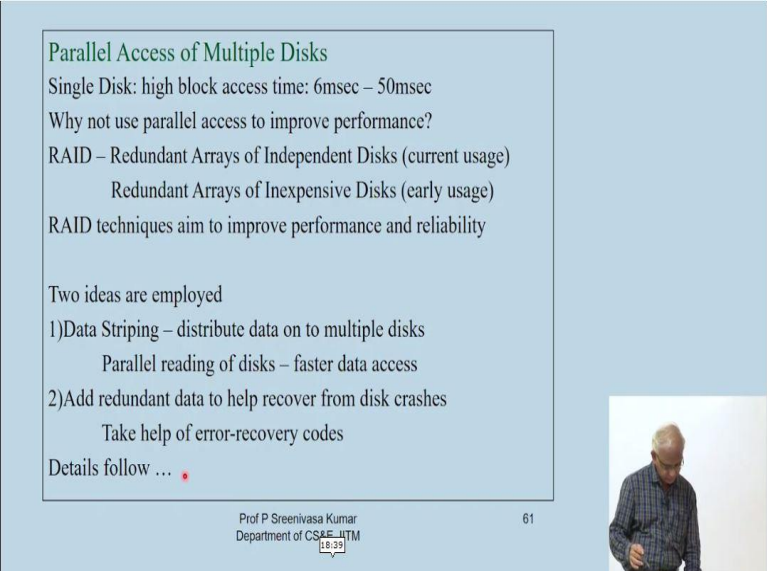


**Database Systems**  
**Prof. Dr. Sreenivasa Kumar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Madras**

**Lecture-32**  
**Performance and Reliability of Multiple Disks**

So, in this lecture, I basically want to discuss very few interesting ideas about how to handle failures of disks and also how to improve the efficiency performance of IO when we have the secondary storage.

**(Refer Slide Time: 00:47)**



**Parallel Access of Multiple Disks**

Single Disk: high block access time: 6msec – 50msec

Why not use parallel access to improve performance?

RAID – Redundant Arrays of Independent Disks (current usage)

Redundant Arrays of Inexpensive Disks (early usage)

RAID techniques aim to improve performance and reliability


Two ideas are employed

- 1) Data Striping – distribute data on to multiple disks  
Parallel reading of disks – faster data access
- 2) Add redundant data to help recover from disk crashes  
Take help of error-recovery codes

Details follow ...

Prof P Sreenivasa Kumar  
Department of CSE, IITM  
[18:39]

61



So, we have seen so far our discussions have not focused on this aspect, we use a single disk and then we also you know notice that with a single disk, sometimes the block access involves seek time, rotational delay, and all that. So it might take some something like 6 milliseconds to a few 10s of milliseconds sometimes so that that is a substantial amount of time. And so one of the first ideas that come up to us when we look at this situation is obviously, why do not we do some kind of parallel data access?

Why do not we put lots of disk together, and then why do not we access them simultaneously, because we can if the bus is speed bus is fast enough, and we may be able to, you know, read off and we can make a wide bus and then we can read off several disk simultaneously, and then gain

speed of access. So in this context, there is a very interesting term called RAID it is if it is called as Redundant Arrays of Independent Disks.

Earlier it used to be you know expanded a redundant array of inexpensive disks whether it is inexpensive or not, now, we were going to treat these disks as independent and then put an array you know, array of these disks and then see how we can actually benefit from this. So, we will this is a RAID is a umbrella term. So, it has a bunch of techniques a bunch of ideas using which we can improve performance as well as reliability of the storage. So, we will briefly discuss the main ideas behind all this RAID in the in this lecture.

So, essentially 2 main ideas are kind of employed in this situation what is called Data striping that means distribute the data on to the on to multiple disks. So, 1 disk data is actually is stored on to multiple disks, I will talk about exactly how we can do that. So, the idea is, of course, it is that if you read those disks parallelly then you will get access to your data, faster access to data. And then since disk can fail, add some redundant information, redundant data, so that we can recover from the this crashes, this is the second idea. So, in fact, we will be taking help from error recovery course in this context.

**(Refer Slide Time: 04:05)**

### Data Striping

Data Striping – distribute data on multiple disks

Bit-level striping:  $i^{\text{th}}$  bit of each byte – stored on the  $i^{\text{th}}$  disk

Use 8 disks for 8 bits of a byte. // higher granularity is also possible

One (parallel) block read – 8 blocks of the data file


Transfer rate – eight times that of single disk

Read/write of a block – involves use of all the disks

Block-level striping:  $i^{\text{th}}$  block of data –  $i^{\text{th}}$  disk

Using  $n$  disks -

- Single block access:  $n$  simultaneous block reads can happen
- Multi-block access:  $n$  fold increase in transfer rate (parallel reads)



Prof P Sreenivasa Kumar  
Department of CS&E, IITM

62

So, let us see how the details of so let us first consider this idea of called Data striping. Basically, we want to distribute data on to multiple disks. So you could do one thing what is called bit levels striping, which is a very, you know, lowest granularity you can think about. So take the  $i^{\text{th}}$

bit of each byte and store it in the  $i$ th disk. Bit, we are talking about a bit of data. If  $i$ th bit of each disk block has 1496 assemble like that, you know bytes, and each byte has 8 bit so we are talking about suppose there are 8 byte, we have 8 bits per byte.

And so let us say we employ 8 disks parallely we employed 8 disks parallely. So for each byte, we stored the 0th bit in the 0th disk first bit in this in the first disk and so on. So we distribute all this bits across all the 8 disks. Now, of course, you can transfer data from the disk, at only block by block. So suppose you now issue a parallel block read for all these 8 disks. So you will end up having 8 blocks of read. Actually, you can see that 8 fold increase in the data speed.

You get the point. See, if you take a block, take a block of data and then you know, take only the 1 bit of it 1 bit goes to 1 disk 1 bit goes to 1 disk so you are operating 8 so 8 blocks 8 blocks will be covered in the first blocks of all these disks. So if you read a give a disk read, which is a parallel display for each of these things are independent, you will end up getting 8 blocks of data, 8 blocks of it.

So of course, you have to assemble all the bytes from each of them, you will get 1 bit from each disk and then you keep on assembling all the bytes. So you will get 8 fold increase in these in the speed of byte one of course, downside of this whole thing is that if you have to read or write a block, then all the 8 disks are involved, because the data is spread across all industries? So it involves users of all that is, as I guess this, there is another idea called block level striping.

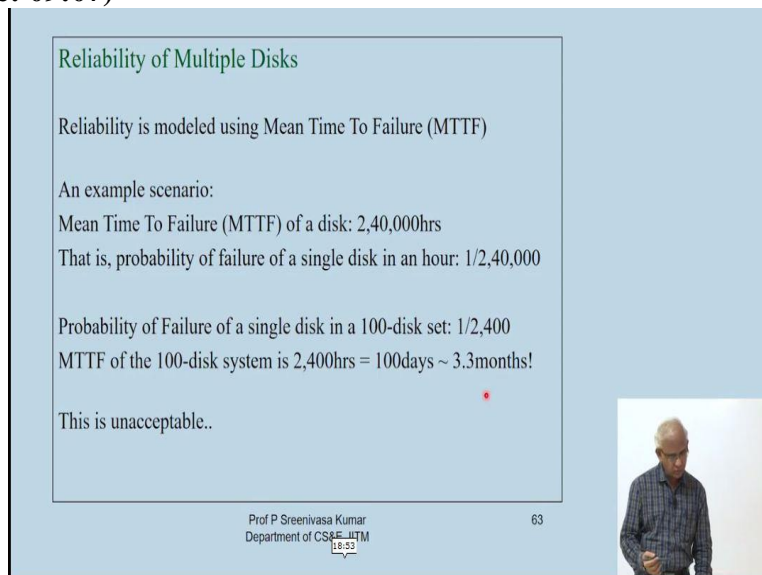
Instead of doing a very minute level, bit levels from striping, of course even higher, slightly higher granularity is also possible. You can decide, I will take every 4 bits of each byte and then we break that as a unit and then swipe across a block level striping, what it does is to do this the 8th block of the data is on the 8th disk  $i$ th disk  $i$ th block on the data you have  $i$ th disk. So you have a bunch of disks.

So, let us say you have some  $n$  number of disks, then there are 2 things, 2 advantages we get out of this block level striping. First thing is you can actually you know, handle  $n$  simultaneous block requests, because the data you know, if it is like it is likely to be there on different disks, and so

you can actually probably be able to handle some simultaneous number of displays because all of them are independent. That is one thing then supposing you got a big data request, which is a multi block kind of request, then you can, do a parallel read.

And then you can get enfold increasingly in the transfer rate. Just like in big level striping. You are also here also you will get  $n$  for increasing transport. Now, these are trying to use multiple disks and then do parallel reads and then we it is not very difficult to see that we will get higher speeds of reading. But what if the one of these disks failed, then we have lost the whole data. Because somewhere, some block of the file is on some disk and that disk failed. And so you know, you cannot get the complete file so the reliability of this set of disks comes down.

**(Refer Slide Time: 09:07)**



**Reliability of Multiple Disks**

Reliability is modeled using Mean Time To Failure (MTTF)

An example scenario:

Mean Time To Failure (MTTF) of a disk: 2,40,000hrs

That is, probability of failure of a single disk in an hour:  $1/2,40,000$

Probability of Failure of a single disk in a 100-disk set:  $1/2,400$

MTTF of the 100-disk system is 2,400hrs = 100days ~ 3.3months!

This is unacceptable..

Prof P Sreenivasa Kumar  
Department of CS&E, IITM  
18.53

63

*(Inset video of a man speaking)*

We will see how do we model reliability. The reliability is actually there are more details to it. But let us, say we will just take 1 parameter called mean time to failure. So this is, so if you have several days, you know how often they fail. That is the kind of thing each individual desk will actually have a probability of failure, which is a little high in when you start using the disk. And then actually, usually the run for a long period like 10 years or something like that.

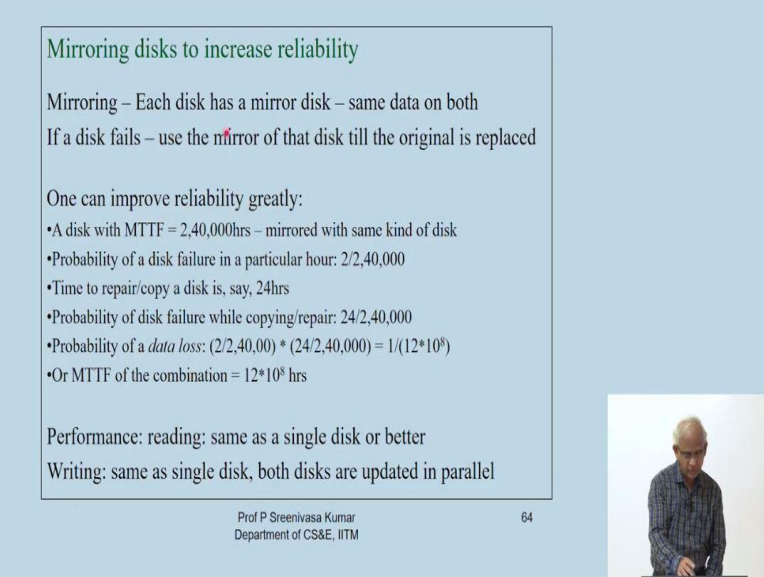
But towards the end, then the probability of the failure actually increases again, because of the wear and tear and also the tiny dust particles coming into the basement. But normally, you can take this you know, if the disk has been after manufacturing it has been tested so that there are no initial errors and things like that it has entered the stable kind of region, then you can imagine

that it has a good time like I have just taken some figures like this, these are typical figures 200,000 hours.

So, one way to imagine one way to convert use this number is to think of the probability of a failure of a disk single disk in an hour is something like one over 240000 now, if this particular disk is part of a 100 disk set, then each of these disks are independent and they can fail independently. So, the probability of a failure of any one of the disk of the 100 disks is 100 times more than this is plus.

So that is about. So if you convert that into mean time to failure, then it is about like 3 months. That is a range like you do not expect a failure to keep coming like 3 months so that is really bad. So, the more number of disks you use, the higher is the possibility of you knows, less failures. So, if you lose if you decide to use more number of disks for parallel access in order to improve performance, we should be worried about how to improve the reliability simultaneously. So let us see now how to improve the reliability of disk.

**(Refer Slide Time: 12:00)**



**Mirroring disks to increase reliability**

Mirroring – Each disk has a mirror disk – same data on both  
If a disk fails – use the mirror of that disk till the original is replaced


One can improve reliability greatly:

- A disk with MTTF = 2,40,000hrs – mirrored with same kind of disk
- Probability of a disk failure in a particular hour:  $2/2,40,000$
- Time to repair/copy a disk is, say, 24hrs
- Probability of disk failure while copying/repair:  $24/2,40,000$
- Probability of a data loss:  $(2/2,40,000) * (24/2,40,000) = 1/(12*10^8)$
- Or MTTF of the combination =  $12*10^8$  hrs

Performance: reading: same as a single disk or better  
Writing: same as single disk, both disks are updated in parallel

Prof P Sreenivasa Kumar  
Department of CS&E, IITM

64



One very simple idea is mirror the disk. For every data disk, you also maintain a copy of that. So every disk the data of the entire disk is copied on to another disk so each disk as a mirrored disk same data on both of them. So if a disk fails, use the mirror of the disk is the original disk, replaced that is the policy. Now, you can see that even though there are, you know, many numbers of disks with each of the disk is actually mirrored like this.

Then the even though the probability of failure of the disks is, high, the probability of data loss actually will decrease. I will show you how exactly it forms. Let us look at this figures. Let us say we have a disk with no mean time to failure up 240,000 hours. And it is mirrored with the same kind of this. So the probability of a disk failure in a particular hour is 2 hour disk why is 2? Because there are 2 disks the data disk and its mirror. So it probability of the single disk failure is 2 hour.

Let us say we have a time to repair or you know, copying procuring a new disk and then making a copy of the whole disk on to that thing as something like 24 hours. Now, you can see that the actual data loss will occur to us when, you know, while we are still not ready with this disk copy or while we are copying If the disk fails again so, that is the if the same disk fails or on which we now have a suppose there is a failure we have been using the mirror. So, the probability of the data loss is the probability of 1 disk failing and the mirror disk failing.

So, the what is the probability of the mirror disk failing when we are copying is that it is you can think of it as the 24 times 24 over this particular number. Because it is probability in within 24 hours per hour is like that and so, it is effective. So, the probability of data loss is actually the sequence of events happening like this, one of the disk fails and then the within the 24 hours, the same dissipates. So, that is something like this. So, you can put and then calculate it will give you this number.

So, if you cannot that into mean time to failure is very high. So, one can actually improve the reliability of this entire system and then the probability of data loss will be very less if you use mirroring. Mirroring is a very nice technique where we can improve the reliability greatly. But then the problem with mirroring is when there are 10 disks you are also again having 10 other redundant disks when the data disks are there, then redundant disks are equal number of redundant disk.

Also, you are kind of using 50% as redundancy. That is a very high amount of redundancy. If you have a mirror disk reading is same as you know reading a single disk or it can be even

slightly better. You might actually read from any of the disks, the copies are there. So you may be actually be able to handle 2 simultaneous block accesses at a time because they are 2 copies of writing can be done parallelly. So in both disk you can write so it is not very worse.

In fact, in practice, you have to probably pause a little before you know, write onto the mirror copy. So it will be a little less performance was, but the main issue here is the amount of redundant disk that we are using.

**(Refer Slide Time: 17:04)**

**Reliability and performance with parity disks**

Mirroring – High reliability; uses 50% more disks!  
Get good reliability & also performance with fewer additional disks?  
Idea: Store additional information to recover data of the failed disk

Error-correcting codes – parity bit (1 if #of 1's is odd, 0 otherwise)  
Data: 1 0 1 1 0 0 1 0 - Parity Bit: 0 ( #of 1's in Data & Parity is even )  
Data: 1 0 0 1 1 0 1 1 - Parity Bit: 1 ( #of 1's in Data & Parity is even )

Parity block: (Assuming block-level data striping with N disks)  
The  $i^{\text{th}}$  bit of the parity block  $j$ : parity of the  $i^{\text{th}}$  bits of block  $j$  on all disks

Parity Disk – has parity blocks for all data blocks

If a disk  $k$  fails: Set the  $i^{\text{th}}$  bit of block  $j$  using  $i^{\text{th}}$  parity bit of block  $j$   
Do this for all blocks to recover data of disk  $k$ !

N – data disks, one extra disk – good performance and reliability!

Prof P Sreenivasa Kumar  
Department of CS&E, IITM

65

So, it gives you higher reliability, but then 50% redundancy. Now, what we will do here in this context, we use the good old parity function from listed from digital logic. And then no surprisingly we will get good performance. So, we will see this; what is parity you. So, this is idea, idea is store additional information to record data of the failures. So, for this additional information all that we will be doing actually is use parity the good old parity function.

So, let us look at parity function parity bit is defined as 1 if the number of once in is on 0 otherwise in a sequence of bits, a sequence of bits. If the number of ones is odd, then the parity bit is 1. If the number of ones is even, then the parity bit is 0. Now so here is the some sequence of bits, you can see that it is just 4 ones and so the parity is 0. So if you now actually take the original bits and the parity bit together, and then look at the number of ones in that, you can see that it will be always even.

That is how it is defined a parity. So here is another example. So you can see 1 1, so there are 5 ones and so the parity bit is 1 and so the over all number ones will always be that is how the parity function works. So, basically we can now make use of this and then you know create what is called a parity block for a block for a bunch of blocks. Let us assume the block level data striping has occurred. That means blocks are stored across some 5 disks or something like that anymore below 5 bunches like that? No.

So some blocks are existing on this 5 disks, now take the 6th disk on each of these first blocks of all these 5 disk construct at bit level parity at the bit level construct a parity consider the  $i$ th bit  $i$ th bits of all the blocks on all these disks compute its parity and make that as the  $i$ th bit of the parity block. So, you consider say block 0 of all the disks in the blocks 0 of all the disks take the  $i$ th bit and compute the parity take the parity bit and make that as the  $i$ th bit of the parity disk parity block.

So, you get the idea of how to construct parity blocks basically 1 blocks are there. So, you just focus on the corresponding base on all the blocks, then take the parity and then construct the big for the parity block the sequence of waves across all the blocks take the parity and then keep that as the corresponding becomes a priority block. Now, the idea is if any of these blocks is inaccessible because the disk failure the parity will tell us whether the missing block bit is either 0 or 1 because we can recomputed that the parity.

So, that is the idea parity blocks so, compute parity blocks like this and then put all the parity blocks on is redundant disk called the parity disk. So, you have a parity disk data disks parity disks the parity disk is the redundant disk actually. So, it has some redundant information, but if any of the data disks fails one failure we can handle if 1 disk crashes, then we can actually reconstruct the disk a fails said the  $i$ th bit of the block  $j$  using the  $i$ th parity bit of the block bit and repeat this for all the blocks.

It has 100 blocks let us say for all the 100 blocks we have parity bits, parity blocks are there. Each parity block will have information about bits in the block, so, set the  $i$ th bit of the block  $j$  using the  $i$ th parity bit of the block  $j$ . So, you will be able to reconstruct 1 block 1 disk from all



the other disk information about the other disk. So this is a very great idea beautiful idea. So, with even though you have  $n$  data disks with 1 extra disk, we get very good performance and reliability.

The more important thing is reliability the disk fails; we are able to recover the data using the parity. And performance of course comes because we have done the data striping. Performance comes because we are done the data striking. We will put the blocks on several disks, so we can. So if you have a large request, say, get me 50 consecutive blocks from the file. Then I can get blocks from all the 5 disks simultaneously. And so I will get a 5 fold increase in my performance. Because you have to work I am glossing over several details.

Exactly which block goes to what disk and all that you have to keep track of. And then when appropriate. I am trying to give you some high level idea of how this whole thing works. Is this idea clear? Or trying to construct this the important thing there is a way of calling it something like a parity block a parity block is bit level parities for all the bits of a bunch of blocks. That is what a parity block is a bunch of corresponding blocks, 0th block on disk 1 and 0th block on disk 2 0th block on disk 5 consider them. For them you constructed 0 parity block.

The parity block will basically have parity bits for each of the bits of that set of blocks. So if any of those blocks becomes inaccessible, because it distracts, we can reconstruct that block. And if we can reconstruct one block we can reconstruct any number of blocks using the same therapy. Now consider writing a block. Consider writing a block, such a basic system. If you write a block, depending on the, striping policy, it will go to some disk and go to some disks.

And then you have to since you have updating the block on the disk, the parity information will change. So you have to update the parity also if you each of these blocks that you are writing is some  $k$ th block on some disk. So for all the  $k$ th block on all the disks, there is a parity block. So if you write the you we update the  $k$ th block then you have to update the parity block. Without updating the parity block, then it will be the parity information will not be correct if you do not update the parity. So each the read is simple.

You do not, nothing will change. Nothing will change if you are writing or updating a block. Obviously, you are changing the bits in the block and so the parity information changes. So you have to update the parity. Now, all the parity blocks are on the parity disk. So anybody changes anything in the blocks in anywhere in the data blocks, they have to go change in the data in the parity block.

So you can now see that the average number of rights on the parity block will be some factor of times the average number of rights on the data blocks on the data disks. There is only one parity this. So, anywhere you write in any of these 10 disks, you have to go and write on the parity disk. So the average number of rights on the parity disk will actually be very high. And if you are you know, using a disk a lot than other days, then obviously the aging will occur. And so, that might fail and then that is what we should not allow it to fail because I saw redundant information. That is the priority disk.

**(Refer Slide Time: 28:52)**

### Distributed Parity

N data disks and 1 redundant (parity) disk

- Very good performance and protection against single-disk crash
- Updating *any* data block – requires updating the parity disk
- Usage of parity disk – high and it ages faster!

Can we distribute the parity information?

Use each disk as a redundant (parity) disk for some *part* of the data!

Say, we have  $D_0, D_1, D_2, \dots, D_5$  – 6 disks with, say, 60 cylinders each

Use each as the redundant disk for 1/6 of data:

Cyl# 0, 6, 12, ... of  $D_0$  – parity blocks for other disk cyl# 0, 6, 12, ...


Cyl# 1, 7, 13, ... of  $D_1$  – parity blocks for other disk cyl# 1, 7, 13, ...

Etc...

This is called *distributed parity* – disk usage is uniform!

Prof P Sreenivasa Kumar  
Department of CS&E, IITM

66



So, here comes the interesting idea. It comes in another interesting idea called distributed parity, why should we put all the parity blocks in 1 disk, if we put all the parity blocks in 1 disk, even though we get very good performance and protection against a single disk crash the parity disk usage increases and it in some sense ages faster. And so, we consider an idea why do not you distribute the parity? Why should we keep it in 1 disk? So each use each of these disks as a redundancy disk for some part of the data.

Use each of this disk as a redundant or parity disk for some part of the data. So that is the idea behind or is called distributed parity. So let me see, let me give you some more details here. Say let us say we have disk D 0 D 1 D 2 D 5 some 6 identical disks are there. Let us say for just for discussion purpose, let us say we have 60 cylinders in each of those disks. 60 cylinders on each of those 60 is actually ridiculously small number of 1000s of cylinders.

No, use each of these disks as a redundant disk for basically  $1/6$  of the data. How do organize this  $1/6$ . One can actually work on so cylinder 0 cylinder 6 cylinder 12 of the disk 0 we will have parity blocks for the same cylinders on the other disks. So, the other disk zeros if this 0 is going to be parity then there are these D 1 D 2 D 5 the other 5 disks are there. So, for the other 5 disks you take these cylinders cylinder 0, 6 and 12 and so on.

For those cylinders each of the cylinders will have lots of blocks the cylinders will have for some blocks compute parity for all of those blocks. So, instead of computing parity for all the blocks now we are only considering these cylinders only for these cylinders alone you could put the parity blocks on the disk 0. So, roughly cylinder the disk 0 will have parity information for  $1/6$  of the cylinders for 10 cylinders if you have.

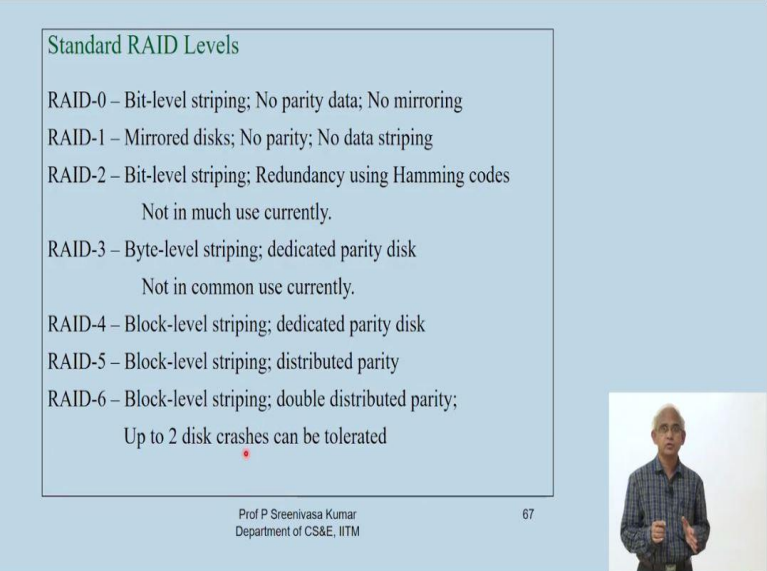
In a similar way so, you basically need see, so, I have not put an etc here you can imagine, so, 0 1 2 3 up to 5 and then 6 7 8. So, all the cylinders are being covered here of all the disks, D 0 D 1 you can also they are being covered here also. So, you can see that, so now, cylinders 1 7 13 of D 1 will have parity blocks for the cylinders 1 7 13 of all the other disks. So that way what we have done is to kind of distribute the parity information across all the disk so keeping it in only 1 parity disk now we are distributing here.

So, as long as you keep track of for which of this cylinders, where are the parity blocks as long as you keep track of that information then your work if any disk fails, if a disk fails a D 1 fails then you know so it has lots of cylinders, so, first cylinders 0 where are the parity blocks you know listen for cylinder something else 6 etc. around there are the parity blocks you can find out. So, like that you will be able to reconstruct that particular disk again, a single disk value can be added so I am not giving you know a formula, you know for this.

But I think you can work it out which cylinder will contain parity for what cylinders. So, the disk usage will be uniform because all the parity information is not held up in 1 particular disk, but it is kind of distributed and so, you are protected against the wear and tear of that parity disk so, these are various, you know bunch of ideas that are made use of in constructing, what is called this RAID levels. RAID is Redundant Array of Independent Disks, techniques.

So there are what are in industry. They are called there are levels that they talk about RAID 0 RAID 1 RAID 2 RAID 3 RAID 4 like that they talk about RAID levels. So, the higher the level, the better is the performance as well as the ability.

**(Refer Slide Time: 35:37)**



**Standard RAID Levels**

- RAID-0 – Bit-level striping; No parity data; No mirroring
- RAID-1 – Mirrored disks; No parity; No data striping
- RAID-2 – Bit-level striping; Redundancy using Hamming codes  
Not in much use currently.
- RAID-3 – Byte-level striping; dedicated parity disk  
Not in common use currently.
- RAID-4 – Block-level striping; dedicated parity disk
- RAID-5 – Block-level striping; distributed parity
- RAID-6 – Block-level striping; double distributed parity;  
Up to 2 disk crashes can be tolerated

Prof P Sreenivasa Kumar  
Department of CS&E, IITM

67

So I will briefly tell you what are the RAID levels standard RAID levels RAID 0 is used as bit level striping. No parity data, no mirroring, so this is actually nothing. It does not have any. But it just does because striping it gives you very good performance. But it is risky stuff. RAID 1 uses mirror disks no parity no data striping it just uses mirrors. So, if you can afford to have lots of storage then RAID 1 gives good performance and good reliability also.

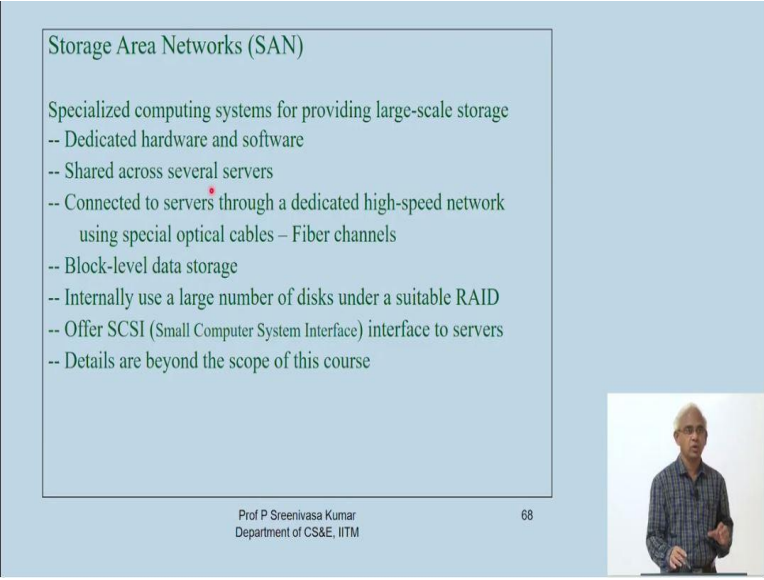
RAID 2 uses bit level striping and redundancy using you know instead of parity they use Hamming codes and this is actually not much in use currently. Then RAID 3 uses byte level striping it uses 5 bytes. So this level it uses byte level striping and there is a

dedicated parity disk dedicated parity that means single parity disk is also not in common use now. RAID 4 5 and 6 use block level striping 4 5 6 gives you block level striping.

So RAID 4 is block level striping with dedicated parity disk RAID 5 is block level striping with distributed parity so, this is a good solution. RAID 6 is much better than RAID 5 what it does is again uses block level striping, double distributed parity. So, it can actually they use what are called read Solomon calls for this thing. So, it can in fact take care of 2 disk crashes almost simultaneously disk crashes 2 disk crashes it can it can tolerate 2 disk crashes. So, RAID 6 is what is the highest one can go with these ideas.

So I am so, I want to give you an idea of this is something which is you know very commonly used in industry and so, the moment you go to any enterprise you start you know have to deal with this architectures have some idea about what RAID are this one.

**(Refer Slide Time: 38:48)**



**Storage Area Networks (SAN)**

- Specialized computing systems for providing large-scale storage
- Dedicated hardware and software
- Shared across several servers
- Connected to servers through a dedicated high-speed network using special optical cables – Fiber channels
- Block-level data storage
- Internally use a large number of disks under a suitable RAID
- Offer SCSI (Small Computer System Interface) interface to servers
- Details are beyond the scope of this course

Prof P Sreenivasa Kumar  
Department of CS&E, IITM

68

The slide features a light blue background. On the right side, there is a small rectangular video inset showing a man with glasses and a beard, wearing a blue and white checkered shirt, speaking. The slide content is enclosed in a thin black border.

So this is one thing, but nowadays actually storage is such a big affair that we have dedicated companies that offer storage. So, you would heard about storage area networks, their concerns. So these are specialized computing systems providing large scale storage with their own dedicated hardware software, everything you know, and it is some is shared across several servers.

And it is connected through a dedicated high speed network to the server using special cables. So these are called fiber channels so to give high speed, access to the disk, so these are block level data stores. Of course, internally, they use a large number of disks an appropriate radar credential. And they offer what is called as SCSI interface to the servers. So I am not going into the details of science. Because this is a really it is a there is a lot of details in that.

So, there are protocols again, because it is networks. So, obviously protocols will come into picture. So, but anyway, again, the abstraction is again block level abstraction block level access all that is converted by disk before SAN's came into picture even you know some kind of automated systems or network attached storage has worked, you know, the storage system lying on the same local area network as a server.

But then the local area network bandwidth was you know not sufficient for handling the data transfers. So, now you go for a dedicated network for storage purpose that is what storage area so these let we close. Next module I will be briefly discussing the various you know algorithms for implementing whereas you know algebra operators ultimately all the files you stored relations data.

But then when (( ))(41:41) submitted to the system it has to be organized as a algebra expression and then let us the issue are compare we have to each other's operators will have you will have options of has to how the these will be exhibited and so that the pretty complex scenario of so you will I will give the big intro that subject in the next syllabus. Thank you.