

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

**Introduction to Storage Systems**

**Lecture – 01**

**Overview of Storage Systems, Introduction, How Storage is different from Processing & Networking, Why study Storage systems as a separate discipline, Basic functions of Storage System (Naming and Storing), Large Persistent Data Structures, Introduction to Storage File Systems, Storage Characteristics, Storage Performance, Storage Protocols, Storage Systems Design, Summary**


Welcome to this course on storage systems. This is a course being offered through NPTEL. My name is Gopinath, I am from Indian Institute of Science. In this course we will study storage systems, why they are important? Why they are needed in the first place? And essentially motivate the study of storage systems. And try to point out some differences between, but you might have studied for example, a network systems or computer systems.

There is some lightly difference things in storage systems, we would like to reconsider that part. Let us get started, for first let us think about why storage systems are become important in the recent past.

(Refer Slide Time: 01:15)

**Introduction**

- **Why Storage Systems?**
  - Earlier (processing + storage) and networking
  - Now: processing, storage and networking
  - Fast networks enable separation from “processing”
- **Devices, Protocols, Layers/Systems**
  - Old and New Devices: Tape, Drum, Disk, Solid State
  - Protocols: NFS, Cloud storage API
  - Layers/Systems: Google FS, Mail storage
- **Issues**
  - Older: concurrency with CPU, handling device diversity
  - Newer: scale, distribution, error mgmt, security, RT, QoS, manageability



For example, if you look at 1980's you find that processing and storage were together and somewhere about 1980's, networking started in a big way. For example, the 3 megabit per second ethernet started an about 1980 and after that networking has become more and more capable and therefore, it is not possible to separate out processing storage. In the past it was the case that a storage device is always connected to the processing device through a electrical bus. Now with networking it is possible to replace the electrical bus with a genuine networking layer. That they can actually be separated out and not be constrained by electrical conditions. For example, the fact that you cannot keep the bus. Let us say away from the CPU by more than certain number of meters. Because, of this you now find that processing storage and networking are now 3 distinct branches of computer systems with her own styles of design and modeling and performance etcetera.

Essentially, I just want to summarize like that fast networks enables separation from processing. And so, storage systems have essentially become decouple from processing. They have the role let us say logic of development. Now, should you know what storage systems you find that you have devices second protocols and certain ways of layering the systems. To give an example if we look at devices right you might heard of tapes, later you might heard of drums, which are no longer used nowadays, disks and now you have solid state devices like flash and in the future, what are called solid state memories.

There also certain protocols I will going to summarize in details later. For example, you can have a this call a network or system storage or you can nowadays with web scale computer systems, you also have what is called cloud storage; that means, that your storage is not locally present, it is available on the web on the on the internet and there is a way to access this storage also and there is a what is called an API that is often published by which you can access this storage. In addition, you can have complete systems large scale systems.

for examples you can have the google file system, which actually is geared for search purposes or you can have storage for email. If you go to any large scale let us say email provider they will have storages very geared for storing large amounts of mail and this things can be quite reasonably complicated. There is a special aspects in the design of these kinds of storage.

If you think about it, there has been varieties of issues that are involved in designing of storage systems. In the past one of the major issues was concurrency with CPU basically because storage devices are slow. Because they are slow we do not want to make the fast guy the CPU keep waiting for the slow guy and therefore, you have to somehow make them go together and so, by definition once you introduce a slow and fast device there has to be some element of concurrency and this managing management of the concurrency with CPU has been an important issue it is still an important issue. But, it was it has been a very potential in a past other aspect is handling device diversity you notice that input, output devices pitch in all the time and they come in all kinds of shape and sizes and capabilities. And so, if a particular device storage device storage system, it turns out that you have to deal with lots of types of devices. As I mentioned already you can have tapes you can have drums which are no longer available disks and etcetera.

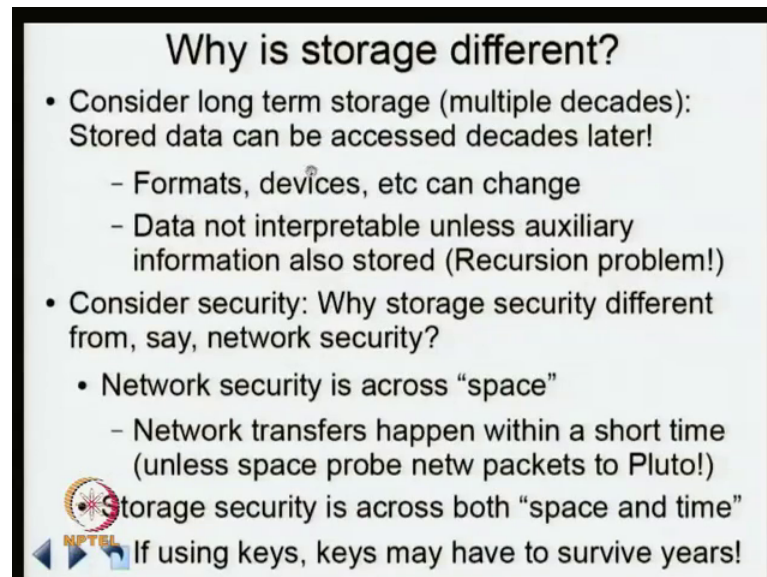
Now, if you think about the newer systems you find that the kinds of things that people worry about is scale. What you mean by scale is? Can I make sure that my system can grow from small size system to very large-scale systems? Like for example, let us say Facebook require some storage system can it scale to that kind of a requirement? That is our distribution where should my storage be for example, if it might be that I am interested in ensuring that some parts of it residing, some places and some other parts ready some other places for ease of access, for performance etcetera and again once you start developing storage systems and very large scale, it turns out you end up getting lot of errors in that devices. There could be errors in the devices, there could be errors in the possibly sometimes in the protocols themselves or in the systems themselves how do you manage these errors?

Again, once you have separated out processing from storage from networking, then there is an issue that there are multiple entities in the system. On a security reconsider important issue. Again, once you have separated out processing through storage through networking there are aspects of real time issues also cropping. Basically, because now you separated things out now if certain things have to be done in in some motion of real time it could be soft real time or hard real time. You know I have to think about all the networking and the storage together provide that capability.

Similarly, quality of service also is important issue and since these storage systems have become fairly complex, the manageability has become also were important issue, of I


will tell you that if you buy 1 gigabyte of storage, the raw price might be let us say 1 dollar, but actual final cost that people pays closer to 7 to 10 dollars; that means, that the cost can be about 7 times as much as a raw cost itself. Because, of the manageability casket.

(Refer Slide Time: 07:54)



**Why is storage different?**

- Consider long term storage (multiple decades):  
Stored data can be accessed decades later!
  - Formats, devices, etc can change
  - Data not interpretable unless auxiliary information also stored (Recursion problem!)
- Consider security: Why storage security different from, say, network security?
  - Network security is across “space”
    - Network transfers happen within a short time (unless space probe netw packets to Pluto!)
  - Storage security is across both “space and time”
    - If using keys, keys may have to survive years!



Let us also look at why storage is somewhat different from other systems you might come across I will mainly contrast with networks. Suppose, you consider long term storage basically I want to keep some data available or multiple decades. Basically, I can access something much, much later than when I created it. Now, what is the issue because of this? My formats and devices can change as I mentioned drums are no longer available.

Suppose I had stored something on a drum I cannot access it now. Other issue is the data may not be interpretable unless auxiliary information also stored. What do you mean by this? If I look at a file system it has certain notions of how to store various parts of a file or the metadata that is the information about the file itself; that means, that I need to know where things are; that means, that I somehow I have also incorporate some understanding about the structure of the data that I am storing.

In the sense that information also has to be somehow stored along with the data itself. You can sense certain recursion problem here. This something also you have to worry about similarly, let us take a look at security you might heard about network security.

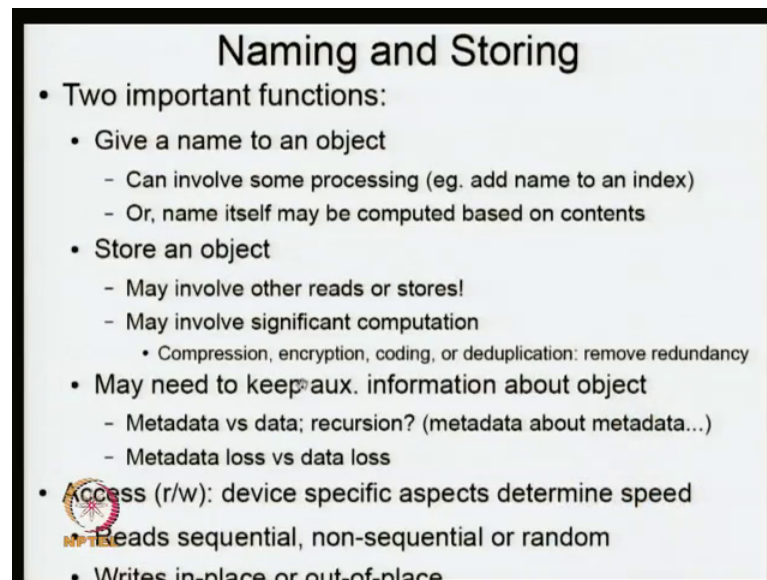
let us just briefly look at why network security is somewhat different from storage security?

Now, I will contain that network security is across space. What I mean by that is? When there is some information that is being transmitted from one area to another area typically it turns out. It is a separated in space, not in typical in time. Of course, it can be in also with respect to time, but usually this very short durations of time. Unless of course, you were thinking about somebody sending a network packet to some very far of place like Pluto or some such place. It may take years, since we are not talking about those kind of systems typically network transfers happen within few maximum seconds typically.

The kind of security that is you have to think about is turns out be somewhat simpler. Because, example the keys that you have to example used to encrypt things have to survive only few seconds. Whereas, suppose I think about storage security where I mentioned that you might have to keep data available for decades; that means, that this storage security is not only across space for example, I want to create a document here I want to ride some place 100 kilometer from here that is across space. Falls across time I create some document I went to see it about let us say 10 years from now. Now I from trying to encrypt things it turns out am I treat required things called keys and these keys may also have to survive years and I mentioned that there is a problem here already of long term storage. This you also have to solve unless giving you some 2 or 3 important issues. Which unfortunately I not yet been solved properly, but it gives an idea of storage is somewhat different.

Again, if you talked cryptographers, they tell you storage security is somewhat different from network security. Why is that? Because, in storage security what happens in often times you aggregate lot of information on single place; that means, you have lot of cipher text in one single place. Whereas, in the case of networks, it is not going to be that dense information on single place. Again, there are lot of different differences you can see with the networks, you can also see differences between other aspects of computer systems you think it about it.

(Refer Slide Time: 11:34)



### Naming and Storing

- Two important functions:
  - Give a name to an object
    - Can involve some processing (eg. add name to an index)
    - Or, name itself may be computed based on contents
  - Store an object
    - May involve other reads or stores!
    - May involve significant computation
      - Compression, encryption, coding, or deduplication: remove redundancy
  - May need to keep aux. information about object
    - Metadata vs data; recursion? (metadata about metadata...)
    - Metadata loss vs data loss
  - Access (r/w): device specific aspects determine speed
    - Reads sequential, non-sequential or random
    - Writes in-place or out-of-place

Again, let us I am going to introduce some aspects of a storage and I try to give some high-level ideas in this particular class and then we will look at most of these things in some detail later on.

Let us just look at what is the issue with storage? There are 2 major functions that you have to do in storage one is naming, one is storing. That is first you have to give a name to an object. Now, what interesting about this? Is that giving a name to an object itself can involve processing for example, you might keeps an index. Can I access things faster than next member that one possibility? Other possibility is that the name itself maybe computed based on contents sometimes what is called content-based storage by which that contents are basically used as a way to locate the information we are looking for. Now, it looks a bit strange, but basically what I can say is that you can store some file by looking at it is contents and actually computing some kind of a hash on it and use that as a name and later on that name is used to access it, because, the storage system can be asked to retrieve that information based on that hash. This also responsible

Now, this is one part of the thing this is naming an object. This as I mentioned earlier can involve quite a bit of processing. Second thing is storing an object. Now, it turns out this itself can involve other reads and store systems. This is some interesting aspect relating to this I come to that later. Storing also can also involve significant computation for example, I can compress things for a store I can encrypt it, I can do coding, for some

time nowadays, because of the large scale large amount of storage that is being generated and stored. You often time do something called de duplication. That is, you want to remove a redundancy that is you have some information you want to store it.

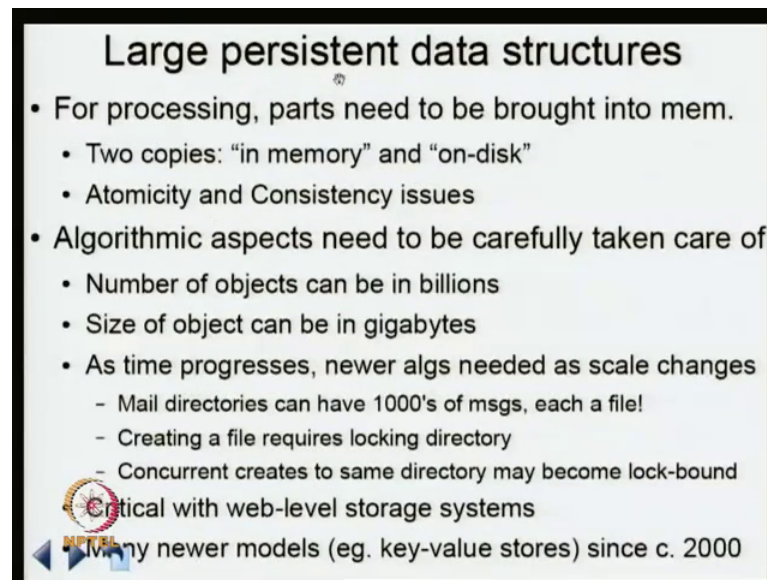
What I would like to ideally do is, see if some part of the information I am trying to store actually already exist in a storage system. I can eliminate having to store that part that also is there. These also becomes quite a big thing; that means, I need to actually check whether I already stored part of it is on somewhere. This also big issue. Basically, storing an object, itself can be substantial amount of time it can be substantial amount of time.

Other aspect is that I need to keep auxiliary information about an object for example, we need a create an object. When did I let us say add something to it? What is the size of that object? That I have kept and nowadays this is become a more important. What is called it is often called provenance? That is how did this object come about. I need to keeps information because often times this is a critical of information for example, in legal areas it is important when something is created when something was sent etcetera all these are important.

In the sense I need to keep some auxiliary information about object and there is an issue about metadata and data. Again, the metadata also has to be stored, again we can see some elements or recursion out here. You can even go further how do you where do you store the on metadata itself where do you store there are other issues for example, if I lose metadata am I lost? Is that an issue? Or if I lose a data am I lost? Which is a more critical thing do I need to have different ways of protecting metadata versus data? These are also issues might have to think about.

Again, I mention naming and storing are important, but if you do not access the information then you are not doing anything useful finally, you have do some access. Again, here is where the device specific aspects determine the speed at you are accessing it. For example, your accesses could be sequential, non-sequential or random and depending on whether sequential, non-sequential or random. Your performance may also change your writes can be in place or out of place again there are different devices which are different aspects relating to this things and there is your storage system has to make sure that you are doing as well as you can given the devices you have.

(Refer Slide Time: 15:58)



**Large persistent data structures**

- For processing, parts need to be brought into mem.
  - Two copies: “in memory” and “on-disk”
  - Atomicity and Consistency issues
- Algorithmic aspects need to be carefully taken care of
  - Number of objects can be in billions
  - Size of object can be in gigabytes
  - As time progresses, newer algs needed as scale changes
    - Mail directories can have 1000's of msgs, each a file!
    - Creating a file requires locking directory
    - Concurrent creates to same directory may become lock-bound

Critical with web-level storage systems

Many newer models (eg. key-value stores) since c. 2000

Now as I mentioned lot of people are generating lot of storage lot of information they storing it. Basically, it has to be persistent so that I can access it to later. Once you have large persistent data structures, if you do wanted to do some processing, I can not do it directly on the storage devices I need to bring it to memory. Usually because of this you have often called in memory copies and on disk copies. Once you have in memory copies and on disk copies, you might have issues relating to what is called atomicity and consistency. We will discuss this in some detail later.

Again, once you have large persistent data structures, you need to take into account organic aspects carefully. Especially, about how to access them how to storage etcetera because the number of objects can be in billions, size of objects can be in multiple gigabytes and as time progresses as a scale keep changings you have to keep developing your algorithms. For example, when I first started using computer systems I think my maximum number of size of things which I used to play along with kilobytes, but nowadays playing along with megabytes is quite common right or you can take a simple example.

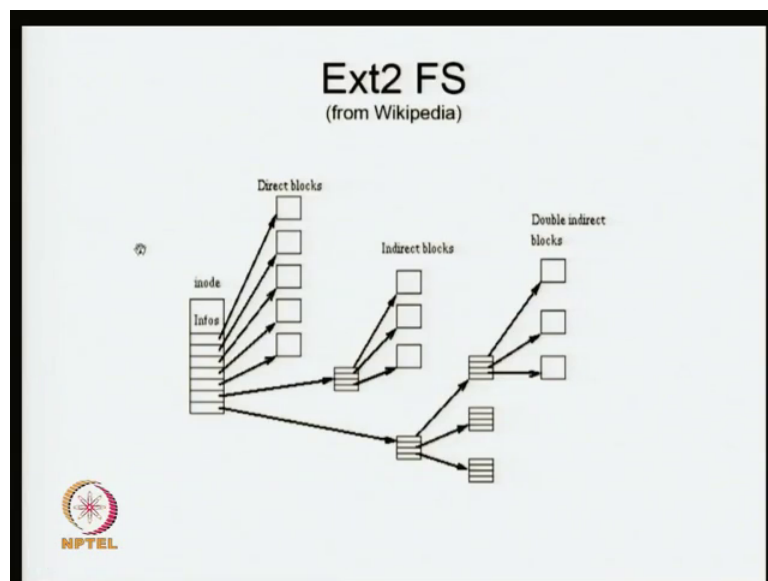
If you look at any email system, you will find that the mail directories can have thousands of messages and typically if you some certain types of formats and mail dif format for example, each message is a file so; that means, at a single directory you can have thousands of messages. Now, if I create a file to ensure certain consistency



requirements, I need to lock the directory. Now, if I 2 or 3 parties wants to concurrently create an email message then the same directory can become lock bound. There are issues with respect to locking can come in again this happen because of the scale. Once you were working with a small scale is not a problem, but once you start going to larger and larger scale systems thins thing becomes extremely difficult. Again, with web level storage system this is absolutely critical part you have to take it in account as part of the as the starting part of design itself and for that reason we will see many newer models.

For example, key value storage become popular since 2000. We will discuss this key stores in some detail later. Again, I am just giving you high level ideas about why this is a slightly different kind of systems that what you have seen before? And some just trying to motivate study of storage system independent of other sub systems.

(Refer Slide Time: 18:58)



You have just to give a quick idea about some of the structure that you often deal with storage system I will give a 2 examples. One is the ext 2 file system I have taken this figure from the wikipedia article on ext 2 FS. This is what is called the data actually resides you in this this blocks? What I called the direct blocks? Or the indirect blocks? Or the double indirect blocks?

And this is the these are all basically index in structures, what you see these are all the index in structures and this particular design is from a file system a is to the file system that has been used in Linux since 1992 or 1993 and this is itself comes from a file system

called the BSD file system. That was designed from 1982 or and the basic idea in this particular design is to ensure that, if you have a small files, you can store them directly in direct blocks. Each direct block is about let us say 4 kilobytes. Example you have a file of only 8 kilobytes you just need 2 direct blocks and rest of it is not used none of it is used needed, whereas, if you want to go for bigger and bigger files you first finish off using all the direct blocks.

Once you done with the direct blocks, then still in more space then you use one level of indirection. One additional level of indirection this is called indirect block. Indirect I should call it I think there should be a small differentiation because direct data blocks. Indirect data blocks double indirect data blocks and this should be called indirect block and double indirect blocks.

This is an indirect block and this will actually have point us to the indirect data blocks and if you run out of this then, you go for one more level of indirection and you will see that typically about 8 of these direct blocks that is about 8 into 432 kilobytes. Go beyond it then it turns out that each 4-kilobyte indirect block can store about, let us say if you are saying about 4 bytes per each indirect data block about 512. You see you can see about 5 12 indirect data blocks 512 into 4 indirect data blocks 512 into 4 kilobyte that will be about 2 megabytes. This will be 2 megabytes.

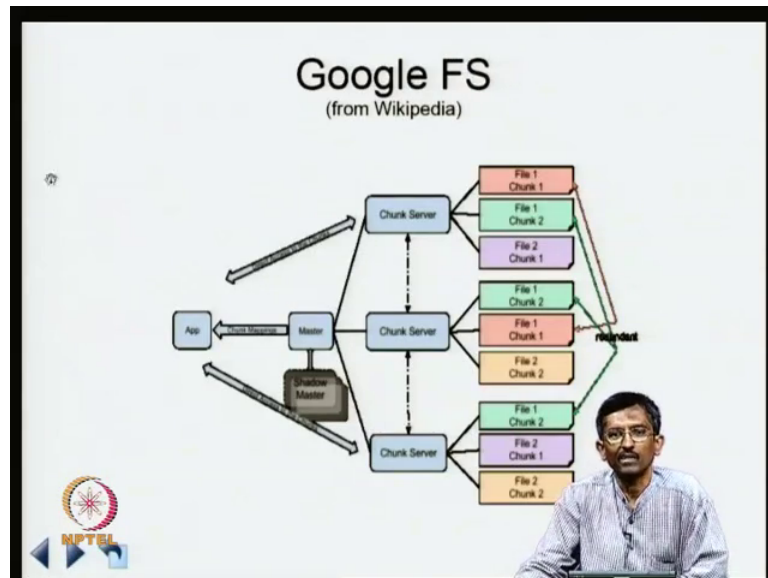
After 2 megabytes you can use this once you go beyond 2 megabytes again this can go into ability to store about gigabytes and this is a design which is trying to solve many problems. Because, I often has small files and big files, I want to see if I can have some structure which can save all this things can be used to store small files and a large files, but you can see there are some difficulties with this.

A good example is, suppose I have a multimedia file the multimedia files are generally time sensitive; that means, that if the time taken to access these double indirect data blocks is different from time to access to this also this. Then, the design of my real system for showing this or looking at this data and showing it for some multimedia purpose right now, it is going to be motor key why is that, because to access this I need to go through one level indirection followed by another level indirection.

This are latency to my access here and therefore, it can create several complications. This is an example of a file system that was designed before large multimedia file systems

large multimedia files became very common. Nowadays, people would like to design something more some other structures by which all the data blocks, whether all the data blocks are about the same place it depends same amount of going through indexing structures. They do not require different types of different units of different amounts of indexes to be traversed before we take to the double blocks.

(Refer Slide Time: 23:24)



Let us take another example, to just see what kind of a file systems can be existing. Storage systems can exist, there is an example of a google file system again I have taken this picture from wikipedia and now this particular file system is geared for doing search. I think all are familiar to google search and the issue is that there are very large amounts of storage that has to be searched fast. In addition, the amount of stop the that is very huge. It is important that you optimize the system for this particular purpose. Basically, google file system is geared for doing one thing.

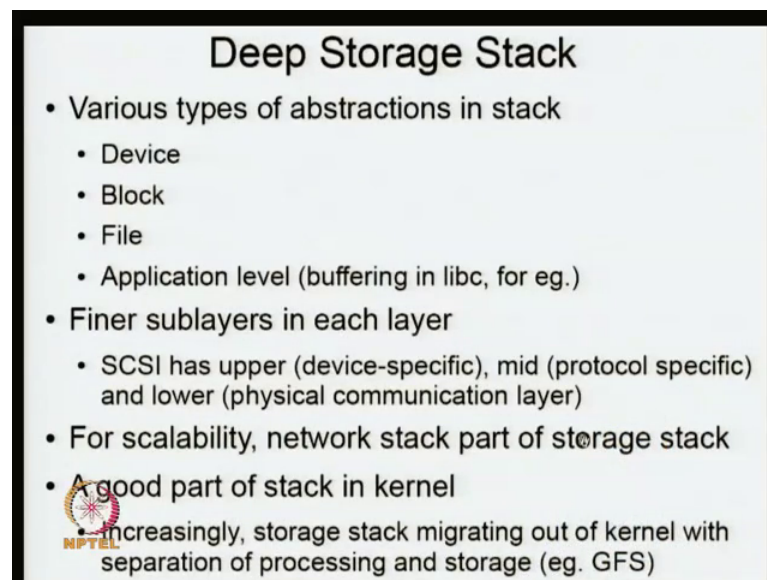
Mostly read only data I want to search it fast and how is that what you is you crawl the web try call the data you crawled and put them in to different files and then each file is split into large chunks that is a 64 mega byte chunks and the master keeps track off the metadata that basically it is the keeps track of where the chunks are. The chunks servers are machines which are able to access those chunks and in application for example, search application Google's search application it will talk to the master and say I want to

know some information about a particular file which has the information which I am looking for with respect to some search quicker as coming.

The master gives you some metadata about where it can access, where it can find it and this application directly access the chunk servers. That is once you get the information about the master it can directly go to the chunk server. Does not have to go to the master again and because the operating at a very large scale, it turns out that there are problems of liability that is there are disk which are storing these things. Let us say, these disks can fail. For that reason, they need to have redundancy in the system; that means, for example, chunk1 is there is one copy here and there is another copy is here. Similarly, chunk 2 will be having another copy. In case one of the copy fails that disk fails then you can actually a master can redirect to some other chunk.


You will find that the google file system has a different notions a what to a from design point of you then what file system has got. Again, in this particular course we will try to explore some of these aspects in some detail.

(Refer Slide Time: 25:59)



**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)

Another aspect that we have to think about in a storage systems is that, the storage stack is fairly deep, basically because there are various types of abstractions, you have can have what is called device abstractions, block abstractions, file abstractions and even application abstractions.

I think this may not be that well not that clear. So, just want to mention that if you write a program in c language in libc for example, there are specific functions which actually buffer what you take from the storage system. So, so even application can also can be depict.

Now, there are each of these abstractions that are given more sub layers for example, if you take a disk let a device for example, there is a protocol called SCSI, SCSI small computer system interface and this device if you look at it carefully, the software stack that is used to access this device, has multiple sub layers for example, it can have something called a upper layer mid layer and the lower layer.

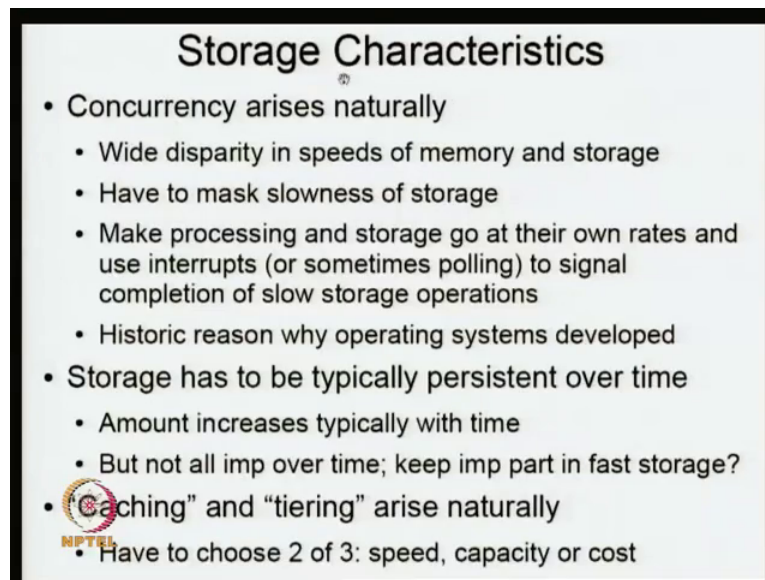
The upper layer essentially takes care of highly device specific aspects for example, tape has to be accessed differently from disk, tape has some notion of a rewind where a disk does not have it, those kind of aspects like rewind etcetera device specific they handle here, mid-level basically protocol specific this actually SCSI protocol itself.

The lower layers basically are the physical communication layer, example your SCSI device can be on a physical electrical connection or it can be through network. So, this part of it is basically that communication part, again continuing on the same communication part of it turns out, if you want to really have your storage system scalable, you need to have essentially the network stack part of the storage stack, because my storage is across multiple devices which could be on in different data centers, or geographical separated by thousands of kilometers; that means, the network is a part of the storage stack.

So, you might have heard about the 7-layer storage stack the internal ISO standard; that means, that almost all of these has to be part of a storage stack, somewhere in the lecture. So, so quite a long time most of the stack was in the kernel, but again given that the networking has become sufficiently pass through separate processing from the storage, a good part of the stack is now migrating out of the kernel a good example is google file system, you see that most of the thing that they do is further out of the kernel, it is not it is not really everything not everything is done by kernel.

Let just briefly look at some aspects of storage.

(Refer Slide Time: 29:07)



**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

First thing I have to mention is, that concurrency arises naturally, why is that, because this is a wide disparity in speeds of memory and storage. So, if you do not want one that keep waiting for the slow guy you, should let everybody keep going at their own place, what you have to do you have to mask slowness of storage somehow very critical. So, you want to make processing and storage go at their own rates and therefore, you need to come up with some new mechanisms like interrupts, or sometimes polling to signal completion of slow storage operations.

And this is the historic reason, why operating systems actually developed? Because you used to deal with devices with highly different let us say speeds, and to manage those things, how a different operating systems started? With this and other aspect of storage is that you are storing something because you want it to be persist over time; that means, that typically it increases the time, but at the same time, not all the stuff you stored is important, somehow there has to be some notion that some parts are important some parts are not that important; that means, that you need to find the way of categorizing storage as something has to be in fast storage and slow storage therefore, caching and tiering arise naturally.

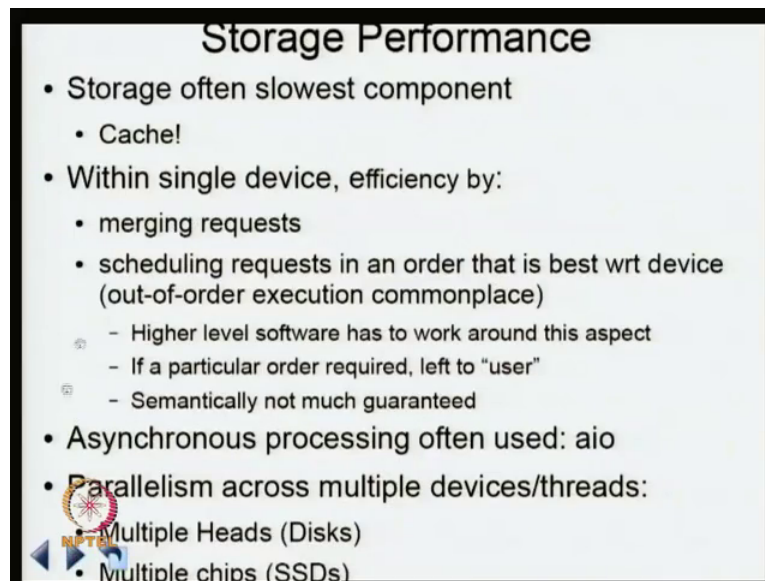
These are the important aspects that have to there because we dealing with slow devices. Now if you think about it, it turns out that there are some serious constraints, either a speed capacity or cost as I mentioned storage keeps increasing over time; that means,

that you can not keep on investing too much money in it, because it just keep on increasing; that means, that you preferably like to have low cost storage.

That it turns out that low cost storage is in conflict with speed. And so, you can not have both high speed and low cost, similarly you can not have high capacity and let us say high speed and low cost it is not possible. So, you have to choose 2 out of the 3 typically, and in storage typical, what you do is? We decide to concentrate on cost a bit more, and capacity bit more, and leaves caching and other methods to take (Refer Time 31:43).

So, this is what we typically try to concentrate on make sure it is as capacity as possible, and as slow cost as possible because the storage amount keeps on increasing with time there is no way to, let us say make it keep decreasing with time is not possible. So, that is why I somehow have to make sure there is low cost.

(Refer Slide Time: 32:06)



**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)

Again, to retreat storage is often slowest component therefore, you need to use caching as an important principle, and in addition to this, you might want to do some other interesting things, for example, if you have single device you may want to merge request. So, that it is possible to deliver the each request more efficiently combine together than (Refer Time 32:31) other thing you can do is schedule request in an order that is best with respect to device, what I am try to say is that right in computer architecture you might heard about out of order execution, that it turns out that that came out much later, but in storage it turns out that this is very, very important and common place thing.

You need to do out of order execution, because you want to make sure that the device is able to respond to request in the way, it finds it best other than the way the application or multiple applications are asking for it. So, the higher-level software has to work around this aspect, if you need a particular order to be honored, it has to be left to high level software, again we will come to this later this is concerning when we talk about atomicity or transaction etcetera, we will find that this is an important issue that we have to re model.

Usually what I am trying to say is that the semantics of storage it is as weak as possible. So, that the system is as high for high performance as possible. So, this introduces lot of complexity into storage, lots of complexity, and for the same reason you often to, what is called asynchronous processing that is you initiate an access and do not wait for it, and you hope that somehow you will figure out when it is finished, and later do something with it for example, you might have some libraries called aio libraries which helps you to do this asynchronous processing.

Again because of the slowness you would like to expert parallelism across multiple devices or threads, and which are multiple disks is possible at, multiple chips nowadays with solid state devices you can do it in the future we will have something called storage cast memories, and this will also become extreme important. Again, the weight of manage so many disks. And so, much iOS happening at same time is through some models of threading in your operating system, or in a application as the application can be multithreaded or the kernel can be multithreaded. So, it also introduces additional levels of complexity, because you need to actually manage the parallelism across these devices.

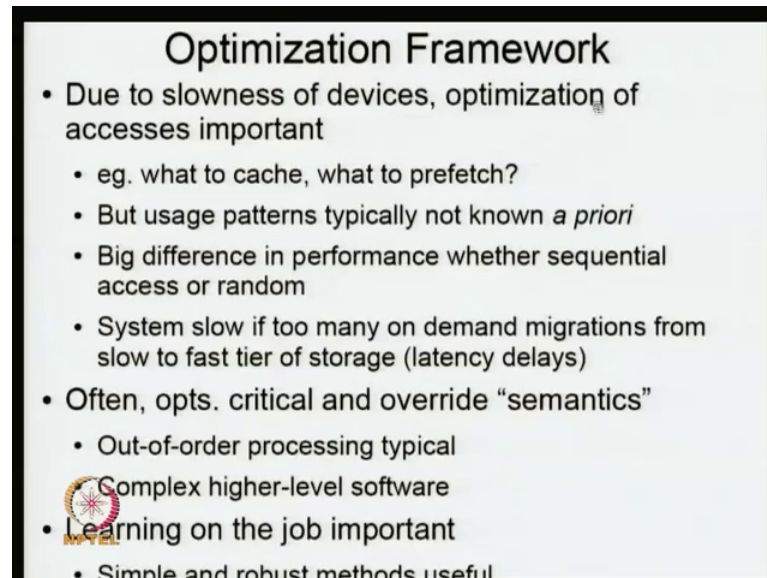
So, again all of these coming because storage is often the slowest component and therefore, you needed lot of things. So, work around this particular problem. And so, your software tends to be slightly more complex, again there are issues with respect to the cancel devices for example, some of the software or much of the software there has been return so far, and seen that disks are being used, therefore, there are something some notion or circular tracks are there.

So, that we will access something, you will find that something near it is track example is preferable to access something close by track than something else. So, that is



essentially assumption about the disk itself whereas, this particular assumption may not be true, when you go to thing say solid state disk or storage cast memories. So, some of the assumptions you are making, sometime unfortunately or let us say deep down in the software's model of how to access this storage devices and the feature that have to be decoupled. So, that these kind of things done do not create complications.

(Refer Slide Time: 36:20)



**Optimization Framework**

- 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

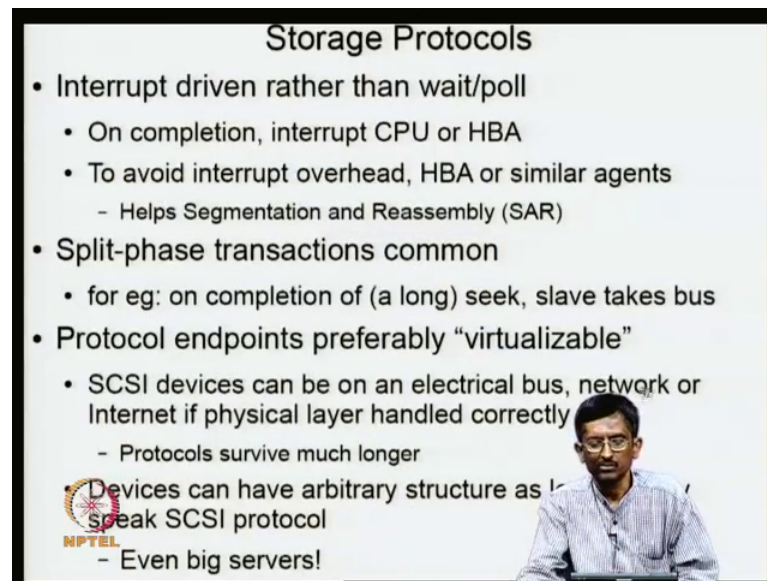
Again, for the same reason because of the slowness of devices, optimization of accesses are important for example, what to cache? What to prefetch? However, usage patterns typically are not known, a priori and it is very clear there are big differences in performance whether sequential access or random. So, a system actually can be proceed to be very slow, if too many on demand like migrations from slow to fast tier of storage there are lot of latency delays, again just as I mentioned earlier optimizations critical and there sometimes override semantics out of order processing is typical, and therefore, this complex higher-level software, again I am this is what I already discussed before.

So, because of this reasons it quite important that you can learn on the job, that is you see certain accesses can you make sense of it, and typically most storage systems has simple and robust methods. So, that you can suppose work around this slowness of devices for example, if you take any file system, if you access any part of a file there is always something called read ahead that happens, and these are some a very simple and method to make sure that if your accesses are sequential, if you do read ahead then you will not

suffer latency for the thing that you already read ahead. So, simple robust methods are widely used in storage system.

But I think as time progresses probably even more complicated, or complex models of learning the kind of application accesses are taking places have to be investigated, and incorporated into storage systems, I think this work is begin just begin about to be just about to be cooperated to the storage systems design.

(Refer Slide Time: 38:25)



**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!

NPTEL

Now, let us look at what are aspect, this is concerning storage protocols. Now if you think of network protocol, usually each party send something and wait for the other, here it turns out that instead you have a interrupt driven, model basically driven that storage devices are slow.

you do not want to keep waiting for them or keep polling them, because it is too expensive again to appreciate the difference in the speeds, memory can be let us say about few nano seconds whereas, access to a disk can be in the region of milliseconds, it is about the differences about 10 to the power 6 or 10 to the power 5. And so, you cannot really afford to wait or poll.

You need to actually tell the storage system when you are done, tell me that you finished. So, this is a typical model, and therefore, what happens is that once the access is completed, you interrupt the cpu, but the CPU if it is get interrupted too often it creates

some complications therefore, you often create what about call as an agent, agents which take care of interrupt processing and agent typical that is used what is called a host bus adapter, this is a name which is a sort of historical basically because this system that electrical bus directing, that is why it is called host bus adapter these are agents which insulate the processing unit from the interrupts that are happening because of storage access completions.

So now because of this host bus adapters, it turns out that you can do some other additional useful things, just like networking has segmentation and reassembly right, because networks have to traverse different types of network different characteristics of different useful sizes of network packets. So, they also do segmentation reassembly, here also we do the same thing. So, that HBA is actually does some type of segmentation of reassembly which helps in efficient ways of translate information from storage device to the CPU through the HBA, other thing that is very common storage protocols is that is called split phase transactions, typically you have a master and slave and usually the idea is the master regulates, how the bus is used.

But in storage system, is slightly different thing that happens, because the master cannot again as I mentioned cannot be keeping on waiting and polling so; that means, that if for example, our disk it takes a let us say few milliseconds to seek to the right spot, then it itself it try to grab the bus it does not have to go through the master to get the bus, in the sense there is a the protocol has been designed in such a way, that the master initiates it, but the slave becomes the in the sense owner of the bus based on when it completes. So, so if we see some certain differences between the protocol actually designed in the storage systems.

Another aspect that is critical about storage, is that the protocol endpoints should be preferably be virtualizable, what does that mean? Notice that SCSI devices preferably should be placed an electrical bus, that is you are trying keep it very close to the SCSI unit because you can not handle too much latency, you're for actually want to physically keep on as close as possible (Refer Time 42:45), device this happens in large scale servers, or you might want to keep SCSI device network or internet itself, if somehow, I can handle physical layer properly.

Now, this is possible if it turns out the protocol endpoints or virtualizer, what is it means? Is that the speaker set on protocol and all that matters is what happens in on the protocol data units?

If those things are handled, then these SCSI devices can be essentially used in various different places in various different formats. So, again we will look at this a bit later in some detail. So, one thing I would like to mention is that, devices can have arbitrary structure as long as they speak the storage protocol, here I am using the SCSI for (Refer Time 43:51) city they can have arbitrary structure, what I mean that is if I have for example, a very large scale storage system, I can have a big server and these big server can manage huge numbers of disks, and it can make it can pretend as if the big server plus all those closed devices, all of them constitute, one big massive disk you want to say the massive disk can be for example, now a days we have one tera byte disks 2 tera byte disks, for suppose I want create a 100 tera byte disks, then I can have then we say 100 and 1 tera byte disks and these big server.

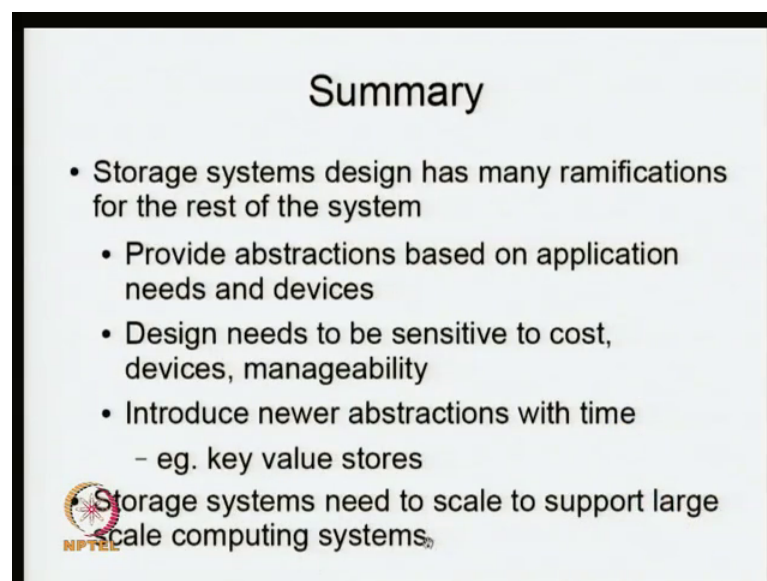
Using some electrical bus or whatever can pool all those devices, and it can pretend as if it is a exporting a 100 tera byte disk, and these big server together the disks can be thought of as a single disk, single device sorry and users using this particular much virtual disk, they would not have to know the structure of it has actually a server with multiple disk somewhere etcetera, all they know is that it is a SCSI device and it uses SCSI protocol and therefore, it can let us say both host, we just assume that this is a SCSI devices, they do not have to know anything about how it is actually constructed. So, this virtualization is a very important aspect of storage systems, and it very extensively used and if you look at SCSI protocol the reason why it has survived. So, long it has for example, SCSI devices started coming out in 80's the first question came in I think 1990 about 1990th in SCSI 2 came about, and SCSI 3 has been resistance same about 2 thousand, and one can see that these kind of devices are let us say widely used (Refer Time 46:16) because the can be virtualized and therefore, you do not lose your investment with respect to your host devices, how they actually talk to these devices.

For example, HBA for example, you do not have to change because as I mentioned to you earlier CPUs have agents, which actually talk to the storage device this HBA for example, they actually do not have to change at all, because the devices if they are virtualizable and they export is SCSI interface HBA is does not have to change. So, I

wanted to point out in this particular slide was that the protocols that you see in slow systems, are often of a different nature than from what you see in other areas like networks for example, and I have not mention it some other aspects here for example, security if you look at storage protocols there are aspects related in security, there also have to be taken into account and typically they will be running on top of the network protocols, and there are some specific protocols for storage security.

Again, we will briefly look into it at some at sort of (Refer Time 47:43).

(Refer Slide Time: 47:44)

A slide titled "Summary" with a list of bullet points. The text on the slide is: "Summary", "• 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", and "Storage systems need to scale to support large scale computing systems." There is a small NPTEL logo in the bottom left corner of the slide.

**Summary**

- 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.

NPTEL

So, I think I would like to summarize, what we have looked at, first of all the storage system design has many ramifications for the rest of the system, why is this? Because if you look at storage systems, they are often used to store some critical information like for example, in the (Refer Time 48:16) system if you talk about virtual memory systems, the storage system actually stores those pages that are has been swapped out. So, if that swap device is not fast enough, then your performance of your programs or applications can be slow.

So, that is the reason why you have to make sure that, your system design is properly well thought out. And so, basically you need to provide abstractions, based on application needs and the type of devices we have, this design has to be sensitive to cost and the capture devices that you have under how to manage them, you need to introduce newer abstractions with time, a good example is key value stores that have become

popular with web scale systems, you will notice that key value stores were not used seriously before 2 thousand, but once search become very important key value stores became quite critical, and this storage systems also have to, scale to support large scale computing systems, and because without the scalability then the current which scales kind of systems cannot really be used at all. And so, they become extremely critical to the whole design of these systems.

And. So, I think I will conclude that this point I think the next class what we will do is we will look at each issue we will take each issue at in sequence and study it in some detail. And so, that you get a complete prospective about this storage systems.

Thank you.