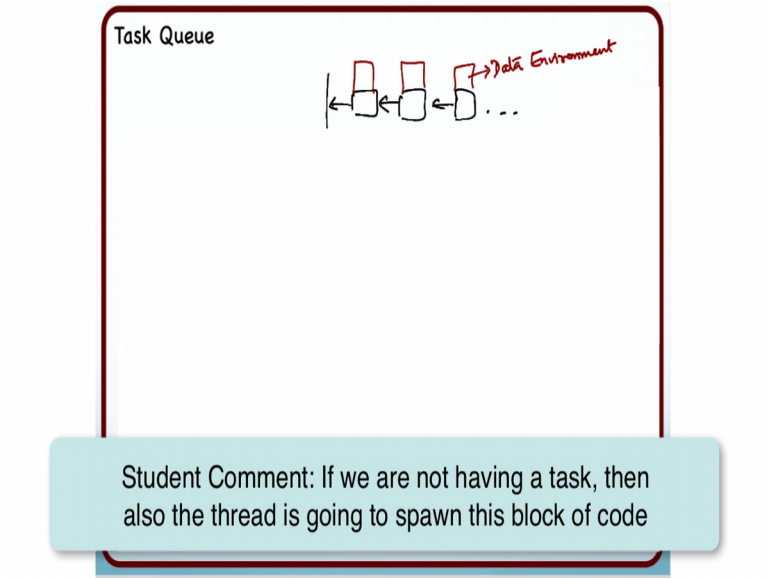
Introduction to Parallel Programming in OpenMp
Dr. Yogish Sabharwal
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture - 28
Task queues and task execution

How do tasks actually work? So, the way they work is that there is something called a task queue that is maintained right.

(Refer Slide Time: 00:07)



And every time you create a new task right you say hash pragma omp task a new task is added to the queue right a queue is maintained and tasks keep on getting added to this queue. Whenever some thread finishes its existing task or has trying to do some work it basically picks up a task from this queue and executes it. So, as and when the work gets created the work keeps on getting added to the task queue, and as and when threads are available they come and pick up the word from the task queue and execute that piece of code.

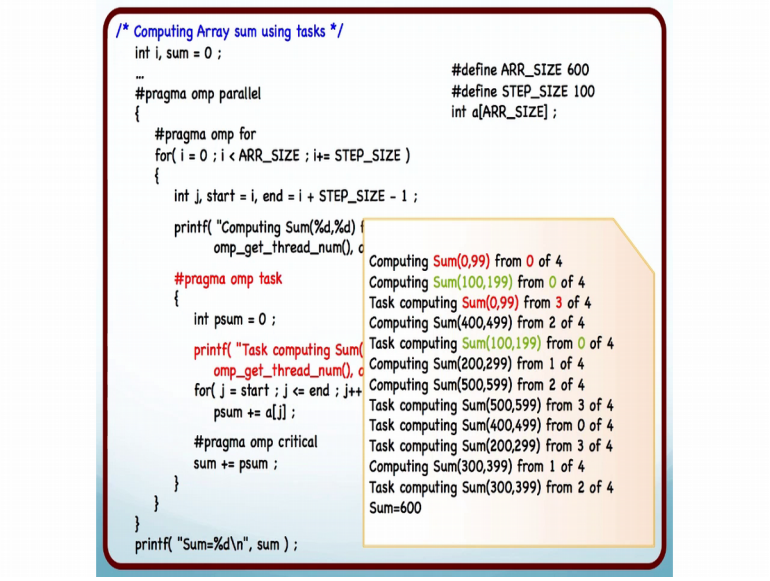
Student: If we are not having a task then also the (Refer Time: 00:49) thread is going to spawn this block of a code.

If we do not say hash pragma omp task?

Student: Yeah.

No they not going to spawn they just going to execute right here.

(Refer Slide Time: 00:59)



So, the thread 0, one which was computing from 0to 99, it would have just executed this piece of code. So, here you would have seen a print task computing from 0 to 99 from 0 or 4 not from 3 or 4 right.

Student: But (Refer Time: 01:17) run (Refer Time: 01:18) my doubt is why it has not done that the.

Student: omp task (Refer Time: 01:22).

So, it is not in your control when that task is going to run, but what is happening here is that thread 0 has created a task that it has put it in the task queue, but he thread 0 realises that it has another iteration of the for loop execute its not free it is going to continuous executing doing its work maybe in the meanwhile, thread three has become free and it does that work from that task queue.

Student: So, why we are saying that it (Refer Time: 01:50) that thread 0 has go back to execute the for loop for another iteration.

That depends on the scheduling mechanism right remember there was a schedule clause whether your schedule in statically or dynamically right. So, here the default is schedule

static the default behaviour when you do not specify this schedule is static. So, what it has done is, it has pre allocated these 6 iterations to the 4 threads. So, the way does it is the divide 6 by 4 sealing of 64 essentially right. So, that is what 2.

So, it will give two iterations to all the threads and to the last few threads it will just give one iteration this divided equally among these threads that is what it is knowing. So, thread 0 has got 2iterations to do right. So, if I had said dynamic then it may have something else I do not know.

Student: To work this I have did the pragma omp task.

Hm.

Student: (Refer Time: 02:50).

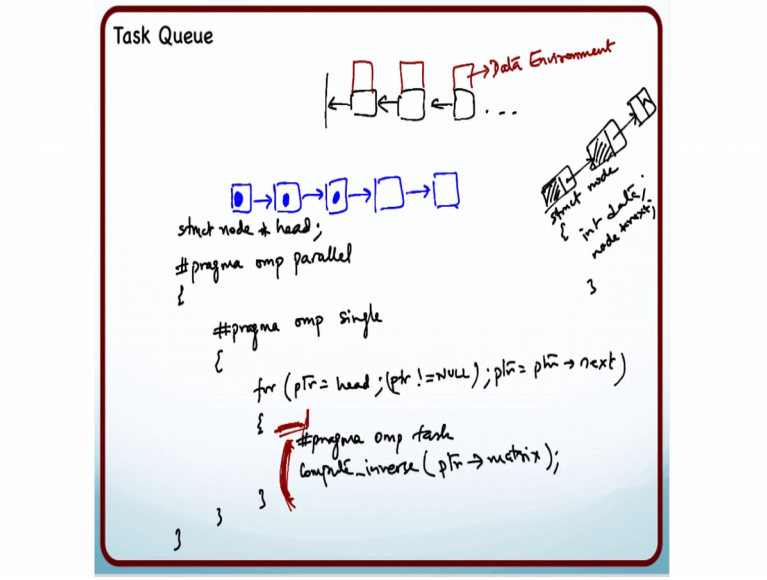
Student: (Refer Time: 02:51).

So, remember there are multiple threads which are executing this code.

Student: (Refer Time: 02:58).

Right lets go through another example

(Refer Slide Time: 03:01)



So, suppose that I have a link list I have to do some processing for each element of the link list all right. So, how do I write code for this and openmp code? I want to distribute that works different threads let us say each no is storing a matrix, and I want to compute the inverse of that matrix. So, for each node I have to do a lot of work, but that work of each node independent the matrix showed that one node does not have anything to do with that matrix showed at another node right. So, how do I divide this works among openmp threads?

Student: (Refer Time: 03:36) either we compute one node at a time and go to the next node (Refer Time: 03:43) or we need to have the addresses of all the nodes (Refer Time: 03:49).

So I need to kind of right maybe convert this link list into an array.

Student: (Refer Time: 03:57).

First right maybe store the pointers of each element of the link list and array, and then I can run a for loop and divide that amongst the openmp threads right that is one way of doing it, but using tasks you can do it in a much much simpler way here is what you do. So, you say hash pragma omp parallel, what happens at this point and time? So, four threads or whatever number of threads get launched right.

Now, inside this code what I am going to do is I am going to say hash pragma omp single, what does hash pragma omp single do? Only one thread will execute this piece of code. So, what is this piece of code now? I will it reverse the link list for ptr is equal to head while ptr is not equal to null ptr equal to ptr next.

So, I am making lot of assumptions over there right I am not defined the data structure for you, but I am assuming that you all worked with link list. So, this should not be any surprise right. So, now, you start with head. So, what is head? Head is let us say something defined over here right stract node star head that is the head of the link list right. So, what do I say for pointer is equal to head while pointer is not null pointer equal to pointer next, that is my for loop what do I do inside the for loop.

Student: (Refer Time: 05:45).

I create a task I say hash pragma omp task I can give a structured blocked or if there is only one statement then I do not even need to put a parentheses right, I can do without parenthesis. So, in this case I will just say compute inverse and I pass ptr matrix maybe matrix is the pointer to the data. So, what is going to happen now?

When I hit hash pragma omp parallel at that time four threads are spawned right I mean I am assuming omp num threads is 4. So, three more threads are spawned totally you have four threads. Now one of the threads goes inside the hash pragma omp single and traverse this entire link list and for every element in the link list it creates a task and what are the other three threads doing.

Student: (Refer Time: 06:30).

There doing nothing. So, they are going to start picking up the tasks and stat executing, and once thread let us say thread number one is executing this once this thread finishes the execution of this for loop it is also going to join them, and start picking up the tasks and doing the tasks it is an extremely convenient way of creating work. So, it save you a lot of troubles that times and trying to yourself figure out what is the work to be done by what thread, how am I supposed to divide it up right it freeze you from all that trouble.

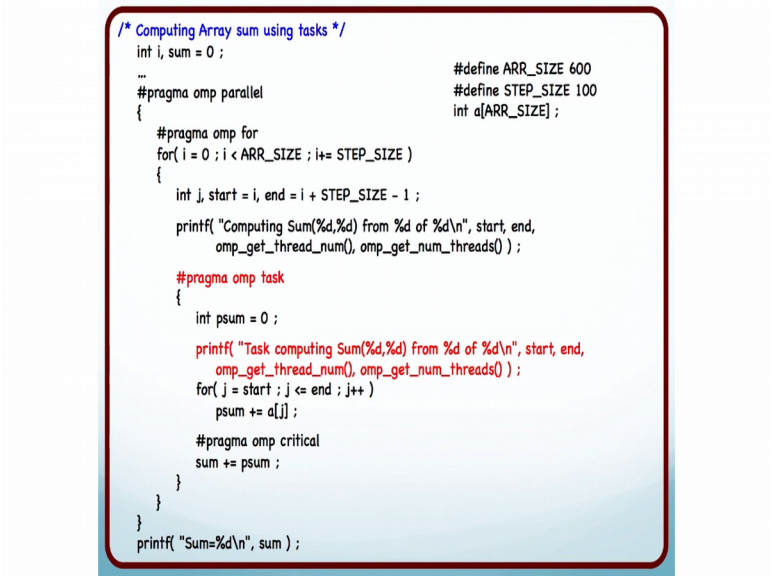
Whenever you find some piece of work go create a task of course, there are some challenges that if the dependencies then you have to be very very careful right because you have no idea how these tasks are going to get executed. You cannot even say that the task that is created first going to get executed first. Now is it going to start first is it going to finish first nothing at all these tasks are independent pieces of code that will get executed at some point and time. You have no idea when they are getting. So, you have to be careful where you introduce the task, and you have to be sure that these are independent pieces of code.

Student: Nothing for on the other threads to do (Refer Time: 07:39).

Student: (Refer Time: 07:42) pragma omp single.

Yeah, but that is not the way I did this previous one right what did I do here?

(Refer Slide Time: 07:47)



And what is the difference between this array creating tasks in this summation of array and the creation of tasks in that link list what is the difference between the two. So, there only one thread was creating the work, creating the tasks here multiple threads are creating the tasks. So, it does not matter right it is just that a lot of times we find that its convenient like in case of a link list multiple threads, cannot even create the work right because you have to sequentially traverse the link list there is no other option.

But in case of an array I had that flexibility, that I can have multiple threads k into work. So, I employed multiple threads to create the work. So, do not get stuck on you know whether I am supposed to created from one threads or multiple threads either this is find any thread at any point of time can create work, then create a task does not matter and any thread could end up executing that task .

Student: (Refer Time: 08:48) here to we are putting a limit of this start, and end and start and end (Refer Time: 08:57).

Student: But if to the task is common (Refer Time: 09:04).

No no what you mean by task is common.

Student: (Refer Time: 09:08).

There is nothing like a common tasks; look the important point here is these tasks are not the same because each one of them is working on a different piece.

Student: Piece

Right.

Student: Yeah.

If they are working on the same piece.

Student: Yeah.

They will be like two different tasks that are doing the same work, but they will be two instances of the task.

Student: They will be 2.

Two instances of the task right if thread 0 and thread one both of them decide to compute the sum over the same part openmp or there now going to figure out for you that you know you are doing something like that that for them it is just a task. One important point is that you have to understand that you can only create tasks inside the parallel region right there is no points of creating tasks also in the parallel region, there have to be multiple threads executing for you to create task.

Student: So, if I cannot do like.

Student: You know I have lot of work (Refer Time: 10:00) tasks and then start a parellal region.

No that is not the way to works right you start a parallel region you have multiple threads and then you start creating the tasks.

Student: So, it will probably be similar stuff, if I want to do say three or four same things (Refer Time: 10:16) stuff tasks just not the way to go.

No that is one necessary.

Student: (Refer Time: 10:26).

I could write anything that inside the code right.

Student: But there will be one piece of task code.

Yeah, but that to that is a code and code I can write anything in a code. So, in that sense I could ask them to do different things does not matter. One thread could be computing a factorial another could be computing the inverse of a matrix, it is up to me I can write the code is just that typically that is not what you do right typically you have some similar stuff going on.