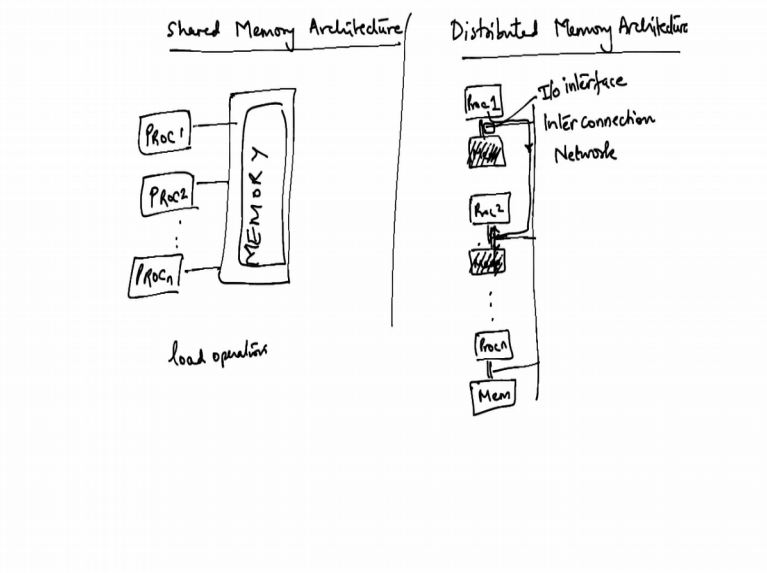
Introduction to Parallel Programming in OpenMP

Dr. Yogish Sabharwal
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 09
Shared and Distributed Memory architectures

(Refer Slide Time: 00:05)



So what are the different parallel architectures? So, there is a shared memory architecture we have already spoken about this, but let us revisit it once again. So, what happens in this case is you have a large memory unit, and you have multiple processors which are accessing this. So, this is analogous to what we just discussed right where all of you were trying to use the blackboard. The blackboard was nothing, but your memory right and you are trying to use the blackboard to communicate with each other.

So, these processes typically do not directly talk to each other except via the memory; like the memory access their communication mechanism serves for communication purposes. This is the shared memory architecture and the other architecture that we commonly come across is the distributed memory architecture, and what happens in the distributed memory architecture is that you have a processor with its own memory right. So, there are multiple processes each one of them having its own memory, and now you need to connect these processes up the processor memory units up and how do you do that? Via some interconnection network.

So, this is your interconnection network. The processor actually needs to use an IO interface right. So, typically on any processor right you have the processor you have the memory and then you can communicate with IO devices, like the printer or the ethernet or any other IO, but the files the disk right all of that happens to go via the IO interface right. So, if you want to perform any IO, it needs to go as the IO interface.

So, similarly the interconnection network is connected to the IO interface. So, if you want to basically communicate with another processor, you have to go via the interconnection network which is behaving like an IO device. So, what is the major difference here? The major difference is that if I want to perform a load operation. If processor one perform the load operation it can access any location in this entire memory right just using a load operation or similarly store some value, it can access memory any location of the memory using normal load store instruction right.

The normal instruction set works whereas, in this case as far as its own memory is concerned, it can use normal load store operations the normal instructions work just as there do, but if it wants to communicate with or get data or store data to the memory of another processor which is remote over here right, it cannot do that using a normal load store instruction. It has to perform an IO instruction it has to actually write data to this device which is the interconnection network and that has to be read over here, and understood by somebody over here maybe the processor which needs to then fetch data from its local memory put it back onto the network device, and then it is read by processor one. Fetching data from remote memory is not an instruction.

It is a proper procedure that has to be executed, where you write a request on to the IO device and it has to be sent to the other professor, it has to be received by the other professor and it has to then deliver the data pick up the data from its local memory, put it on back onto the network and then this processor has to pick it back from the network.

So, as you can expect performing memory operations in a distribute memory architecture is expensive right you if you are going to remote memory. If you are going to local number it is not an issue right then a shared memory architecture going to any location in the memory is the same thing.