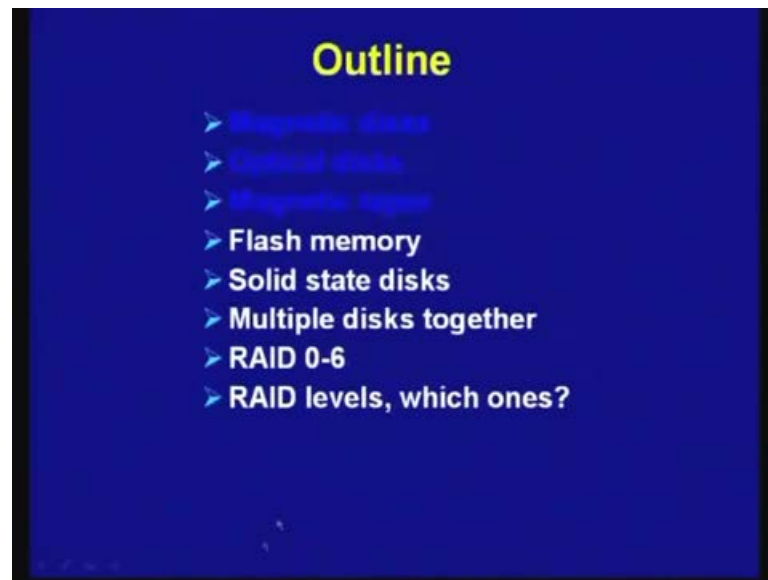


High Performance Computer Architecture
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
Storage Technology (Contd.)

Hello and welcome to today's lecture on Storage Technology, this is the second lecture on this topic. And in the last lecture we have discussed about magnetic disks, optical disks and magnetic tapes, which are used as storage devices.

(Refer Slide Time: 01:03)



And today I shall focus on flash memory, which is one of the storage technology that is used, and that is flash technology has led to solid state disks, I shall discuss about those solid state disks. And then to improve the liability and availability, we shall discuss how multiple disks can be used together, and leading to what is known as 0 to 6 RAID levels redundant array of independent disks. That is the disk acronym that has been used to this technique, I mean where multiple disk are used together, and there are 6 different levels 0 to 6, I shall discuss about them, and then I shall conclude my lecture by considering, which ones to be used in what situation.

(Refer Slide Time: 02:07)

Flash Memory

- A form of EEPROM:
 - However, allows a block to be erased or written in a single operation (in a flash).
- Floating Gate Avalanche-injection Metal Oxide Semiconductor (FAMOS)
- Electrons are trapped in a floating gate.
- Writing a byte requires creating a new block:
 - Old block is copied along with the byte to be written.

I have already discussed about flash memory in the context of a main memory, we have seen that flash memory is a form of electrically erasable programmable ROM, and a only difference with electrically erasable and programmable ROM is that. If this flash memory allows a block to be erased or written in a single operation, and as you know it is based on floating gate, avalanche injection metal oxide semiconductor technology and we have seen how electrons are trapped in a floating gate.

(Refer Slide Time: 02:43)

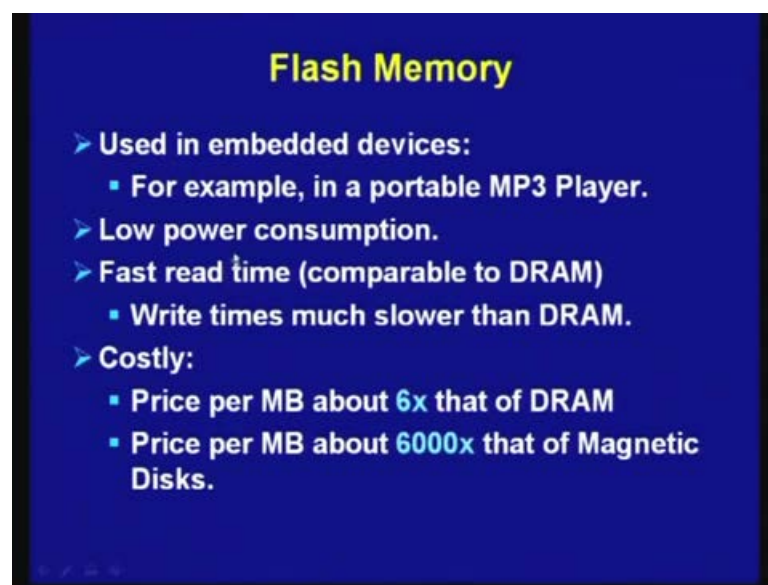
Flash Memory

- Performance:
 - Reads at speed of DRAM (~ns)
 - Writes like DISK (~ms). Write is a complex operation.

Avalanche injection. Removing programming voltage leaves charge trapped. Programming results in higher V_T .

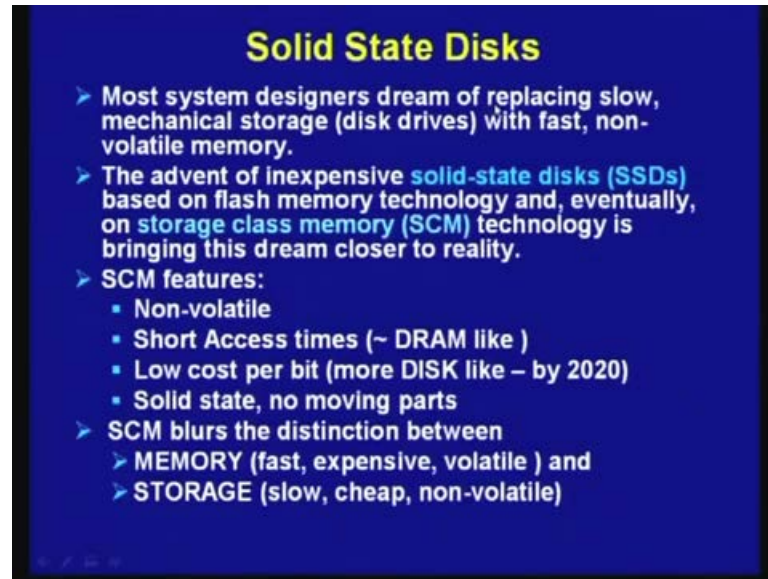
As, it is shown in this particular diagram, how the electrons are trapped and as the electrons get trapped in a floating gate, the threshold voltage is reduced, and that is how reading and writing can be done. And particularly in for flash memory reads can be done at the speed of dynamic RAM, that is at the which is in the range of nanoseconds on the other hand writing is rather slow, which is done I mean at the speed of milliseconds. So, write is a complex operation, because you have to I mean allow the accumulation of electrons in the floating gate, and that that is why the writing is complex process and takes longer time.

(Refer Slide Time: 03:31)



And flash memory has become very popular in recent years, particularly in embedded systems in portable M P 3 player, because of their low power consumption fast read. And as I mentioned write times is much slower than DRAM, but in the embedded applications like M P 3 player, where it is primarily used for reading. It is rarely used for writing most of the time you are reading, and it is costlier than dynamic RAM about 6 times costlier than dynamic RAM, and price per mega byte is about 6000 times that of magnetic disks; however, price is falling.

(Refer Slide Time: 04:11)



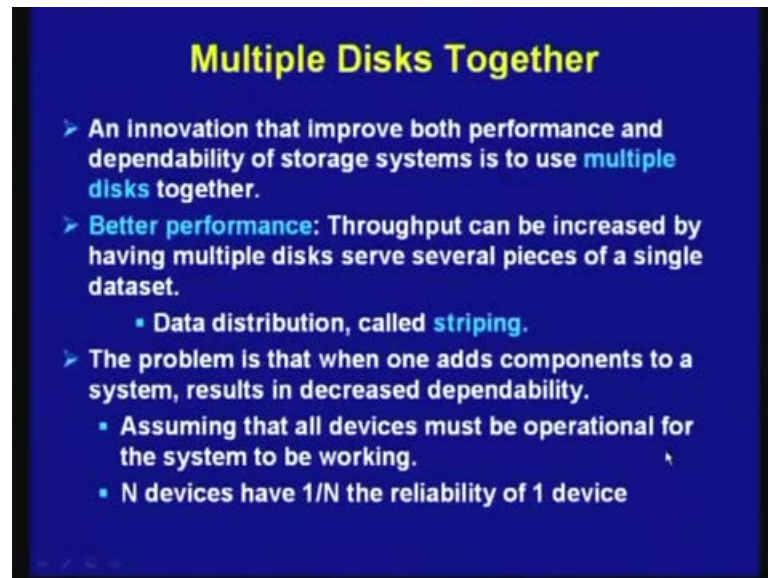
And that has led to the development of solid state disks, so most systems designers as you know dream of replacing slow mechanical storage. As, you have seen the magnetic storage is based on you know driving a magnetic, I mean plate disk by motor and as a consequence it is very slow, and it is not very reliable. So, this slow and mechanical storage to be replaced with fast and non volatile memory, that is the dream of most of the system designers, and particularly the advent of inexpensive solid state disks, based on flash memory technology.

And eventually on storage class memory, which has evolved based on this flash memory is beginning is bringing this dream closer to reality. So, it has become that the dream is coming true, and particularly these storage class memory based on this solid state disks or flash memory technology has the important properties. Like it is non volatile short access time, low cost per bit, and solid state, because there is no and it has got no moving parts, because it is based on solid state technology.

And we have already seen the discussed the use of flash memory in the context of main memory, so main memory requirement is fast expensive and volatile, you know main memory is much more expensive. So, it satisfies more or less both the requirement, that is the reason why this solid state disks blurs the distinction between memory and storage, the requirement for storage is slow, cheap and non volatile. So, in case of this storage

class memory that distinction between memory and storage has reduced, in other words it can be used for both the purposes with this background.

(Refer Slide Time: 06:23)



Now, let us focus on how we can improve the reliability and availability by using multiple disks together, so the innovation that improve both performance and dependability of storage systems is to use multiple disks together. So, this is not a very new concept, as we know whenever we try to improve the availability we use multiple devices, if one falls other one will serve the purpose. Similarly, for the purpose of reliability we can use multiple disk, where you know we can have some redundancy, which will allow you to have higher reliability.

So, that is a basic concept behind use of multiple disks together, so it gives you not only reliability and dependability, it gives you better performance, because throughput can be increased by having multiple disks are several pieces of a single dataset. That means, since we are using multiple disks, the through put can be increased by parallel access of the disks, and serving and then that a single dataset can be accessed from multiple disks.

And that is how the throughput can be increased, and we shall see that distribution and it is data is done by technique known as stripping, actually different blocks of the same data is placed in different disks. And that approach is known as stripping in the context of this, thus multiple disks memory system, so the problem is that when one and its components to a system results in decreased dependability.

As, you know whenever the number of components increases, since instead of one component; if you have n component; obviously, possibility of failure of the entire thing I mean one of the components increases. So, reliability I way in that way decreases however, this is based on the assumption that all devices must be operation for this system to be working. Only, then you know this reliability or dependability decreases, whenever we have n devices, and have and the liability becomes $1/n$ of the reliability of device; however, we do not use the multiple disk in this manner.

(Refer Slide Time: 08:59)



Multiple Disks Together

- **Better availability:** If a disk fails, then while it is being repaired data can be obtained from another disk:
 - Data is fully replicated on other disks.
 - Data can be reconstructed from data on other disks.
- **One issue:** what if another disk fails while a disk is being repaired?
 - **MTTF** of a disk in tens of years
 - **MTTR** of a disk can be as low as hours
 - Therefore, statistically there is no problem.

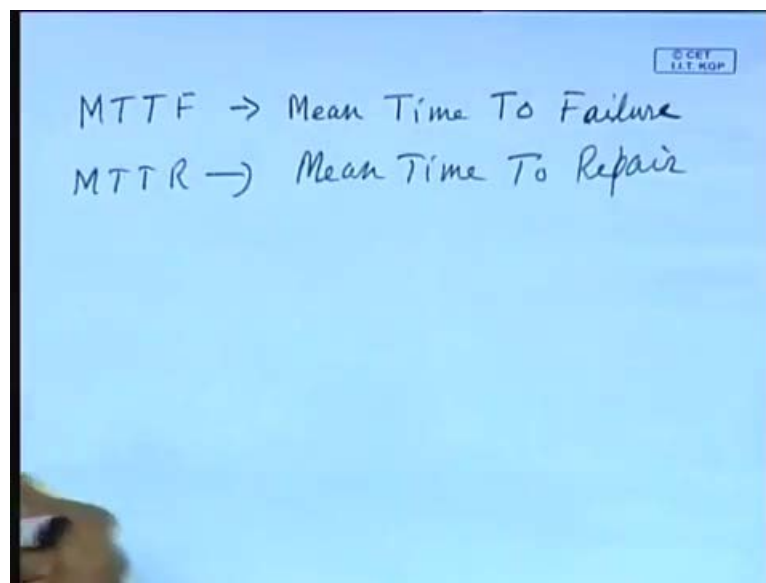
As, we shall see, so better availability is achieved in this way, if a disk fails then while it is being repaired data can be obtained from other disks so; that means, whenever we are using multiple disks, we have you can access one disk. And if a particular disk fails, then instead of accessing from that disks, we can access from other disks, so that is done, because of the replication of data is fully replicated on other disks. So, either you will read from the replicated data, or there is another approach which can be used that data can be re-constructed from the data on other disks.

So, if a particular disk fails, then from the replicated data, that is where the duplicate information is stored, because by incorporating redundancy you can read it from there, or from the remaining disks, which are available, which have not failed from those disks data can be reconstructed. We shall see how it can be done, one important issue is what if

other disk fails while disk is being repaired; that means, so far as the single fault is concerned.

It can be easily I mean tolerated, or in the sense that availability on reliability can be improved for a single disk failure; that means, whenever one disk is getting I mean regeneration of a particular disk is taking place from other disk. Say, if another disk fails then what will happen, now let us see whether this is possible or not, it has been found that the mean time to failure, so MTTF is a very important parameter.

(Refer Slide Time: 10:53)



MTTF Mean Time To Failure, so this figure is quite large, it has been found that of a disk is tens of years; that means, a particular disk does not fail in alternate days. So, failure is very uncommon and rare, so one seen may be several years tens of years it happens; however, that MTTR Mean Time To Repair that time is very low.

(Refer Slide Time: 11:45)

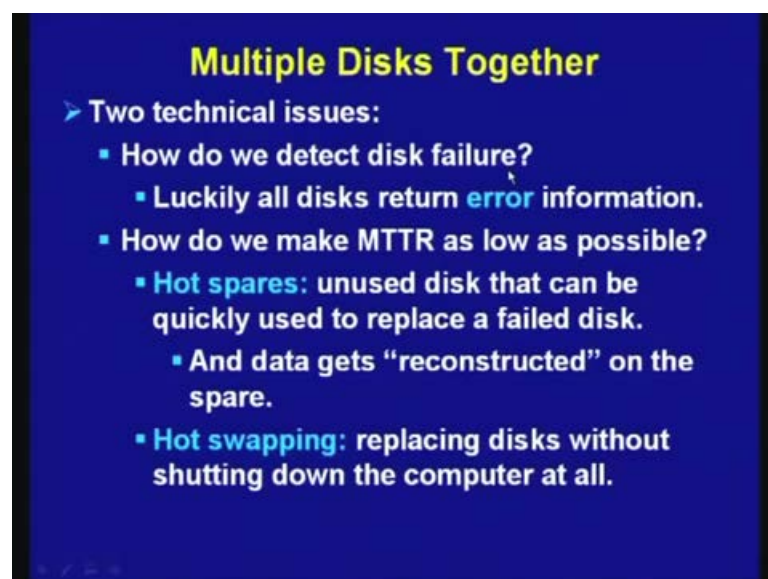


Multiple Disks Together

- **Better availability:** If a disk fails, then while it is being repaired data can be obtained from another disk:
 - Data is fully replicated on other disks.
 - Data can be reconstructed from data on other disks.
- **One issue:** what if another disk fails while a disk is being repaired?
 - **MTTF** of a disk in tens of years
 - **MTTR** of a disk can be as low as hours
 - Therefore, statistically there is no problem.

So, MTTR of a disk can be as low as hours; that means, if a particular disk fails, that failing that it will fail may be once in ten years, but whenever it fails within few hours, it can be regenerated or it can be I mean a new disk can be placed in place of that faulty disk. So, in such a case you know since the mean time to failure is very long, and mean time to regeneration repair is only few hours, it does not pose any problem; that means, there is a the possibility of multiple disk failing is extremely low, because of these two figures. So, therefore, statically there is no problem, the possibility of two disks failing together will not happen in practice.

(Refer Slide Time: 12:43)

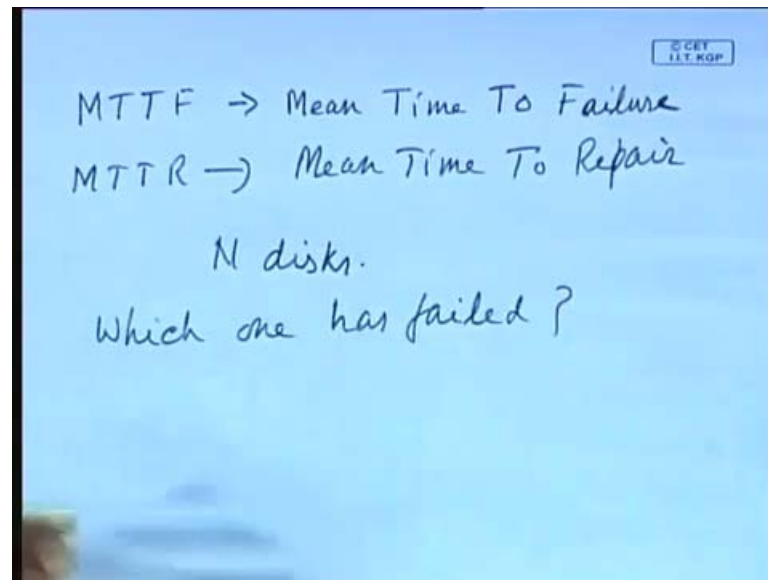


Multiple Disks Together

- **Two technical issues:**
 - How do we detect disk failure?
 - Luckily all disks return **error** information.
 - How do we make MTTR as low as possible?
 - **Hot spares:** unused disk that can be quickly used to replace a failed disk.
 - And data gets “reconstructed” on the spare.
 - **Hot swapping:** replacing disks without shutting down the computer at all.

Now, question is there are two important technical issues by which, mean time to repair is reduced, so how do you detect whenever a disk fails. So, there are you have to understand the technical aspects behind the operations of these multiple disks, say you have got n disks in a multiple disks.

(Refer Slide Time: 13:12)



Question arises, which one has failed, and how to find it out, fortunately the disk which has failed generates a flag. So, it is a self generated signal; that means, the failed disk, which has failed will generate a signal that it has I have failed.

(Refer Slide Time: 13:46)

Multiple Disks Together

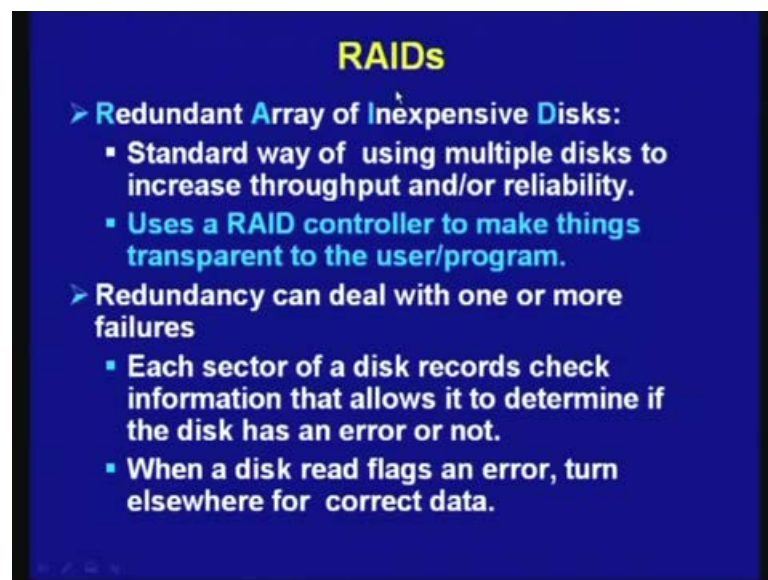
- Two technical issues:
 - How do we detect disk failure?
 - Luckily all disks return **error** information.
 - How do we make MTTR as low as possible?
 - **Hot spares**: unused disk that can be quickly used to replace a failed disk.
 - And data gets “reconstructed” on the spare.
 - **Hot swapping**: replacing disks without shutting down the computer at all.

So, that is very good thing luckily all disks return error information, so flag is generated and so identification of a faulty disk does not pose any problem. Now, how do we make MTTR as low as possible that is few hours, that can be done in two ways, one concept is known as hot spares, in such cause unused disk that can be quickly used to replace a failed disk.

That means, what can be done few disks are already in the rack, which are not being used, so those are known as hot spares, it is not that a disk has been kept in the [FL], that if the disk is already there, it is in the system, but it was not being used. So, as soon as a particular disk fails, it can be made operational very quickly, that is known as hot spares, another possibility is to use hot swapping.

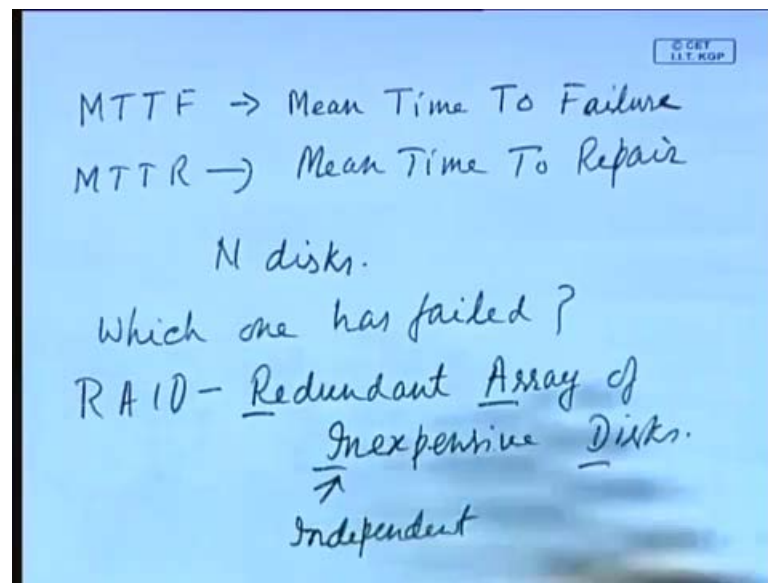
So, whenever a particular disk fails it can be taken out and a new disk can be placed, so hot swapping is without shutting down the system, you can put in a new disk a I mean a disk which is not faulty. So, replacing the disk without shutting down the computer at all, so these two techniques are used to reduce the MTTR, it can be very small.

(Refer Slide Time: 15:13)



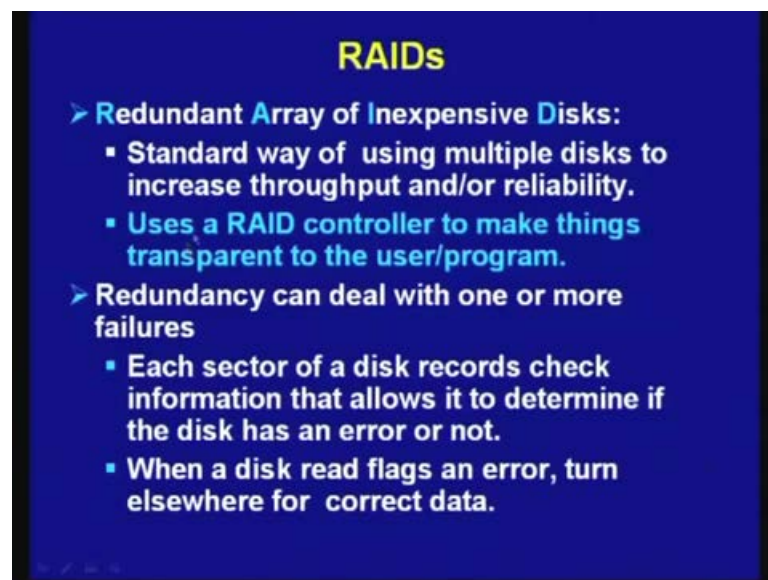
Now, we shall discuss about a technique, which is used for the purpose of improving the reliability and availability, and the technique is known as redundant array of inexpensive disks or in short it is RAID, so Redundant Array of Inexpensive Disks.

(Refer Slide Time: 15:33)



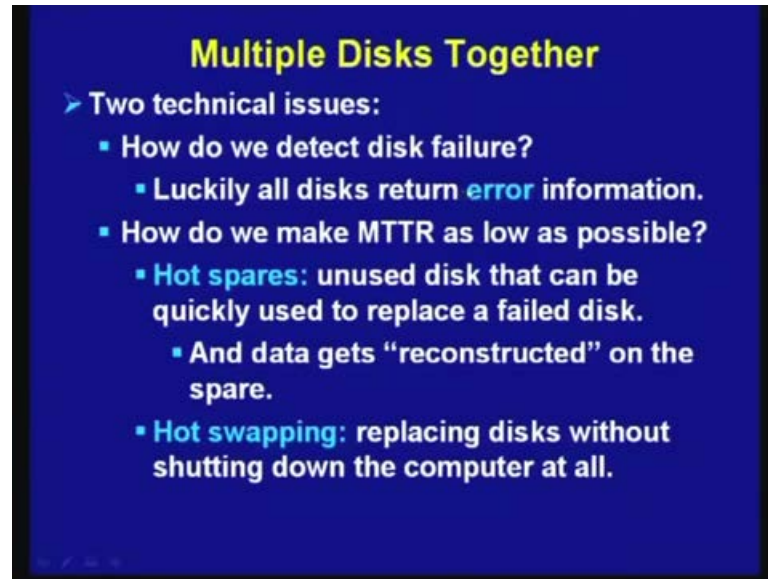
So, redundant RAID, this is the acronym that is being used, and this I sometimes is instead of inexpensive, it is also used in for independent.

(Refer Slide Time: 16:21)



So, independent or inexpensive I mean either any one of them or both of them can be used it does not matter and, so this is the standard way of using multiple disks to increase throughput and or reliability. So, this uses a RAID controller to make things transparent to the user or program.

(Refer Slide Time: 16:37)

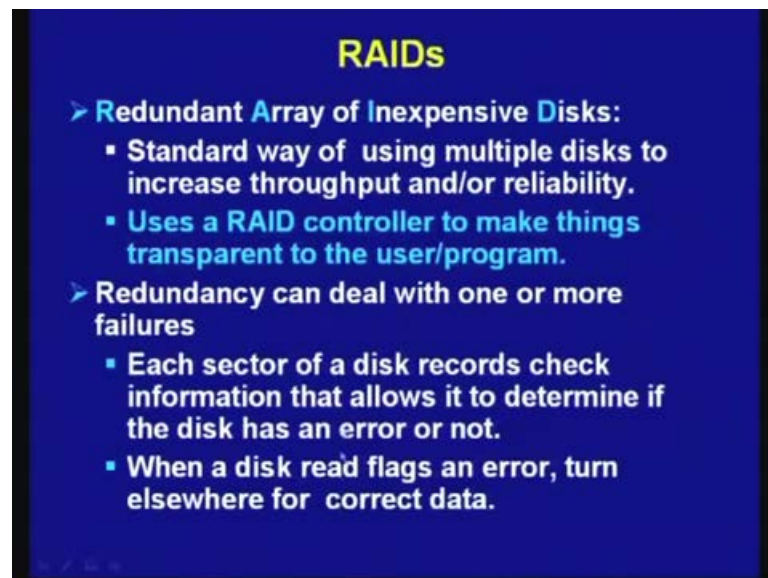


Multiple Disks Together

- Two technical issues:
 - How do we detect disk failure?
 - Luckily all disks return **error** information.
 - How do we make MTTR as low as possible?
 - **Hot spares**: unused disk that can be quickly used to replace a failed disk.
 - And data gets “reconstructed” on the spare.
 - **Hot swapping**: replacing disks without shutting down the computer at all.

In the previous cases we have seen whenever, we do hot spares or hot swapping sometimes manual intervention is required.

(Refer Slide Time: 16:47)



RAIDs

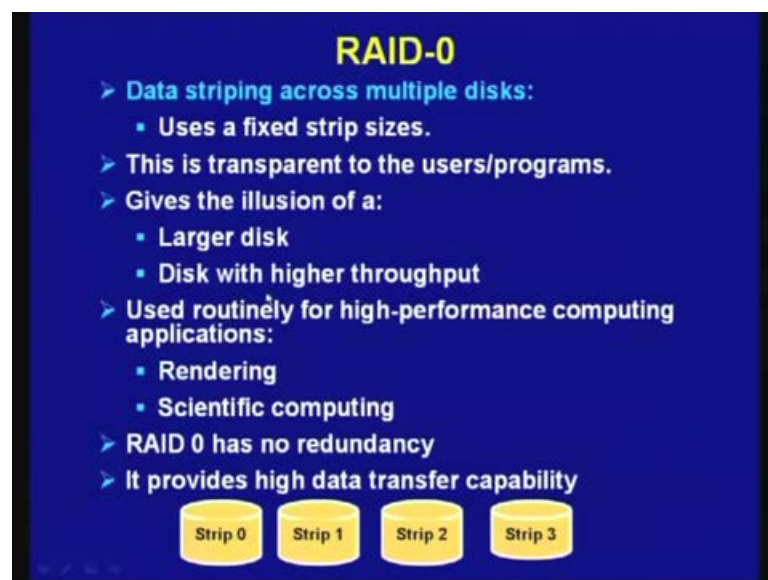
- **Redundant Array of Inexpensive Disks**:
 - Standard way of using multiple disks to increase throughput and/or reliability.
 - Uses a **RAID controller** to make things transparent to the user/program.
- **Redundancy can deal with one or more failures**
 - Each sector of a disk records check information that allows it to determine if the disk has an error or not.
 - When a disk read flags an error, turn elsewhere for correct data.

But, in this particular case whenever we are using RAID's then there is no need for any manual intervention the automatically the RAID controller, will that some electronics that is present in a system, will take care of will make thing transparent to the user or a programmer that is running. So, redundancy can deal with one or more failures, so what can be done, as we shall see either single failure or multiple failures can be tolerated, and

each sector of a disk records check information that allows it to determine if the disk has an error or not.

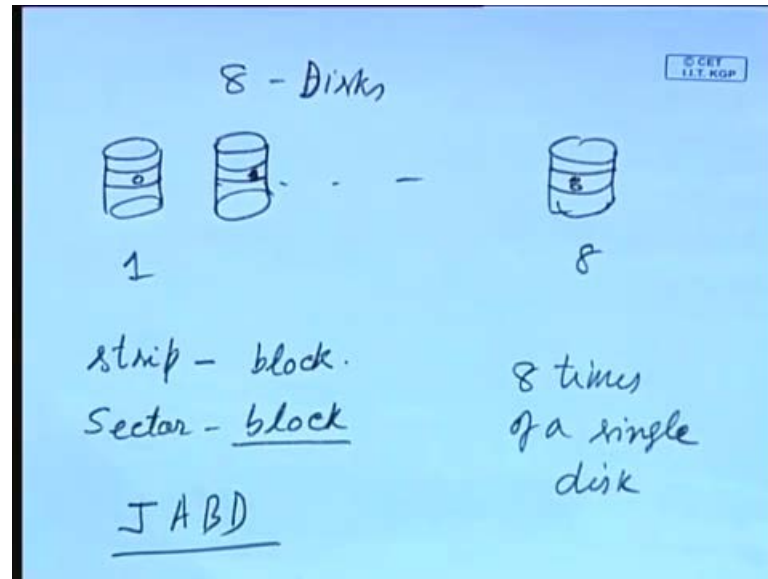
So, as we know some error detecting codes or error detecting I mean codes are usually present for each sector, so either parity or some other techniques check some, whatever it may be that is being used to detect any error for reading a sector. So, whenever that happens we know that a particular disk is has become fault, and when a disk reads flags an error turn, elsewhere for correct data. That means, whenever that flag is generated, because of incorrect reading from of a sector, then it generates a error message or you can say disk flags an error, and we have to turn to other devices for getting correct data how it can be done we shall see.

(Refer Slide Time: 18:24)



So, as I mentioned there are 6 different RAID levels, first one is known as RAID 0, so although you are using the terminology redundant, as we shall see in case of RAID level 0. There is no redundancy, and the main thing that is being done in RAID level 0 is data stripping across multiple disks, so uses a fixed strip sizes and this transparent to the users or programs, and what is being done.

(Refer Slide Time: 19:24)



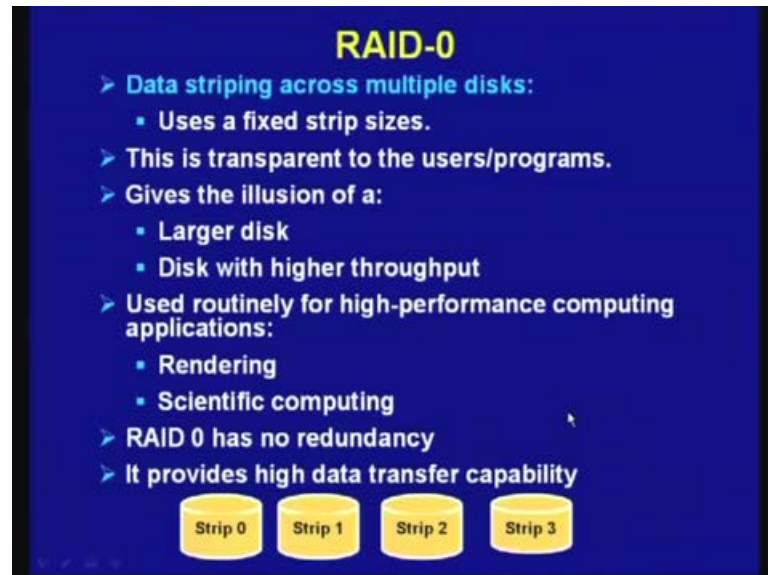
Say suppose you have got 8 disks, so this is one disk, and in this way you can have 8 disks, so one strip is stored here, that one strip, that is usually called a block. We have already seen that a data that it can be accessed from a hard disk minimum unit is a sector. So, a sector of data can be called as a block in the context of hard disks, or if we go for bigger size block then it may involve multiple sectors, that is why that we do not use it a sector, but you can have a block, which is bigger than a sector.

So, 0th block is placed here, then the first block is placed here, then the eighth block is placed here, so in this way data is distributed in 8 disks. You may be asking what purpose it will serve, the main purpose that will serve is first of all it will give the illusion of a larger disk so; that means, you can store large amount of data, and it will give the illusion of the single disk.

So, since it is transparent to the user, they will feel that the capacity is now 8 times that of a single disk, and also it gives you higher throughput, as we know we have already seen whenever you read data from a disk. It will involve several I mean parameters of times, seek times, then rotational time, then transfer time, all this times are involved, but here all these things can be done parallel. And as a consequence it will give you high it gives you performance, so it uses routinely for high performance computing applications like rendering scientific computing.

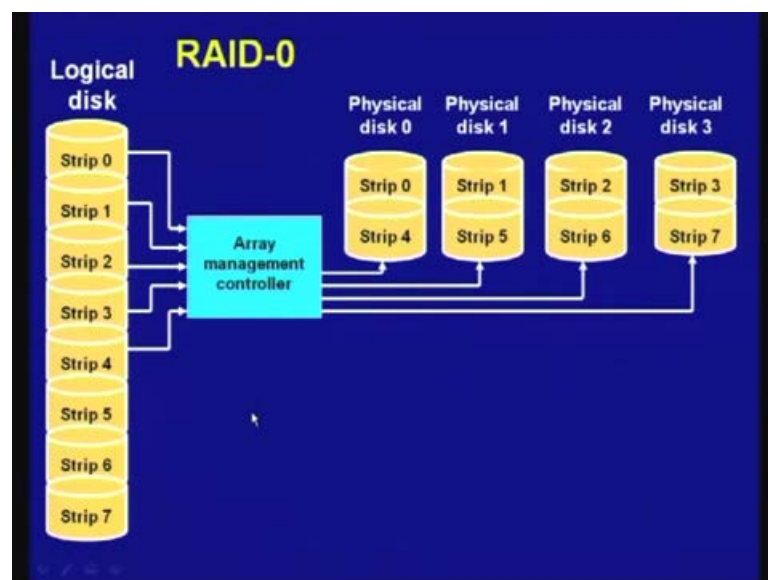
So, as I mentioned ironically although you are using the term RAID it has no redundancy, so it provides high data transfer capability, so it is just a bunch of disks, so that is a terminology sometimes used just a bunch of disks JABD.

(Refer Slide Time: 22:11)



So, this is RAID level 0.

(Refer Slide Time: 22:15)

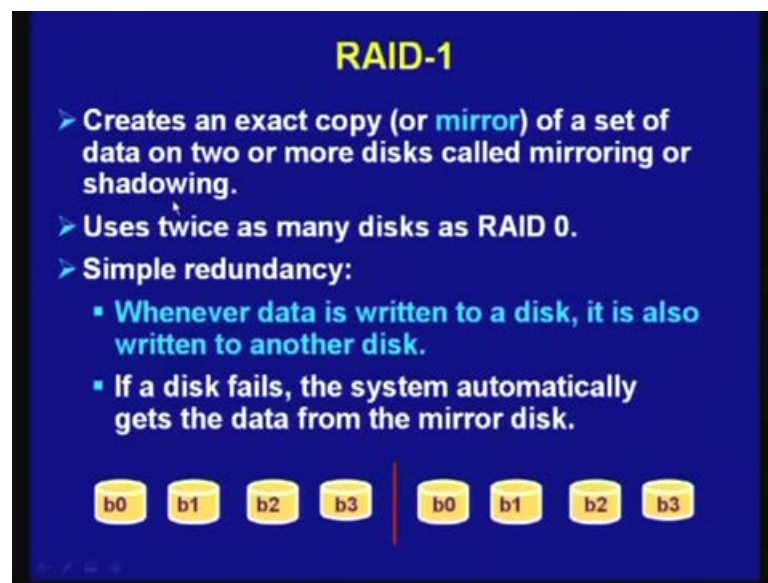


Now, let us come to how it is being done, this is only here, so here the user have the illusion of a single logical disk of large capacity, then there is a array management controller, which distributes the data to different disks. Say this is got in this particular

diagram 4 different disk 0, disk 1, disk 2, and disk 3, and where data is distributed to a strip 0 goes to disk 0, strip 0 goes to disk 0, strip 1 goes to disk 1 strip 2 goes to disk 2 and that distribution is done by this array management controller.

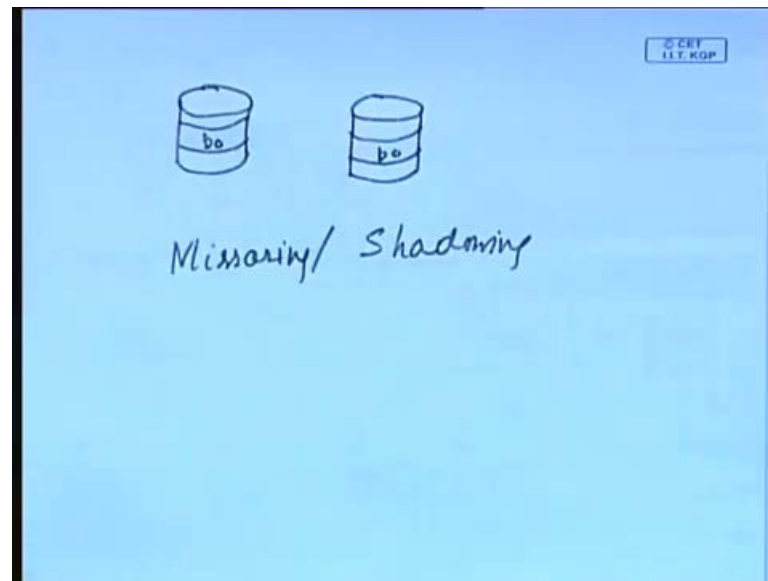
And the user have a logical disks phase, so whenever it reads then the array management controller will read them from different disk, and provide the data to the user or the program which is running.

(Refer Slide Time: 23:18)



So, this is your RAID level 0, and a this was how it really works coming to RAID level 1, so this creates an exact copy of a set of data on 2 or more disks called mirroring or shadowing, so in this case what we are doing to improve the reliability and availability.

(Refer Slide Time: 23:36)



You are doing what is known as mirroring, so suppose this is one disk, and here you have got another disk, so same data, so if you are storing block b 0 here, you will be also storing block b 0 here. So, you are using a concept known as mirroring or which is also known as shadowing, so this is the in context of only two disks, and since you are using mirroring and shadowing, although you are having two disks the capacity is equivalent to that of a single disk.

(Refer Slide Time: 24:32)

RAID-1

- Creates an exact copy (or **mirror**) of a set of data on two or more disks called mirroring or shadowing.
- Uses twice as many disks as RAID 0.
- Simple redundancy:
 - Whenever data is written to a disk, it is also written to another disk.
 - If a disk fails, the system automatically gets the data from the mirror disk.

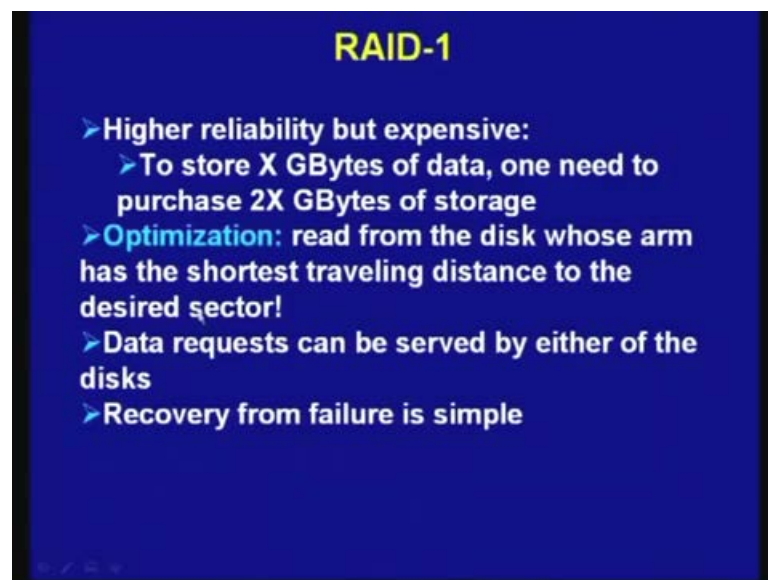
A diagram showing two identical rows of four yellow rounded rectangles, each containing a label. The left row contains labels "b0", "b1", "b2", and "b3". The right row also contains labels "b0", "b1", "b2", and "b3". A vertical red line is positioned between the two rows, separating the two disks in the RAID-1 configuration.

So, in this case this is the very simple redundancy and very simple, and that widely used in many applications in the early years of redundancy, and whenever data is written to a disk it is also written to another disk. So, in this particular case I mean reading can be done either from this disk where block 0 is stored or can be you can read it either from the disk.

So, there is a possibility of reading from both, but usually data is read from one main disk not, the mirror disk only, when the main disk fails, the data is read from the mirror disk, and in the mean time the regeneration of data takes place in the failed disk. So, this is how it really works, this is automatically gets data from mirror disk, and regeneration takes place in the other disk, this is how RAID level 1 works.

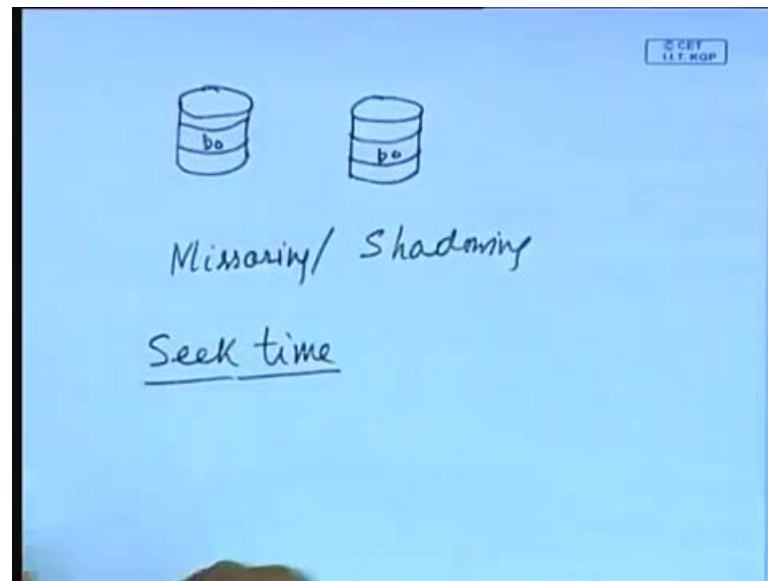
So, in this particular case, as you can see the redundancy is maximum, you are require double the number of disks, so whenever you are using this type of RAID level 1 for the purpose of improving the liability and availability. So, because of this large requirement I mean requirement of, so many disks, this is rarely used.

(Refer Slide Time: 26:01)



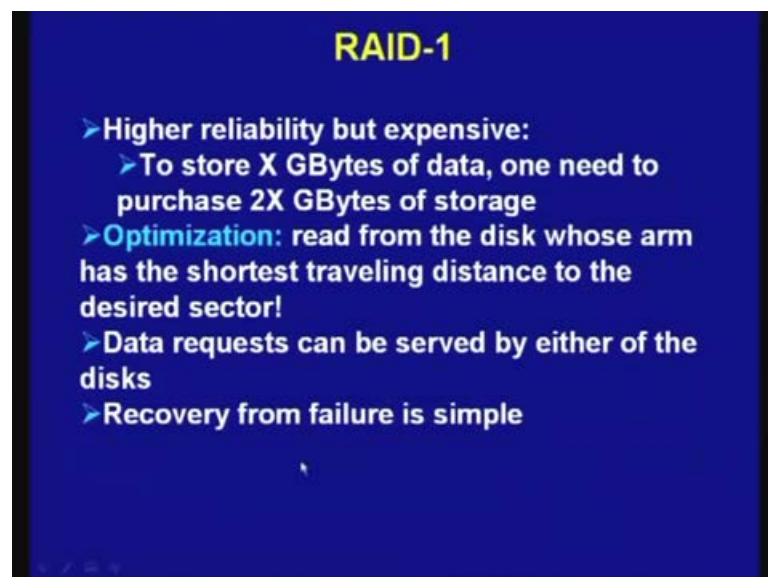
and so, as I mentioned higher availability, but expensive to store x gigabytes of data one need to purchase, 2 x gigabytes of storage, so you can do some optimization read from disk, whose arm has the shortest travelling distance to the desired sector.

(Refer Slide Time: 26:29)



So, this kind of optimization can be done, as you have seen, there is a seek time, seek time is dependent on where the position of the arm, and since we are having two disks storing the same data. So, you can identify, for which one arm is nearer to the track that has been referred by the access, so then you can access from that particular disk, so that kind of optimization can be done.

(Refer Slide Time: 26:57)




And data requests can be served by either of the disks, and recovery from failure is very simple as I have told already explained.

(Refer Slide Time: 27:07)

RAID-2

➤ RAID 2 is not being used:


- Was defined in the original standard.
- Expensive in terms of disks.
- Expensive in terms of the controller.
- The main feature was the use of Error Correcting Codes (ECC).
- But most disks provide ECC by default nowadays.



The diagram shows seven yellow disk icons arranged in a row. The first four disks are labeled b0, b1, b2, and b3. A vertical red line separates them from the next three disks, which are labeled f1(b), f2(b), and f3(b). This illustrates the parity structure of RAID 2, where data is striped across disks and parity is stored on separate disks.

So, this is RAID level 1, then coming to RAID level 2 we have started with a sentence RAID level 2 is not being used, but since in the original document of RAID, this was also mentioned for the sake of academic interest. We shall be discussing it very briefly, this since this was defined in the original standard, and this is also expensive in terms of disks, say and also expensive in terms of the controller, why it is expensive.

(Refer Slide Time: 27:43)



The diagram shows two hand-drawn disk icons, each labeled 'b0'. Below them, the text 'Mirroring/Shadowing' is written. Underneath, the words 'Seek time' and 'Error Correcting Codes' are written and underlined. A hand is visible at the bottom right, holding a white marker and pointing at the underlined text 'Error Correcting Codes'.


You can see we are using the conventional technique of error correcting codes, for the purpose of regenerating data.

(Refer Slide Time: 28:03)

RAID-2

➤ RAID 2 is not being used:

- Was defined in the original standard.
- Expensive in terms of disks.
- Expensive in terms of the controller.
- The main feature was the use of Error Correcting Codes (ECC) .
- But most disks provide ECC by default nowadays.

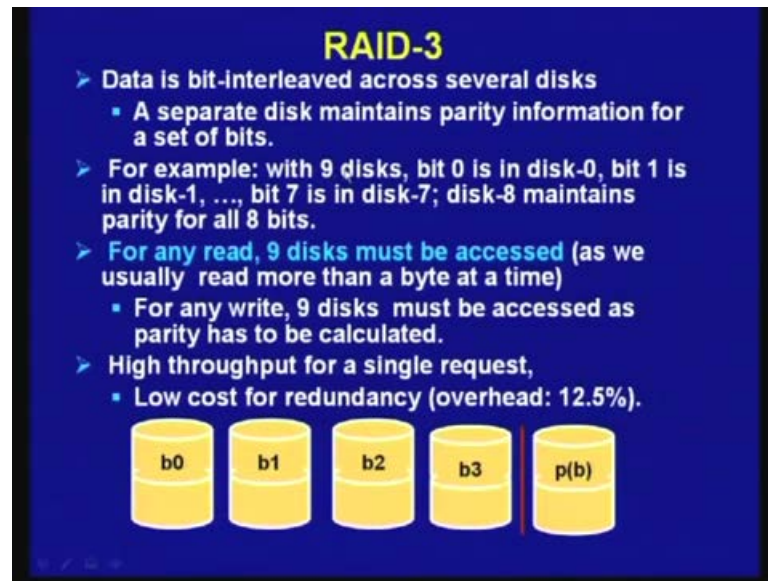


The diagram illustrates the RAID-2 disk layout. It shows seven yellow cylindrical disks arranged in a row. The first four disks are labeled b0, b1, b2, and b3. A vertical red line separates these from the next three disks, which are labeled f1(b), f2(b), and f3(b). This represents the data and its corresponding error-correcting codes.

That means, in this case as you can see, here you have got 4 disks and to make it feasible for correcting code automatically, you will require 3 additional disks, where you have to store those redundant information. So, that you can regenerate data whenever a one of the disks fails, so in this particular case I am not going into that details of error correcting codes you may have studied it.

And you will require large amount of redundancy for 4 disk, you will require three additional disks to store information, and the redundancy requirement is quite heavy. So, that is why this is not used; however, most disks provide ECC by default, nowadays this is done not in the disk level, it is done within the disk. So, that is why error correcting codes not provided in the this way, but within the disk can use error correcting code, because you know disks are unreliable medium, an error correcting code is provided within the disk.

(Refer Slide Time: 29:15)



Then coming to the RAID 3 data is bit interleaved across several disks, so if separate disk maintains parity information for a set of bits, so here you are using the concept of parity bit, for the purpose of error detection and correction. Actually, error detection is done by the disk itself, as I have already told the disk will give a send a flag, so whenever a disk has failed that is known. Now, from the remaining disk, you should be able to regenerate data by using the concept of parity bit, that is the basic idea of this RAID level 3.

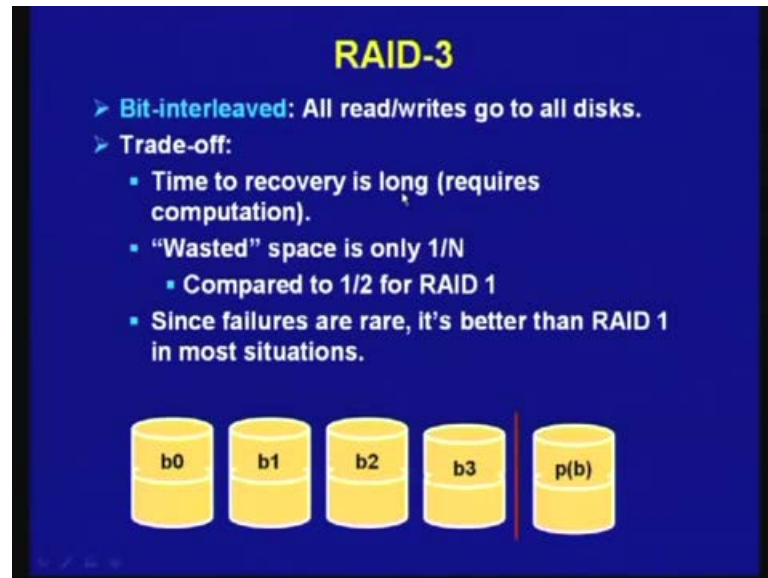
So, for example, whenever you are using nine disks, you are storing 8 bits of data in a 8 disks, bit 0 in disk 0 bit 1 is in disk 1 bit 7 in disk 7 and so on, and then disk 8 is used to store the parity bit. So, for any read you will require nine disks to be accessed, this is a very important point you must notice, so whenever you are performing reading or writing you have to access all the disks.

That means, whenever you are reading you have to read all the 9 disks must be accessed, whenever you are using 9 disks or if you are using 5 disks, you have to use read all the 5 disks, because after reading those bits. You have to check whether you have read correctly or not that parity bit has to be compared, so for any 9 disks must be accessed as parity has to be calculated.

So, this gives you high throughput for a single request, since you are reading from multiple disks, so you are reading parallely, so it will give you high throughput. And

redundancy overhead is much smaller only one 2.5 percent whenever you are using 9 disks; that means, one extra you required for 8 data, so 1 by 8 in percentage divided is 12.5 percent.

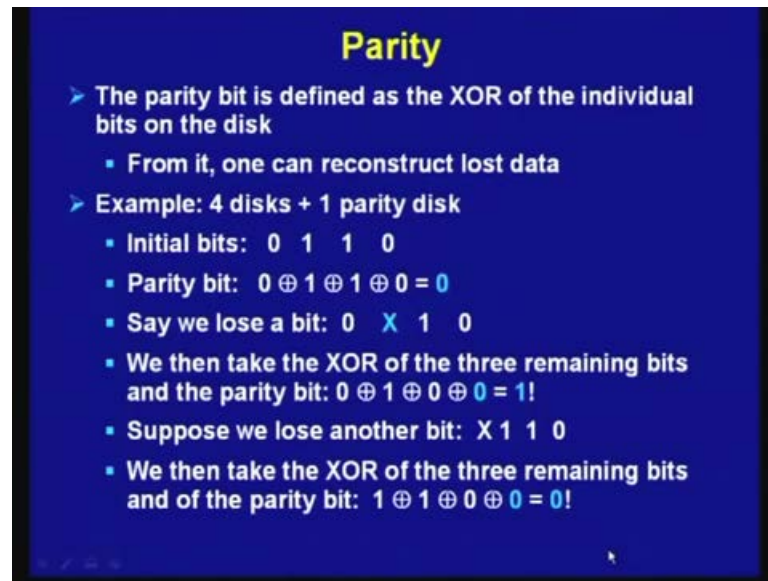
(Refer Slide Time: 31:33)



So, since the interleaving is done interleaving is; that means, you have distributed bit wise, that all read write go to all disks, so time to recovery is very long, because you will require computation, you have to read from all the disks. Then if a particular disk fails, then you have to read all the disks then you have to regenerate the other disks, so the time to recovery is very long, because of the calculation of the parity bit.

And wasted space is only one by n, as I have already mentioned compared to 1 by 2 that is used in RAID level 1, so since failures are rare it is better than RAID 1 in most of the situations. So, if you compare RAID level three with RAID level 1, RAID level 2, actually it has to be RAID level 2, where redundancy is used here this is a better scheme.

(Refer Slide Time: 32:37)



Parity

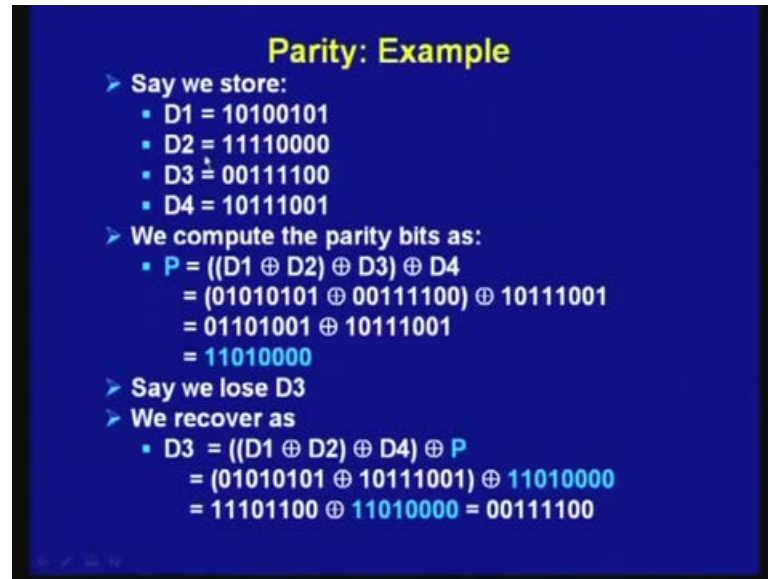
- The parity bit is defined as the XOR of the individual bits on the disk
 - From it, one can reconstruct lost data
- Example: 4 disks + 1 parity disk
 - Initial bits: 0 1 1 0
 - Parity bit: $0 \oplus 1 \oplus 1 \oplus 0 = 0$
 - Say we lose a bit: 0 X 1 0
 - We then take the XOR of the three remaining bits and the parity bit: $0 \oplus 1 \oplus 0 \oplus 0 = 1!$
 - Suppose we lose another bit: X 1 1 0
 - We then take the XOR of the three remaining bits and of the parity bit: $1 \oplus 1 \oplus 0 \oplus 0 = 0!$

So, this is the simple idea of parity bit, how it is being used for the purpose of regeneration of data, so the parity bit is defined as a XOR of the individual bits on the disk, from it 1 can reconstruct lost data. For example, whenever you are using 5 disks 4 disks for storing data, and 1 for parity disk, let us assume the initial data is 0 1 1 0, so parity bit can be easily calculated, since you are using even parity, so 0 1 1 0 even parity bit is 0 bit.

So, we will store 0 in that, disk where you are storing the parity bits, suppose we have lost a bit; that means, this particular disks that first the disk number 1 has failed. So, disk has failed, you can reconstruct the data from the other disks including the parity bit disk, so we can take the XOR of the 3 remaining bits, and the parity bit; that means, 0 exclusive OR 1 exclusive OR 0 exclusive OR 0, and that gives you 1, and indeed this is the data that was present here.

So, you can easily reconstruct the data from this, suppose you have lost another bit, say this bit has been lost after this, and here also again you can calculate the find out the bit value to that was present here. In the same way and you get 0 and indeed 0 was present, as you can see here, so in this way this bit interleaved parity can be easily used for the purpose of regeneration of the data, whenever a particular disk fails.

(Refer Slide Time: 34:33)



Parity: Example

- Say we store:
 - D1 = 10100101
 - D2 = 11110000
 - D3 = 00111100
 - D4 = 10111001
- We compute the parity bits as:
 - $P = ((D1 \oplus D2) \oplus D3) \oplus D4$
 $= (01010101 \oplus 00111100) \oplus 10111001$
 $= 01101001 \oplus 10111001$
 $= 11010000$
- Say we lose D3
- We recover as
 - $D3 = ((D1 \oplus D2) \oplus D4) \oplus P$
 $= (01010101 \oplus 10111001) \oplus 11010000$
 $= 11101100 \oplus 11010000 = 00111100$

It can be done in the block level as well, suppose you have got data 8 bit data D 1, D 2, D 3 and D 4 stored in 4 different disks. So, you can compute the parity D 1 exclusive of D 2 exclusive of D 3 exclusive of D 4 to get the parity 1 1 0 1 0 0 0 0, for the data which is found here. Then suppose the D 3 has failed, you can regenerate the data to be stored in D 3 in the same way, D 1 exclusive of D 2 exclusive of D 4 exclusive of P gives you the data that was present in the disk D 3, so this is how you can do it in the bit level or in the block level.

(Refer Slide Time: 35:21)



RAID-4

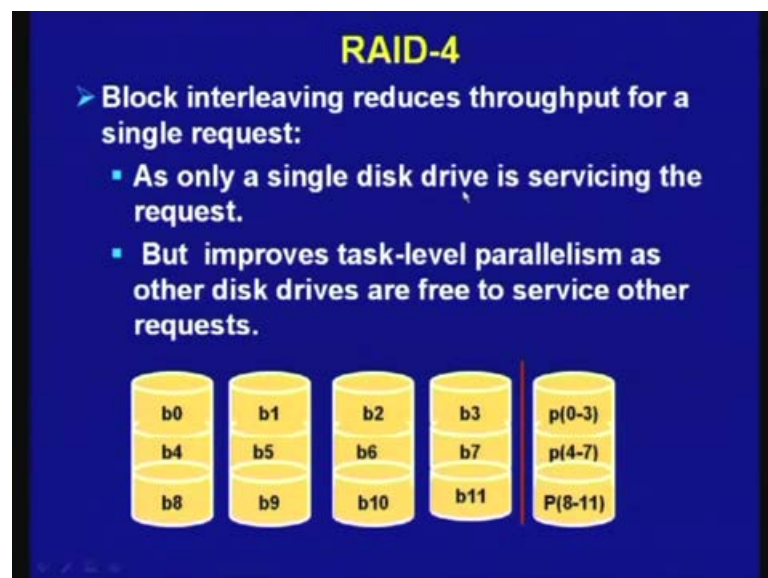
- **Block-interleaved Parity**
 - Basically the same idea as RAID 3 but improved
- In RAID 3, every read/write goes to all disks because data is interleaved at the bit level.
- In RAID 4, data is striped in a block fashion:
 - That way a (small) read involves only one disk.
 - An application can then do many small independent reads to multiple disks.
 - Sometimes more desirable than the bit-level scheme in RAID 3.

And that is the reason why instead of single bit, in the bit level the RAID level 4 has gone for block interleaved parity, so in this case the idea is same as RAID level 3, but improved. How it has been improved, because in RAID level 3, we have seen every read write goes to all disk, because data is entirely at bit level, so you have to read some meaningful data, you have to read all, you have to access all the disks, but that need not be done, whenever it is block interleaved.

So, whenever you are reading small amount of data, that may be present in a single disk, instead of reading from all the disk for small data, I mean for large data say more than a block. If it is only one block, then you can read it from single disk, so that idea is being used in RAID level 4, so data is striped in the block fashion, so that way a small read involves only one disk.

So, an application can then do many small independent reads to multiple disks, so this allows you to have multiple reading for multiple applications. So, in case of multiple environment multiple users can read from multiple disks in a block level, and that is a reason why it is more desirable than bit levels scheme of RAID 3; however, the time needed for reading data will be longer same volume of data.

(Refer Slide Time: 36:56)

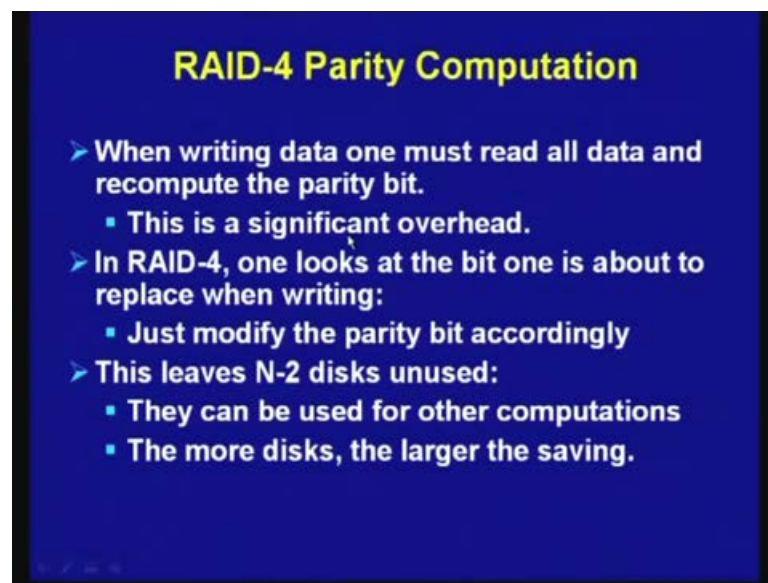


So, block interleaving reduces throughput for a single request, as I have already mentioned as only a single disk drive is servicing the request, so you have to transfer the entire data from a single disk. It will take longer time, but improves the task level

parallelism as other disk drives are free to service other request, as I have already told in a multi programming environment, whenever you are using task level parallelism, that can be helpful in this in the RAID level 4.

So, this is how the data is stored is shown here, block 0, block 1, block 2, block 3, because 1 parity block level parity is stored here, 4 5 6 7 here, in 4 different disk corresponding parity is stored in another disk and so on.

(Refer Slide Time: 37:53)

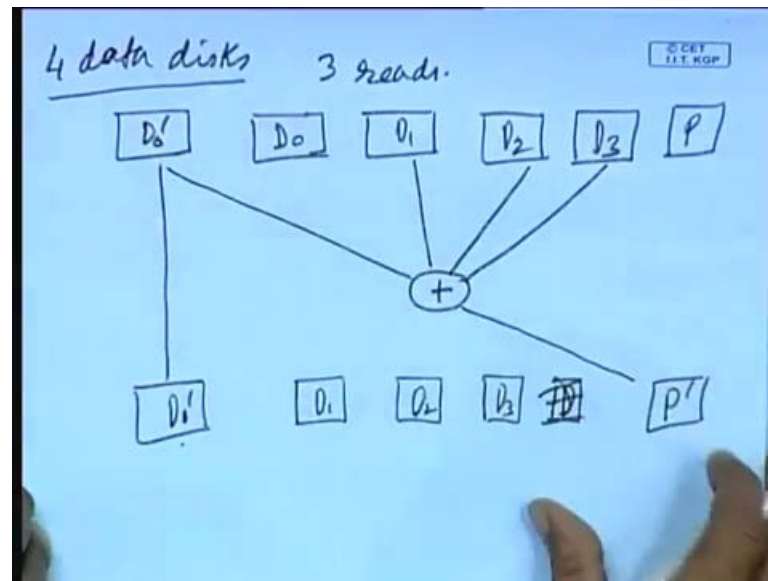


RAID-4 Parity Computation

- When writing data one must read all data and recompute the parity bit.
 - This is a significant overhead.
- In RAID-4, one looks at the bit one is about to replace when writing:
 - Just modify the parity bit accordingly
- This leaves N-2 disks unused:
 - They can be used for other computations
 - The more disks, the larger the saving.

So, this is your RAID level 4, now coming to when writing data, one must read all disk and recomputed the parity bit, so this is a significant overhead as we have already seen.

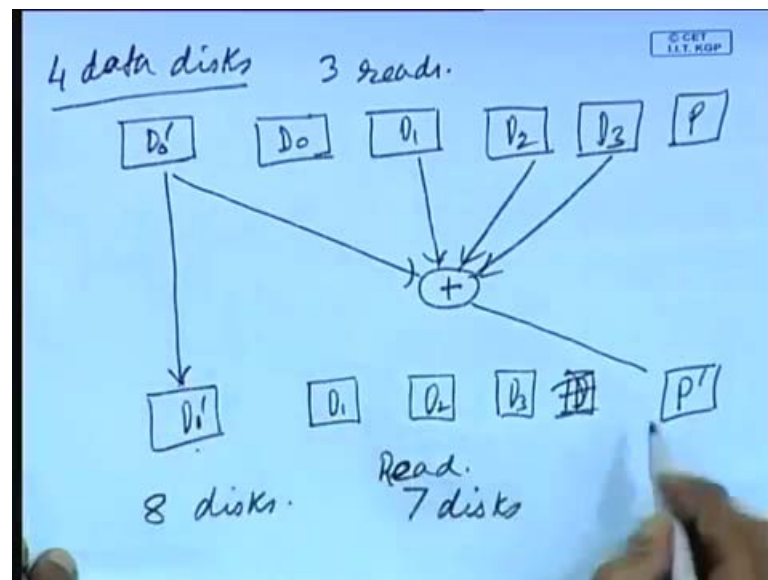
(Refer Slide Time: 38:11)



Suppose, you have got 4 disks say D_0 , D_1 , D_2 , D_3 and this is your parity, so suppose this is the new data you have to write, so what you have to do, you have to calculate the new parity by using all the data that you will perform exclusive operation. Then you will write in that new parity bit, it will go to the new parity bit, this data will go to the new disk, so that is your D_0 dash.

Then of course, you will be having D_1 , D_2 , D_3 and D_4 the D_0 , D_1 , D_2 and D_3 , so in this case as you can see you have to read all the disks. So, you will require 3 reads, if you are having say 4 data disks, if you write on one of the disk, you have to perform 3 reads, then of course you will be writing into disk that is required. So, the number of reads that you have to perform is long, can it be reduced, so whenever you are using this RAID level 4, what you can do you can use a better scheme.

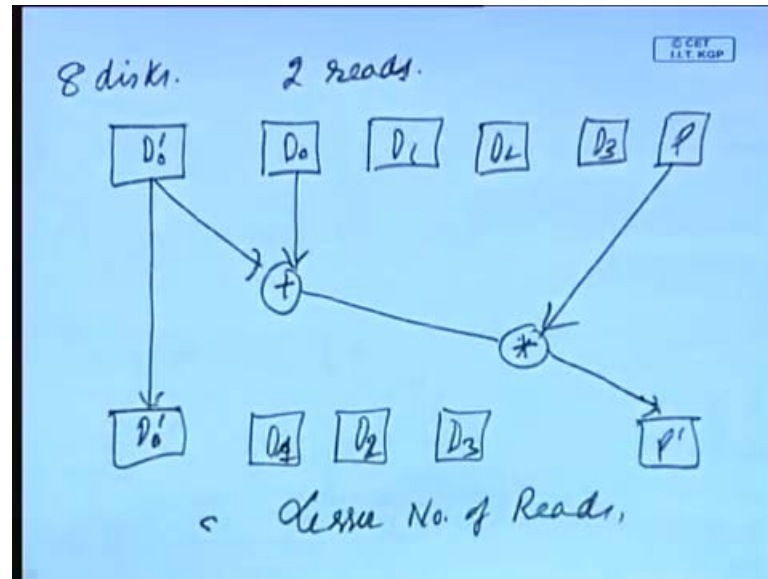
(Refer Slide Time: 39:55)



So, better scheme is like this you do not have to read so many disks, so suppose it is disk D_0 , then D_0 D_1 D_2 and D_3 , and this is your parity bit, now you have to write in D_0' dash, in any case. And for the calculation of parity, what you can do, you can use these 2 bits exclusive OR, and also this can be you can perform exclusive OR with this old parity, and you can write a new parity.

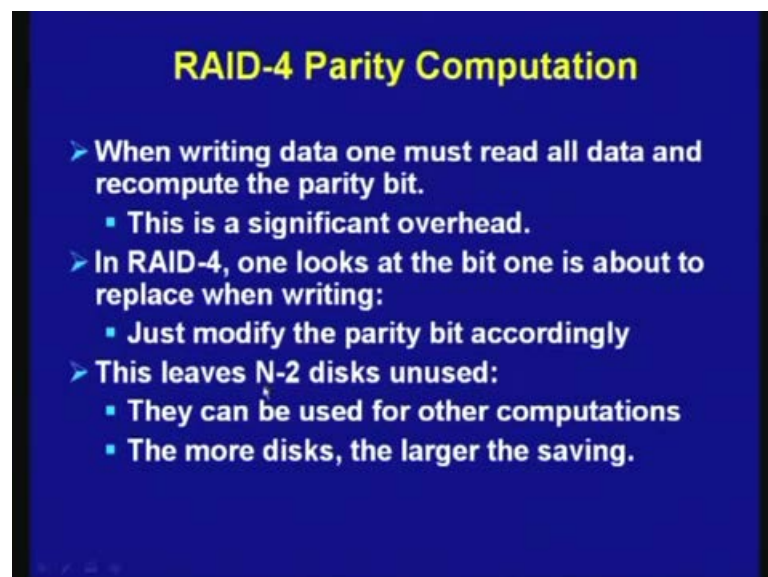
And of course, D_0 D_1 D_2 , D_0 is already there D_1 D_2 and D_3 , so in this case as you can see the number of reads is two, but whenever you have got 8 disks. In the previous case your requirement was you had 8 data disks, then you had to read from 7 disks, I mean read you have to read 7 data disks, but whenever you use this scheme.

(Refer Slide Time: 41:18)



You can see you have to read only 2 disks, number of reads is reducing, if it is 8 even when you have got 8 disks, you will require only two reads by this enhanced scheme, and you can calculate the parity. It can be proved that you get the same parity, new parity whenever you do it in this manner; that means, the parity that is being generated in this scheme, will be same as the parity that is being generated in this scheme.

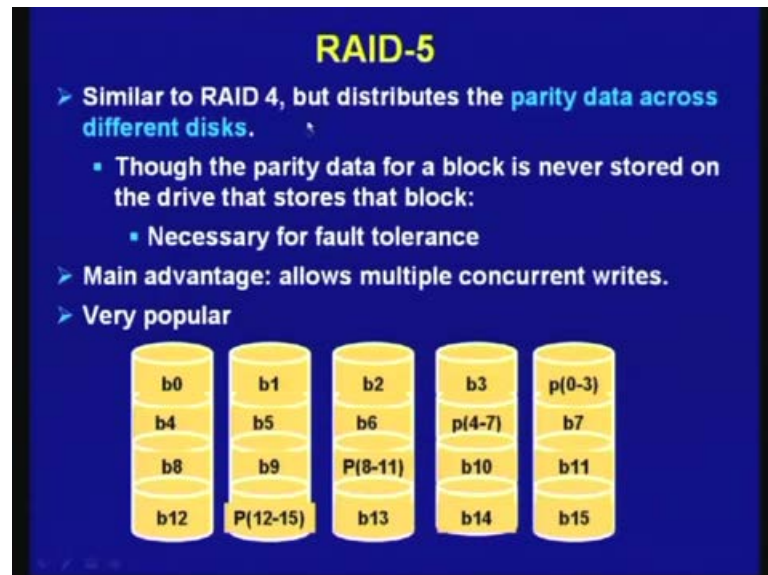
(Refer Slide Time: 41:58)



So, you will require lesser number of reads, and that is what is being done in RAID level 4 and, so in this particular case whenever you are writing, you have to perform writing

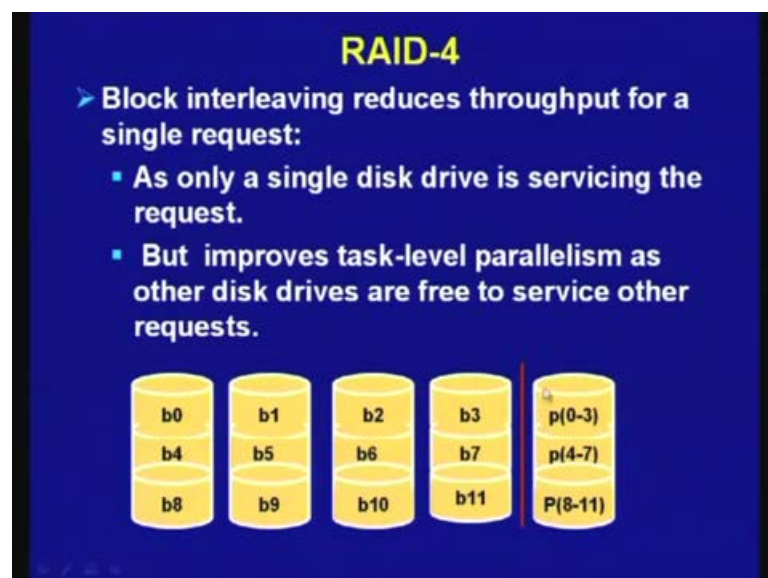
only on the $n - 2$ disks. If you have got n disks, this leaves $n - 2$ disks unused, so they can be used for other computations, the more disks the larger the saving.

(Refer Slide Time: 42:32)



As, I have already explained within the context of 8 disks, 9 disks, so this is your RAID level 4, then coming to the RAID 5.

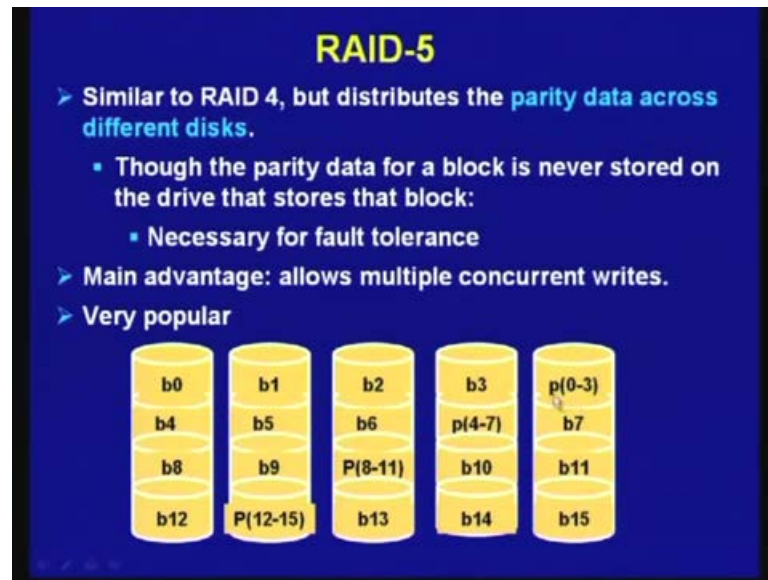
(Refer Slide Time: 42:40)



In the previous case we have seen, that data was parity bits all the parity bits were stored in a single disks; that means, whenever you are performing trying to perform writing. Then you have to access that particular disk, and also suppose you are reading block b 0,

and block b 5 writing block b 0 and writing block b 5, so what you have to do you have to access block b 0, you have to access, disk d 0, disk 1. And also you have to access the parity disk twice, because you have to modify this p 0 to 3 and p 4 to 7, so this leads to bottleneck.

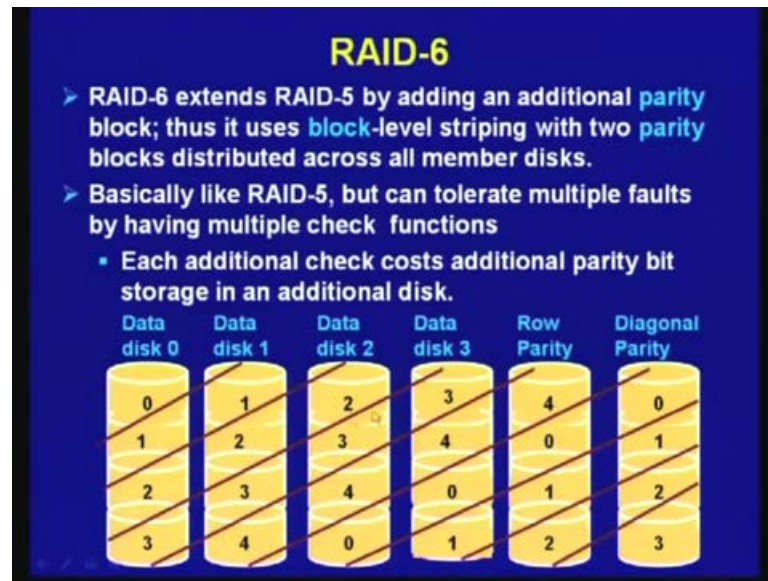
(Refer Slide Time: 43:27)



This bottleneck can be overcome by using this RAID 5, where the parity data is distributed across different disks, so though the parity data for a block is never stored on the disk drive, that stores that block. So, what you were doing here, the you are using say for example, p 0 to 3 stored in this disk 5, p 4 to 7 stored in disk 4 p 8 to 11 in disk 3 and so on.

Suppose, you are reading, writing b 0 and b 5, so you can see you have to modify b 0 and b 5 means you have to access b 0 and disk D 0 and disk D 4, and also whenever you are modifying writing block 5, then you have to access the disk D 1 and disk D 3. So, you can see you can do it parallelly at this, so the bottleneck arising in RAID level 4 is overcome. So, main advantages is allows multiple concurrent write, writes as I have already explained, and is the reason, why this RAID 5 is very popular and it is widely used in most of the commercial systems.

(Refer Slide Time: 44:50)



I believe our discussion will not be complete without considering RAID level 6, so RAID level 6 extends the RAID level 5 by adding an additional parity block, what is the why do you need additional parity block. The need for having additional parity arises to take care of multiple failures, we have seen from RAID level 1 to RAID level 5, they can tolerate single disk failure, but there are situation, where you have to handle multiple failures.

For example, an user instead of replacing a faulty disk has taken out a good disk, correct disk, so in such a case it will lead to multiple failures, so it will be difficult to regenerate the data, whenever multiple failure occurs. So, to take care of that situation RAID level 6 is used, where two types of parities are used, one is that row oriented parity that is a row parity, so you are using 1 disk to stores the row parity; that means, the row parity for 0 1 2 and 3.

That row parity storing this disk 4, in addition to that you are storing another parity which is known as diagonal parity, so diagonal parity, how it is calculated diagonal parity is calculated for all the diagonal elements. For example, that diagonal 0 corresponds to disk 0, disk 2, disk 3, disk 4, and disk 5, and that diagonal 1 corresponds to disk 0, disk 1, disk 3, disk 4, and disk 5.

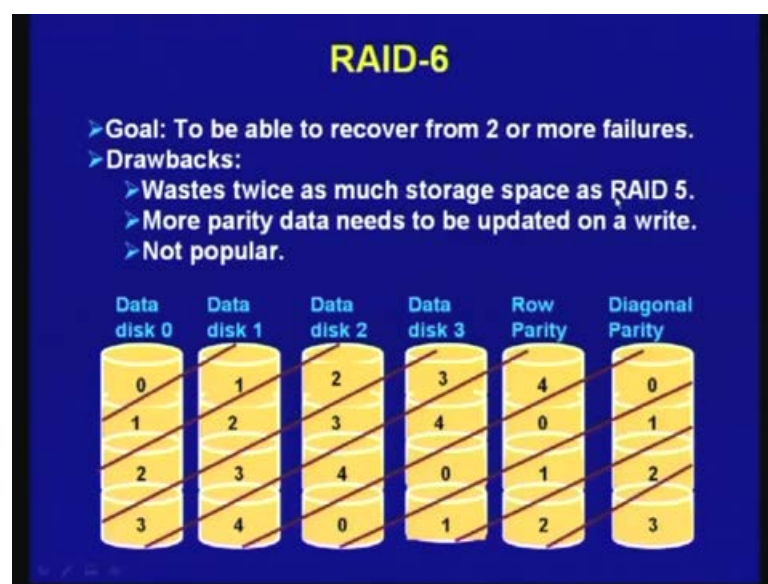
So, in this way diagonally the number that is being shown here the parity is calculated for those diagonal elements; however, you can see if you are having n disks the parity is

restricted to $P - 1$. So, if you got P disks altogether $P - 1$ parity is calculated on $P - 1$ disk, so whenever multiple disks fail, for example disk 1 and disk 3 has failed together. So, how do you take care of that situation, so whenever both disk 1 and disk 3 has failed, but that simple row parity calculation will not allow you to regenerate data in disk 1 and disk 3.

So, in such a situation you have to resort to this diagonal parity, for example this disk 1 and disk 3 fails, then find that diagonal parity 0 does not involve disk 1, so you can regenerate data for third disk by using the 0 diagonal elements. And, because one of the faulty disk is excluded from there, similarly you can use say diagonal 2 for regenerating data for disk 1. So, in this way two failures can be I mean whenever 2 more than 1 failure occurs, you will be able to reconstruct data using the diagonal parity, that is a basic idea of RAID level 6.

And if you want to have I mean tolerate more than 2 faults, then you have to keep on increasing the a number of parity disks, so if you want to tolerate 3 faults, then you have to add another additional disks. But, as you have already seen that the possibility of failure is I mean that mean time to failure is very long, so this RAID level 6 although theoretically, it has been proposed it is rarely used. Because, RAID 5 satisfies most of the requirements, because single failure is that what occurs occasionally, so double failure is very rare and so, RAID level 6 is not commonly used.

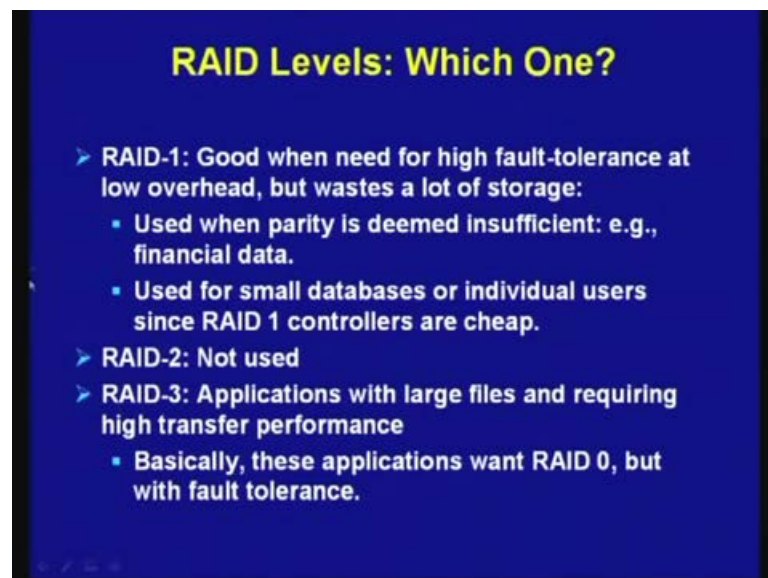
(Refer Slide Time: 49:33)



Also, as I have already mentioned recovers 2 or more failures, you have to go for RAID level 6, so drawback is that waste twice as many storage as RAID 5, and more parity data needs to be updated on a write. So, whenever you are writing data, this will be more complex, because you have to write data ah not only in the row parity also, you have write data in diagonal parity. And as you can see you have to perform I mean access of multiple disks to generate, that the row parity in the block level, and diagonal parity in a block level that calculation will involve access of several disks.

And that is why writing is very complex, and it will take time, and one that is the reason why this RAID level 6, although it has been proposed after RAID 5 it is not commonly used.

(Refer Slide Time: 50:43)



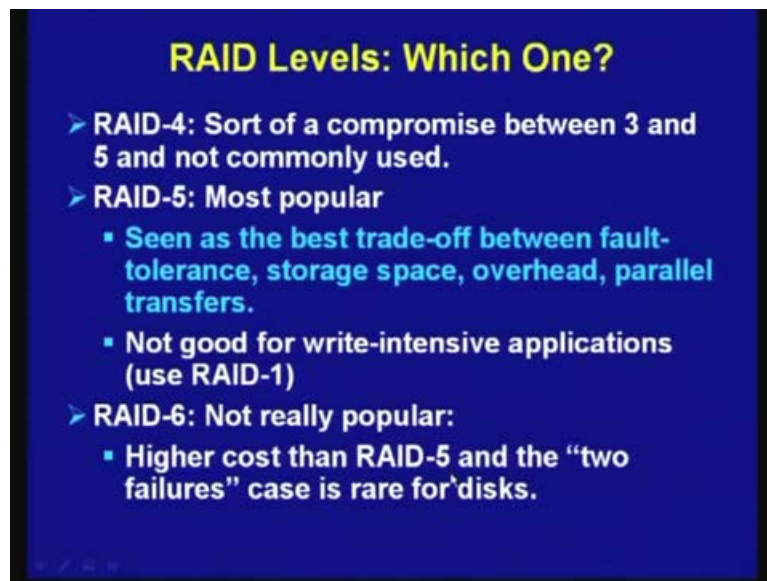
So, now the question is which one to use in what situation, so RAID level 1, this is good when need for high fault tolerance at low overhead, but wastes a lot of storage. So, we have seen, where we are using mirroring or shadowing that thing can be used, when you want very high fault tolerance, and used when parity is deemed insufficient. That means, parity bit the other techniques are based on parity bit, but RAID level 1 is not based on parity, it is based on mirroring or shadowing.

That is why when parity a when parity bit is deemed insufficient, this particular technique is used, and used for small databases or individual users, since RAID 1 controller are cheaper. So, it is simplest and RAID level 2 is not commonly used, as I

have already mentioned, because it is based on error correcting codes, and it is not at all used and RAID level 3 applications with large files, and requiring high transfer performance.

So, this is bit oriented interleaving that is used, when you are dealing with large files and that requires very high performance, so basically this application want RAID 0, but without fault tolerance.

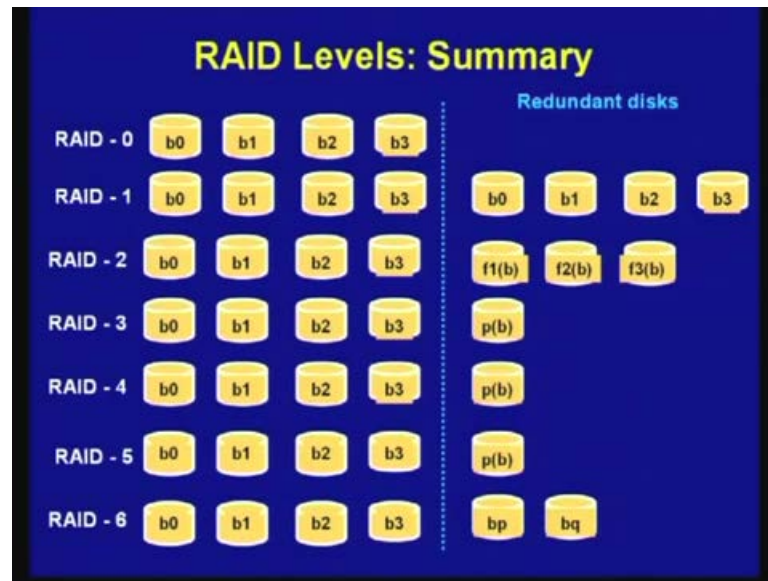
(Refer Slide Time: 52:14)



Then coming to RAID level 4 sort of compromise between RAID level 3, and RAID level 5, and not commonly used. And RAID level 5 is I have already mentioned is the most popular, seen as the best trade off between fault tolerance storage space overhead and parallel transfers. So, this is the most popular one and not good for write intensive applications, because we have seen whenever you are doing writing, then it involves more overhead, and I mean in that case it is better to use RAID level 1, and RAID level 6.

As I have already mentioned is not really popular, and higher cost than RAID level 5, and the two failures case is rare for disks and that is the reason why RAID level 6 is not used.

(Refer Slide Time: 53:03)



So, this gives you a summary of the different redundancy that is being required, RAID level 0 there is no redundancy, RAID level 1 has the largest redundancy, so where you used mirroring or shadowing and RAID level 2, also has got high redundancy. You can for 4 disks 3 additional redundant disk on the other hand, RAID level 3, 4 and 5, all the 3 will require single additional disk to store the parity bits.

So, now, out of the 3 we have seen raid-five is the choice of the day, because of many good features RAID level 6 requires 2 additional disks, one for that that row parity, and another for that diagonal parity. So, one additional disk is required here, which can tolerate more than one I mean tolerance of the double fault; however, it is rarely used and not commonly used, so this summarizes the various RAID levels.

And commercially in all the disk or systems you will find particularly RAID level 5 is commonly used, so with this we have come to the end of storage technology. And in the next lecture we shall discuss about some you know that of the processors, some processors, which are commercially available we shall discuss about them.

Thank you.