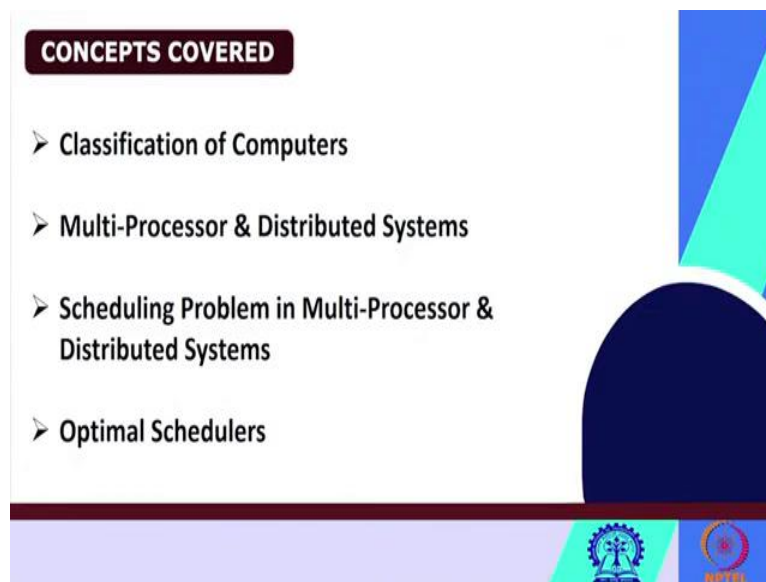


Real Time Systems
Professor Durga Prasad Mohapatra
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Lecture 32

Introduction to Multiprocessor and Distributed Systems

Good afternoon to all of you. Hope you have already learned the basic concepts of Real Time Systems, various scheduling techniques, then this scheduling in presence of task dependencies, et c. in the earlier classes. So, this class will take up a new chapter that is how to perform task scheduling in multiprocessor and distributed environment.

(Refer Slide Time: 00:49)



We will first see, what concepts we will cover in this class, we will first look at the classification of the computers, because you will see, you have to use the concept of multiprocessor systems and distributed systems. So, that is why I want to give a brief classification of the computers, then little bit about multiprocessor and distributed systems. Then we will see some brief ideas about the scheduling problem in the multiprocessor and distributed environment and finally we will look at, very briefly, on the concept of the optimal schedulers. We will see about the optimal schedulers.

(Refer Slide Time: 01:29)

KEYWORDS

- UMA & NUMA
- Distributed Systems
- Multi-Processor Systems
- Real-time distributed systems
- Optimal Schedulers

So, let us just start with the keywords that we will use in this lecture. We will use UMA versus NUMA, then distributed systems, multiprocessor systems, real time distributed systems, optimal schedulers. These keywords we will use in this lecture.

(Refer Slide Time: 0:01:48)

A BROAD CLASSIFICATION OF COMPUTERS

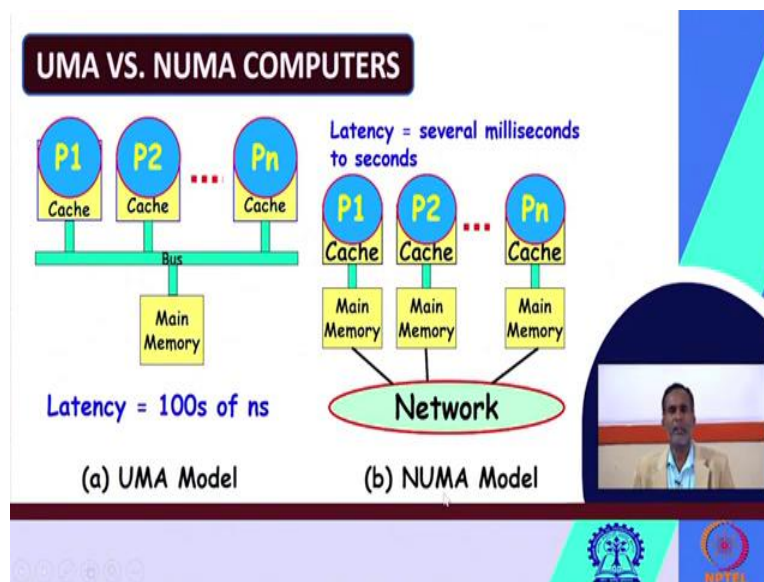
- Shared-memory multiprocessors
 - Also called **Uniform Memory Access (UMA)**
- Distributed memory computers
 - Also called **Non-Uniform Memory Access (NUMA)**:
 - Distributed Shared-memory architectures (DSM)
 - Clusters
 - Grids, etc.

We will start with the initial concept. That means first we will start with a broad classifications of the computers. Why? Because this lecture is, or this module is specifically dedicated towards real time task or real time scheduling in multiprocessor and distributed system. So, I think it will be helpful for you if I will cover the broad classification of computers.

So, normally we will discuss here two important broad classifications of the computers. One the shared memory multiprocessors and the other one are the distributed memory computers. So, after discussing those fundamental concepts, we will see how this real time task scheduling can be done in this multiprocessor environment as well as this distributed environment.

The shared memory multiprocessors, they are also known as uniform memory access, UMA computers and the distributed memory computers they are also called NUMA computers that means non-uniform memory access. Also, these examples if we will see which are coming under NUMA, distributed shared memory, clusters, grids, etc., they all are coming under distributed memory computers.

(Refer Slide Time: 03:01)

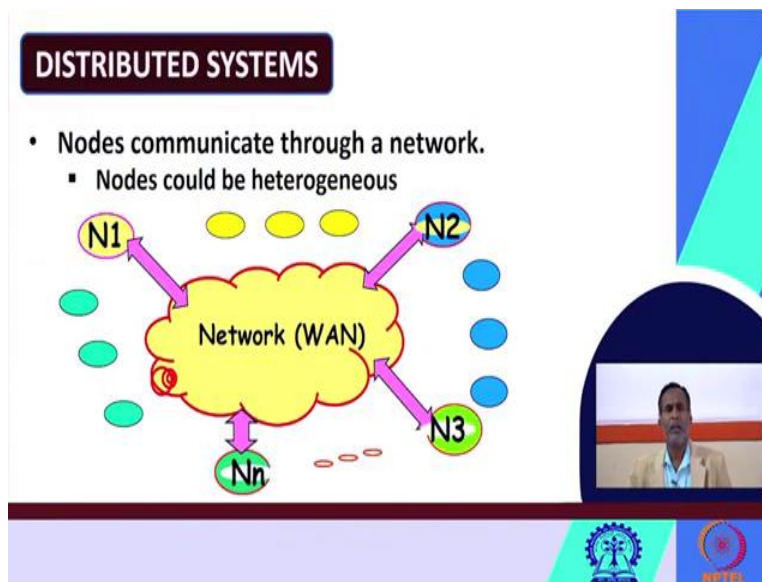


Let us look at the architectures of UMA versus NUMA, how they differ. In UMA model what is happening, so one main memory is there and this memory is shared among several processors, P1, P2, Pn and these are different processors, each one having individual cache. So, they are connected with a bus and here this latency is of the order of 100s of nanoseconds, in case of UMA model.

Let us see how that is different in case of NUMA. So, as its name suggest, non-uniform memory access here, instead of having one memory as in case it is happening in UMA here multiple main memories are there, they are connected through network and each processor is having a separate cache, and each cache in turn they are connected to individual main memories.

Here the latency is of the order of several milliseconds to seconds. So, this is how the UMA computers, they are different from NUMA computers, and these concepts we will use while scheduling the task in real times systems, are both, in case of multiprocessor systems, as well as these distributed systems.

(Refer Slide Time: 04:16)




So, let us now look at about, so this distributed system. Distributed system is a system which consisted of several nodes, and here nodes will refer to obviously computers. So, these nodes, they communicate through a network and these nodes, they may be heterogeneous, they may be homogeneous, no problem, so here nodes, they could be also heterogeneous. So, all those nodes they are connected through a network we call as WAN, wide area network.

(Refer Slide Time: 04:45)

DISTRIBUTED MEMORY COMPUTERS

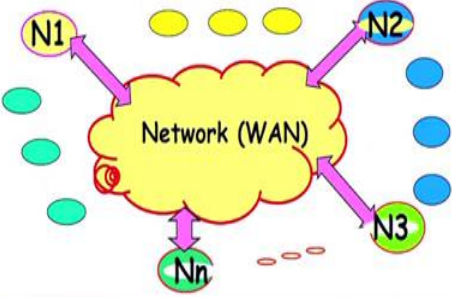
- Distributed computers use:
 - **Message Passing Model**
- Explicit message send and receive instructions written by the programmer.
 - **Send:** specifies local buffer + receiving process (id) on remote computer (address).
 - **Receive:** specifies sending process on remote computer + local buffer to place data.




The slide features a dark blue header with the title 'DISTRIBUTED MEMORY COMPUTERS' in white. The content is a bulleted list. A small video inset of a man in a suit is positioned on the right side of the slide. At the bottom, there are two logos: a blue gear-like logo on the left and a red circular logo with the text 'WPTCL' on the right.

DISTRIBUTED SYSTEMS

- Nodes communicate through a network.
 - Nodes could be heterogeneous



The diagram shows a central yellow cloud labeled 'Network (WAN)'. Four nodes, represented by colored circles (yellow, blue, green, and green), are connected to the network by pink arrows. The nodes are labeled N1, N2, N3, and Nn. There are also several small colored circles (yellow, blue, green) scattered around the network cloud, representing other nodes in the system.



The slide features a dark blue header with the title 'DISTRIBUTED SYSTEMS' in white. The content is a bulleted list. A diagram illustrates a network of nodes connected to a central 'Network (WAN)'. A small video inset of a man in a suit is positioned on the right side of the slide. At the bottom, there are two logos: a blue gear-like logo on the left and a red circular logo with the text 'WPTCL' on the right.

In distributed memory computers let us say how it works. The distributed computers they use message passing model that means, as here I already told you that the nodes, they communicate through a network so in distributed memory computers these distributed computers, they use a message passing model for passing messages among the nodes or among the computers.

So, there are explicit instructions known as send and receive. So, explicit message send and message receive instructions, they are written by the programmer for message passing among the computers. So, what does this send instruction will do? This send instruction, it will specify the

local buffer, plus the receiving process IDs or the remote computer addresses. So, send instruction will perform this.

What this receive instruction will do? This receive instruction it will specify how the sending process on the remote computers and the local buffer to place the data. This will be carried out by this receive instructions. So, in this way the programmers, they explicitly write message send instructions and message receive instructions to perform the following activities.

(Refer Slide Time: 06:08)

ADVANTAGES OF MESSAGE-PASSING COMM.

- **Hardware for communication and synchronization are much simpler:**
 - Compared to communication in a shared memory model.
- **Explicit communication:**
 - Simpler programs, which helps to reduce maintenance and development costs.
- **Synchronization is implicit:**
 - Associated with sending/receiving messages.

Then let us quickly look at the advantages of the message passing communication. This is quite easy, you can redo yourself. Like hardware for communication and synchronization are becoming much simpler nowadays. So, in comparison to communication in the shared memory model, so it is becoming simpler, which one? Hardware for communication and synchronization. They are becoming simpler as compared to the communication in a shared memory model. This is the number one advantage.

Then explicit communication. So, here we are using much simpler programs which help to reduce maintenance and development cost. So, this explicit communication, this we are using very simple programs, we are writing, the programmers write very simple programs to achieve explicit communications. This helps reduce the maintenance and the development cost.

Then synchronization is implicit. So, no explicit synchronization is required. Here the synchronization is implicit. It is associated with sending and receiving the different messages. So, these are some of the advantages of the message passing communication.

(Refer Slide Time: 07:12)



DISADVANTAGES OF MESSAGE-PASSING COMM.

- Programmer has to write explicit message passing constructs.
 - Also, precisely identify the processes (or threads) with which communication is to occur.
- Explicit calls to operating system:
 - Higher overhead.

The slide features a dark blue header with the title in white. The main content is on a white background with a blue and green geometric design on the right. A small video inset shows a man in a tan jacket. At the bottom, there are logos for a university and NPTEL.

So, inspite of these advantages there are several disadvantages also of the message passing communication. Here the programmer has to explicitly write the message passing constructs, also he has to precisely identify the processes or the threads you can say, the processes are also known as threads.

The programmer has to also precisely identify the processes with which the communication has to occur. Obviously here the programmer is making explicit calls to a program operating system. The programmer has to make explicit calls to the operating system. So, which ultimately, what incurs higher overhead. So, these are some of the disadvantages of the message passing communication.

(Refer Slide Time: 08:00)

WHY REAL-TIME DISTRIBUTED SYSTEMS?

- **Applications themselves are distributed:**
 - Example: command and control, air traffic control.
- **High performance:**
 - Work distributed among multiple nodes.
- **High availability (fault-tolerance):**
 - No single point failure.

Now let us move towards our real time aspects. So, why real time distributed system is at all required? Previously we have seen real time system in a uniprocessor system also, so why real time distributed system at all required. These are some of the reasons high real time distributed systems are required.

The applications themselves they are distributed in nature. Now, that you say the modern day applications they are themselves distributed in nature. So, we have to use a real time distributed system in order to handle those modern days application. For example, you take the command and control systems. Air traffic control for where the flights are running. So, air traffic control, et c. These are some of the examples of what modern day applications which are by default, by nature they are distributed. So, we have to use real time distributed system.

Similarly, another example you can say that this, where these refineries or mining there, what locations, there you need a distributed all over India or even all over the world. So, they are communicated through networks, these are also good examples of distributed systems. So, in order to handle the transactions in those cases also we require real time distributed systems.

Similarly, if we will use real time systems, the performance will be much higher, why? Because the whole work, the total work will be distributed among multiple nodes or multiple computers which relate to high performance. We will get high performance if we use real time distributed systems.

Similarly, another achievement we will get, that is high availability, why? Because no single point fieldwork. In the traditional system when only one processor we are using, if that is failed due to some reason then you lose the data, the whole process will be stopped, but here, since you are using multiple number of what computers, multiple number of nodes, even if one node fails, no problem. The load will be shared by other nodes. The other nodes will take care of the responsibility.

So, the whole system, it will not be stopped. So, there will be high availability of the data and we will say that it will be having much more, what, it will have a better fault-tolerance ability, because a single point failure will not what affect the system. The whole responsibility will be now taken up by the other computers or by the other nodes. These are some of the reasons why real time distributed systems, nowadays it is required.

(Refer Slide Time: 10:39)

PROBLEMS WITH DISTRIBUTED SYSTEMS

- **Efficient resource management is difficult:**
 - No global knowledge of workload.
- **Synchronized clocks difficult to implement.**
- **Communication problems:**
 - Out of order delivery of packets, packet loss, etc.
- **Difficult to identify intermittent node/link failures**

The slide features a video inset of a speaker in the bottom right corner and logos for IIT Bombay and NPTEL at the bottom.

But it has also some problems. So, distributed systems they have also some problems, like efficient resource management is difficult. If you are using only one processor, managing the resources is quite easy, but if you are using multiple numbers of nodes in distributed systems you know there are several nodes that are connected among, with this network.

So, if you are using distributed systems, managing the resources efficiently is a difficult task, because we do not have global knowledge of the work load. What is the total work load? How much work load is assigned to which processor, we do not have those information. No global

knowledge of the work load is available. So, that is why efficient resource management is becoming quite difficult in case of distributed systems.

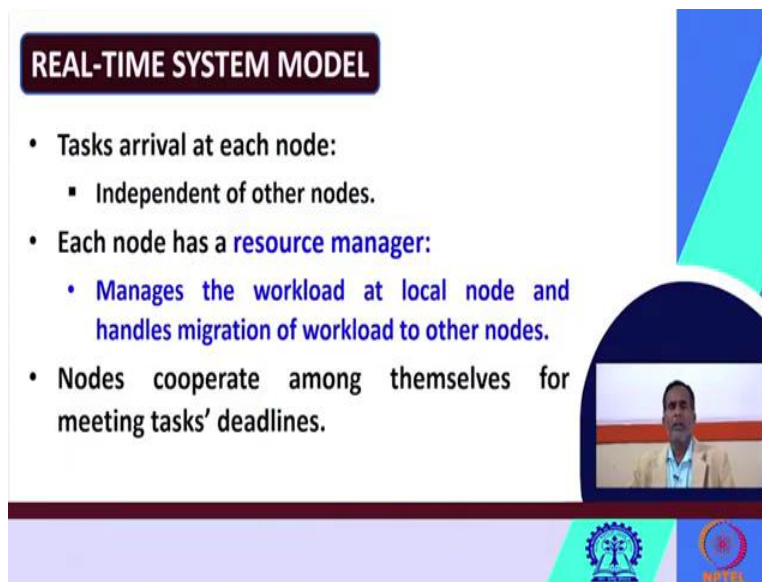
Similarly, synchronized clocks difficult to implement, if you are using one system, one processor, one computer sure we do not require synchronization of the clocks, but if you are using multiple clocks, multiple nodes distributed at different locations, we have to synchronize their clocks and in a distributed systems, synchronizing clocks is very much difficult.

This is another difficulty and third problem is communication problems. So, since these different nodes of the computers they are connected through network there is every possibility that some packets might be lost during the transmission. Or some packets they will be delivered but by that time they are delivered they will be just, what, they will lose their value.

Out of order delivery of packets, they will be delivering too late by that time, they do not carry any, they will not carry any importance, they lost their significance, they already lost, so it will lead to some communication problems, so in distributed systems some communication problems might arise due to loss of packets, due to out of order delivery of the packets.

Similarly, another problem is that difficult to identify the intermittent node of the link failures. So, when the intermittent nodes or the links they fail, it is very much difficult to track and it is very much difficult to identify, which intermittent node or which intermittent link it has failed, very much difficult to identify. As a result the whole communication will be stopped, so which are some of the problems, which might be occurring, which might be what happening, which might occur in distributed systems. So, we have seen the advantages and the disadvantages of the distributed systems.

(Refer Slide Time: 13:11)



REAL-TIME SYSTEM MODEL

- Tasks arrival at each node:
 - Independent of other nodes.
- Each node has a **resource manager**:
 - **Manages the workload at local node and handles migration of workload to other nodes.**
- **Nodes cooperate among themselves for meeting tasks' deadlines.**

The slide features a dark blue header with the title in white. The main content is on a white background with a blue and green geometric design on the right. A small video inset shows a man in a light-colored shirt speaking. At the bottom, there are logos for a university and a research center.

Now, we will come to the real time aspects. The real time system model you will look at here. So, we will see the simplest model for this real time system. Here the tasks they arrive at each node and these tasks are independent of the other nodes. So, in our real time system model we will assume that the task, the task arrival at each node, they are independent of other nodes, that means if there are three nodes, node 1, node 2, node 3, which task is arriving at node 1.

Suppose task A is arriving at node 1, task B is arriving at node 2 and task C is arriving at node 3, so these task arrivals are these different nodes, they are independent of each other. Each node has also a resource manager. So, every computer, every node has a resource manager. So, this resource manager what it will do, it will manage the workload at the local node and it handles a migration of workload to other nodes.

So, the job of the resource manager which is present at each node is to manage the workload at that local node and it will also handle the migration of workload, the migration of the job, the migration of the task from one node to other nodes. The nodes cooperate among themselves for meeting task deadlines. So, since there are multiple nodes, these nodes, they must cooperate among themselves for what, why they should cooperate? They should cooperate so that the different tasks, they can meet their deadlines. So, the different nodes they must cooperate among themselves for meeting the deadlines of the tasks.

(Refer Slide Time: 14:51)

WORKLOAD ASSUMPTIONS

- Mixture of periodic and aperiodic tasks.
- Task usually have :
 - Specific precedence constraints, resource and fault-tolerance requirements.
- Message transmission times are assumed to be:
 - Constant and bounded.

The slide features a dark red header with the title 'WORKLOAD ASSUMPTIONS' in white. Below the title, there are three bullet points. The second bullet point has a sub-bullet. A video inset in the bottom right shows a man in a light-colored shirt speaking. The slide has a decorative background with blue and green geometric shapes and logos at the bottom.

Now, let us see what assumptions we will take in this real time system model. So, first assumption, so the task set we are considering it is a mixture of period and aperiodic task. The task set that we are considering in this real time system. So, they are mixture of periodic and aperiodic task.

The task, they usually have a specific precedence constraints, resource and fault-tolerance requirements. So, each task that we are considering, so one assumption is that the tasks usually have specific precedence constraints. What is the precedence constraint, which task will arrive before which task, like that.

The resource constraints, what resources are available, the resource requirements and fault tolerance requirements. First should be the level of the fault-tolerance that must be satisfied, that must be required. So, every task, it must usually have a specific precedence constraints, resource and fault-tolerance requirements.

So, next assumption is that the message transmission time are assumed to be constant and bound. So, the message in order to transmit the message from one to another node it will definitely take some time. These message transmission times, we assume that these message transmission times, they are constant and they are bounded, they do not change.

(Refer Slide Time: 16:10)

CLASSIFICATION OF TASK SCHEDULING SOLUTIONS

- Broadly, there are two types of scheduling solutions:
 - **Static:** Tasks are statically assigned to processors
 - **Dynamic:** The processor which would execute a task is dynamically decided.

Now, let us quickly look at the classification of the task scheduling solutions. So, broadly there are two types of scheduling solutions, here we will discuss. One is static, another is dynamic. In static scheduling solution what do we do? The tasks are statically assigned to processors, we are discussing multiprocessor systems. So, here, these tasks, suppose there are 10 tasks, these tasks are statically assigned to processors, so before this first we will assign the tasks to the processors, statically and which tasks, how many tasks will be assigned to which processor that is pre-decided.

In dynamic scheduling solutions what is happening? The processor which would execute a task is dynamically decided. In this case which processor would execute which task that is not statically decided, as in case of static techniques. In static techniques the tasks are statically assigned to the processors. We know which processor will execute which task in advance, but in dynamic scheduling the processor, a processor will execute which task, it is not pre-decided, it is dynamically decided. So, which processor will execute which task it is dynamically decided. These are the difference between the static scheduling techniques and the dynamic scheduling techniques.

(Refer Slide Time: 17:36)

CLASSIFICATION OF TASK SCHEDULING SOLUTIONS

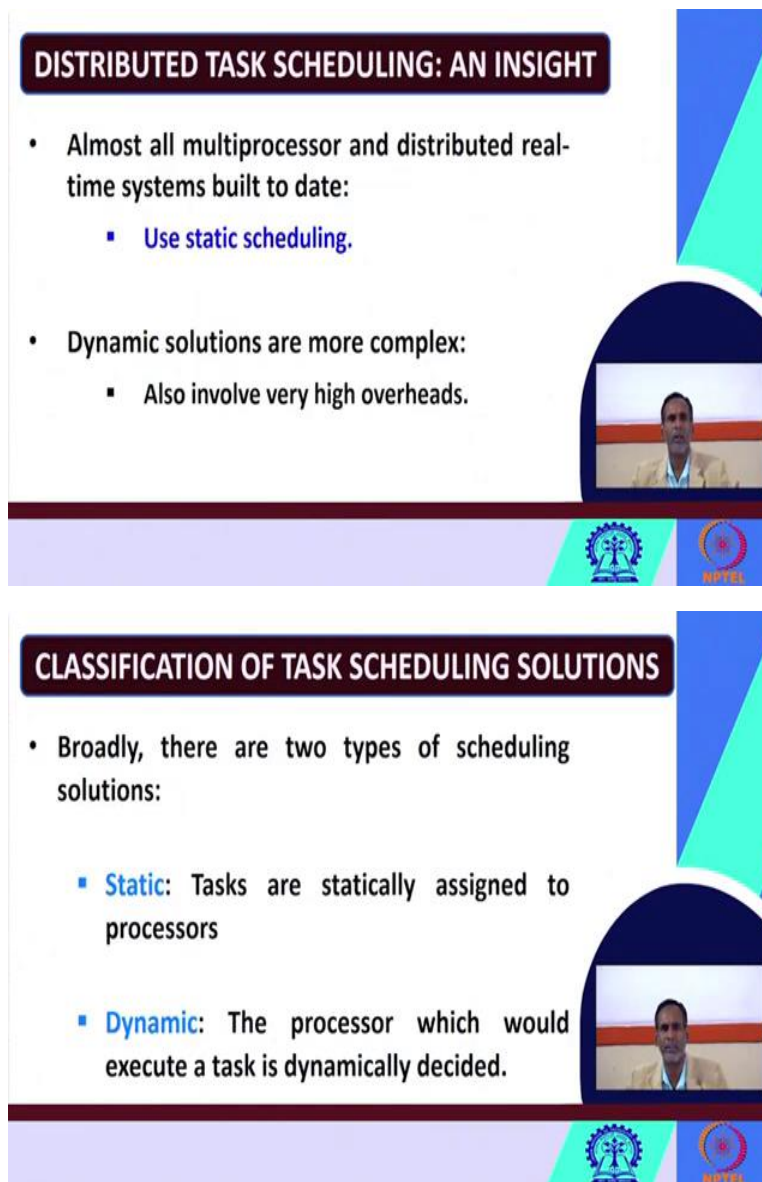
- **Local scheduling:**
 - A resource manager is associated with each node.
 - Task scheduling, and load transfer.
- **Global scheduling:**
 - A single scheduler handles balancing load across nodes.
 - Transfer policy.

The slide features a video inset of a man in a tan jacket speaking. At the bottom, there are logos for a university and a group labeled 'NUTEL'.

Another scheduling technique we will say local scheduling. Here a resource manager is associated with each node. Every computer, every node is associated with a resource manager. I have already told you what is the function of resource manager in the earlier slides, and this resource manager may be used for the task scheduling and the load transfer, etc. Here the emphasis is on the task scheduling and load transfer.

Then global scheduling, here a single scheduler it handles balance load across the nodes. So, here in global scheduling we will use a single scheduler and this single scheduler will handle the balancing of the loads, across what? Across the different nodes. And here we use a specific transfer policy, so you have to use a particular transfer policy. So, these are the different classifications of the task scheduling solutions.

(Refer Slide Time: 18:28)



The image shows two presentation slides. The top slide is titled "DISTRIBUTED TASK SCHEDULING: AN INSIGHT" and contains two bullet points. The bottom slide is titled "CLASSIFICATION OF TASK SCHEDULING SOLUTIONS" and contains two bullet points. Both slides feature a small video inset of a man in a tan jacket speaking. The slides have a blue and green geometric design on the right side and logos at the bottom.

DISTRIBUTED TASK SCHEDULING: AN INSIGHT

- Almost all multiprocessor and distributed real-time systems built to date:
 - Use static scheduling.
- Dynamic solutions are more complex:
 - Also involve very high overheads.

CLASSIFICATION OF TASK SCHEDULING SOLUTIONS

- Broadly, there are two types of scheduling solutions:
 - **Static:** Tasks are statically assigned to processors
 - **Dynamic:** The processor which would execute a task is dynamically decided.

Now, we will look at the distributed task scheduling, we will just briefly look, an insight we will see into the distributed task scheduling. So, almost all the multiprocessor and the distributed real time systems built today, they use static scheduling techniques, static scheduling solutions, I have already told you what is static scheduling solution.

So, till now what multiprocessor systems are what, distributed real time systems you are observing or which are available to date, till now. They normally use the static scheduling solution. Why? Because they are very simple, easy to develop. But the dynamic scheduling solutions they are much more complex, they also involve very high overheads, they require much

more time to develop. So, that is why, almost all the multiprocessor and the distributed real time systems are developed so far, they use the static scheduling techniques.

(Refer Slide Time: 19:27)

THE SCHEDULING PROBLEM

- Actually two separate problems.
- Task assignment problem:
 - How to assign tasks to processors?
- Scheduling problem:
 - How to schedule the tasks on the processor to which it has been assigned?

The slide includes a video inset of a man speaking, and logos for IIT Bombay and NPTEL at the bottom.

Now, let us see, what you mean by the scheduling problem in case of the multiprocessor or what distributed system. So, normally the scheduling problem, actually it consists of two separate problems. What are the two separate problems? Number one the task assignment problem, number two, the task scheduling problem.

Let us see what do you mean by task assignment problem? Task assignment problem means, how to assign the tasks to processors. There are say 10 tasks but available only 5 processors. So, how will we assign the 10 tasks to the 5 processors? So, here task assignment problem is we have to divide the method so that we can effectively, efficiently assign the different tasks to the processors.

Then what do you mean by scheduling problems? How to schedule the tasks on the processor to which it has been assigned. To first what we are doing, we are first assigning the different tasks to the processors. Say I have already told you there are five processors P1 to P5. There are 10 tasks T1 to T10. How to decide which task will be assigned to which processor.

I mean which tasks you will assign to processor 1, which tasks will you assign to processor 2, which tasks will you assign to processor 5, like that. This is the first step. Then in the second step

what we are doing? We are deciding how to schedule the task on the processor, to which it has been assigned.

Suppose P5 gets say two tasks, then how to schedule those tasks, like P5 suppose gets T3 and T4. Now we have to decide how to schedule the task on the processors. So, P5 it will execute say T3 and T4. Which one will execute it fast? First T3 will execute it or T4 will be executed or how, or you can little bit enhance. Suppose more number of tasks is there. Say 15 tasks and 5 processors.

And in the first step what we are doing in task assignment problem? We are dividing these 15 tasks, we are assigning the 15 tasks among the 5 processors. Now coming to the second step what are doing, we have to schedule the task on the processor to which it has been assigned. Supposed this processor P5 it has been assigned task T1, T2, T3.

Now in the second step, that means in the scheduling problem we have to decide, we have to schedule the task in such a way that these tasks will be executed efficiently on the assigned processors P5, that means whether we will first assign T1 or T2 or T3, in which sequence. We have to assign, we have to execute the task in the processor so that they can, the utilization can be maximum. So, how to assign the different tasks, how to prepare this schedule, how to prepare the sequence of execution of the task on the processor, say here processor P5, to which it has been assigned so that maybe the utilization will be efficient, the utilization will be maximum.

(Refer Slide Time: 22:38)

OPTIMAL SCHEDULERS

- There are optimal schedulers for uniprocessors:
 - Static --- Rate Monotonic Algorithm (RMA)
 - Dynamic --- Earliest Deadline First (EDF)
- What are their complexities:
 - Linear for RMA
 - Log n for EDF
- Real-time task scheduling in multiprocessor/distributed systems:
 - NP hard

The slide features a video inset of a man in a suit on the right side. At the bottom, there are logos for a university and a company named 'HOTEL'.

Then we will see about these optimal schedulers I hope in this, while discussing about the scheduling, you have must have known the terms optimal schedulers, efficient schedulers proficient schedulers, those definitions you just have known earlier. So, here we will see about little bit basic concept of the optimal schedulers. So, there are optimal schedulers for uni processors systems, you have already seen. They can be, while you have already learned in the earlier classes on the optimal schedulers for uniprocessors, you might have seen there are two important types of optimal schedulers. One is the static scheduler; another is the dynamic schedulers for the uniprocessors.

The example of what a static scheduler, you have seen for uniprocessors, it is RMA, which stands for Rate Monotonic Algorithm another is the dynamic. The dynamic scheduler for the uni processor, the example is EDF that is the earliest deadline first. I hope in the earlier classes you have already seen the details of RMA and EDF and the variations of RMA and EDF. Now the complexity is also we have already seen for RMA and EDF. So, you know that the complexity for RMA normally it is linear, while complexity of EDF is of the order of log n. So, I again what advice you to see these details of RMA and their variations and EDF and their variations along with their complexities advantages disadvantages. The cases where they can be applied.

Please see those things in detail from any book or from the slides. So, you have also known earlier that normally real time tasks scheduling in case of uniprocessor systems is quite easy. It is

quite simple, but in case of this real time task scheduling in multiprocessor systems or distributed systems it is NP hard. I hope what is NP hard, NP complete you have already known earlier. So, I want to conclude that the real time task scheduling in multiprocessor or distributed systems is NP hard in nature.

(Refer Slide Time: 24:57)

SUMMARY

- Discussed broad classification of computers.
- Explained the concepts of Real-time distributed systems.
- Presented classification of task scheduling solutions in distributed systems.
- Explained the concept of task scheduling problem.
- Briefly discussed the concept of optimal schedulers.

THE SCHEDULING PROBLEM

- Actually two separate problems.
- Task assignment problem:
 - How to assign tasks to processors?
- Scheduling problem:
 - How to schedule the tasks on the processor to which it has been assigned?

OPTIMAL SCHEDULERS

- There are optimal schedulers for uniprocessors:
 - Static --- Rate Monotonic Algorithm (RMA)
 - Dynamic --- Earliest Deadline First (EDF)
- What are their complexities:
 - Linear for RMA
 - Log n for EDF
- Real-time task scheduling in multiprocessor/distributed systems:
 - NP hard



CLASSIFICATION OF TASK SCHEDULING SOLUTIONS

- Broadly, there are two types of scheduling solutions:
 - **Static:** Tasks are statically assigned to processors
 - **Dynamic:** The processor which would execute a task is dynamically decided.



So, today let us say what we have discussed? We have discussed so far the broad classification of computers. There I have already explained you the differences between UMA and the NUMA computers. Then we have also explained the concept of the real time distributed systems. Why real time distributed systems are required I have already told you.

The advantages of real time distributed systems I have already told you. Also, we have presented the classification of the different task scheduling solutions in the distributed systems. For example the static, dynamic and global scheduling solutions, we have already told you earlier. I have also explained the concept of task scheduling problem. I have already told you, that this task scheduling problem consists of two separate problems.

One is the task assignment problem, where the emphasis is on how to assign the task to the processors and the second problem is scheduling problem where the emphasis is on how to schedule the different tasks on the particular processor to which it had been assigned so that also we have seen regarding the scheduling problem.

We have also briefly discussed the concept of the optimal schedulers. I have already told you that for you uniprocessor systems you have seen different types optimal schedulers or static schedulers and dynamic and examples of static you have seen, static scheduler is RMA and example of dynamic scheduler that we have already seen, for uniprocessors systems. I am again repeating this is for uniprocessor systems.

For uniprocessor systems, for the dynamic scheduler you have already the EDF, and the time complexity also you have known that for RMA it is of the, it is linear, whereas for EDF it is of the order of $\log n$, but real time scheduling in case of multiprocessor and distributed systems, it is NP hard in nature. So, these things we have discussed in today's class. Next class we will discuss about the particular types of these task scheduling techniques, for multiprocessor systems. So, maybe we will, if I will start with next class we will discuss about this static testing. We will discuss about this static scheduling.

So, on the static scheduling next class we will discuss three important techniques, we will see for static scheduling of task in multiprocessor systems, one is in, what, uniform, what, scheduling then next fit scheduling and bin packing scheduling, so those are coming under the static scheduling techniques for multiprocessor systems, those static testing techniques we will discuss in the next class and after that we will see the available dynamic scheduling techniques for multiprocessor systems. So, that we will discuss maybe after one class.

(Refer Slide Time: 28:06)



REFERENCES

1. Rajib Mall, Real-Time Systems: Theory and Practice, 1st Edition, Pearson Education, 2007.
2. C.M. Krishna, K.G. Shin, Real-Time Systems: Theory and Practice, Tata McGraw-Hill Education, 2010

The slide features a dark red header with the word 'REFERENCES' in white. Below the header is a list of two references. To the right of the text is a video inset showing a man with a beard and glasses, wearing a light-colored shirt, speaking. The background of the slide is white with a blue and green geometric design on the right side. At the bottom, there are two logos: a tree logo on the left and a circular logo with the word 'NOTEL' on the right.

This is for today, and all those concepts, these contents I have taken from this Rajib Mall's book on real time systems, written by Professor Rajib Mall of IIT Kharagpur. This book was published by Pearson Education. Also some of the important concepts we have taken from this real time systems book by C. M. Krishna and K. G. Shin, published by Tata McGraw-Hill, so you may refer these books. You may also refer some additional materials from internet or so and like this, so if any doubt you can post it. So, this is what we have discussed today. Thank you very much for your patient hearing.