there are 48 of these that can run or be live inside one SM at the given time which means the 1500 something 1536 or something threads whose contacts can be maintained together okay you made a reasons why you cannot all 1500 of them there is either they use these much memory more memory together then you have or register usage is high or so on hence so forth.

So in principle you can rub that many thing threads together no number of course is only 16 on a single SM. Number of SP's is 16 okay in one SM there is 32 of them no within one SM you can have 48 SIMD groups right 1 SIMD groups is 32 threads and you have 48 of them live 16 processor 16 execution units but they actually run 32 because you do 2 every clock. So effective

you can think of it of having 32 execution units although that is not physically low but logically you can think of that.

And so 32 threads run in a given clock in 1 instructions okay on 1 SM but 47 other groups are 32 threads may be waiting to run their instructions while this guy running one instructions right. So 48 such live groups may be their at any given time that is the maximum alright. So let us come back to so this discussion which just have yeah 512 total course there were 16 SM's.

(Refer Slide Time: 21:48)



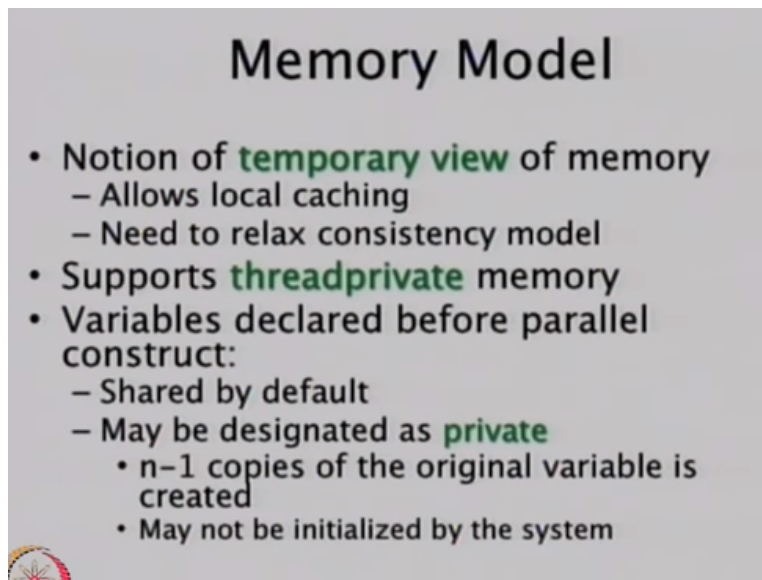
So if you look at this slide this showed the SM's right the pair of each column on so one SM you can rate at the top and rate at the bottom alright. So getting back to open MP very little variable in their some details that are different but the overall high level organization let us not change it. We will talk a little bit more about it when you start to program things but in compiler you actually can provide output it is a cross compiler.

You can compile this different architecture and there is this intermediate language which is above sitting above the assembly language and below the high level language PTX okay and idea is that it will compile to PTX which then will subsequently compile your instructions if that changes which is does from testla family. If the user takes that into account it can do much better job of it from the compiler well okay and the truth is that is could have programming model exposes to the user all those details like how many SIMD while thinks you want to run right.

Of course you can say more than 32 but then it is going to break it down into 32 so once you and other similarly how memory is accessed and so on if you organize that memory access or efficient on a given architecture then you will be better half you will on better architecture right it is possible to one thing I have said earlier I should repeat it is not that different NVidia GPU architecture even they are changing are that start different.

So similar optimization will work on most of them to it is possible to do the same thing that you would be doing by a compiler interpret and do it and there is a research on going on getting the compiler to do better and better at it. Right now the state of architect is state of the compiler can do better than the human and I suspect it is never going to be there but at some point it is going to good enough and then you start using.

(Refer Slide Time: 24:41)



Memory Model

- Notion of **temporary view** of memory
 - Allows local caching
 - Need to relax consistency model
- Supports **threadprivate** memory
- Variables declared before parallel construct:
 - Shared by default
 - May be designated as **private**
 - n-1 copies of the original variable is created
 - May not be initialized by the system

Okay now back to the open MP programming model we have talked about the execution model which said that there is notion of threads and at any instruction runs so many threads in parallel okay. You need to understand what the model in terms of the shared memory is in what kind of consistency is going to get provided very expressively in the specification of the design is given that different processors may have local cash.

May be sharing memory somehow talking to bus through whatever interconnect through whatever sequence of switches or other processors and so it is understood that sometime memory access may be slow and something will have to be cached and so there is this notion that a processor has a local view of the memory. So that same way same variable it may be sitting in cache with earlier going to 39 but in the main memory its value is something else okay.

And the consistency model accounts for that there may be different variance of that memory location and that's why it is very weak consistency model. When we as I had said that will discuss in at a theoretical level what consistency model means will realize that this is in that spectrum of weak to strong consistency model among or weaker than what is officially called the weak consistency model okay it is we can call it the weaker consistency model.

There is a notion of private memory meaning that I will generate a thread and generate shared memory but there may be something that sits in the main memory but is only visible to this one and private to this thread. So you can create thread private variable and there are this shared variable which means that others have access to it but it does not mean that it is in consistent state memory on line.

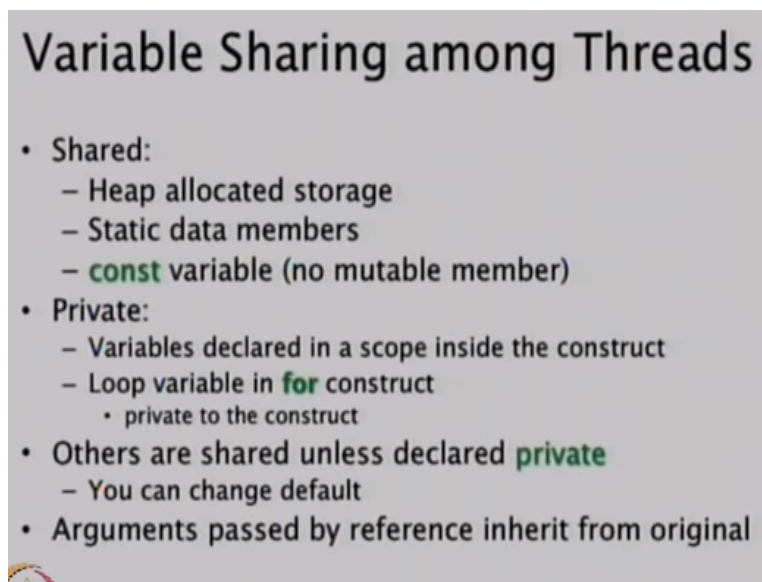
There is features to make it consistent if you want to but it's possible that different people are seeing different values of that variable because they have it locally cached. When an so the time up of shared the temporary view means that whenever the parallel regions get created right which says that now am going to start 4, 5 threads each will get local address to have to do local work on that shared variable and at some point they are going to consistent side may consistent those local copy okay.

In some says this is also true but right only thing is in private there is notion that are going to make consistent unless at the end of the parallel consistent thread you said that I want these private copies to be collated in the certain fraction but there is clear notion there is N copies of that variable and you have specify what to do with N different values okay. Let us some region running in parallel you can still do it with thread but you will get a single process okay they are going share that process table.

Not using the shared memory open MP program model okay you that using in MPI type programming model which we will talk about may be among from now and we get to assignment to. And then this I will repeat later on so when these N-1 copies of the private variable are created again it is user again say something how to initiate those copies it is just that this is address allocation.

So has controlled over all though guys some default behavior define user has control over how those memory copies get initialized and how they get later collate. That is for the private the shared case there is a consistency model okay now there also explicitly saying this is a private variable and this is a shared there also something that are inferred okay.

(Refer Slide Time: 30:44)



Variable Sharing among Threads

- Shared:
 - Heap allocated storage
 - Static data members
 - **const** variable (no mutable member)
- Private:
 - Variables declared in a scope inside the construct
 - Loop variable in **for** construct
 - private to the construct
- Others are shared unless declared **private**
 - You can change default
- Arguments passed by reference inherit from original

On this slide you see some restrictions on what must be inferred what can be inferred what has to be specified. So anything that you have to allocate and heap is can be shared okay the static data numbers inside a function inside whatever the variable that has been declared to be static are shared. Constant declared variables something are shared as long as sometimes you have a constant variable that has to members it may be constant type obvious members that are mutable.

So when I said the constant things must be shared I mean things are truly constant do not have any fields and you remember that are long constant. Anything that's they cleared inside as a local

variable function right in this case there is parallel construct that are talking about if it inside scope of the parallel constant it is private okay because it is essentially you are saying made copies of it.

Everybody has a local copy of it when you make a loop what should happen this is without saying anything sorry. It depends on what kind of loop right if there is loop that is being shared by saying I am going to run $I = 0$ to 3 you run from 4 to 7 this guy will run from 8 to 10 then you got to have different copies of it compile. If you say in fact this will make a little bit more sense when we talk about the actual construct details there is a parallel construct and there is a loop construct.

When inside a parallel construct you write a loop it is says $I = 10$ do something and that I is shared what will happen I want to go to 0 to 1 and that guy is 3 to 4 and this other guy is doing something else they are all running from 0 to 10 because we are working of and everybody doing the same thing. I also want to run from 0 to 10 you also run 0 to 10 but we are not at the same page we are not running at the same international level at a given time and so we better maintain our own copies right.

Also whenever the loop is inside the parallel region right for some reason multiple people or executing either this entire loop or a part of this one you need the variable that is being used as a loop iteration variable to be unique meaning that you want to loop construct want to shared. See there are specified loop constructs okay and for that they compile the open MP is preprocessing base scheme right so the compiler has to be generate code by inferring certain.

So if you specify that is going to make that inference much harder although you can write non loop construct codes and then you can do shared variable and do whatever you want with it and that are arguments that are going to be passed from function to function if we pass a shared argument then it is a shared passed private argument that is a private thing.

(Refer Slide Time: 35: 30)

Relaxed Consistency

- **Unsynchronized access:**
 - If two threads write to the same *shared* variable the result is undefined
 - If a thread reads and another writes, the read value is undefined
- **Memory atom size is implementation dependent**
- **Flush x,y,z .. enforces consistency. Specs say:**
 - If the intersection of the flush-sets of two flushes performed by two different threads is nonempty, then the two flushes must be completed as if in some sequential order, seen by all threads.
 - If the intersection of the flush-sets of two flushes performed by one thread is nonempty, then the two flushes must appear to be completed in that thread's program order.

Alright this is about the consistency and this remember that this is in the context of having temporary views temporary copies of variables even when it is shared is not even talking about private so if it is two threads right to the same shared variable and they are not synchronized with respect to each other. Then after the two have written I do not know the value of the variable it will be the one of those two but I do not know which one okay.

So it is undefined for the final result is going to be and if the thread reads the another writes again the same thing the unsynchronized I do not know whether the read happened before the write or after the write. So whatever what is being read is undefined it mean undefined does not mean junk undefined means I cannot precisely say what would I happen okay. And the amount is going to get read from the shared memory this is about again putting it in cash is system dependent.

If it read a struct then different fields of the struct different members of the struct because the entire field of the struct was not atomic will have some from before that right somebody else was writing that struct and some field and some members from after the write okay. So what atomic if the system says that you can write 16 bytes at a time. So if I struct that has 4 hints then am going to be guaranteed that the entire read is atomic.

So either am going to get all 4 hints from before the other write or after the other right but if the atom size is 4 bytes one in (()) (37:28) then I have no idea which of the fields come from before the update and which of the field is from after the update okay. I think that byte is definitely suppose to be atomic and I do not think that expressively said in the specification. It is system dependent right you will know what its item size is.

It is not that atom size is supposed to be unknown to you it is that atom size will differ from implementation to implementation okay which means that if you have a windows open MP you may get slightly different behavior versus running it unlimited okay. Although it turns at its really determined by Intel and so atom size are in this cases 4 bytes and there is additional construct which makes the temporary view and the real view and the memory view of a variable consistent.

So you can at any time say flush something is not the same flush that you may have seen in the C++ library. I was not flushing something in on to the file into the desk okay this is flushing to memory so this is not that flush from this is the construct flush to be inside ash pragma open MP flush. And in that flush at one time you can say these are the variables I want to flush okay is not necessarily a single variable flush.

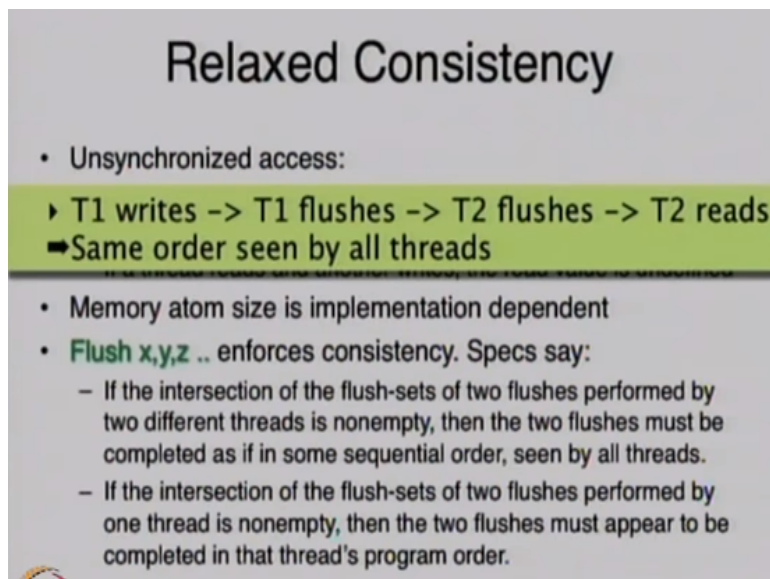
And will come back to it this is this something needs to be understood properly because that tends to this is one place you will typically find many bugs and so now am reading from specs the actual definition of consistency. So recent intersection of the flush set is thinks that are inside the flush directive. Say flush X, Y, Z so that is the flush right if the intersection of the flush sets of 2 flushes which may be performed by different threads are nonempty right.

So if some variable here also exist in that flush set then their relative ordering is important in specifically this fashion the two flushes is must be completed as if there is entire flush has happen before the other one or entire flush happen before this one you cannot say A got flushed and the remaining things that guy got flushed and this guy remaining is things got flushed. So the entire flush is atomic and if it is empty then does not matter which order you determine okay and that part is not listed here although is same simplicity.

If two flushes are empty nothing is specified which are they have atomic if you are talking about flush sets of the given program their flush must happened in the order that they appear in that circle not a given program but in the given program thread execution order that must appear in the given threads execution order. So if a thread says flush A. B. C and then flush X, Y, Z cannot order A, B, C okay.

This is very similar to the consistency model I had mentioned earlier this is called the ZV consistency model reason it is not the weak consistency model is only among flush sets the real V consistency model says any access anywhere has to be similarly order with respective every other access that is being obstructed out or moved at higher level or saying this is only going to apply of the special constructs that the use are inserts because that is necessary and it is only among those special flush constructs that the V consistency model of us okay.

(Refer Slide Time: 42:28)



Relaxed Consistency

- Unsynchronized access:
 - T1 writes -> T1 flushes -> T2 flushes -> T2 reads
 - ➡ Same order seen by all threads
- Memory atom size is implementation dependent
- Flush x,y,z .. enforces consistency. Specs say:
 - If the intersection of the flush-sets of two flushes performed by two different threads is nonempty, then the two flushes must be completed as if in some sequential order, seen by all threads.
 - If the intersection of the flush-sets of two flushes performed by one thread is nonempty, then the two flushes must appear to be completed in that thread's program order.

So what that means is that which was not supposed to hide but in any case so this is showing you 4 flush sequence on a single variable. This is going to guarantee that something that recommend to variable that was read by any other variable it says that T1 is thread 1 is going to write to this variable then it is going to flush the variable right and because that is inside the same program it has to be ordered right happens the flush happens.

Now we are talking about T2 is also going to flush this variable and as long as the flush is ordered this flush any read after that flush from T2 will be able to read this variable okay. So this if you want data from here to reach there you are going to have guarantee this now there are sometimes I probably we will show you a couple that we can look at the specs also some things are automatically flushed from the context which is given that you are requiring the flush.


Other cases after flush for example at the end of the parallel construct everything is flushed ok everything that is in the shared class okay. No nothing ensured it is up to you that is users problem right user have to ensure that whatever order is being desired is enforced all this is saying is this is the ordering that will ensure that T2 read happened after T1's right okay. How you enforce this order? No you make sure that T1 flushes and then T2 flushes that by doing some lock right.

So the lock may be shared lock may be separate consistency lock is by default flushed lock is one of those things otherwise it would not make any sense you would not call it as lock right.

(Refer Slide Time: 45:19)

Beware of Compiler Re-ordering

<i>thread 1</i>	a = b = 0	<i>thread 2</i>
b = 1		a = 1
if (a == 0) { <i>critical section</i> }		if (b == 0) { <i>critical section</i> }



So now let us look about one example what I was talking it I have got two sets the threads on the left it says thread one thread on the right is thread two and in the beginning some variable will be both and these are shared variable MBI shared menu are said to 0 okay. And thread 1 says thread

$B = 1$ and which it is my indicator when $B = 1$ am indicating that I am doing something special am going with the critical section.

And it checks now $A = 0$ in the other guy has not set A to 1 then go into the critical section do whatever you want to do after you come out you probably will set B to 0. Similarly the other fellow which also want to allow into the same critical region sets indicates that it is going to go into the critical region checks that the other person is not in the critical region and goes if it is not okay.

Will it work? So my goal is that both of the should come into the critical region at the same time which order there is the thing that's is why the title says compiler can reorder your statements okay. So the compiler sees oh you are setting $B = 1$ but nobody is using it this piece of code does not anywhere access it B . So what do I bother? Does not said it work may be later on it uses its B because something else it is going to happen it will reorder it.

So it can always go one move it here without any risk it might fail similarly the other guy also reorder $A=1$. And then there will be problem and that they have not indicated that I am going to go in the critical region and I have gone into the critical region. When I came to $A=1$ other fellow have not reached there yet I go in but I did not go that I am going in so other guys are come in and go on alright.

So you want to compiler to turn up to optimization B is shared variable B is so shared variable should not be in order you are saying that is the position you might tell. But it is not the open MP positions which means that if you do not specify any ordering then the compiler is going to reorder assuming that you do not need it you know that it is critical region you have decided that is what means that set that I am going in which means you have enforce that thing remains it okay.

And that is why flush comes in now if you say flush again unless the compiler goes something special about flush it will say these two together right $B=$ flush B both of them go there is the same promise okay. So how you would solve is using this flush set if you make sure that flush A

B is here and flush B is there then you know that these two as sum how related because they have common they have intersection that is non-MP.

So these flushes must be order because with respect to each which in turn means the $B=1$ must be in respect to the second guys flush okay. And so now you can figure out while we take this big why it should or whether should work and then we will start the next section after that okay only the flush. So the compiler does not say what has to be ordered the compiler says that the system is going to guarantee that flush of A and B and that other flush of A and B are order with the respect to each other.

There is known relationship between these two flushes after the fact when the flushed are happen you know what is the relationship? Other flush yes yeah inside there is set it is not just an order inside the set. Yes it does not guarantee there this is only ensuring that both cannot come in at the same time. If it is both of them think that the other guy coming them go through probably try again.

It says that $B=1$ has to be happen before flush right and that flush has to happen either before flush right or after flush so you take those two cases. If it is happen before that flush then it is the way was intendant if it is happen after the flush what does it mean that means $A=1$ that flush happen before this flush $A=1$ happen before that and so when it comes here A will equal to 1.

Yes A and B both can be 1 you can do that but it is better the use simply the mechanism that is provided flush it that happen you can also enforce compiler ordering by using it exactly okay.