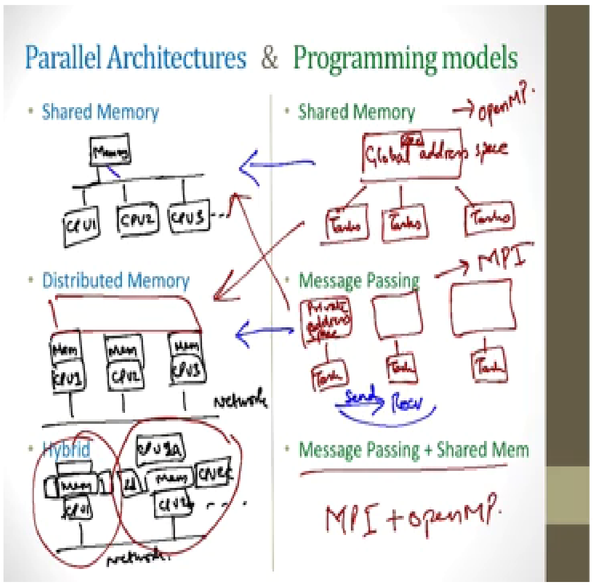
Introduction to Parallel Programming in OpenMp

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

Lecture - 02
Parallel Architectures and Programming Models

(Refer Slide Time: 00:03)



Parallel Architectures & Programming models

- Shared Memory
- Distributed Memory
- Hybrid

- Shared Memory
- Message Passing
- Message Passing + Shared Mem

So, I will just go a little bit deeper or into the parallel architecture than programming model, just give you a very bird's-eye view of that to understand that what we are going to focus on in this course. So, if we talk about parallel architectures right there are primarily two architectures shared memory and distributed memory, and then there is a third architecture which combines them both which we call hybrid.

So, let us quickly have a look at what that is. So, in shared memory architectures what happens is that you have a memory unit right and this is accessible to multiple CPUs. So, there is CPU 1, CPU 2, CPU 3 and so on. So, there is a single memory unit that is shared by all the processors, that is why it is called shared memory.

So, this typically there is a common bus that is it is between the memory and the processing units and all the CPU share that bus. So, they have to you know you require an arbiter to figure out how the CPUs, which CPU gets access to the memory right. So, anyways we will get into more details of this later, but just to understand that does it that is what shared memory means at a high level.

So, what happens in distributed memory? So, in distributed memory what happens is that every CPU has it is own memory unit and now what this, but obviously, if you are trying to solve a large problem and you are trying to do it in parallel these, you need to divide the problem and you know you need to sink up at various points.

So, these needs these professors need to talk to each other right. So, how do they talk to each other? So, typically what happens is that there is a network. So, there are lots of different types of networks again we will we will get into that later, but for now it is just important to understand that there is a network that connects all the processors that and each one of them has it is own memory.

So, now what is the issue with shared memory the basic issue with shared memory is that it is not very scalable right because how many processors can you add because they are going to use the same bus to access the memory right. So, there is a limit to the number of cpus that you can add. So, typically in modern day processors you see 8 course, 16 course right some research professors are pushing it to about 100 of course, but that that is you know there is a limit to it that is their it will end right you cannot go beyond that.

But in distributed memory because each CPU has it is own memory right. So, you can just replicate this and you can have as many processors as you want. So, the bottleneck is going to shift to the network. So, it all depends on what kind of a network you design right. So, there are different kinds of network systems networks completely connected network connecting all the processors together or you may have some simple network like similar to Ethernet or something which is shared across all the processors right.

But the important point here is that them the bus is not a bottleneck right the bottleneck shifts to the network and depending on how you design the network, you can scale this to a large number of processors and finally, what we use in real life what we see in real life is neither the shared memory or the distributed memory system, but a hybrid of the two right.

So, typically what happens is that you have; you have different nodes which are connected together by a network, and each node has multiple CPU sitting on it CPU 2 a CPU 2 b and so on right and similarly you have multiple CPUs sitting over here right.

So, locally on each node it is a shared memory architecture right if you look at one of these nodes, it is a shared memory architecture, but if you look at the cross nodes these two nodes are connected together using a network. So, that is the distributed memory part of it right. So, this is called a hybrid system.

So, just is you have parallel architectures there are different programming models to write code for these right. So, this is the shared memory model and what the shared memory model assumes is that there is one huge global address space. So, think of it as the memory it. So, you assume that there is one global memory that is visible to all the processors or all the tasks.

So, all the tasks get to see the same global address space which means that if one of the tasks writes some something to the global address space that is visible to everybody else and the other is the message passing. In message passing what happens is that each task gets to see it is own address space right. So, it is private it is only visible to this task.

So, each task will get to see it is own private address space, and now if one tasks wants to you know get some data which resides with another task, how does it get that data. So, you have to explicitly do message passing, that is why it is called the message passing model you have to explicitly send data. So, the programmer has to write code to send the data across to another task and the other task has to write code to receive that data right

So, this task will have to receive that data. As you can see right in shared memory with there is no send or receive involved because everybody gets to see the same data, all you need to do is write into the memory it is visible to everybody else they can just read it off from the memory right and in case of message passing, you have to do explicit send receive communication.

So, as you can see right if you look at the left hand side and the right hand side is quite obvious that the shared memory programming model is kind of like suited to the shared memory architecture, and the message passing model is suited for the distributed memory architecture right, because the send receives happen over the network. And here because all of them have access to a local memory, all of them can use that as the global address space ok.

But that is not necessarily true. So, you can actually have a shared memory model which works on top of distributed memory, and you can have a message passing model which works on shared memory architecture. So, what is that mean? So, let us take this case right message passing working on shared memory, a process has it is own address space even on the same node on the same CPU right you can run multiple processes each having it is own address space.

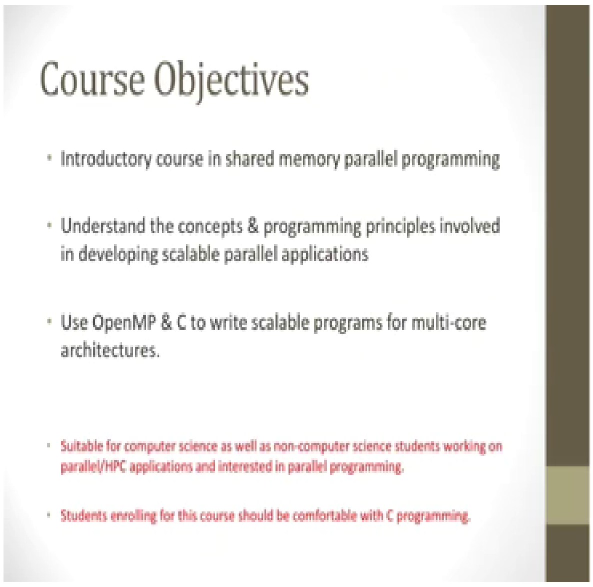
So, they do not get to see each other the address place unless you do some explicit calls right; and then you can actually encode message passing you can encode send receive and so on using the shared memory right. So, you can do that and similarly when you do shared memory you can do it across distributed memory, where there is one global view of the entire memory right, but here what happens is that when you access a variable which is lying somewhere else, it is the responsibility of the underlying operating system to in somehow get that data for you ok.

So, to the programmer it is its not visible that this is a distributed system the only issue the programmer is going to have is that some data is going to come back to it very quickly whereas, other data will take very long to come back right because underneath it there will have to be a communication over the network and then data will have to be transferred over the network back to this computer right

But again the most common way of using these architectures is that you use the shared memory model on shared memory architectures, and you use message passing on distributed memory architectures. What do you do on a hybrid system? So, you use a combination of message passing and shared memory. So, typically you have different processes running on different nodes of the distributed memory architecture, and within that you use a shared memory model to use the multiple cores that are there on the node.

For shared memory there are lots of models, but one of the most common models is OpenMp right and for message passing again one of the most common models is MPI right. So, you have to write your code using MPI which is a message passing interface library, and in OpenMp you have two adhere to certain directives right in order to program in parallel and when you use it a hybrid system then you have to code up in both MPI plus OpenMp you have to make use of both of them right, ok.

(Refer Slide Time: 09:16)



Course Objectives

- Introductory course in shared memory parallel programming
- Understand the concepts & programming principles involved in developing scalable parallel applications
- Use OpenMP & C to write scalable programs for multi-core architectures.
- Suitable for computer science as well as non-computer science students working on parallel/HPC applications and interested in parallel programming.
- Students enrolling for this course should be comfortable with C programming.

So, what are we going to do in this course? So, this is intended to be an introductory course in shared memory parallel programming right. So, we will be focusing on the shared memory aspects that is slightly simpler, in the sense that you know you can start off with a sequential program and you can incrementally convert it you know introduce your OpenMp commands into that, directives into that and make it run in parallel and you do not require a huge infrastructure you can do it on a node which has multiple cores even your laptops do they have multi cores, right.

So, you can it will be easy to work with that on the other hand MPI requires some redesigning of the code right it is not intuitive that data is not is distributed across different nodes right. So, you have to explicitly write send receive or you have to make calls to send receive and so on. So, you have to redesign your code if you want to use MPI.

So, we will focus on OpenMp shared memory programming. So, we will understand the concepts and programming principles involved in developing scalable parallel applications right we will see how much we can scale we will try to scale up to 8 to 16 course and we will use OpenMp and C to write the scalable programs for multi core architectures that is the plan.

So, this is going to be for both computer science and non computer science students, I am going to cover whatever basics of operating systems or architecture is required in this course, what is expected is that you have reasonable understanding of C, you are reasonably compatibility with C. I am not going to go into the basics of C programming, right.