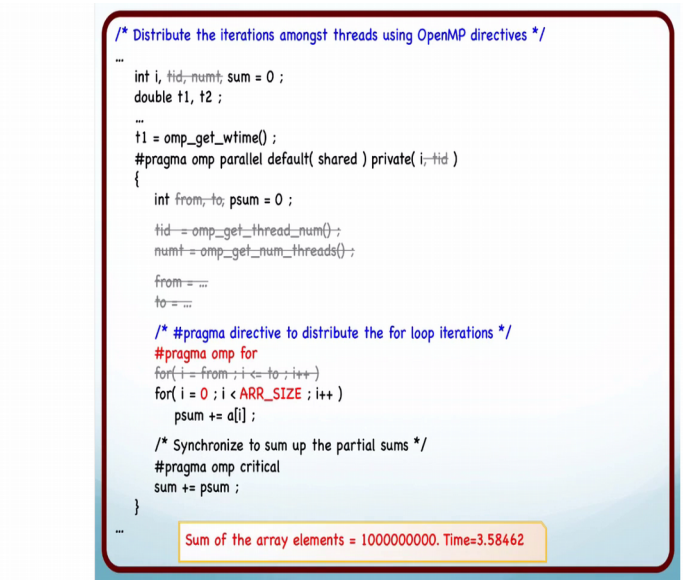
Introduction to Parallel Programming in OpenMP
Dr. Yogish Sabharwal
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

Lecture – 23
Distributing for loops and reduction

(Refer Slide Time: 00:06)



So, one other issue is that I went through all this trouble; rate of computing from what is the from? What is the to? Where am I supposed to start from? Where am I supposed to end? That I went through all that trouble, but if you have a simple for loop typically that is not something that you have to do openmp can take care of that for you.

So, what do you do for that? So, all you need to do is you just need to say hash pragma omp for just before the for loop. And now I am not dividing up the work; this is the whole loop; for i is equal to 0; i is less than ARR size i plus plus 0 to billion minus 1. And I am doing the addition into psum; psum is what? psum is a private variable; it is local to the thread.

So, in this case what openmp does for you is that the compiler introduces code to ensure that this is split up. The splitting may actually happened statically or dynamically, we will come to that, but openmp takes care of the splitting for you; you do not even have to specify how the work is divided; openmp will divide the for loop for you. You write it as a normal for loop and this is not a new parallel regenerate; just understand that new

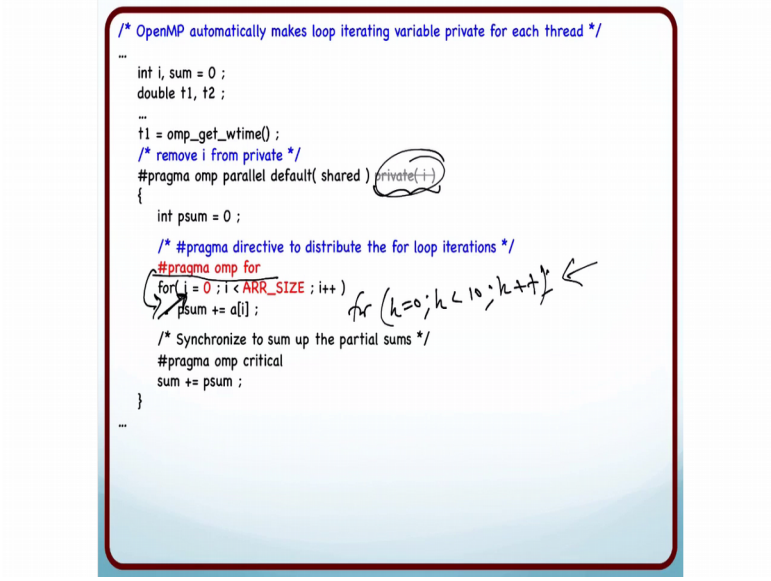
threads are not being launched over here that happens over here. But this is the parallel region; new threads got get launched at this point in time and they get joined at this point in time.

This is just a construct within this to say that this for loop is supposed to be divided amongst the threads; that is all this is saying. And then you write you code normally as you do and something are implicit over here; like I am working on i equal to 0 to i is less than ARR size. So, I am not doing any division; I have just written into the normal loop, openmp will take care of dividing this loop.

I am writing this psum plus equal to ai; psum is private to its thread; this is a single loop, but each thread is accessing a different psum when its executing this loop its part of the loop. And the remaining code is the same and I run this code and I get pretty much the same time that I got earlier.

So, I do not need to do the work division myself I can have openmp do it for me.

(Refer Slide Time: 02:10)



Another thing you should keep in mind is that this variable i; earlier I had mentioned i to be private. So, now I have removed this I is no longer private; so I do not need to declare i to be private; I do not need to scope it. So, openmp automatically in shows that if I have a hash pragma omp for; then the variable for that for loop is automatically made to be private.

It has to because each thread will be executed its own set of iterations

Student: How does this work if we have (Refer Time: 02:46) cascades of for loops?

No, this only applies to the for loop appearing immediately after hash pragma omp for. So, if I have a for loop inside the for loop that for loop is it is like a loop which is being executed by every thread from it start to end; that is not shared amongst the threads. This hash pragma omp for only applies to this for loop; not to any for loop which appear inside it. If I write another for loop for k equal to 0, k is less than 10; k plus plus. If I have this kind of a loop inside this I loop; then each thread will execute this loop from 0 to 9.

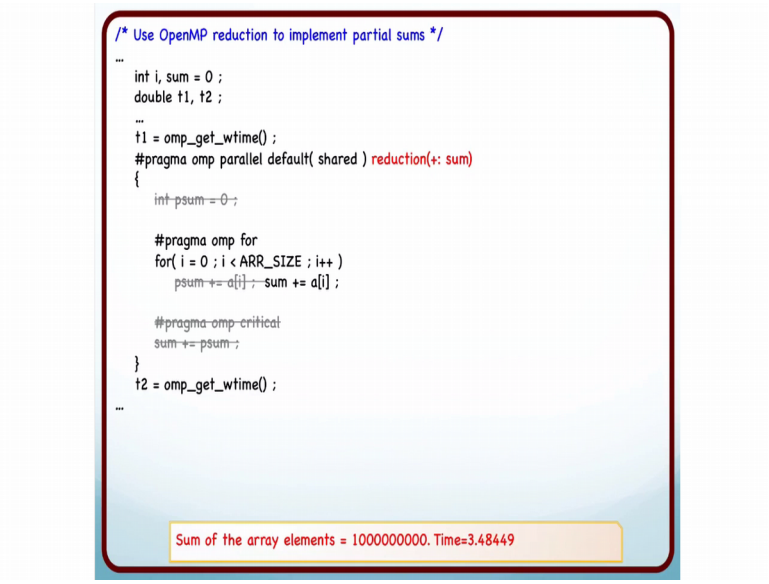
Student: Is it possible that we have multiple for loops and we want the job of multiple for loops to be divided amongst the threads (Refer Time: 03:40) 10 times (Refer Time: 03:42) processes and one for loop contains another for loop (Refer Time: 03:46).

Yeah

Student: (Refer Time: 03:47) idea also to be generate two (Refer Time: 03:48).

It is possible; so, I mean this near some things once you understand the basics near something that you can look up the openmp reference manual and see how to do that right, but it is possible in openmp you can specify how many levels you want to task it down to distribute the work amongst the threads.

(Refer Slide Time: 04:08)

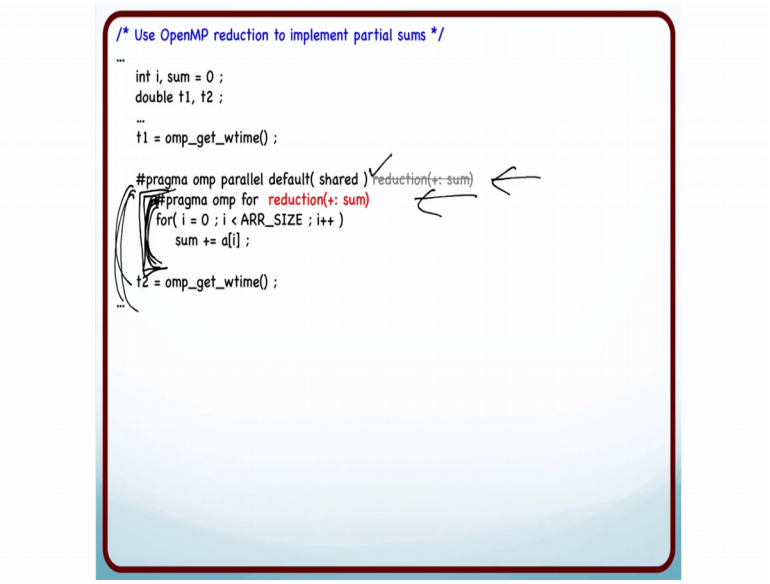


Another thing that we can get rid of; so I was doing all this psum business; I was maintaining a private variable psum for every thread and actually I wanted to accumulated in sum. So, I can actually ask openmp to do all this for me; so, all I have to do is I have to say reduction, the operation; reduction operation and the variable and then I am going to write my code as hash pragma omp for i equal to 0; i is less than ARR size i plus plus; sum plus equal to ai.

Internally worked openmp is going to do what the compiler is going to substitute this code with this is going to introduce these partial sum quantities, do the partial summing and then at the end accumulate them together into sum. It will take care of all the things that we did implicitly; I am specifying its sum is a reduction variable; I want to values being used for the fits to add up into sum.

It will allocate partial sum variables for its thread and in the end; add them up together; how do you know what to do the final pursues sum variables; also run specifying here plus. It will add up the partial sum into the final sum; I again execute the code; it is same time right; so, looks good.

(Refer Slide Time: 05:16)



So, this reduction plus colon sum that something that I can specify at the directive; at the parallel region with hash prgama omp parallel or I can specify it hash pragma omp for. In this case, it will allocate the partial sums and combine them at the end of this for loop. In this case it will add them up combine them at the end of the parallel region. So, in this particular case I can define this at either case; for depending upon you application, you can do it at whichever place it makes more sense.

In this code; I have this hash pragma omp parallel; directive immediately followed with the hash pragma omp for directive and there was nothing else that this hash pragma omp parallel was doing other than this part which is inside hash pragma omp for. So, I can actually combined them both together and the way to combine them I say hash pragma omp parallel for. That says that; this is the parallel region which is the following statement and it is nothing, but a for loop and I want to paralyze this for loop amongst the threads. So, that is it to shortcut way of saying that and everything else remains the same.