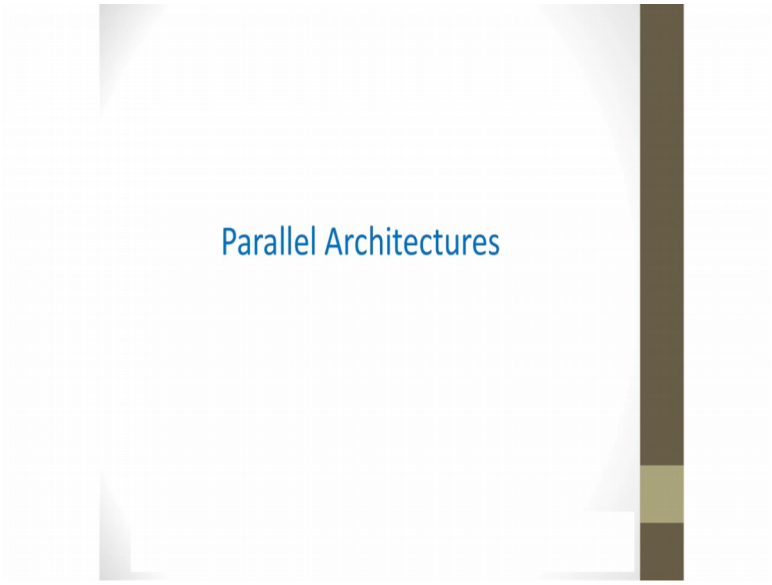
Introduction to Parallel Programming in OpenMP
Dr. Yogish Sabharwal
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 03
Parallel Architectures

(Refer Slide Time: 00:25)



Last time we spoke about single processor architectures and today we are going to talk about parallel architectures. How we pick up single processors system and we combine multiple processors together to form parallel architectures. So, before we start this lets have a small activity. So, I want each one of you to think of a random number, determine whether it is odd or even.

Now, I want you guys to tell me whether there are more odd integers or there are more even integers that you have collectively thought of. It is not some psychological games of some you have to talk to you are going to figure that out. I want to know the total number of odd and even integers or I want to know whether there are more odd integers or there are more even integers.

Student: You need to another processor to combine the data (Refer Time: 01:21).

No I do not need another processor, come on yes how many professors are sitting here.

Student: (Refer Time: 01:27).

One thing is we want to come up with different ideas of how to do it and then we actually want to do it. So, I would like you guys to first actually do it, then we will come up with ideas of how to do it.

Student: (Refer Time: 01:36) we have to communicate (Refer Time: 01:40).

Yeah you have to communicate with your neighbors.

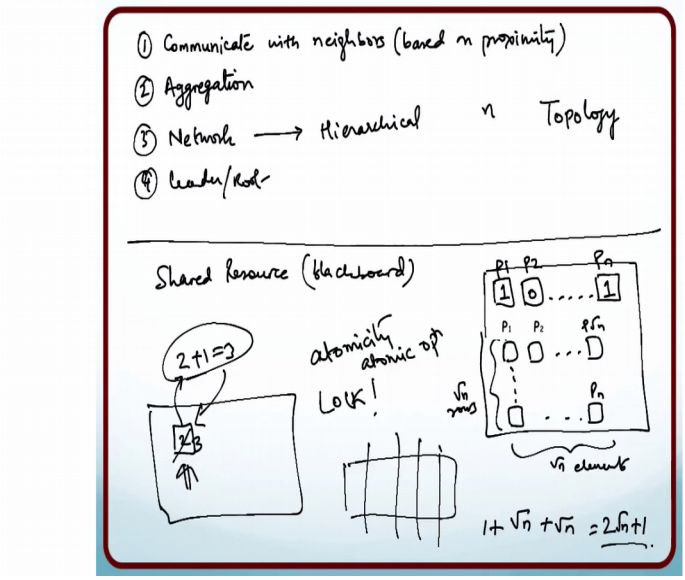
Student: (Refer Time: 01:44).

So, everybody with the; who were thought of an odd number put your hands up.

Student: (Refer Time: 02:09).

You got it, right good all right good. So, let us try to jot down the different things that you had to do.

(Refer Slide Time: 02:13)



Student: Communicate.

You have to communicate, communicate with who?

Student: (Refer Time: 02:22) neighbor.

Communicate with neighbors. So, how did you communicate with your neighbors?

Student: (Refer Time: 02:29).

You just sent him some data which is essentially whether the number you are thought of was even or odd; what did your neighbor do with that. So, who was your neighbor, what did deep what did you do with that?

Student: I sent my data to someone else.

Your data or your data combined with his data.

Student: (Refer Time: 02:47).

So, you basically did some kind of aggregation.

Student: (Refer Time: 02:51).

You did some kind of aggregation and then you pass from that information all right, how did you guys figure out who is your neighbor?

Student: Proximity (Refer Time: 03:00).

Proximity.

Student: (Refer Time: 03:03).

So, what did you guys do you essentially form the network.

Student: Hm

Right.

Student: Hm

What was the shape of this network, how do you guys figure out what is the network you have to form? It was a little bit at random was it the most efficient way of forming the network probably not?

Student: Finally, all the data be committed at one point.

At the end of the day all the data had to be cumulated at one point. So, there was one kind of a leader or the route right where the data had to be accommodated.

Student: (Refer Time: 03:52).

Let us say each upon one of these things a little bit more that is the network. So, how many of you were there were 20 students here right you form the network at random you could solve the problem quickly fine, but what if we have like 1000 or 100 1000 processors trying to solve a problem let us say trying to do such an aggregation.

Student: (Refer Time: 04:12).

So, what is the network topology, that you will use can you think of who should communicate with who in such a situation.

Student: Make small block and then again (Refer Time: 04:25).

Make small small blocks and again make a network. So, you are it is kind of like a hierarchical network.

Student: (Refer Time: 04:30).

Right. So, if there are let us say n nodes n processors, if there were n of you how quickly could you do this, what is the best communication network I can form. so that this can happen very quickly? So, this is something called the topology of the network we will come back to it, now let us talk about another situation. So, here you are basically exchanging messages with each other right it was some kind of message passing, where each one of you was passing some message to your neighbor, which was essentially a number right.

So, if there was no shared resource, you did not have some resource in common where you could share information instead you chose to talk to each other right. But if you have a shared resource let us say that you have a blackboard what does that mean? That means, that now anybody can write to the blackboard and anybody else can read from the black board right and let us say you are not talking to each other now. Because you do not have any mechanism of talking to each other, you only have the blackboard how do you solve the problem now let us say this is the blackboard.

So, the first question is each one of you writing at a different location or at the same location?

Student: Same location.

Same location.

Student: Different location.

Different location good differing views, different algorithms, different solutions yeah. So, how would you solve it if you are writing at different locations?

Student: I will then found out in.

So, first of all of you are going to come and write down write each one of you comes and writes down processor 1, I am just calling you processes right do not mind. So, processor one and comes in writes down one over here, processor 2 says my number is even. So, I am going to write a 0 right we are counting the number of words and then professor n comes and writes one.

Now, now what do you do? So, each one of you has written onto that shared resource on to the blackboard, that what is the number you have now what do you do?

Student: The leader will come and aggregate.

Whose going to do that.

Student: (Refer Time: 06:43) leader (Refer Time: 06:44).

All of you will count our one of you will count

Student: One the leader (Refer Time: 06:47).

Now that is that a very efficient way of doing it?

Student: Hum.

What if you had 1, 00,000 processors well shared resource is a very difficult you know thing to have in a when you have a 1, 00,000 processors, but still. So, as suppose that you have 1000 processor. Even with 8 processors or sixteen processors it becomes a

challenge. So, it is not the most efficient thing you are doing right one processor is going through all the data sequentially, right.

So, you have lost the parallelism what was good about the message passing thing that you did. What was good about it was happening in parallel, while you guys were computing your numbers these guys were computing their numbers and then it was exchange right. Even here like the front row was calculating it separately the back row was calculating it separately and so on, right.

So, everything was happening in parallel. So, here there is a problem right you are not doing it in parallel, sure everybody is writing it in parallel to the shared resource, but after that one person is sequentially going through and reading all the numbers and trying to sum it up right. So, how can you resolve that?

Student: (Refer Time: 07:53) for one row one leader aggregates for another row another leader aggregates and so on.

So, what you are suggesting is you divide up the processors into groups.

Student: (Refer Time: 08:07).

P 1.

Student: P 1.

And so on right up to maybe P root n, and I am just arbitrarily taking a number right you have not told me what the number is I am just arbitrarily choosing a number and then P n. So, this is going to be root n rows of root n elements each right and by doing this you have brought down the time into how much? How many steps do you take now? So, one step is in if when all of you are writing to the board,

Student: Hum

Right that is one step because it is happening in parallel. So, it is just one step right after that one person in each row goes and adds up all the entries in the row, how much time does that take root n step.

Student: (Refer Time: 08:55).

Right. So, one plus root n, now there are root n people root n leaders of one in each row who has the sum. So, what do you do next?

Student: (Refer Time: 09:04).

So, now, you need to sum across that right. So, that will be another.

Student: Routine.

Another routine steps. So, total is about.

Student: 2.

2 root n plus one steps right

Can you do it even more efficiently yeah you can. So, we will talk about these things later right. So, what are the other challenges that you could possibly have? How does P 1 know that t 2 through P and all of them are done with their counting and. So, it can start counting now either the processors work in a very lockstep manner where you know they are all doing everything at the same time. So, you know that in the previous cycle I did the counting. So, everybody did the counting, but that is not typically the way these processor work. When the leader wants to do the counting how does the leader know that all the other processes are done.

Student: I think asks (Refer Time: 09:54).

It asks how do you ask on the black board how do you ask?

Student: (Refer Time: 10:02) we can allocate a memory location.

You can allocate a memory location.

Student: Whenever a processor writes we can increment it by one.

Whenever a processor known. So, that is not good, what does that mean whenever a processor writes to it we will increment it by 1.

Student: (Refer Time: 10:19).

So, how do you increment very good. So, this is what we want to talk about how do you increment a number which is sitting in the memory, which is sitting in the shared resource.

Student: (Refer Time: 10:28).

You cannot just go and tell the blackboard that increment this by one, where does the incrementing actually happen where is the numbering number incremented.

Student: (Refer Time: 10:36).

In your brain.

Student: Hum.

Right the number is incremented at the processor not in the memory.

Student: Hum

Right. So, how do I increment the number?

Student: Read the number.

You read the number, you add one to it and then you put it back.

Student: Hum

So let us say the number was 2. So, what did you do you write the number right. So, you got 2 you added one you got 3 and then you wrote three back here

Student: Hum

What can go wrong?

Student: Somebody else (Refer Time: 11:11).

Somebody else may write to that memory location, while you are doing this right.

Student: Yeah.

So, what do you want to do then?

Student: (Refer Time: 11:21).

Each processor has to have it is own memory allocation.

Student: Memory address where he is only allowed to write.

Huh, but that defeats the purpose of a shared resource, I can always divide the memory up into n chunks and give it to each of the processors right that is not an issue that is not a problem I can do that I would not allow anybody else to write on the other processors memory space not an issue.

But then how do I communicate? My end goal is to communicate without any problem right the problem here was not that I had to increment a number, the reason somebody else is overwriting it is because he was trying to communicate something to me.

Student: Hum

Right I wanted that collective number, otherwise if n processors want to increment their own n numbers not a challenge at all. Each one of them has it is own memory they have the number sitting in their own memory, they go get their number is incremented by one storied back right. There is no nobody else is going to come and write to their memory right.

The point here is that there is this one in shared location which is shared by all the processors, where we are trying to communicate we are trying to count how many processors have updated the memory. Have up updated the number of odd integers that they have right yeah. So, what do you do?

Student: Instead of each and every one professor writing to it (Refer Time: 12:48).

How do you tell that processor? You only have the shared resource right the only way you can tell the processor it is again going back to the blackboard. You again have to use the blackboard to communicate to tell that processor, there is no other channel that you have to tell that processor.

Student: Whenever somebody is writing it will put a lock.

Ok.

Student: (Refer Time: 13:09) rule will be if somebody in writing do not (Refer Time: 13:14).

Right. So, what is the lock? A lock is basically you are trying to say that while I am performing this operation nobody else should be touching this location right that is what you are trying to essentially say right. So, you lock the memory location or you can lock anything you want, you can lock the entire memory you can you can choose this is your algorithm right. So, you can choose what my lock means right, but essentially you lock whatever is shared right where there is there is a problem of conflict where somebody else may also be writing.

So, we lock that location and then you do whatever operations you want to do on it, and then you unlock it and while it is locked. Nobody else can come and lock it right everybody has to wait for this to be unlocked before they can lock it only one person can have a lock at a time right or you are trying to perform an atomic operation. The memory location is not changed while the update is taking place, while all those operations are being completed nobody else is coming and touching the memory right. So, you are doing it atomically.

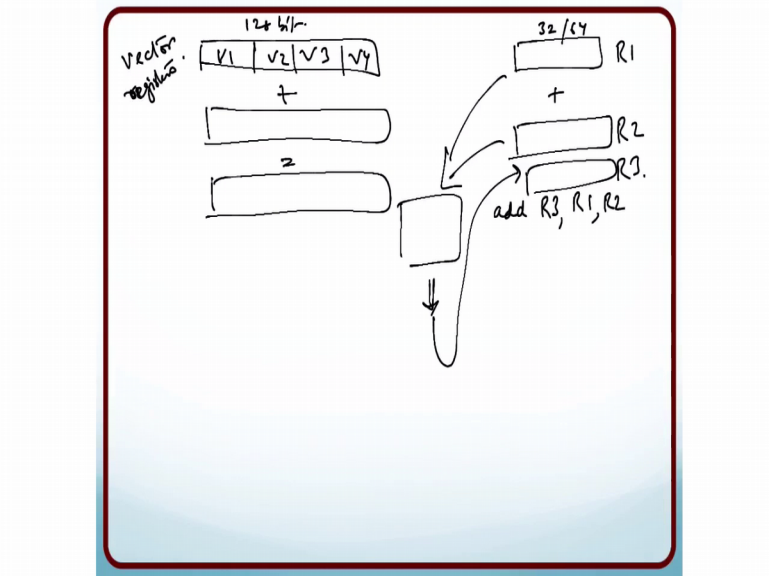
Student: (Refer Time: 14:19) if this (Refer Time: 14:21) locked somebody locking it and therefore, everybody is not done.

Know. So, between the time that. So, let us say somebody is locked it right and somebody else wants to write something to it, but he has to wait for this to be analog. So, that he can lock it. So, for a momentary period between the two locks, it is going to be unlocked for a little while right. When he unlocks it before the other guy lock it is going to be unload in the unlock state right. So, that cannot be a you know used to determine whether it is a finished or not.

Student: (Refer Time: 14:57) on the processors (Refer Time: 15:01).

No. So, there are architectures which do that, there are vector processes I mean let me just give you a just of it, but we are not going to talk about it in detail right.

(Refer Slide Time: 15:12)



So, what is the register? Register is 32 bit or a 64 bit a register, a location where you are storing your data right and where you perform the operation. So, you pick up like if you want to add you take two registers let us say register called R 1 and register called R 2, and you give a command add and store into R 3, the value of the add addition of R 1 and R 2 right. So, what does this do? This picks up the value from register R 1 the value from register R 2, sends it to the math unit and the result is then stored into add register R 3 right.

And what is the size of these registers? 32 or 64 depending on what is the word size of that architecture, but it can be used for performing operations on integers or longs or whatever right a lot of architecture support vector operations. So, in these what you have is something called vector registers. So, these are registers which are like quite long right.

So, these are like instead of let us say 32 bit they are 32 into 4, 128 bit and these registers can actually store 4 values right V1, V2, V3, V4 these are 4 different values they are storing these 4 different values. And now you can do the same thing on these on these registers right.

So, all of these operations are essentially performed synchronously exactly in the same cycle right they go through the math unit all of them proceedings lockstep in doing

exactly the same thing. In each step in this course we will assume that the processes are working independently on their own.