

Storage Systems
Dr. K. Gopinath
Department of Computer Science and Engineering
Indian Institute of Science, Bangalore

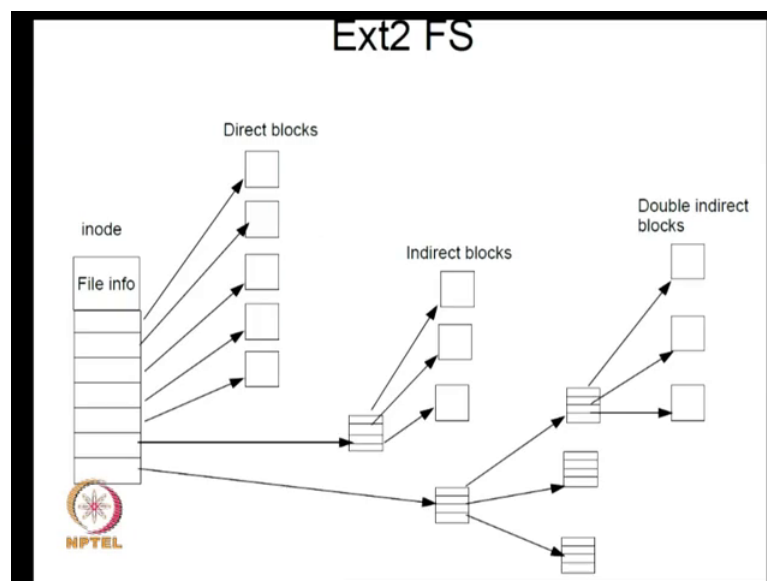
Introduction to Storage Systems

Lecture – 04

**Introduction to Storage Filesystems, Storage Stack, Storage Characteristics,
Storage Performance, An Optimization framework, Storage Protocols, How to
design**

Welcome to the NPTEL course on a storage systems. Last time you are standing to look at some specific systems. I believe I came to this point, I was talking about the file systems that exist in Linux and basically we are talking about some way of locating the blocks on the file.

(Refer Slide Time: 00:34)



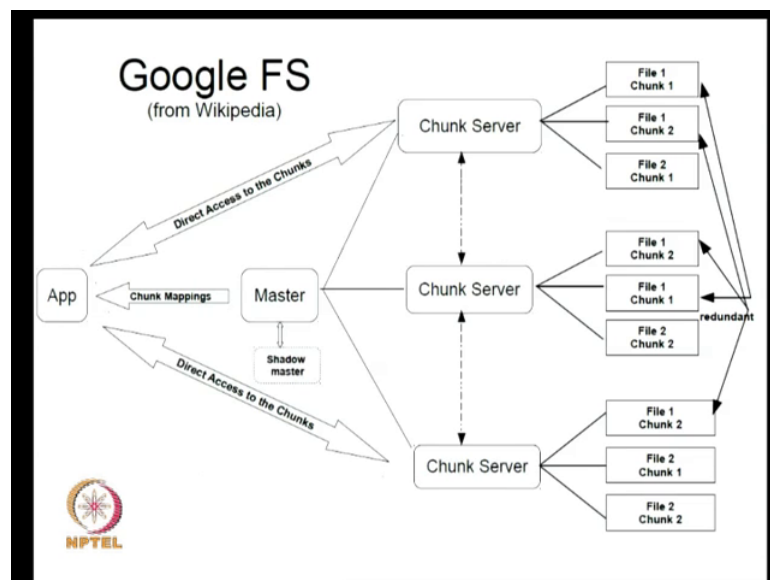
And what we are talking last time was that the size of the files can be varying from small to very large. For example, you can have some files are exactly 0 bytes, 1 byte all the way to terabytes. So, we should be able to scale to all this levels and it turns out that this scaling is achieved in this kind of systems through multiple pointers through a structure like this where there are direct blocks, indirect blocks and double indirect blocks and there are some systems which also have triple indirect blocks so that any one go ahead.

Now, think about this particular file system it is a color based file system it is in the corner that is its not outside its not in the user space; that means, that this kind of systems

are very difficult to linear change and typically these are modified only at with great care because color code is extended to difficult to engineer look at it right. So, therefore, you find that there is more version of a file system, but that happen every. So, often for example, we had Ext2 FS this came in 1992 approximately or 93 somewhere about that time, then your Ext3 about early 2000 I think, then about 2005 or 6 design for Ext 4 which is what is being used right now. So, now, there is something called beta FS which is. Now, also being looked at these things happen every few years.

The next one which I am talking about is what I might call is a user based file system. This is the Google file system.

(Refer Slide Time: 02:42)



Now, what we are talking about here is a system that uses information stored on this side and being accessed through an application. Now, it turns out the Google file system is a user level file system means it is not in the kernel its outside the kernel and actually it uses a Linux file system as its base file system.

So, all this file chunks are actually stored in the Linux file system. A file chunk in Google file system is 64 megabytes right huge one 64 megabytes. So, it has files much much bigger than that. So, just like a block here for example, you had notion of a block a typical block in this kind of systems about 4 kilo bytes. Here in Google file system something like a block I should not call it block thus they called it in the different name that is why we call a chunk what looks like a block in Google file system is a 64

megabyte, chunk is a huge big piece of thing and this has been decided because you want to optimize on the performance of the disk base system. This basically disk base system with respect to the storage information. Why is that? Think about it.

If I am in a disk base system every time I want to access something I have to first do what is called seeking to a particular place on the system, on the disk, on the platter and then later I have to wait for the head to come below the place for the information stored. Now, that that is why there is 2 aspects to it, one is what is called the seek time and then there is something called the rotation time. These 2 things are not easy to decrease because it is a electro mechanical system, it has been for quite some time between 30 milliseconds nowadays I can (Refer Time: 04:54) till about 10 milliseconds, since the last 20 30 years.

It has decreased only from 30 to 20 milliseconds to 10 or 8 milliseconds they combined aspect combined positions. And that means, that the progress in this is very slow compared to the other amazing progressive seen other areas like CPUs disk capacity those things are doubling increasing and almost twice every year or something that we get. Whereas, because the disks being electro mechanical we are not able to make such amazing advances there, so one aspect of disk technology is that it is fundamentally slow; that means, if I want to seek someplace say at 10 milliseconds. Now, just imagine the (Refer Time: 05:41), I want to read one block for kilobytes at one place and then I want to move to another place. So, let us look at.

(Refer Time: 06:02).

Sir desktop.

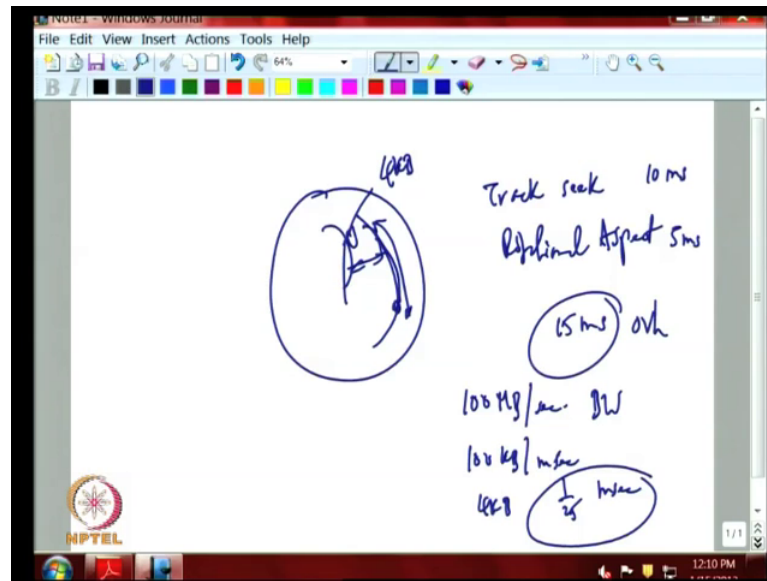
Desktop.

Desktop here (Refer Time: 06:08) forgot where it was.

First one sir, first program window channel.

So, see if you think about it you have a (Refer Time: 06:21).

(Refer Slide Time: 06:25)



Let us say this is a disk, you have a block here and you want to move to the block. So, this could be on one track this could be another track; that means, that I have to first move from this track to this track, so that is the track seek time, followed by I have to move my head from here to it was somewhere here right and it has to move this much this distance. So, there is also here rotational aspect.

Now, track 6 have let us say we will just conservative (Refer Time: 07:05) 10 milliseconds. Rotational aspect full rotation let us say 10 milliseconds typically we will do about half, so its 5 milliseconds, 15 milliseconds total. Now, if you have talking about 4 kilobyte blocks right what is that mean it means that for every 4 kilobytes you have a cost of 15 milliseconds. Now, how much is the time taken for 4 kilobytes? Typical disk now a days you can let us just say be conservative we will say 100 megabyte per second, this is typical bandwidth.

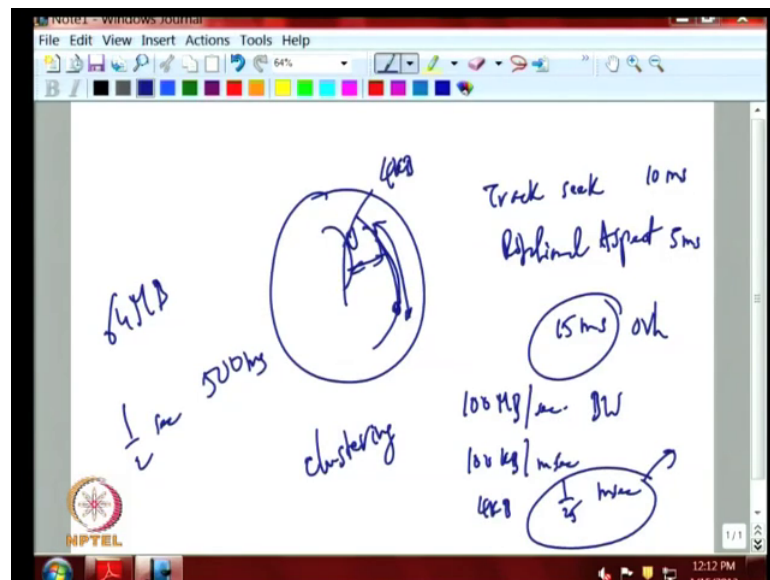
So, if you take 100 megabyte per second bandwidth how much will 4 kilobyte take. So, you can easily see that it is about 100 kilobytes for millisecond. So, we talking about 4 kilobytes till one 25th of a millisecond. So, what we are doing is we are saying that if I want to go from here to here I have an overhead of 15 milliseconds, but actual transfer time is 125 millisecond this totally off it makes no sense, but that is how systems are built right now, that is in why that is a lot of emphasis on all file systems to bunch things

together, to make things as let us say cluster as possible clustering, you want to cluster the blocks its very critical.

So, you can say the 15 milliseconds 125 millisecond these are useful activities overhead, we are talking about a utilization of barely less than percent if you do a seek every time. So, a critical thing about a disk is that if you want to really get any performance we have to avoid 6 as much as possible if you able to do a work 10 20 30 blocks sequentially before you again do a second.

Now, this has been carried to much bigger levels in the Google file systems there this 64 megabytes.

(Refer Slide Time: 09:08)



Now, just see what is 64 megabytes you can see its approximately about half a second, it is like more a half a second. So, we are talking about half a second which is about 500 milliseconds of data time versus 15 milliseconds overhead. That means that they have to do one critical thing, what should the Google file system do? They should put lot of things together in one place in 64 megabytes you have to put lots of things in it. So, when doing a search they have all kinds of information about various web pages they are going to put all of the web pages and put them into 64 megabyte chunks because they talking about a very big system some 100s of petabytes or 10s of petabytes. So, for them they have that information they try to packet in and there is some other index structures you have to build from top of it, it should locate those web pages within that 64

megabyte chunk and that is being done by some other system the file system gives you the ability to pull it out from the disk and write it back etcetera. But there has to be another system which actually looks into finer let us say (Refer Time: 10:16) inside this 64 megabyte chunks and that is what is called big table which is I think we will come to that also later.

So, that two systems are operation. So, basically these things are 64 megabyte chunks and they are basically being serve through chunks available; that means, this is Linux file system here. Chunks are what is the server on the Linux system, which knows about the locations of these chunks, so there is one Linux system here is one more probably another Linux system it is mentioned they are quitting chunks. And you notice also something we have written redundant something what happens is the file 1 chunk 1 is also available here also in some other Linux server as another copy. So, there could be 2 copies or 3 copies, typically 3 copies are kept. So, that even if this particular system dies the other things are available.

The reason why they have to do it is because at the scale at which you are doing it you will find that failures are very common. So, what you should do is you should design a system which is somewhat tolerant to failure that is the critical aspect to the system. Again why is that the case?

(Refer Slide Time: 11:50)

The image shows a screenshot of a Windows Notepad application window. The window title is "Notepad - Windows Journal". The menu bar includes "File", "Edit", "View", "Insert", "Actions", "Tools", and "Help". The toolbar shows various drawing tools and a 64% zoom level. The main text area contains handwritten notes in blue ink:

- 10^{-20}
- 10^{-5} BER
- CERN
- $10^5 \rightarrow 10^6 X$
- 10^4 Bytes
- 10^{12}
- end-to-end guarantees ECC
- NPTEL logo

A small inset image of a man in a green shirt is visible in the bottom right corner of the Notepad window.

It is because if you once again let us think about that part again. So, normally it turns out you can a disk has got a bit error rate of 10 to the power of minus 15. What is it 10 to the power of minus 15 bit error rate; that means, that if you in spite of all the coding that is done reed Solomon coding, all those sophisticated error coding, techniques are used in spite of that there is a residual error bits always there that is 10 to the power of minus 15 bit error rate. What is that mean? It means the 10 to the power of 15 bits will have one bit wrong one bit is wrong whatever you do that is what the disk will give you the disk specification will give you, 10 to the power of minus 15 bit error rate.

So, now, how much is 10 to the power 15 bits? It is about 10 to the power of 14 bytes that is about 100 gigabytes sorry terabytes, there is a mistake terabytes right 100 terabytes. So, if you are talking about a system which is about 10s of petabytes; that means, that within that system there are at least about 100 or more bit errors bit flips some were there out we are not able to figure out these have these have bits in error which you can cannot figure out nothing whatever you do you cannot figure out there in whether they are or not because the disk has no where finding out.

So, these are somewhat dangerous errors because there is no way to figure out this. So, you have to build some additional things on top of it and that you can do by having some other extra error coding techniques on top of it. So, basically it is not only because of the disks. It is also the case that you will have errors when transmitting it from multiple places because here as mentioned the servers out here you are transmitting it through some network, they are going through some memory right.

This memory also can do bit less you normally is what is called ECC, this is error correction codes even this once their bit error rate is somewhere about 10 to the power of minus 12 or something like that or minus 13 something like that; that means, that even memory can get corrupted and there is no way the ECC also will not able to tell you the some things wrong with it. That means, that there is a problem here itself. So, there is a problems from the disk side, problems on memory side, the problems on network side. So, if basically you want what is called end to end guarantees, end to end guarantees; that means, I need to able to say I am sending something from this side in spite of whatever is there in between I should able to get it in tattoo on to the side, how do we do that that is the big issue.

So, one way to do it is in addition to the coding that is done I should mention that there is coding everywhere. Now, memory already does some coding ECC memory is there one servers enormous almost always use ECC memory. Networking also has some coding for example, Ethernet is very good they have very good coding for example, gigabyte Ethernet their error bit, bit error rate theoretical is somewhere about 10 to the power minus 20 its very good. So, actually the medium will not do any problems, but when it comes to when you attach two things sometimes there are problems.

I think some of you know that electrical connections are sometimes not very good right some diodes can be found sometimes, if you have studied any electrical engineering you will notice, sometimes connections will not made well. So, that is why even though the medium itself is very good they could be other some somewhere else when it comes to electrical contact.

Similarly the disk have a problem, they are all engineered extremely well. So, well that most of us do not know that this kind of problems will exist, but in random laptop the amount of storage that you have is sufficiently small that probably during the lifetime of the laptop you might see only 1 or 2 difference that is all. And that you will result in some random hanging of the machine or some crazy things. Hopefully it does not spoil your thesis whatever writing. Mostly does not do this kind of things basically because the amount of stuff you write and I write is so small compared to what systems are there. I think last time I talked about it even if you spend all our time typing right barely you can come to megabytes right. So, what is there on your laptop is mostly other stuff other people have created two programs your browser which is about my on my laptop my browser takes as much as something like 2 gigabytes.

I did not write it right, there was a program written on that was collecting all kind of information that was browsing things (Refer Time: 17:10) such 2 gigabytes. So, even in that larger number like 2 gigabytes the chances of the errors small that is why usually it will not hit your critical information whatever your address whatever your written most likely it was because we are talking about a megabyte kind of things versus gigabytes. So, chances of you are hitting that error is 1 into 10 to the power of 3 or less. So, you will not normally see those things.

If hits its go and hit some system program typically the big browsers is I told you 2 gigabyte Firefox. So, if there is a byte it will hit that part and it will randomly crash, you must have seen some crashes on your systems sometimes of course, you cannot blame all these guess for you it could be software error also, but normally it attacks those things. So, there are errors all through the system, luckily there are very small, but the minute you go to large scale systems like Google file systems or remember that in there is something called c r n right where they were trying to find out some new bosons right you must heard about this one. They essentially I do not remember the exact details, but they collect a petabyte per day or something (Refer Time: 18:27) or 10 petabytes per day I can call the exact number. So, that is somewhat information that comes in.

For those guys that information collecting every day will have about at least 100 of 100 or 200, 1000 bits flipped. They have to something serious. How do they do it? On top of the things they try to do this end to end guarantee. How do you it? At the time it is generated you put one more error coding on top of it and then when you get on the other side somehow you have to make sure that that error code will tell you whether it was transmitted from this side to that side intact. So, a big subject on some of you may be interested in this area, but there is a big area which about coding, coding theory which will help you to understand this parts. So, you need also the overall coding that happens if you want to give end to end guarantees.

Now, what these guys are doing Google kind of systems. They are essentially saying that if there are any undercover able bit errors it is being handle through copies there are going to take coding on top of this, but if you are talking about newer systems they might also involve coding on top of it. So, this I have basically Linux servers, this file chunks are sitting on Linux file systems most were Ext3 file systems and the application what is the application like as a Google it is going to be a search application. And what it does is it talks to a master what is a master, master keeps track of the meta data it knows for example, which chunks has over got which file chunks. So, what happens is that this master has all this information sitting in memory it is not there in disk everything is memory based here everything and it has got a shadow copy just in case something happens to the master.

It is a single master system, so this is single point of failure, but because this is a shadow master it can taken if some bad things happens to this guy. You do not have a system

which is got multiple masters at the same time because that design is little more complicated we will talk about these things later. Having multiple master at the same time this is slightly more difficult design, it has to be handled you know advanced systems do that do that part, but in this particular Google file system of 2003 they have only a single master and they have a shadow master just in case this master phase, it will come in only when the master phase otherwise it is a passive (Refer Time: 21:05).

There are now systems with active mass space which can distribute a load also and this master has a chunk mappings. So, when application want some details it tells or ask the master tell me give me the locations of where this things are that is what the chunks maps are. Then application does not have to go through master again it can directly access the chunks it knows that from the information it got from the master it knows that this file chunk which I am interested is there in chunks about 2 7 and 8 at that offset whatever it is, and then it talks to a chunks server. Chunks server user space program it goes talks to the file system u s d 3 file system and says give me that file which you are stored chunk 1 file 1. So, that might not be files in the here just saying f 1 c 1, f 1 c 2, f 2 c 1 etcetera some means over that kind. So, if the chunks are just give me f 1 c 2 from Ext3 file system, something like that.

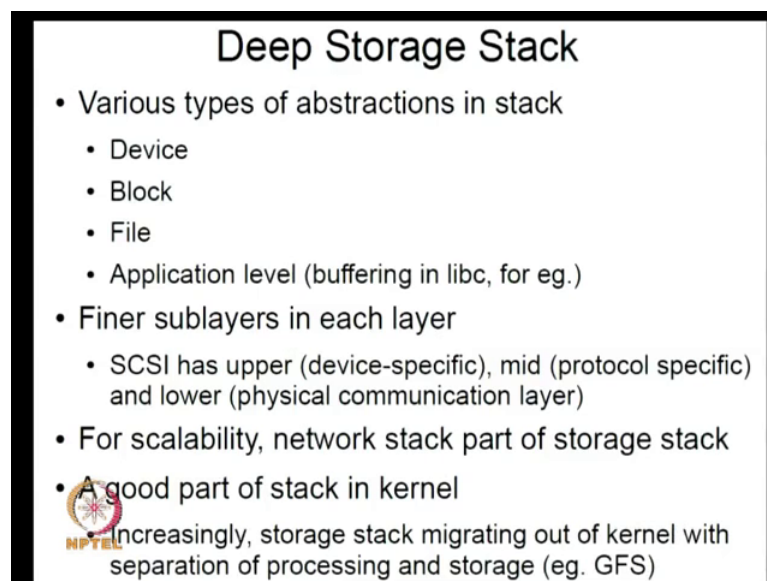
And then those things are again send back it does not go through the master again. So, they goes to the master only at the time when as when it is seeing something for the first time or otherwise this guy will crash things etcetera. Most of the attempt is to make sure that this things are put up in large sizes kept in memory so that you do not have to suffer these 15 millisecond delays this seek times. And the master index is completely in memory so that it is almost instantaneous. There is reason why we are able to see so many searches coming out very quickly its possible because most of it is memory based. So, this an example of a user based file system on top of a base file system, the base file system being Linux system. This is the different set of constraints compared to the kind of constraints that a kernel based file system will have.

So, we will look at some of these things in more detail. I think at this point I just wanted to give you two examples one is kernel based kernel based file system and user based file system and that design aspect are quite different. The concurrency what you want can handle in the kernel is different in the concurrent that is possible in a user based file system. For example, the concurrency here can be depends just on how many chunks are

possible whereas, in a single you know kernel base file system you cannot go beyond a certain a number of kernel because it starts eating up memory it becomes slightly more difficult to scale. This is much more scalable. But the scalability is limited by the master cells the master because everything is in memory this part of it decides how big the systems can be because everything all metadata is stored in memory. So, this can have about let us say in those days beginning that some you can guess as some approximately 100s of gigabytes in those days 2003 approximately. Now a days it will be terabytes of memory.

Some people are talking about even if you talk to high performance computing areas they will talk about (Refer Time: 24:09) petabytes of memory, but I just want to mention that there are some scalability issues in every of the designs that will decide exactly how far it can go.

(Refer Slide Time: 24:20)



Deep Storage Stack

- Various types of abstractions in stack
 - Device
 - Block
 - File
 - Application level (buffering in libc, for eg.)
- Finer sublayers in each layer
 - SCSI has upper (device-specific), mid (protocol specific) and lower (physical communication layer)
- For scalability, network stack part of storage stack
- A good part of stack in kernel
 - Increasingly, storage stack migrating out of kernel with separation of processing and storage (eg. GFS)

So, one thing you should you know think about all things is that there is a deep layering in the system. So, as I mentioned already we have already mentioned that the Google file system is sitting in top of Linux file system that is one example of layering. But if you think more carefully it turns out the layering is much much more extensive. For example, there are various steps obstructions on this track, when I talk about stack I am talking about the software that is learning all this things, this are device specific stack, the block specific stack, file specific stack, application level stack.

So, (Refer Time: 25:08) can be what is called a SCSI disk and it turns out SCSI has itself multiple layers in it. For example, SCSI as what is called upper layer mid level and lower level. Now, there is a why it is like this because SCSI itself is a very generic protocol. I think you might be aware of this because now a days you get USB sticks right not only can use it to store memory it can be used for networking right, you must have come across this internet access things which also happen through USB. So, USB is a specific hardware format for interconnection for (Refer Time: 26:01) interconnection, for giving some service. Your camera for example, has some way in which (Refer Time: 26:06) devices right. It has got some connector by which it can convert whatever you want to transfer there into USB sockets right.

So, you will see that USB has ability to interconnect lots of things not just storage it can be the communication it can to other things printers etcetera right. That means, that this USB is so let us call it flexible it can talk to storage devices, it can talk to network devices, it can talk to printing devices, whatever; that means, at somebody has to mixes up all these things. How is it done? It is done by layering system and the layering system first says the lower part of it is strictly what is say connected with sending bit patterns zeros and ones, always giving a one type of layering its going on USB devices.

We will come to that later in more detail, but lower level at the lowest level will in that on the stack will be it will be just communication of bits. On top of it you need to have some other things why because when you are for example, you already put your USB stick you know in a laptop right, you want to say I want to write to it right I do not write to it that is the spirit CPU can give it the spirit which it can be put into the magnet media, if it is magnetic media or if it is solid state device at it has got a particular spirit to it can accept it the information. That means, that the CPU cannot just say I just through that you to take it right because other part is to do willing to accept it or particular or whatever create its capable of or you imagine a printer also right. Printers you know that it is slow compared to electronic devices like memory sticks; that means, the CPU also has to slow down with respect to slow devices. Somebody has to do this part that is sort of a protocol specific thing they are talk to each other.

One guy sending bits are other guy on top has to watch and see at what rate has to be sent and one can say I finish writing it you can. Now, send me the next one these are all the protocol specific what is called there is the way in which 2 guys are communicating

they are talking to each other and saying yes I am done. Now, you send the next one always of that kind or it could also be something like I am a tape, I can (Refer Time: 28:42) guy please do rewind go to the beginning right, somebody has to send this that is also the protocol specific act right. So, it turns out there are device specific things like for example, tapes you know they have device specific things there is this protocol specific things, sorry.

There are communication layer specific things; USB has similar things SCSI also similar things. SCSI is basically students protocol USB is slightly more general protocol it is not (Refer Time: 29:15) gate for storage, but SCSI is very much gate for storage. We will discuss why SCSI has a certain storage aspect with it. What do you mean by single storage specific aspect? Basically what we are talking about it what mean by that is that you know the storage is slow, when you dealing slow devices you might have a different way of dealing with than if a dealing with fast devices. Good example is suppose I want to seek to a place it is going to take 15 milliseconds in (Refer Time: 29:52), but 15 milliseconds you can say something like almost like to (Refer Time: 29:59) phrase because what is the side what is the unit of time for CPU it is about nanoseconds. What is the nanoseconds on milliseconds? It is a factor of 10 to the power of 5 or 10 to the power of approximately 10 to the power 5.

If you think about it, it is like 1 second versus 4 to 5 6 8 weeks; that means, that the CPU when it makes a request it can take something like such a large time, the best thing is to let the device tell you when it is done. I cannot be keeping on checking whether the guy is done or not I initiate it and then I let, let us slow device whatever time it want to do it when it finish will be it, it is job to do (Refer Time: 30:47). I will not the pulling it I will not may keeping on checking you I will be done. That means, that suppose it is a bus all these devices are sitting on it CPU, slow devices all those can sitting on this. Normally the fast guy usually the bus master that guy decides when you can use it, but in this case of SCSI protocol because you know it is so slow it also can slow device also can become bus master because only it knows when it is done otherwise you have to keep on pulling it.

So, you will notice that there are some specify protocols not protocol device specific aspects in the protocols. So, such as mentioned about block devices I think we looked at this previous system, this is an example at which there is some protocol between chunk

servers, application, master there is some protocol here also. Not only at the device level it is also at the user level there is also another protocol going around. There are different things there are different systems this is a Linux system 1, Linux system 2, Linux system 7 applications when I somewhere else master is somewhere else all this things are this protocol going on.

Now, what is more interesting also is that you might heard about the network stack right, you heard about this 7 layer stack etcetera it turns out storage actually uses because you look at it there is some networking going here also. That means, if you look at the whole thing not only is the storage track there is also in a network stack also as part of your system and there are very specific network stacks for storage. For example, there is something called fiber channel which is a storage which is a stack specifically for transferring large amount say data, you know doing backup etcetera it turns out that the amount of stuff that you have to send its so huge, it make sense to have a specific protocol which is gate for storage only.


So, there is we have in addition to various things we also has the storage (Refer Time: 33:07) our stack is can be as many as 10 to 15 layers deep that many we can do that. Though thing which is interesting for us is why storage system has been part of the operating system for a long time has been that a good part of the stack is in the kernel. I think all of you know about the TCP IP processing happens in the kernel, some of this file systems things here all happen in the kernel right. So, good portion of the kernel good portion of the storage stack is the kernel, but because of the difficulty of working in the kernel and the newer all kinds anythings are happening at all the time people want to have a slightly easier time working in the kernel is hard. So, only for real professionals that is it.

So, people are trying to figure out that is easier ways of doing it that way is the one way to its doing it by writing sitting on top of it like the Google file system and this will be done through probably who knows pythons, scripts whatever they can do all kind of things (Refer Time: 34:15) here, on top of it. So, the various languages available. Now, you can do it java, you can do it to pythons, script, whatever, which actually coordinates across these things. So, this is slightly more easier this storage stack migrating kernel is separation of processing a storage its makes slightly easier.

(Refer Slide Time: 34:37)

Storage Characteristics

- Concurrency arises naturally
 - Wide disparity in speeds of memory and storage
 - Have to mask slowness of storage
 - Make processing and storage go at their own rates and use interrupts (or sometimes polling) to signal completion of slow storage operations
 - Historic reason why operating systems developed
- Storage has to be typically persistent over time
 - Amount increases typically with time
 - But not all imp over time; keep imp part in fast storage?
- “Caching” and “tiering” arise naturally
 - Have to choose 2 of 3: speed, capacity or cost



One thing about storage also is that by definition concurrency some most by definition r s s natural, why is that because you dealing with the slow things. When it dealing the slow things what is the idea will let all the slow things you initiate all the activities must slow devices and then you do other things right.

So, because you cannot often to wait for them if you wait for them; that means, the struck we cannot do anything this way. So, what is the idea? The idea is basically make sure we have large inverse of things you can initiate in parallel with other thing I will do right. So, waiting for something we want to do some useful, because of wide disparity in speeds of memory and storage we have to mass slowness of storage and the only way it is do it is by paralology concurrency. Make processing and storage go at their own rates and use interrupts or sometimes polling to signal completion of slow storage operations.

Actually interrupt versions were device in 1957 for the exactly the same this the only reason by we have operating system is because of this difference in speeds the first interrupt base mechanism is invented by (Refer Time: 35:58) in 1957, that is why here actually started operating system guy. And finally, we moved off from operating system to theory of concurrency, how to write programs because once you have concurrency it turns out very difficult reasons about it. Operating system is all about reasoning about concurrency good part of operating systems. Whole of concurrency theory come from this issue about matching disparity in speeds of memory and storage.

So, this is historic reason why operating system developed basically because of this part. If everything is working extremely fast this know it for you to let us say think about concurrency that much only one because something so disparate then you have to something serious. The other thing about, one is concurrency is important; that means, that your file systems are example are engineered at extremely careful carefully so that you are as concurrent as possible because you are trying to be as concurrent as possible it is very difficult to get the product working right.

So, that is why on my people do not usually write file systems of (Refer Time: 37:12) typically they want to avoid it as far as possible because it is very difficult to debug it. Reason why it is few more difficulties because if you want to debug it you have to be your own (Refer Time: 37:24), it has to work for you before you can tell other (Refer Time: 37:28) and while a debugging it, it will be hearing experience nothing will work. So, that is why because why file systems or storage systems change slowly they do not change very fast because you have to guarantee that the data you write is actually there when you want to access it should be there. If you were not able to guarantee even that minimum thing it looks like a small thing, but actually it is a very tough thing. You cannot even get start it because you are going to store your, if you are developing system you have to depend on it you have to use it yourself.

As I said there is a famous word in we have to eat around dog food that is why I call it there is the principle. You create something, you better use it before another is start using bit, you cannot unless you are able to design systems which are bullet proof right you can ask any bullet us to it use it. So, we have to be your own we have to suffer all the consequences of your program not working. And the other thing is that storage has to be typically persistent over time what is that means, it means that you write something you expect it to be access it later.

We already discussed it sometime back we are not talking about persistence over centuries even ability to persist over a year (Refer Time: 38:42) this are told you last time about CDs they go bad within 1 year right because that is mentioned what is the reason it is because of dies that are used chemical dies that are used which have degrading over time and finally, you cannot even read them right you need to get to much much better dries then you can talk about 10 years, series for same series can do it.

And tapes for example, there been categorized that there server 50 years that is why anybody serious about storing anything usually stores on a tape they do not even trust disks they do not know what when it will go back. But same reason why if you are you heard of all about digital effects and digital movies and everything right, but surprising or interesting to for all some of you to know is that the master copies of this things are still stored in unlock form in some deep case somewhere. You do all those things, but finally, you have to keep it because nobody knows how to till today nobody knows how to take your bits and make it survey for 100 years.

For example, they just to be let them for studies they find that cost of storing it analogue, if fraction of the cost of storing a digital because digital means that you have to have a server which is running, every time a new operating system comes in, every time a new disk comes, in every time a new technology comes in you have to transfer the whole thing to that side somebody has to be on a job all the time watching that I have just stuff here if I do not transfer from this side to this side from this floppy disk to this new amazing new contraption that has come now. If I do not track the technology as it happens one day it will become bits you cannot read. So, people have founded the cost of storing it analogue is fractionth of a cost of digital system. But digital is very good from the act accuracy point of view. So, they want to keep both certain keep analogue must convert somehow from the digital to analogue keep that as this one, but also try to do this also keep on transferring from one generation to next generation. It is a endless thing, we have to keep on doing it otherwise you lose it.

So, amount increases typically with time that all are I think all of us familiar with right you had extra 64 megabyte memory stick I though as big. Now, I carry around it gigabyte of memory strike, but I am pretty sure many of you have 64 gigabyte memory stick with you some of you might have 64 or 32 gigabytes I do not have it I am not at got. So, this increase on the time. That means, that as you keep adding more things most of the stuff is often useless often that also another fact, not important over time something that (Refer Time: 41:43). So, the trick is somewhat figure out which is useful stuff to keep and because of the amount is increasing its too difficult to even figure out what I have to keep or sometime people just give up, I cannot figure out what is important very important right. Same thing that happens right in you have lot of papers you know what a through or what a keep.

So, because of these there is an interesting thing that since not everything is important can I keep some important part in fast storage and keep the unused part the slow storage in the natural thing. So, concurrence is natural keeping it in different types of storage is natural in large storage systems. So, that is a caching and tiering arise naturally. Caching means you keep it in faster storage, tiering also same thing same but you might have in different types of storage. For example, I might say that if not everything is important over time I will tell myself the following I will keep it in less reliable storage what is an important. I will keep it in more reliable storage what is important.

One way to do it is I have older disks which were done in a older technology which are do not I am not as trusting of that I know it will fall apart. I will not keep most important stuff on that old disks I keep between the new disks or it could be something else also. Nowadays you know we have something called solid state disks people are not sure about how good this solid state this is our predictor or like or CDs also nobody knows whether it will be last one here or to use.

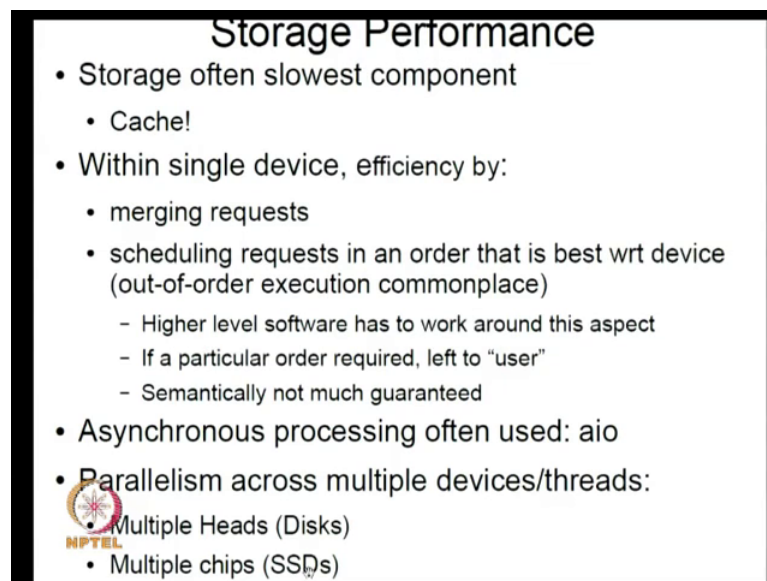
So, even that may be it may be newer technology, but if I am not sure about it I want to keep my stuff in the reliable part of the storage system. So, I have tiers, it could be reliability tier based on reliability considerations it could be performance tiers based on performance characteristics it could also be security considerations right. I cannot tier it to all levels or multiple levels all these things are multiple possibilities are there, I can have a cross product of these possibilities.

But interesting is also is that typically you have to choose between 2 out of 3 things you cannot get all 3 at the same time. If you want to get all 3 you have to spend any in cent amount of money. You either decide to choose speed or capacity or cost. For example, let us take the case of speed and capacity I can always have 10 peta bytes of memory right it is got capacity it is got speed, but the cost will be (Refer Time: 44:32) I will be bankrupt or I can have capacity of cost, I can have easily nowadays 3 terabyte disk cost also is cheap but I will not get speed, I can have speed and cost but not capacity.

I can have for example, I can even notice the registers right CPU registers right, I get 64 registers fine fantastic right are extremely fast I can access it every fractions of a nanosecond speed is there cost is also I can effort 64 bit, but capacity is not there, I just have 64 registers of 64 bytes, 64 bits. So, you will find that this invariably in the case


speed capacity at cost, that is a similar thing you can also talk about convenience and security. If you wanted convenient typically it has to be slightly relaxed with respect to security, you want to be really secure then you have it has to got an inconvenient like going to an airport and remove your shoes and taking your laptop all that (Refer Time: 45:40) here right this is the same thing. Here also seems to do if you want to keep something secure things by definition are going to do non (Refer Time: 45:48).

(Refer Slide Time: 45:50)



Storage Performance

- Storage often slowest component
 - Cache!
- Within single device, efficiency by:
 - merging requests
 - scheduling requests in an order that is best wrt device (out-of-order execution commonplace)
 - Higher level software has to work around this aspect
 - If a particular order required, left to "user"
 - Semantically not much guaranteed
- Asynchronous processing often used: aio
- Parallelism across multiple devices/threads:
 - Multiple Heads (Disks)
 - Multiple chips (SSDs)

 NPTEL

So, storage performance is often the slowest is often, storage often is slowest component that is why I have to do caching these are the single device efficiency is by merging requests as I have told you that 4 kilobytes per kg is too small if you want to go to larger sizes to use the bandwidth of the disks. We have to merge request, you want to scheduling requests in an order that is best with respect to device that is called out of order execution is common place. You might not request come in you may want to execute them not even often times not in the same order, it is helpful if you do it in same order because that is called a FIFO order, but often times it turns out that you may want to do things out of order.

So, higher level software has to work around. Suppose basically in some sense lower level guys will be ordering reordering things and the upper guy has to figure out is it creating any complications. A good example is when a disk for example, if there is a piece of data here and nearby there is a piece of data here, but my request requires this

data plus this data here, but under a request has come only for this, this is made to the previous request. So, should I serve this guy go for seek and come back here I have should serve this guy and even it will do this guy even this guy came later right. So, that is where out of order execution comes in. You also have what is called if a particular order required left to user.

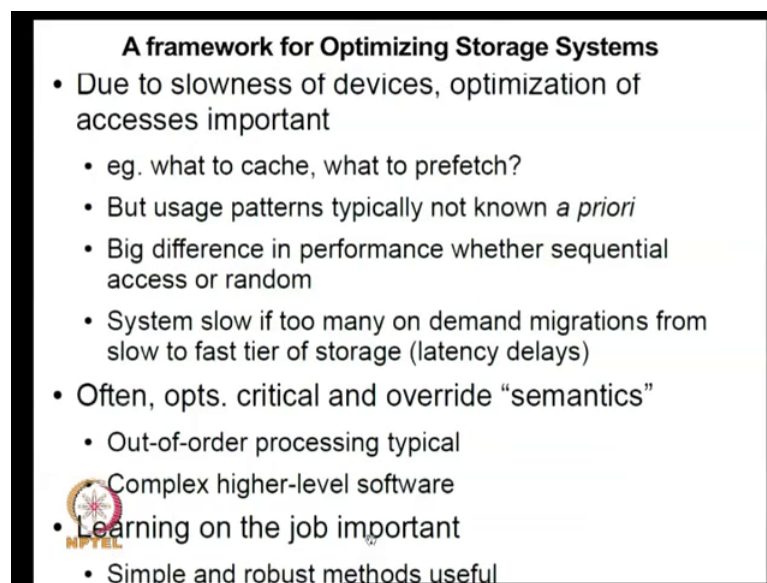
So, basically you have to tell the user you are going to suffer it if you want a particular order because I am going to do go from here sit here, come back go here come back go here. So, I am just going to kill you with 15 milliseconds (Refer Time: 47:43) every single time 4 kilobytes. Do you want such a bad performance; if you wanted I will give it to you. Whereas if you do not take anything system assumes that you are not going to tolerate such slope behavior I will do the best in terms of out of order execution which is good for you.

While keeping myself same if the file system it has to have its metadata intact. So, it turns out for updating its own metadata it will actually do it in the order its required it will not do out of order if it is going to create problems for itself. So, for sanitize the system it will do it might do strict order, but if it is not told anything about what you are doing it will do what it thinks it is good with respect to out of order. So, semantically not much guaranteed basically because it is going to do the out of order if you do not tell it anything, sometimes upper level guys they might think something is performance, but actually it is getting reorder. So, you have to be careful about this similarly asynchronous processing is often used.

For example in Linux systems in the recent past not last 5 years on them there is something called asynchronous as a frame work; that means, that you can initiate a request and then the kernel goes in this other things and later when it is completed it gets and entrapped. And basically a kernel threads are used in the kernel so that they are watching what is called when the things finish who has to process it once the process once that particular IO is completed. So, there is lot of asynchronous aspect that goes on here. You can have parallelism across multiple devices on threads, a multiples heads multiple chips this solid state disks have multiple chips for example, the basic size will be 8 gigabits you put multiple 8 gigabits and all of them can be access at the same time.

So, you can have multiple heads; that means, you have multiple disks for example, you can have in high performance response systems you have something like 48 disks, 1000 disk, 8000 disk you can have (Refer Time: 49:55) things forgot to tell. So, all the disks that are working at the same time you have parallel file system it can go and try to use all the disks at the same time. It is difficult to exercise all the parallel that is available, but that sort of parallel file system constituted. With the SSDS people have also now, beginning to figure out how to use all the concurrence that is available. So, that also is (Refer Time: 50:14) on the (Refer Time: 50:15).

(Refer Slide Time: 50:15)



A framework for Optimizing Storage Systems

- Due to slowness of devices, optimization of accesses important
 - eg. what to cache, what to prefetch?
 - But usage patterns typically not known *a priori*
 - Big difference in performance whether sequential access or random
 - System slow if too many on demand migrations from slow to fast tier of storage (latency delays)
- Often, opts. critical and override “semantics”
 - Out-of-order processing typical
 - Complex higher-level software
- Learning on the job important
 - Simple and robust methods useful

So, due to slowness devices optimization of access is very important what to cache, what to prefetch. Caching is demand base, prefetch means you get it before it actually requested. So, that you somehow guess where is this is going to be required print to prefetch. The big problem is usage patterns typically not known a priori, nobody tells you what are going to be excessive for example, you might be using Firefox 1 minute suddenly you start the Skype session suddenly you want to do some terminal session or whatever right and it is going to have a different types of storage access is depending on what you have to do, nobody tells you that you are not going to tell the operating system what is going to happen right it has to figure out based on your access is you have to figure out what you are doing.

So, let us say there is a big difference in performance whether sequential access or random I already mention to you if you keep pulling of 4 kilo bytes followed by seek, your performance really is going to be pretty full. So, if it is sequential somehow you can guaranteed sequential then it is much better. So, whole operating system and slowed system they sit and figure out how to make sure that you are pulling out 100s of blocks in one short it is very big issue in operating system with design slash total system design.


Again similar to that if you are doing tiering a system is slow if too many on demand migrations from slow to fast tier of storage that determine latency delays. Now, for optimizations critical and override semantics I have already mentioned this out of order processing typical that is why we have complex higher level software and for some reason why databases and file systems are slightly complex things may not that simple to design once it is designed people do not tend to change it.

So, typically because of the complexity simple or robust methods are useful because they there is a certain let us say just avoid the complexity you find that this is something useful.

(Refer Slide Time: 52:24)

Storage Protocols

- Interrupt driven rather than wait/poll
 - On completion, interrupt CPU or HBA
 - To avoid interrupt overhead, HBA or similar agents
 - Helps Segmentation and Reassembly (SAR)
- Split-phase transactions common
 - for eg: on completion of (a long) seek, slave takes bus
- Protocol endpoints preferably “virtualizable”
 - SCSI devices can be on an electrical bus, network or Internet if physical layer handled correctly
 - Protocols survive much longer
 - Devices can have arbitrary structure as long as they speak SCSI protocol
 - Even big servers!



I think we will just mention a few things about protocols already I mentioned that there could be protocols in the device level it could be at the other levels of the stack like, for example, at the intermediate levels or at top levels.

So, example a typically storage protocols we are interrupt driven rather than wait poll on completion interrupt CPU or a CPUs agent. A CPUs agent is typically in storage system called host bus adaptor. A host bus adaptor is an agent of the CPU which instead of the CPU wait let us say processing interrupts HBA addressing for example, let us say the CPU wants a buffer of 16 kilobytes then without the HBA a CPU will interrupt it 4 times for every kilobyte transfer. So, to avoid the CPU can interrupt a so often they are host bus adaptor which is another small little system of its own it is another CPU what call it a some kind of a small system. That this subsystem will interrupt the HBA, HBA will take each of this 4 blocks that I have been write put them together what is called resembling, in networking you might have heard something called segmentation and resembling SAR right. So, this HBA basically does the SAR for storage devices. So, you have HBA or similar agents, helps segmentation and reassembly.

HBA to other things also we will not get into it, but it one of the things it does this what is called split phase transactions should also common, what are split phase transactions? Instead of doing a send something you return found back right, you send something and you essentially of the transaction of the sometime and later sometime get again you initiate this transaction. It is not accessed when I send it and waiting for the other get to respond I send it I do other things I just completely loose track off the beginning part of it I just do not do other things, sometime later I will be told that you did not complete transaction you better finish it now, and completion of a long slave takes the bus.

Other inclination think of a step protocols is protocol endpoint preferably virtualizable. Now, what is that meaning? You can see the explosive size of this systems when a first tracked teaching in this institute the size of disk was (Refer Time: 55:18) we talk of 4 megabytes. I remember in 1995 or 96, I bought the biggest disasters available cheap that 495 megabytes, 495 megabytes.

Now, think about it 490 megabytes how much it is close to a gigabyte gigabit that is about 31 bits, you want to prefer to each bit the required 31 bits or 30 bits or whatever right. Now, 32 bits is not enough you have to hold 64 bits; that means, that everything in the system is changing with respect to sizes, beginning we started with 16 bits or not enough you went to 32 bits that is not enough maybe want to 48 bits before you go 64 bits also.

Now, what it means is that if you have any specificity with respect to each of those end points what this guy requires what that guy requires you can do problem, that you can also see in the case of 2 terabytes and 3 terabytes. Between 2 terabytes and 3 terabytes this is a problem there is some size limit that is coming in, in the bias of that or in some lower level part of the subsystem that is why we will find that 2 terabytes are acceptable in some systems is the trouble. The many a triple 3 terabyte system that guy contricate there is a 3 terabytes it will give you some little only give you some 900 megabyte gigabytes something all right because it is chopping of it is being some truncating of the higher bits or something all that kind.

So, what is important is that since evolution of this devices are taking place all the time you wanted to be what is called virtualizable; that means, that you do not wanted to be a hard let us say numbers which are fixed once more of. You wanted to be able to do something which depending a situation we can change certain things. So, for example, SCSI devices right originally start an electrical bus, now they are networkable. These are something called SCSI devices, SCSI access, SCSI based storage systems.

Nowadays you can even access a disk through internet, there is something called SCSI these are disk sitting out there I can send commands through internet to talk that disk there is called SCSI. So, the thing is it could be an electrical bus it could be network internet etcetera. So, what is important is that somehow you want to make sure that the aspect relating to storage and access that should be independent of which way I actually access it whether I have accessed the electrical bus network (Refer Time: 58:08) in the sense that sort of inverting it has in virtualizable it should be designed in such a way in a client server kind of model, let us call it client server model which somehow abstracts away from these issues which are able to do a successful abstraction the same SCSI protocols will keep revealing off.

For example, the SCSI devices came in 1980 it came in 82-83 somewhere about that time some new versions came out, the SCSI to came in about I think mid 90s, SCSI 3 came in early 2000 in spite of all these changes in spite of the split changes in fact, all those things in terms of the SCSI protocol not change mostly minor additions have been made here under because it is robust with respect to changes in the device space. That is one of the things thought of.

Devices can have arbitrary structure as long as this peaks SCSI protocol. Actually what interesting is that we will find that if we talked as big system, storage systems, a big sort about with hundreds of disks and multiple servers means tens of serves disk etcetera they will all present itself as a SCSI device to the base systems.


All of the stuff will be speaking this SCSI protocol. So, a SCSI any kernel system is knows about a SCSI they will give a SCSI commands and this system is to understands what this SCSI command is with does some processing and finally, pretend (Refer Time: 59:36) it a SCSI device and (Refer Time: 59:38). So, I would like to conclude.

Storage system design has many ramifications for the rest of the system.

(Refer Slide Time: 59:45)

Storage Systems Design

- Storage systems design has many ramifications for the rest of the system
 - Provide abstractions based on application needs and devices
 - Design needs to be sensitive to cost, devices, manageability
 - Introduce newer abstractions with time
 - eg. key value stores

 Storage systems need to scale to support large scale computing systems

We want to provide abstractions based on application needs and devices design needs to be sensitive to cost devices and manageability. It also includes new abstractions with time we will talk about this soon for example, key value stores an example a good example is of this is the Cassandra file system from system from (Refer Time: 60:04) file system, Cassandra system from Facebook that is a key value storage. It is not a file system kind of model it is a slight different model.

And you need to scale to support large scale computing systems this has to sale like anything because the storage what you are storing and what you are returning was also become bigger and bigger. This is some critical things that is a reason why nowadays

storage systems are becoming critical parts of computing infrastructure and there are completely new designs, completely new things we are looking at these things and that is what we will be looking at this course.

So, next class we will start looking at some very simple storage systems to start a more coming in some detail so that you can get an idea.