

Real Time Systems
Professor Durga Prasad Mohapatra
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Lecture 61

Concurrency Control in RT Databases and Commercial RT Databases

Good afternoon to all of you. Today we will take up this lecture on some more concurrency control protocols in real time databases importantly this OCC Optimistic Concurrency Control Protocols. And then we will discuss a few commercial real time databases.

(Refer Slide Time: 00:41)

CONCEPTS COVERED

- Optimistic Concurrency Control (OCC) Protocols
- Speculative Concurrency Control (SCC) Protocol
- Comparison of Concurrency Control Protocols
- Databases for Hard Real-Time Systems
- Commercial Real-Time Databases

The slide features a video inset of Professor Durga Prasad Mohapatra in the bottom right corner. At the bottom of the slide, there are logos for NITRR (National Institute of Technology, Rourkela) and NPTEL (National Programme on Technology Enhanced Learning).

So, first we will discuss some popular optimistic concurrency control protocols then we will discuss another protocol named as Speculative concurrency control SCC then we will see about the comparison of the different concurrency control protocols. So, far we have seen little bit of all the databases for soft real time systems so now we will see little bit on the databases for hard real time systems. Finally, we will see some of the commonly popular available real time databases.

(Refer Slide Time: 01:08)

KEYWORDS

- Optimistic Concurrency Control
- Conflict Set
- Shadow Version
- MDARTS
- eXtremeDB

The slide features a dark blue header with the word 'KEYWORDS' in white. Below the header, five bullet points are listed, each preceded by a right-pointing arrowhead. The slide has a decorative background with blue and green geometric shapes on the right side. A small video inset of a man is visible in the bottom right corner. At the bottom, there are logos for IIT Bombay and NPTEL.

So, these are the keywords we will use OCC or optimistic concurrency control, conflict set, shadow version, MDARTS, extreme DB. Etc.

(Refer Slide Time: 01:20)

CONCURRENCY CONTROL IN R-T DATABASES

- Concurrency control protocols allow several transactions to access a database concurrently,
 - but, leave the database consistent by enforcing serializability.
- Two main categories of protocols:
 - **Pessimistic protocols:** Disallow certain types of transactions from progressing.
 - **Optimistic protocols:** Allow all transactions to progress without any restrictions, and then prune some of the transactions.

The slide has a dark blue header with the title 'CONCURRENCY CONTROL IN R-T DATABASES' in white. The main content consists of two bullet points. The first bullet point has a sub-bullet point. The second bullet point has two sub-bullet points. The slide has a decorative background with blue and green geometric shapes on the right side. A small video inset of a man is visible in the bottom right corner. At the bottom, there are logos for IIT Bombay and NPTEL.

Let us start with the what OCC protocol? Last class I have already told you the concurrency control protocols what do they do? They allow several transactions to access a database concurrently without affecting the consistency property. How? By ensuring serializability. So, two major categories of concurrency control protocols I have told in last class that is pessimistic protocols

and optimistic protocols. And in the last class we have already discussed about so many pessimistic protocols.

(Refer Slide Time: 01:51)

OPTIMISTIC CONCURRENCY CONTROL (OCC) PROTOCOLS

- They do not prevent any transaction from accessing any data items it requires.
- A transaction is validated at the time of its commitment and any conflicting transactions are aborted.
- If there is little contention and interference among transactions,
 - Most transactions would be successfully validated.

The slide features a dark blue header with the title in white. The main content is on a white background with blue and green text. A video inset on the right shows a man speaking. Logos for a university and NPTEL are at the bottom.

So, today we will discuss about the, some popular optimistic protocols. Before going to these popular optimistic protocols let us first see about some basic characteristics. So, these popular protocols are known as OCC which stands for Optimistic Concurrency control protocols. So, these protocols as I have already told you they do not prevent any transaction you need not have to take any permission.

So, they do not prevent any transaction from accessing any data items it requires that allows all the transactions to access the database. As I have already told you later on some of these what transactions may be pruned. Let us see how it is happening.

A transaction is validated at the time of its commitment. First, these protocols they do not prevent any transaction they allow all the transactions to access the database, to access the any data items in the database later on some of the what transactions may be pruned. How? By using some validation test. So, my transaction is validated at the time of its commitment and if there are any conflicting transactions found they are aborted.

So, in this way these optimistic concurrency control protocols they work. If there is a very little contention and interference among a little interference among transactions till then most transactions, they would be successfully validated no problem will arise.

(Refer Slide Time: 03:11)

OCC PROTOCOLS cont...

- **Under heavy load conditions,**
 - **There can be a larger number of transactions that fail the validation test and are aborted.**
- **This might lead to severe reduction in throughput and sharp increase in deadline misses.**
- **Thus, the performance of OCC protocols can show a marked drop at high loads.**

The slide features a dark blue header with the title 'OCC PROTOCOLS cont...' in white. The main content is a list of three bullet points in red text. The first bullet point is followed by a sub-bullet point. The slide also includes a small video inset of a man in a suit on the right side and logos for IIT Bombay and NIPTEL at the bottom.

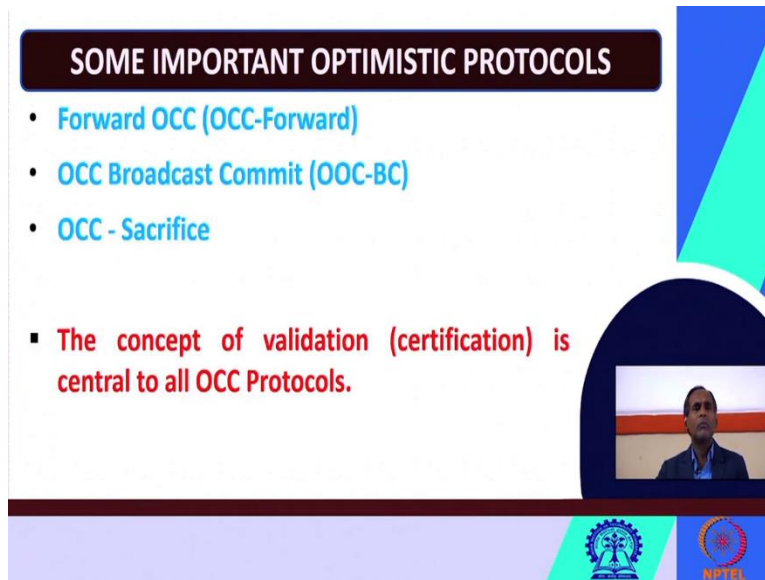
But there will be problem under heavy load conditions. If there is only little contention and interference among transactions then most of the transactions they will be successfully validated. But, in under heavy load conditions what will happen there can be a large number of transactions that failed the validation test.

I have already told you so a transaction has to be validated at the time of commitment and if any conflicting transactions will be found then they will be aborted. So, what may happen under heavy load conditions there can be a large number of transactions which may failed the validation test. And if they will be failed, they have to be what aborted.

So, if they will be aborted what will happen this might to severe reduction in the throughput then the throughput will be significantly reduced there will be sharp increase in the deadline misses and you will get a greater number of deadlines misses so many transactions, they will be miss their deadlines. So, if there will be reduction in throughput and so many transactions will miss their deadlines then what will happen the performance of the OCC protocol obviously, they will show a marked drop at high loads.

Whenever there will be high loads the performance will be degraded, the performance will be dropped. So, what will happen under heavy load conditions the performance of the OCC protocols maybe significantly what reduced. Thus, the performance of the OCC protocols can show a marked drop at high loads.

(Refer Slide Time: 04:38)



SOME IMPORTANT OPTIMISTIC PROTOCOLS

- Forward OCC (OCC-Forward)
- OCC Broadcast Commit (OCC-BC)
- OCC - Sacrifice

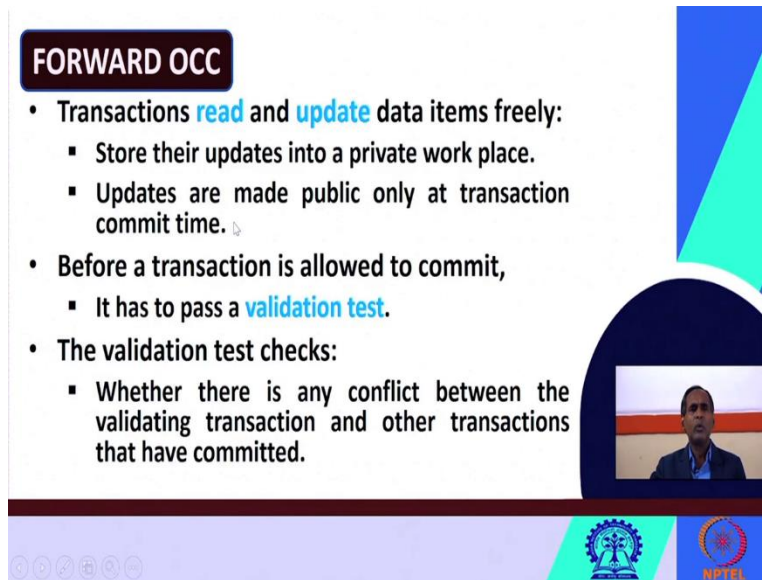
▪ **The concept of validation (certification) is central to all OCC Protocols.**

The slide features a dark blue header with the title in white. The main content is on a white background with blue and red text. A small video inset shows a man speaking. The footer contains logos for IIT Bombay and NPTEL.

Now, let us see some important optimistic protocols. First, we will discuss about a protocol called as a forward OCC in short, we will call as OCC forward, then OCC Broadcast Commit or OCC BC and then we will see OCC Sacrifice. These are the three important protocols now we will discuss but please remember for all these three important protocols the concept of validation is central to all OCC protocols.

I have already told you in OCC protocols a validation check, a validation test has to be performed at the time of Commit. When the transactions will commit, they have to undergo a validation or certification test. So, the concept of validation or certification is central to all OCC protocols. This validation test or certification test has to be carried out for all these three important optimistic protocols.

(Refer Slide Time: 05:26)



FORWARD OCC

- Transactions **read** and **update** data items freely:
 - Store their updates into a private work place.
 - Updates are made public only at transaction commit time.
- Before a transaction is allowed to commit,
 - It has to pass a **validation test**.
- The validation test checks:
 - Whether there is any conflict between the validating transaction and other transactions that have committed.

The slide features a video inset of a man speaking, a blue and green geometric design on the right, and a footer with navigation icons and logos for IIT Bombay and NPTEL.

We will start with the forward OCC. In forward OCC let us see how does it work in forward OCC the transactions they read and update data items freely. So, as this name suggests OCC or optimistic concurrency control protocol in these protocols the transactions they read and update the data items freely. They store their updates into a private work place. So, instead of storing the updates in a public work place these protocols store their updates in to a private work place the updates are made public only at transaction commit time.

So, the updates when they will be made public? The updates will be made public during the transaction commit time. Before a transaction is allowed to Commit it has to pass a validation test. This I have already told. Then what you are performing in validation test? Or what is performed in the validation test? The validation test checks whether there is any conflict between the validating transaction and the other remaining transactions that have committed.

I am repeating again. What is performed during validation test? The validation test it is checks whether there is any conflict between the validating transaction and the other transactions which have committed so it will check this validation test.

(Refer Slide Time: 06:44)

FORWARD OCC cont...

- A transaction is aborted:
 - If it does not pass the validation test.
 - This guarantees the **atomicity** and **durability** properties.
- Since writes occur only at commit time,
 - the **serialization order in OCC is the order in which the transactions commit.**

The slide features a video inset of a man speaking in the bottom right corner. At the bottom, there are logos for IIT Bombay and NPTEL, along with navigation icons on the left.

Now, let us see the outcome of the validation test and what action will be taken. A transaction is aborted if it does not pass the validation test. So, if the transaction does not pass the validation test it has to be aborted. Why? Because this will guarantee the atomicity and durability properties. I have already told you the its properties. So, if a transaction does not pass through the validation test it has to be aborted in order to guarantee the atomicity and durability properties.

Since, writes occur only at commit time I have already told you once the write will occur writes occur only at the commit time. The serialization order in OCC is the order in which the transactions commit. So, since the writes normally they occur only at the Commit time the serialization order in the optimistic control protocol it is the order in which the transactions commit. So, this is something about the forward OCC. How does it work.

(Refer Slide Time: 07:43)

OCC BROADCAST COMMIT

- In OCC Broadcast Commit (OCC-BC) protocol:
 - When a transaction commits, it notifies its intention to all other running transactions.
- Each running transaction carries out a test to check:
 - If it has any conflicts with the committing transaction.
 - If any conflicts are detected, then the transaction carrying out the check immediately aborts itself and restarts.

The slide features a video inset of a man speaking, a navigation bar at the bottom with icons, and logos for IITM and NIPTEL.

Then we will see another OCC protocol that is OCC Broadcast Commit. In short, we will call it as OCC BC. So, in OCC BC protocol, what happens when a transaction wants to commit? It notifies its intention to all other running transactions. So, in OCC, when any transaction commits, it notifies its intention to all other running transactions. Each running transaction carries out a test to check.

So, each of the running transactions carries out a test to check. This might be the same validation test. Why does it carry out a test? In order to know that if it has any conflicts with the committing transaction. So, each running transaction has to carry out a test in order to check whether there are any conflicts with the committing transaction.

Now, let us say there are two possibilities: there might be some conflicts, or there might not be some conflicts. So, if any conflicts are detected, then the transaction carrying out the check immediately aborts itself and restarts. So, in OCC BC, if any conflicts are detected, then the transaction carrying out the check immediately aborts itself and then it is restarted.

(Refer Slide Time: 08:57)

OCC BROADCAST COMMIT cont...

- There is no need for a committing transaction to check for conflicts with already committed transactions:
 - **Because if it were in conflict with any of the committed transactions, it would have already been aborted.**
- Thus, in OCC-BC once a transaction reaches its validating phase,
 - **It is guaranteed commitment.**

Please remember, here that is no need for a committing transaction to check for conflicts with already committed transactions. This is the advantage of OCC broadcast Commit. Here there is no need for a committing transaction to again check for a conflict with the already committed transaction. Why? Because, if it were in conflict with any of the earlier committed transactions then it would have been already been aborted why it will come so far.

So, there is no need for a committing transaction to check for conflicts with already committed transactions. Thus, in OCC BC protocol what will happen? Once a transaction it reaches its validating phase it is guaranteed to Commit. Thus, in OCC BC protocol once a transaction it reaches its validating phase that means it is guaranteed commitment it will definitely commit.

(Refer Slide Time: 09:50)

OCC BROADCAST COMMIT cont...

- Compared to OCC-forward,
 - It encounters earlier restarts and less wasted computations.
- This protocol performs better than the OCC-forward protocol in meeting task deadlines.
- It does not consider the priorities of transactions.

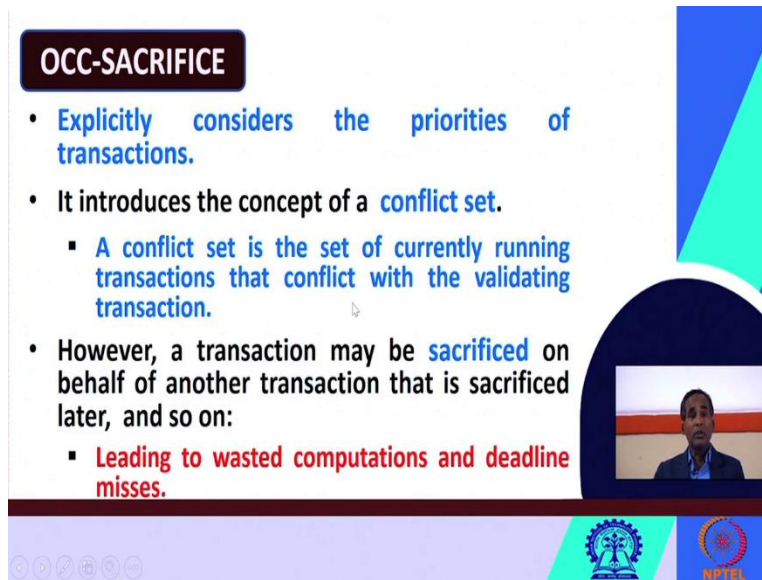
NIPTEL

Now, let us quickly look at a comparison between this OCC broadcast and OCC forward. So, compared to OCC forward this OCC broadcast it encounters earlier restarts and less wasted computations. So, in comparison to OCC forward in OCC broadcast if there is any what restart it is performed early. It encounters earlier restarts and if this restart thing procedure is carried out much earlier obviously it will result in less wasted computations.

The computations or very less amount of computations it will be wasted. So, this protocol performs better than OCC forward protocol in meeting task deadlines. So, if there will be a less amount of computations so, they will be wasted? Obviously, the chances of meeting the deadline will be more or the chances of missing the deadline very less. So, that is why this protocol performs much better than the OCC forward protocol in meeting the deadlines of the tasks.

But, one drawback is there in OCC broadcast what is that drawback? It does not consider the priorities of the transactions. So, what are the priorities of transactions OCC broadcast Commit protocol does not consider these protocols do not take into account the priorities of the transactions. Now, let us see how this problem can be overcome. That means how to consider the priorities of the transactions.

(Refer Slide Time: 11:12)



OCC-SACRIFICE

- Explicitly considers the priorities of transactions.
- It introduces the concept of a conflict set.
 - A conflict set is the set of currently running transactions that conflict with the validating transaction.
- However, a transaction may be sacrificed on behalf of another transaction that is sacrificed later, and so on:
 - Leading to wasted computations and deadline misses.

The slide features a video inset of a man speaking, a navigation bar at the bottom with icons, and logos for IIT Bombay and NPTEL.

So, another protocol is there called as OCC Sacrifice. This protocol overcomes this earlier problem regarding handling of the priorities of the transactions or considering the priorities of the transactions.

This OCC Sacrifice protocol you can consider it as an extension of the OCC BC protocol. This protocol overcomes this problem of not considering the priorities of the transactions by OCC BC it explicitly considers the priorities of the transactions. This OCC Sacrifice it introduces the concept of a conflict set. So, this OCC Sacrifice protocol it introduces the concept of a conflict set. And what is a conflict set?

A conflict set is the set of currently running transactions which conflict to the validating transactions. However, a transaction may be sacrificed on behalf of another transaction that is sacrificed later please see what is the problem with this OCC Sacrifice. Why the name is OCC Sacrifice? In this protocol a transaction maybe sacrificed on behalf of another transaction which is sacrificed later on. And so on this may continue.

And if this will continue this will automatically lead to wastage of computations and there will be chance heavy chance that the deadlines of the tasks will be missed. So, since here a transaction may be sacrificed on behalf of another transaction that is sacrificed later on and so on this process carried out that is why the name is OCC sacrificed.

And since a transaction is sacrificed on behalf of another transaction and this process maybe continued so there is a chance that it may lead to heavy amount of wasted computations and this will result in deadline misses. Many of the transactions may meet that deadline. This is the problem of OCC Sacrifice.

(Refer Slide Time: 13:00)

OCC-SACRIFICE cont...

- A transaction once reaches its validation stage,
 - Checks for conflicts with the currently executing transactions.
 - If conflicts are detected and one or more of the transactions in the conflict set has higher priority than the validating transaction,
 - **then the validating transaction is aborted and restarted.**
 - Otherwise, all transactions in the conflict set are aborted and restarted.

Now, let us see how OCC Sacrifice work. A transaction once reaches its validation stage the transaction then checks for conflicts with the currently executing transactions. If conflicts are detected if during this validation test if conflicts are detected and one or more than one transaction in the conflict set have higher priority than the validation transaction.

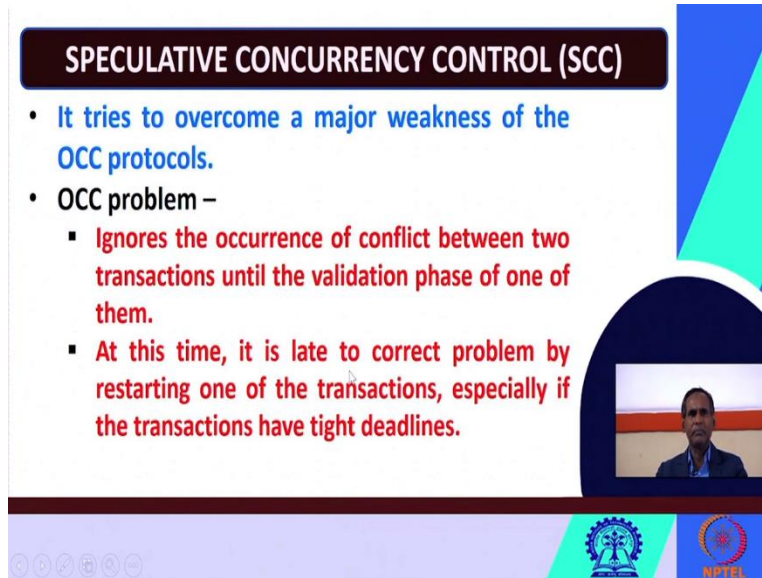
Please see what I am saying. If during the validation test it is found that some conflicts are detected and one or more of the transactions in the conflict set, they have higher priority than the validating transaction and what action will take? Immediately the validating transaction is aborted and restarted.

So, if during the validation test some conflicts are detected and one or more of the transactions in the conflict set, they have higher priority than the validating transaction then what action will be taken? Then the validating transaction is aborted and restarted.

Otherwise, that means if there is no transaction which has a higher priority than the validating transaction that means your validating transaction is having the highest priority. If no transaction in the conflict set has higher priority that means what? The validation transaction has the highest

priority. Then all the transactions in the conflict sets they are aborted and restarted. This is how the OCC Sacrifice protocol works.

(Refer Slide Time: 14:38)



SPECULATIVE CONCURRENCY CONTROL (SCC)

- It tries to overcome a major weakness of the OCC protocols.
- OCC problem –
 - Ignores the occurrence of conflict between two transactions until the validation phase of one of them.
 - At this time, it is late to correct problem by restarting one of the transactions, especially if the transactions have tight deadlines.

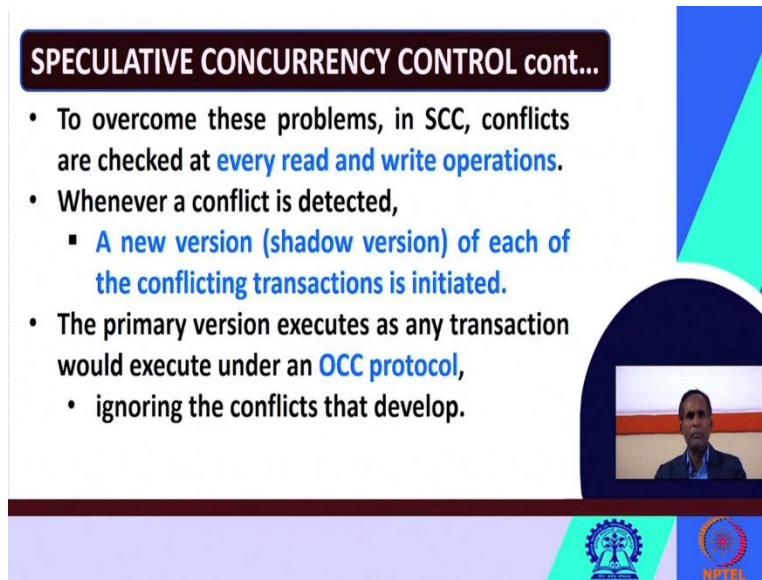
The slide features a dark blue header with the title in white. The main content is on a white background with blue and red text. A small video inset in the bottom right shows a man speaking. The bottom of the slide has a blue bar with navigation icons and logos for IIT Bombay and NPTEL.

Now, we will see about a new protocol called SCC which stands for Speculative Concurrency control protocol. So, what is its objective? The objective of this protocol is to overcome a major weakness of the OCC protocols. It overcomes major weakness a major limitation of the OCC protocols. Let us see what is that major problem.

In OCC you know what happens OCC protocols they ignore the occurrence of a conflict between two transactions until what point? Until the validation phase of one of them. Until the validation phase arises this OCC ignores the occurrence of conflicts then what will be consequence? By this time, it will be too late to correct the problem by restarting one of the transactions. So since, the conflict is not known until the validation phase so at this time it will be too late to correct the problem by restarting one of the transactions.

Especially, this problem will be very much acute if the transactions are very closed deadlines, deadlines are very tight deadlines. Then in that case the transaction they missed they are deadlines. So, this is one of the major problems in OCC. Let us see how SCC overcomes this problem.

(Refer Slide Time: 15:51)



SPECULATIVE CONCURRENCY CONTROL cont...

- To overcome these problems, in SCC, conflicts are checked at **every read and write operations**.
- Whenever a conflict is detected,
 - **A new version (shadow version) of each of the conflicting transactions is initiated.**
- The primary version executes as any transaction would execute under an **OCC protocol**,
 - ignoring the conflicts that develop.

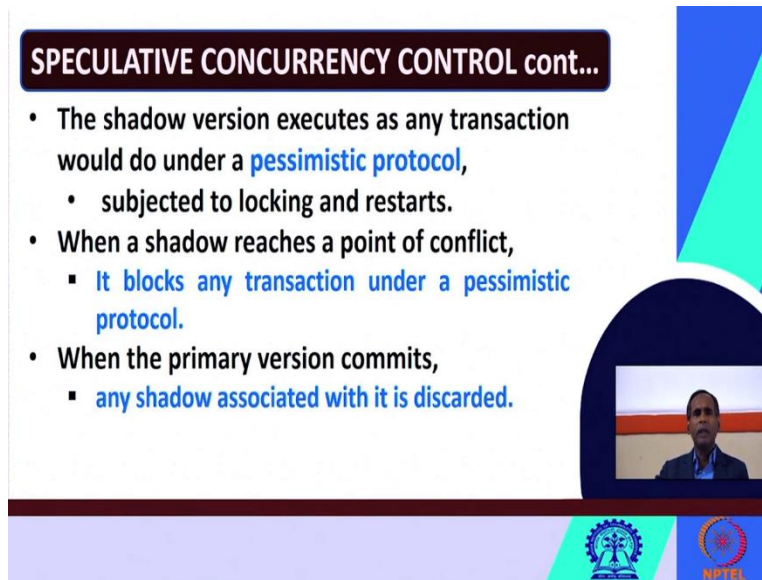
The slide features a video inset of a man speaking, set against a background with blue and green geometric shapes. At the bottom, there are logos for IIT Bombay and NPTEL.

To overcome this problem in SCC conflicts are checked at every read and write operations. We have seen in the OCC the conflicts are checked the conflicts are found out during the validation phase of one of them. And hence, there is a chance that some of the transactions may miss their deadlines to overcome these problems in SCC the conflicts are when checked?

The conflicts are checked at every read and write operations. Now, let us see after this checking if conflicts are detected what will happen? Whenever a conflict is detected then a new version of each of the conflicting transactions is initiated. So, whenever a conflict is found a new version of each of the conflicting transactions is initiated this new version we will call as shadow version of the transaction.

Now, there are two versions the primary version and the shadow version. Let us see how the primary version will execute and how the shadow version will be executed. The primary version it executes just like as any transaction would execute under an OCC protocol. So, the primary version will execute just like as any transactions which may execute under an OCC protocol. By ignoring the conflicts that develop. Now, let us see how the shadow version will execute.

(Refer Slide Time: 17:12)



SPECULATIVE CONCURRENCY CONTROL cont...

- The shadow version executes as any transaction would do under a **pessimistic protocol**,
 - subjected to locking and restarts.
- When a shadow reaches a point of conflict,
 - **It blocks any transaction under a pessimistic protocol.**
- When the primary version commits,
 - **any shadow associated with it is discarded.**

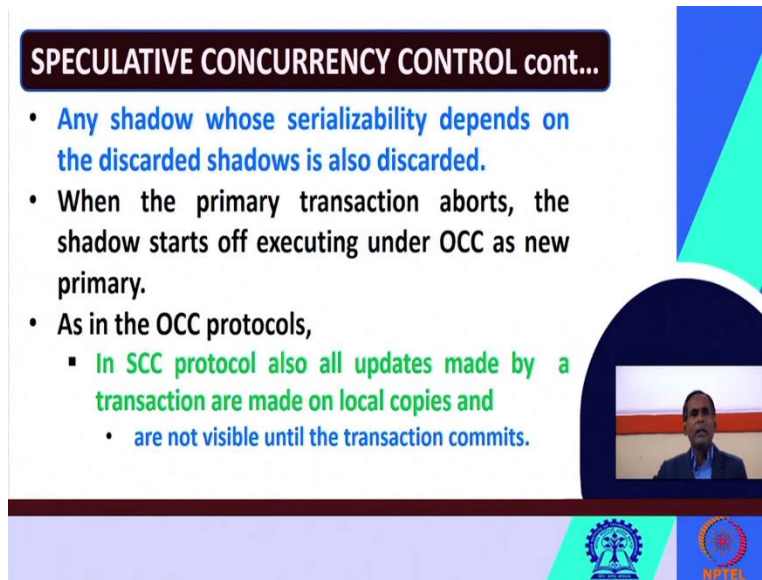
The slide features a dark blue header with the title in white. The main content is a bulleted list with blue text. A video inset in the bottom right shows a man speaking. The slide is decorated with blue and green geometric shapes and logos for IIT Bombay and NPTEL at the bottom.

The shadow version that will execute as any transaction would do under a pessimistic protocol. So, we have seen that the primary version will execute as any transaction would execute under an OCC protocol and the shadow version will execute as any transaction would do under a pessimistic protocol. And you know that in pessimistic protocols we use the concept of locking.

So, subject to locking and restarts. When a shadow reaches a point of a conflict, now let us see when after the shadow version is executed what will happen? When a shadow reaches a point of a conflict then it blocks any transaction under a pessimistic protocol. And when the primary version commits so what will happen let us see.

Suppose this shadow version it has reached the reached a point of conflict and it blocks any transaction under pessimistic protocol. Now, when the primary version commits now if the primary version commits what will happen? When the primary version will Commits any shadow associated with the primary version it is discarded. Because the primary version is also committed so no need of considering shadow.

(Refer Slide Time: 18:25)



SPECULATIVE CONCURRENCY CONTROL cont...

- Any shadow whose serializability depends on the discarded shadows is also discarded.
- When the primary transaction aborts, the shadow starts off executing under OCC as new primary.
- As in the OCC protocols,
 - In SCC protocol also all updates made by a transaction are made on local copies and
 - are not visible until the transaction commits.

The slide features a video inset of a man in a suit and glasses. The background has a blue and green geometric design. Logos for IIT Bombay and NPTEL are visible at the bottom.

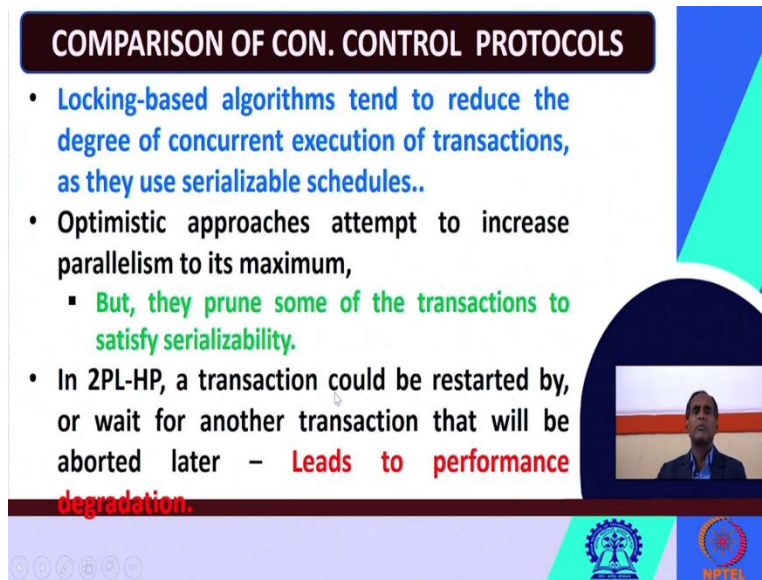
So, when the shadow version commits any shadow associated with it will be discarded. Any shadow whose serialization depends on the discarded shadow is also discarded. So, note to once I have already told you when primary version commits any shadow associated with the primary version is discarded. Not only that that shadow version but also any shadow version whose serializability depends on the earlier discarded shadow that is also discarded.

When the primary transaction aborts, the shadow starts off executing under OCC as new primary. Suppose, now there are two cases if primary version commits then the shadow is discarded not only that shadow but also any shadow whose serializability depends on the discarded shadow that is also discarded.

Now, the second option the primary transaction may abort. So, what will happen if primary transaction aborts? When the primary transaction aborts then the shadow starts off executing under OCC as new primary. So, when the primary transaction aborts then this shadow version it immediately starts off executing under OCC as new primary. Just like in OCC protocol this shadow version it immediately starts executing under OCC as the new primary version.

As in the OCC protocols let us see one similarity between OCC protocols and SCC protocols. As in the OCC protocols in SCC protocols also all the updates made by a transaction they are made only of the local copies. And hence, they are not visible until the transaction commits. So, was happening in OCC so this is happening in SCC.

(Refer Slide Time: 19:58)



COMPARISON OF CON. CONTROL PROTOCOLS

- Locking-based algorithms tend to reduce the degree of concurrent execution of transactions, as they use serializable schedules..
- Optimistic approaches attempt to increase parallelism to its maximum,
 - But, they prune some of the transactions to satisfy serializability.
- In 2PL-HP, a transaction could be restarted by, or wait for another transaction that will be aborted later - Leads to performance degradation.

The slide features a video inset of a man speaking, a navigation bar at the bottom with icons, and logos for IIT Bombay and NPTEL.

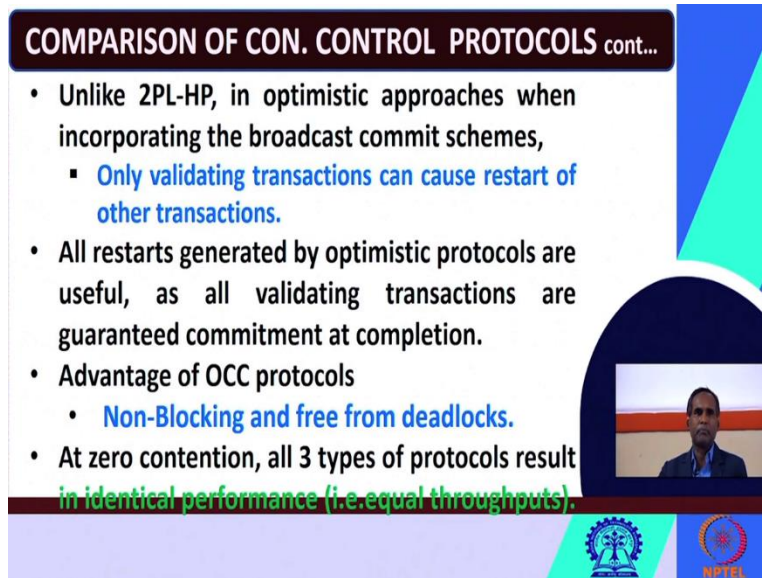
So, we have successfully seen three OCC protocols, OCC forward, OCC BC and OCC Sacrifice. Then also we have seen the SCC speculative concurrency control protocol now let's quickly compare the different concurrency control protocols we have seen. So, we have seen the pessimistic protocols.

I have told you that the pessimistic protocol they are using locking based algorithms. The locking-based algorithms tends to reduce the degree of concurrent execution of transactions or they reduce the degree of Concurrency. Why? Because, they use serializable shadows. On the other hand, the Optimistic approaches the Optimistic algorithms they attempt to increase the concurrency to increase the parallelism to its maximum.

But we have already told you the Optimistic protocols they prune some of the transactions to satisfy the serializability property. We have last class we already have seen about the pessimistic protocols such as 2PL-HP, 2 PL WP etc. In 2PL HP a transaction could be restarted by, or wait for another transaction that will be aborted later.

In 2PL HP what happens a transaction it could be restarted by or wait for another transaction which will be aborted later. This will lead to performance degradation. In this case in case of 2PL HP there would be much degradation of performance.

(Refer Slide Time: 21:19)



COMPARISON OF CON. CONTROL PROTOCOLS cont...

- Unlike 2PL-HP, in optimistic approaches when incorporating the broadcast commit schemes,
 - Only validating transactions can cause restart of other transactions.
- All restarts generated by optimistic protocols are useful, as all validating transactions are guaranteed commitment at completion.
- Advantage of OCC protocols
 - Non-Blocking and free from deadlocks.
- At zero contention, all 3 types of protocols result in identical performance (i.e. equal throughputs).

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 speaking. At the bottom, there are logos for IIT Bombay and NPTEL.

Unlike 2PL, HP in the optimistic concurrency protocols in optimistic approaches when incorporating the broadcast commit schemes only validating transactions can cause restart of other transactions. Not all the transactions can cause the restart of other transactions. Only the validating transactions they can cause restart of the other transactions.

All restarts generated by Optimistic protocols are useful. Please remember, in case of OCC all the restarts generated by the OCC they are useful because all the validating transactions are guaranteed to Commit. Some of the advantages of OCC protocols are as follows they are non blocking in nature. We have already known what is a blocking and non blocking they are non blocking in nature and they are free from deadlocks. So, they do not suffer from deadlocks.

At zero contention if you will see the performance of these protocols you may observe that at zero contention all the three types of OCC protocols such as OCC forward, OCC BC and OCC Sacrifice they result in identical performance. At zero contention all these three types of OCC protocols they result in identical performance that is almost equal throughputs you will get in zero contention for all these three types suppose OCC protocols.

(Refer Slide Time: 22:36)

COMPARISON OF CON. CONTROL PROTOCOLS cont...

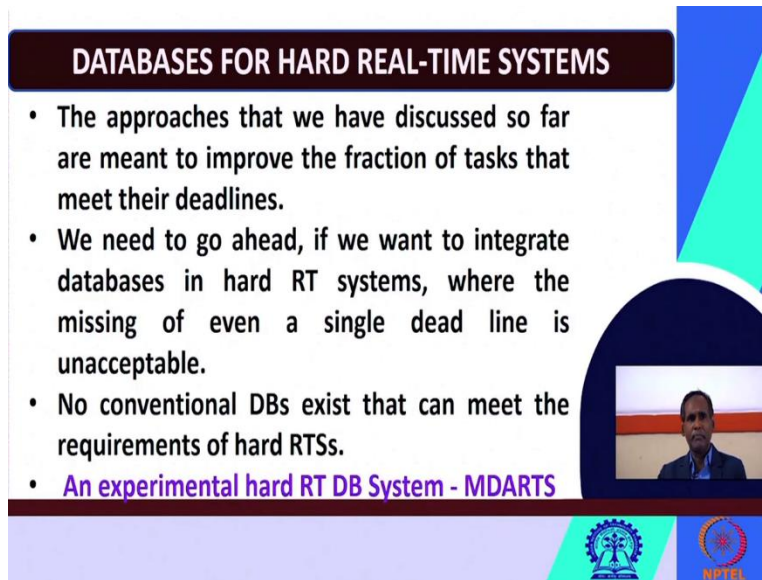
- At low conflicts, OCC outperforms Pessimistic protocols.
- However, Pessimistic protocols perform better as the load becomes higher.
- SCC performs better compared to both OCC and Pessimistic protocols,
 - when transactions have tight deadlines and the load is moderate.

The slide features a video inset of a speaker in the bottom right corner. At the bottom, there are logos for IIT Bombay and NIPTEU.

At low conflicts, OCC outperforms the pessimistic protocols. However, the pessimistic control protocols they perform better as the load becomes higher. When the load is becoming more the pessimistic protocols they performed better than the OCC protocols.

The SCC protocols they perform better compared to both OCC and the pessimistic protocols, when transactions have tight deadlines and the load is moderate. So, when the transactions they are very closed deadlines they have very tight deadlines and the load is moderate not so high not very low then SCC performs better compared to both the OCC and pessimistic protocols. This is a brief comparison of the different concurrency control protocols that we have discussed so far.

(Refer Slide Time: 23:25)



DATABASES FOR HARD REAL-TIME SYSTEMS

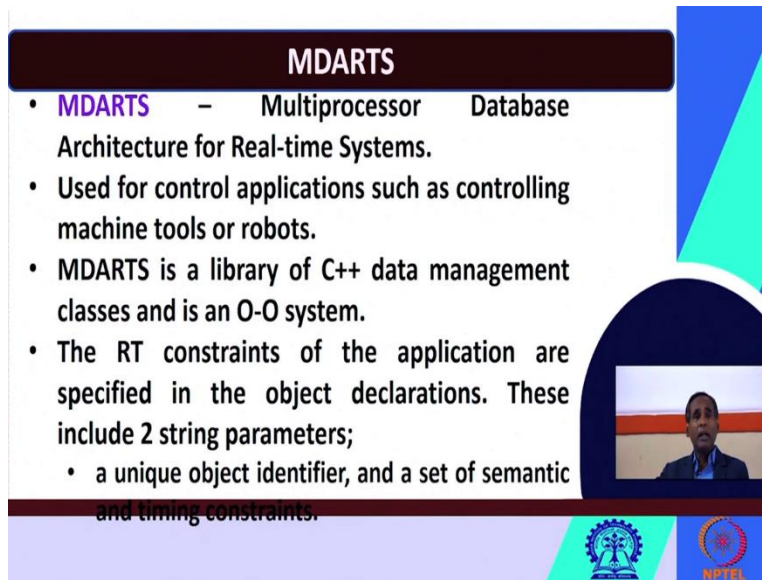
- The approaches that we have discussed so far are meant to improve the fraction of tasks that meet their deadlines.
- We need to go ahead, if we want to integrate databases in hard RT systems, where the missing of even a single dead line is unacceptable.
- No conventional DBs exist that can meet the requirements of hard RTs.
- **An experimental hard RT DB System - MDARTS**

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 video inset shows a man speaking. At the bottom, there are logos for IIT Bombay and NITEL.

So now let us quickly look at the databases for hard real time systems the approaches that we have discussed so far, they are normally meant to improve the fraction of the tasks that meet their deadlines. If we need to go further then we have to integrate database in hard real time systems. We need to go ahead if we want to integrate the databases in hard real time systems where the missing of a even a very a single deadline it is unacceptable.

You cannot simply miss a single deadline also that is unacceptable. No conventional databases exist so far that can meet the requirements of hard real time systems. I have taken a very small example of a hard real time system very rare are there. An experimental hard real time database system is MDARTS. This is good example of a what hard real time database systems we will see little bit about this hard real time database systems MDARTS.

(Refer Slide Time: 24:25)



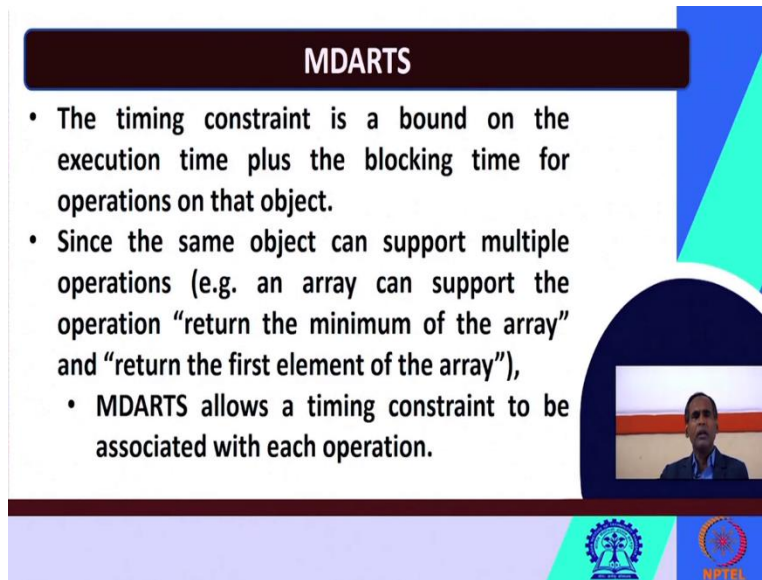
MDARTS

- **MDARTS** – Multiprocessor Database Architecture for Real-time Systems.
- Used for control applications such as controlling machine tools or robots.
- MDARTS is a library of C++ data management classes and is an O-O system.
- The RT constraints of the application are specified in the object declarations. These include 2 string parameters;
 - a unique object identifier, and a set of semantic and timing constraints.

MDARTS stands for multiprocessor database architecture for real time systems. This MDARTS it is used for control application systems such as controlling machining, controlling machine tools or robots. Normally, MDARTS is a library of a C plus plus data management classes. And it is an object oriented system. Since, we are discussing about real time systems so let us see how the real time constraints they are specified in MDARTS.

The real time constraints of the application are specified in the object declaration. So, these object declarations include 2 string parameters. One a unique object identifier and the other one is a set of semantic and a timing constraint.

(Refer Slide Time: 25:14)



MDARTS

- The timing constraint is a bound on the execution time plus the blocking time for operations on that object.
- Since the same object can support multiple operations (e.g. an array can support the operation “return the minimum of the array” and “return the first element of the array”),
 - MDARTS allows a timing constraint to be associated with each operation.

The slide features a dark blue header with the title 'MDARTS' 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 speaking. At the bottom, there are logos for IIT Bombay and NPTEL.

Now, let us see what thus timing constraints it is. The timing constraint is a bound on the execution time plus the blocking time for the operations on that object. Since, I have already told you these RT constraints of the application they are specified in the object declaration. These object declarations includes 2 string parameters. One is a unique object identifier and the other one is a set of semantic and timing constraints.

Since, this is a real time database we must know about little bit this timing constraint what is it. The timing constraint is a bound on the execution time the timing constraint is a bound on the execution time plus the blocking time for the operations on that object. Since, this same object it can support multiple operations. One object, an object it can support a multiple operations. For example, if you taking an array, an array it can support the operation what return. An array can support the operation return the minimum of the array also it can support the operation return the first element of the array.

So since, the same object can support the multiple operation this database MDARTS it allows a timing constraint to be associated with each operations. So, MDARTS it allows a timing constraints a timing constraint to be associated with each operation.

(Refer Slide Time: 26:36)

MDARTS

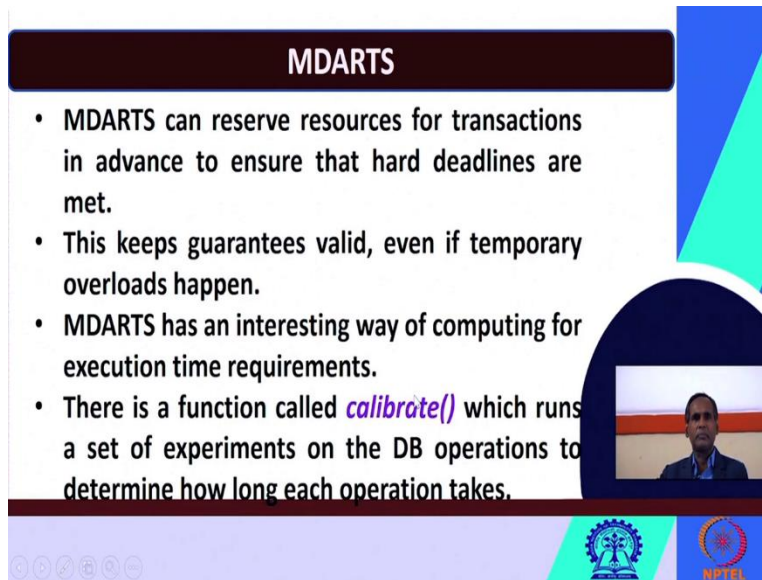
- MDARTS does not use the standard relational database model.
- Instead, each object provides context and identity for its data.
- Conventional DBs require that indexes be searched to locate data.
- Instead, MDARTS objects use direct memory pointers to their data.

The slide features a dark blue header with the title 'MDARTS' in white. The main content area is white with a dark blue border. A video inset in the bottom right shows a man speaking. The footer contains the IIT Bombay logo and the NPTEL logo.

Some important characteristics of MDARTS let us see MDARTS does not use the standard relational database model. It does not you already know what is a relational database model in your database model so this real time database system it does not this hard real time database system does not use the standard relational database model then what does it do, instead here each object it provides the context and the identity for its data.

You know that the conventional databases they require that the indexes we searched to locate the data. You have to use some indexes to locate the data. But, in MDARTS what is happening? Instead, the MDARTS objects they use direct memory pointer to their data. They use direct memory pointers to their data. They do not use this what index to locate the data. You know that the conventional databases requires that indexes be searched to locate data. But MDARTS objects they use the direct memory pointers to that data.

(Refer Slide Time: 27:38)



MDARTS

- MDARTS can reserve resources for transactions in advance to ensure that hard deadlines are met.
- This keeps guarantees valid, even if temporary overloads happen.
- MDARTS has an interesting way of computing for execution time requirements.
- There is a function called *calibrate()* which runs a set of experiments on the DB operations to determine how long each operation takes.

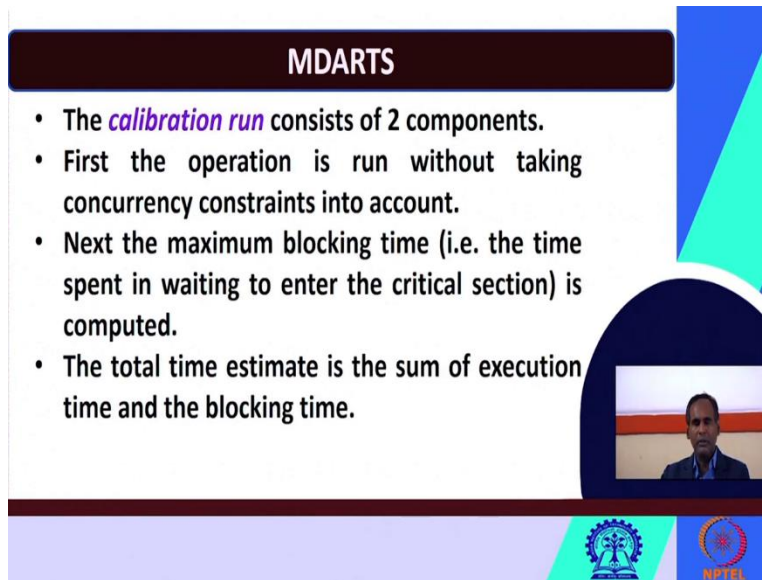
The slide features a dark blue header with the title 'MDARTS' in white. The main content area is white with a dark blue border. A small video inset on the right shows a man speaking. The bottom of the slide has a light blue footer with navigation icons and logos for IIT Bombay and NPTEL.

MDARTS can reserve resources for transactions please remember we have already seen resource reservation protocols. MDARTS can reserve resources for the transactions in advance to ensure that the hard deadlines are met. This will keep guarantees valid.

Since, MDARTS can reserve resources for transactions in advance to ensure that the hard deadlines are met this keeps guarantees valid even if temporary overloads happens. So, even if temporary overloads happens this keeps the guarantees valid. The MDARTS has an interesting way of computing for execution time requirements. Let us see how MDARTS computes the execution time requirements.

So, in MDARTS there is a function called calibrate. In MDARTS there is a function called calibrate which runs a set of experiments so this function calibrates it runs a set of experiments on the database operations to determine how long each operation can take.

(Refer Slide Time: 28:41)



MDARTS

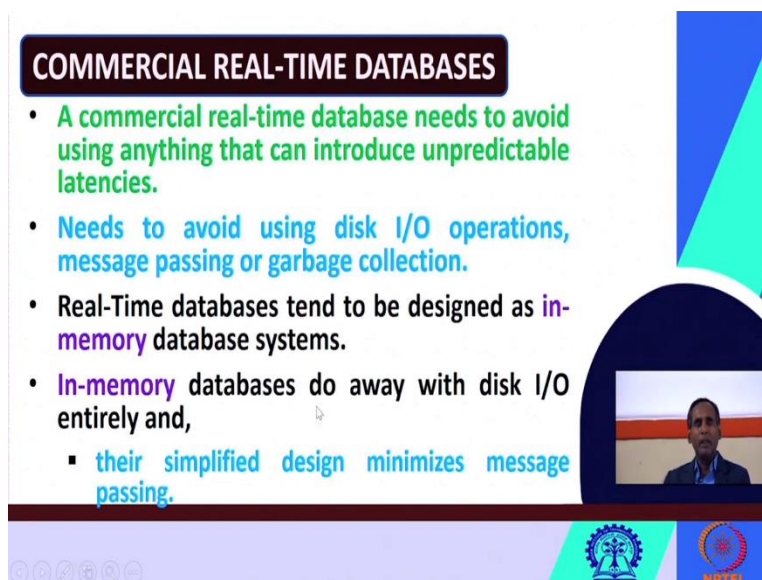
- The *calibration run* consists of 2 components.
- First the operation is run without taking concurrency constraints into account.
- Next the maximum blocking time (i.e. the time spent in waiting to enter the critical section) is computed.
- The total time estimate is the sum of execution time and the blocking time.

The slide features a dark blue header with the title 'MDARTS' in white. The main content area is white with a list of four bullet points. A video inset in the bottom right shows a man speaking. The footer contains logos for IIT Bombay and NPTEL.

In this way my MDARTS can compute the execution time requirement. So, finally we will see how this calibrate function it runs it executes. The calibration runs consists of 2 components. First the operation is run without taking the concurrency constraints into account. And then the maximum blocking time is computed.

What is meant by blocking time? The time which is spent in waiting to enter the critical section. So, now if you will find out the sum of the execution time and the blocking time that will be the total estimated time. The total time estimate is the sum of execution time and the blocking time.

(Refer Slide Time: 29:16)



COMMERCIAL REAL-TIME DATABASES

- A commercial real-time database needs to avoid using anything that can introduce unpredictable latencies.
- Needs to avoid using disk I/O operations, message passing or garbage collection.
- Real-Time databases tend to be designed as in-memory database systems.
- In-memory databases do away with disk I/O entirely and,
 - their simplified design minimizes message passing.

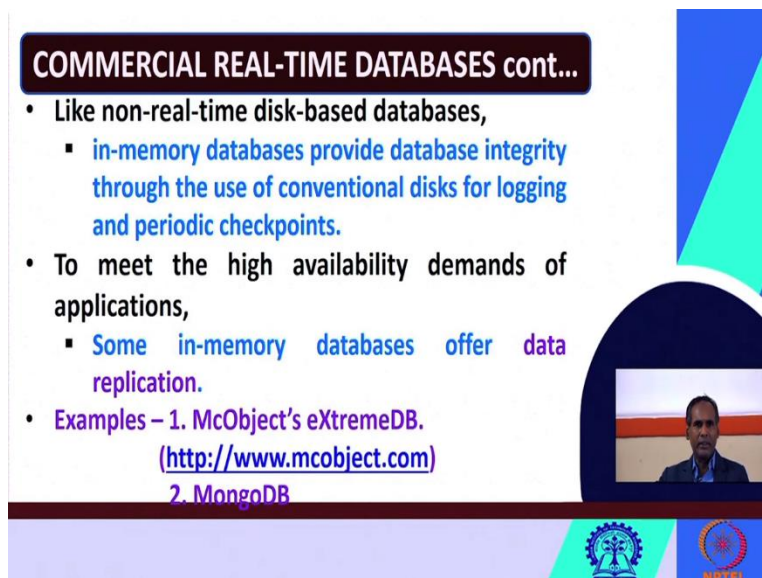
The slide features a dark blue header with the title 'COMMERCIAL REAL-TIME DATABASES' in white. The main content area is white with a list of four bullet points. A video inset in the bottom right shows a man speaking. The footer contains logos for IIT Bombay and NPTEL.

Let us quickly look at this few of the commercial real time databases. So, before going to some sample or example commercial real time databases let us quickly look at their features. A commercial real time database it need to avoid using anything which will introduce unpredictable latencies. For example, the commercial real time databases that need to avoid using disk I O operations.

In the last class I have already told you why Disk I O operations should be avoided why you should use in memories for storing the databases. So, the commercial real time databases they need to avoid using disk I O operation, message passing and garbage collection. I have already told you real time databases tend to be designed as in memory database systems. You should use the in memories to store the entire database.

You have known that in memory databases they do away with they completely remove the disk I O entirely. The in memory database they do away with the disk I O entirely and their simplified design minimizes message passing. So, since you are using in memory and they are having a very simplified design their simplified design minimizes the message passing. Message passing can be minimized if you are using what the in memory databases.

(Refer Slide Time: 30:29)



COMMERCIAL REAL-TIME DATABASES cont...

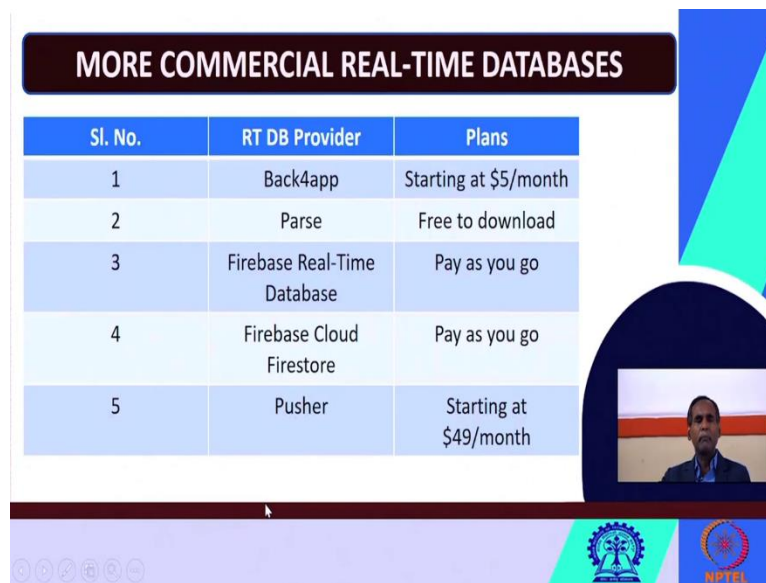
- Like non-real-time disk-based databases,
 - in-memory databases provide database integrity through the use of conventional disks for logging and periodic checkpoints.
- To meet the high availability demands of applications,
 - Some in-memory databases offer data replication.
- Examples – 1. McObject's eXtremeDB.
(<http://www.mcobject.com>)
2. MongoDB

The slide features a video inset of a man speaking, set against a background with blue and green geometric shapes. Logos for IIT Bombay and NPTEL are visible at the bottom.

Like non real time disk based databases, the in memory databases they provide database integrity by using the conventional disks for logging and periodic checkpoints. To meet the high availability demands of applications some in memory databases offer data replications. What I am saying here?

To meet the high availability demands of applications in order to meet the high availability demands of applications so some in memory databases they also offer data replication. Data can be replicated let us say some of the popular examples of commercial real time and databases one such example is McObject's eXtreme Database you can get it from this website. And you have might have known MangoDB. Nowadays MangoDB has also some features for which it can be used as a real time database.

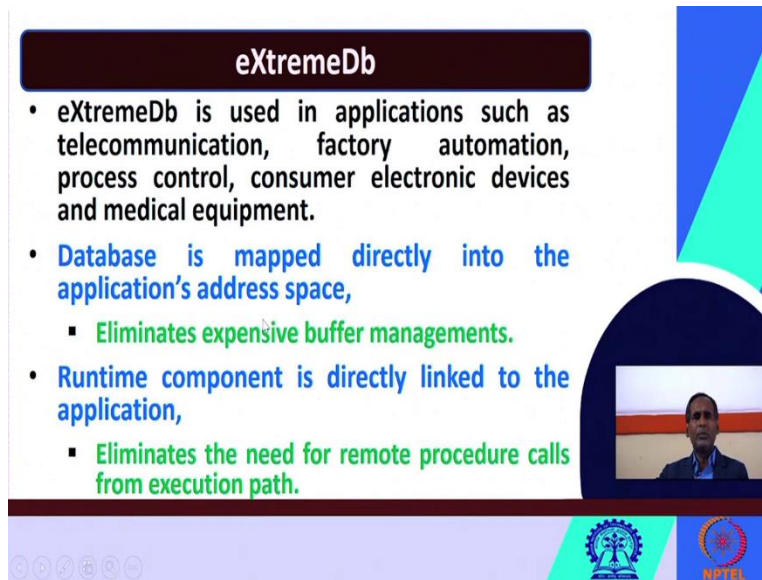
(Refer Slide Time: 31:19)



Sl. No.	RT DB Provider	Plans
1	Back4app	Starting at \$5/month
2	Parse	Free to download
3	Firebase Real-Time Database	Pay as you go
4	Firebase Cloud Firestore	Pay as you go
5	Pusher	Starting at \$49/month

These are some more commercial real time databases with the real time database provider and what is the plan like Back4app it is available by this payment term starting at Dollar 5 per month, Parse, Firebase Real Time database, Firebase Cloud Firestore, Pusher these are some of the common commercial real time databases and these are plan by which you can avail these real time databases.

(Refer Slide Time: 31:44)



eXtremeDb

- eXtremeDb is used in applications such as telecommunication, factory automation, process control, consumer electronic devices and medical equipment.
- Database is mapped directly into the application's address space,
 - Eliminates expensive buffer managements.
- Runtime component is directly linked to the application,
 - Eliminates the need for remote procedure calls from execution path.

The slide features a dark blue header with the title 'eXtremeDb' in white. The main content is a bulleted list with blue and green text. A small video inset on the right shows a man speaking. The bottom of the slide has a light blue footer with navigation icons and logos for IIT Bombay and NPTEL.

We will see only few minutes on this most popular real time database that is eXtreme database. Extreme DB, extreme DB normally it is used in applications such as telecommunication, factory communication, factory automation, process control, consumer electronic devices a medical equipment.

Here the database is mapped directly in the application's address space which eliminates the expensive buffer managements. So, since the database here it us mapped directly into the application's address space this eliminates the expensive buffer managements. The runtime is directly linked to the applications so in extreme DB the runtime component is directly linked to the application which eliminates the need for RPC which eliminates the need for remote procedure calls from execution path.

(Refer Slide Time: 32:30)

eXtremeDb cont ...

- **eXtremeDB uses its own memory manager for allocations and deallocations made by the database runtime.**
 - **No bottleneck of paging data in and out during I/O operations.**
- **eXtremeDB supports five priority levels that can be assigned to transactions.**

Extreme DB uses its own memory manager. It does not depend on the other what memory manager extreme DB uses its own memory manager for the allocations and the deallocations made by the database runtime. There is no bottleneck of the paging data in and out during I O operation. Since, extreme DB it uses its own memory manager there is no bottleneck of paging the data in and out during the IO operations.

Extreme DB supports five real time levels that can be assigned to transactions. So, in extreme DB it supports five priority levels which can be assigned to the transactions. This is little bit about the extreme DB you can more get from that website.

(Refer Slide Time: 33:08)

CONCLUSION

- Reviewed few important optimistic concurrency protocols like Forward OCC, OCC Broadcast Commit and OCC-Sacrifice.
- Discussed about Speculative Concurrency Control.
- Compared Concurrency Control Protocols.
- Explained MDARTS as a Databases for hard RT Systems.
- Discussed some commercial real-time databases.

IIT Bombay NIPTEI

Today we have discussed some of the important optimistic concurrency protocols forward OCC, OCC Broadcast Commit and OCC Sacrifice. So, today we have reviewed the few important optimistic concurrency protocols are like forward OCC, OCC Broadcast OCC BC and OCC Sacrifice. We have also discussed about this SCC protocols speculative concurrency control protocol. We have compared the available concurrency control protocols.

We have explained that the database for hard real time systems. What is the MDARTS? Multiprocessor database architecture for real time systems. We have explained MDARTS as a database for hard real time systems. We have discussed some commercial real time databases. As well as their payment terms for use.

(Refer Slide Time: 33:54)

REFERENCES

1. Rajib Mall, Real-Time Systems: Theory and Practice, 1st Edition, 2007, Pearson Education
2. C. M. Krishna & K. G. Shin, Real-Time Systems, 2017, Tata McGraw Hill Education

The slide includes a small video inset of a man in the bottom right corner and logos for IIT Bombay and NITEL at the bottom.

These are the reference we have taken. Thank you very much.