

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

**Introduction to Storage Systems**  
**Lecture – 03**  
**Basic functions of Storage System: Naming and Storing**

Welcome again to the NPTEL course on storage systems.

(Refer Slide Time: 00:28)

### 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

In the previous class we discussed some basics and you are looking at some aspects relating to naming and storing, and storing system these are the 2 important things that it does, one is how to name something, and how to store it and later we have to retrieve of course, and when I mean storing I am also meaning storing and retrieve, both I am taking about. So, as I mentioned in the last time, naming is a fairly can we quite complicated it is not that simple thing.

And that also you will see in a real life example you have if you are a student right, you have an regular name, you might have a roll number as a correspondence, and how the roll number is decided might be depending on some structure right? And if you think about a bit more carefully that is why the Indian government wants something called UID, what is UID? Unique ID they believe that somehow it helps in reducing lots of complications that are there in our society with respect to getting benefits etcetera right?

If there is some that naming somewhere suppose to do help a lot and you will also see this also, if you study anybody who has studied the rec computation right.

You notice that the naming is an important part, why do I show this has an anybody heard of something called as lambda calculus anybody has heard of it, you will see that lambda calculus has got only 3 rules something called alpha rule, beta rule, and theta rule. Alpha rule is about naming right, that what is called it is a way of let us somebody called alpha rule is basically talks about which variables have to be are being discussed, that is all that is all it is and how to do renaming, and the beta rule is all about how to do function application.

Beta rule is some I do not want get into it, but it is something it turns out that having these 2 major rules alpha and beta rules, you can do everything at turing machine can do, anything and everything. So, hidden between these simple sounding thing called naming and storing, lies the complete power of a turing machine, it may come and surprise to you if you studying lambda calculus we will find it. So, that is a reason why a storage system not only does seemingly simple thing like naming storing, it also is do lot of computation that is what were talking last time it can involves significant computation it is compression encryption coding de duplication etcetera.

Second, we just look at this, what is compression? Compression is way of saying if I see something repeated some segment of something, repeatedly coming in instead of repeating it every time I use a pointer or some a much shorter way of and referring to that, and basically, I am trying to reduce a size space that is taken of it, in the encryption what I am a doing? Now the latest of possibilities I can use what is called there are simplest schemes and there are slightly more complicated schemes, for example, if you use something like a s right.

It uses just a single key whereas, there are other systems which is 2 keys for example, private and public keys, or a different type of cryptography, and nowadays you have something called identity base encryptions. So, I d that also is that there are varieties of systems also here, now for each of those things different kind of solutions had been adopted, because fundamentally the keys have to be stored and this keys have to be made available in the future whenever you want decrypt it. So, there is a the fact that you use

encryption, does not solve a storage problem you have to also figure out how to store and retrieve the keys so, it is a recursion problem here.

Similarly coding also is there it turns out encryption the form of coding, you use encryption or the term encryption then you what to say it cannot be broken easily by anybody, else unless you have the key. Coding is a simple way or saying that I am transforming the data into some over variety, for reasons of convenience the reasons of if I loose one piece I should able to retrieve the information back in some other way, I am not interested in secrecy or unbreakability of the code.

So, for example, you have things like red 1, or red 5 what is red 1 you typically keep a another copy extra copy. So, means if you are data is  $x$  it is going to be  $2x$  the total amount is specifically call, or you can have what is called red 5 what you do is you have data something, what you do is you store one segment of the data in one disc, second data second (Refer Time:05:59) second disc, third disc, 4th disc and fifth disc you do a sort of all and keep it there now  $x_1, x_2, x_3, x_4$ , because you doing xor which is a low cost operation, xoring is not that complicated exclusive or.

So, if any of them is lost one of the disc is breaks down, if I xor the rest of them I get what has being the disc that has been lost. So, these an example of coding which is not cryptography quality, this is a trivial thing anybody can do, it only need a keys are (Refer Time: 06:35) of it. So, this is widely used and we will see in many pieces nowadays you get, red as part of your system actually some laptops already have red systems in it; that means, they have 2 copies of data 2 discs, and then they do this why do you need this kind of coding because the disc can fail. So, this is an example of failure management, and this quite critical for very large scale systems, if you look at google they have data in the region of tons of data bytes or more and they cannot afford to lose the data.

So, they usually use 3 copies, every single piece of data is actually typically copied 2 extra copies are made, and they are often can be done, this distributed one piece of it is staying in europe, one is second Asia, one it is in u s the company, and since it is a wide spread thing and the reason why you need to do it also is because errors cannot be eliminated, one disc failure another is disc may not fail, but the medium gets corrupted here and there, last time we talked about sector errors and this is not uncommon, even if you use a best error coding schemes, you will find that there are still uncorrectable

errors, for example, in discs there you something called reed soloman coding widely, and for example, for every 512 bits as much as about 40 to 50 or 60 bits are used to encode that 512 bits.

So, that even if the best of error takes place, for example, 4, 5 bits get lost or even 10 bits get lost, I can still recover the whole thing, actually this has been carried to very interesting levels, if a take a c d rom for example, you can drill a hole in it somewhere anywhere, as long as mechanical mechanically you do not destroy it any way which prevents it to from putting it inside the c d drive you can drill a hole in it, is still be able to read it because there is the use what is called longitudinal and horizontal essence, the 2 dimensional encoding let me not call it longitudinal, there is 2 dimensional encoding that is the sectors you can think of it as sector 1, sector 2, sector 3, sector 4 they do encoding this way they do encoding this way.

So, they use fairly statistically techniques, but in spite of that you notice the you know that, series cannot be get sometimes, why does it happen? Not because of the coding problem, it will cause you use certain types of dies and the cheap discs you often find, the dies gets spoilt, they decompose under some sunlight or some light con conditions. So, over after one year or so, they die, you can not do anything that is why do not ever trust anything valuable on a c d that is bought without knowing where it is coming from, if you are very careful you should buy the best quality cds it may last you for our 5 to 10 years, that is all nobody will guarantee anything or.

Whereas you buy the cheapest brand somewhere right, mostly like it will not last for more than 6 to 8 10 months right. So, be careful about cds, disc are better because they have been designed for use in in let us see business kind of applications. So, you need to make sure it is how is that long. So, what I am trying to suggest is the coding is an important part of you know disc systems or storage systems, and there that is reason why if you look at a disc, you will see that it has a got a processing power, which is approximately that of a in those days sometime back they should call it about pentium 400 megahertz processor.

That means that if you have laptop, let us say you have an atom based transistor let us say, 1.6 gigahertz or whatever, it is about 1 4th or even slightly more, or the processing power of the laptop is sitting in the disc itself, here it is doing all this reed solomon

coding it is doing buffering it is doing various kinds of things, that amount of processing power sitting there, again if it open a disc you will see a mother board, there not (Refer Time: 11:03) the circuit board we will see and there circuit board has got a processer.

So, actually it can even go even to much higher level, de duplication, what is de duplication? Nothing I think I have briefly mentioned it last time the idea is that, you tend to have multiple copies of same thing, I keep forwarding my mail right, you sent me some mail with an attachment with the 507 photographs, I forward it to you right, and I keep a copy of my sent mail I might even have so many copies right. So, or I might when you are working on a document, you keep modifying it let us say there are 3 people working on a document, you make modification that person makes a modification I make a modification.

So, most of the times I keep modifying different parts of it; if I can figure out how my document differs from yours, what is the delta that I have cost, then I can say that I can represent the current version of this as some set of basic parts of it with that smaller addition made by that person, small addition made by you, and small addition made my me, that kind you (Refer Time: 12:10) it that is while where many people say, now a days that close to I think I quoted the figure last time 90 to 80 percent of all storage often you hold is often duplicate, data on big enterprise systems this. So, we can cut it out by 1 10th nothing like it.

So, but I think I mentioned last time, there is a problem in de duplication what is that; that means, at every time you get a piece of data you have to compare it with every of that piece of data you have, I am already talking about 100 petabytes kind of systems, I get one file I have to check it with a 100 petabyte file system, how to do it fast, is not trivial, again I think all of you by now should notice that this minimal storage is natural thing, because that is one thing we human beings also do a bit, we try to store something in a heads right, if you think that it is trivial then we are not you do not what all we go through that a room for [laughter] something right.

Because now the complexity of our life is all about kind remembers certain things all right, you forget certain things sometimes it is just now a tip of our tongue we calculate where it is there is, some complication there right, it is not a trivial thing, I think we gonna say the if you find that the brain is one of the most sophisticated systems that has

been designed, if it has problem so, what remembering it showing because of the large numbers of things, we can say corresponding situation also in large scale storage systems, things can get corrupted somebody has to corrected things, can be in multiple pieces you have to join them together, it might be de duplicated you have to pick up some piece from there from here, even put them together all kinds of that things will be in corner.

So, even your brain might be doing. So, of this things for example, a brain might be doing some compression; that means, that it sometimes stores somethings just a pointer, and then you think a bit more about a thing, then you remember more details; that means, you have a initial handle only rest of it is sitting somewhere else, do not know, where and then you keep thinking and thinking you finally, get this thing and that is what example some of people say hypnosis right they say that if you hypnotize you can remember things, which you can not rather remember in your conscious state in the sense right.

Same thing about of course, I do not know whether you do encryption. So, they are certain kinds of things that we human beings also do, and the same complexity that that attends to store system is there in our systems also, and more important thing for us in storage systems is that, you always need to keep auxiliary information about object, but is if not necessary just to store the thing, I need to know I need to keep an index about it, I think one of the big one of the big problems you often have is that, you kept the thing somewhere, but only where we kept it, alright so, that where we kept it is a critical part not only where you kept it, but sometimes you need to know what kind of thing have we kept right.

So, if you do not have the some auxiliary information, then you have to go through the whole document again to know what it is right. So, this auxiliary information about an object is basically called a metadata, that is data about data metadata, now you can see that this is a already is a (Refer Time: 15:34) concept why if I said that there is data about data, then I can do recursion, what about data, about it why does metadata about metadata about metadata, I can do this and we have to do it has to be done. So, typically we do about 2 or 3 (Refer Time: 15:46) recursion, when file storage systems for example, we have piece of a file you know inodes, but the somebody has to figure out where the inodes are.

That means you also have to now start thinking of where you kept a thing, of course, you can have continuing further, but typically you go about 3 or 4 levels more than that is going to be tricky. So, reason why this happens is that, again for reasons which you discussed earlier, suppose I have some piece of information kept in 3 places, the data is here, I lose this part, then I need information saying that I have got 2 copies here and here right, now if this information itself is lost, then all that amazing thing I kept on is also useless, right I have metadata about where I kept things, but nothing says that I cannot lose this I can lose a metadata itself.

Then I am almost like a lobotomized guy, I do not even know where to start right; that means, that I have to replicate even the metadata, you know replicate a metadata, then again you are saying who is going to keep track of where things are you again, you can start thinking how do we do this, and these things are extremely challenging problems there are no obvious solutions, there are in practical engineering solutions for all these practical engineering solutions, that is why even why google file system works, not because they are solid and impossible problem there are lots of impossible problems in this area, now idea is that you solve a problem to the extent it is feasible. So, that it does whatever our job is what we need to that, is it do not try to solve impossible problems.

So, some lot of it is about fundamentally there are some delicate issues here. So, you normally what you do is you, want to break the recursion. So, basically you go 2 or 3 levels and finally, there is to be place where you have to fix it, you say that the root of the information is here, and you have only kept in 2 exactly known places, and it is no, is like your on a disc is something called master update record, it is known to be term sector 1, or sector 2, let us say that is known and there is no of course, you have to know there is there sector 1, sector 2, where is there information coming from, it is because are the particular level of technology somebody has it is white spread knowledge somewhere.

And there is one of the kinds of knowledge that gets lost over centuries right, we should try to remember that the systems suppose I want to carry it across, there is some information of metadata it may be known commonly now right, example almost everybody knows that we use something called ext 2 file systems, and it stores the super block in a particular place, probably some block might be there in sector 3 and sector 4 whatever it is, now that may not be constant across all file system, but it is known around right now.

So now it is not like does not look at a very critical piece information, but once you have gone few decades in the future, if this common knowledge will not be there. So, then you have to get that piece of knowledge in a very hard way, it is not gonna be trivial I think you must have come across this one of the reasons why lot of you do all this archeology, we do all kinds of trying to figure out what happen in the past, is at you have to take this clues from here, and there scattered here, and they because the metadata of is lost somewhere, you can take this clues and somewhere reconstruct the past, that is some kind of detective work.

So, I just wanted to mention that there are issues think about all this is a bit carefully, you will see that some of this things are nontrivial, and for the same reason I mentioned, you have this ability to withstand a data loss there is this metadata loss again there are different issues here some people value greater more than metadata, I will given example, suppose you are doing your kept their information about your bank account right, now what is important?

How much money you have is more important or the point or which keeps where the how much money you have is important, many people which say the more important thing is even the pointer is lost, if somewhere I am able to figure out how much money have that is more than enough right. So, basically people will say data is very critical, put at it you just all cost at all cost even if the pointer is lost, I will scan the whole thing in the most laborious fashion imaginable, and hopefully I will able to see that my name is somewhere and my account is somewhere there.

Some of you might have heard of about some thing called fsck, what is it? Fail system check right, that does basically what it is doing, it is like I am simplifying it too much, but basically what it is doing is it saying that some point has been lost, it says that yes point has been lost things are looking bad, but I still have the data the disc itself is still lucky readable, most of the disc is still readable, some the part of the disc is gone that is why I lost pointers, but most of the disc is still readable.

So, what I am going to do I am going to say, I am going to do the most dumb or the most harrowing think possible, I will read every single block, see if the information what I am looking for what is important to me, can I say that can I tag the these are all important let



us somehow create a new index structure, even what is say even though I lost somethings, because I am still valuing the data.

So, some people will value data much more than metadata, and a good example is I told you anybody in the finance section financial sector they will (Refer Time: 22:04) there are these people who value data more than metadata, metadata they will say will figure out somewhere, because it is valuable stuff I have do something about it, the other people we will say the opposite, who are the people suppose you are file system guy, what is your business? Your providing a service to various users, you are in a single system with 105 people using a system, your data set your sector died, something happen to us, what is my responsibility? Should I shutdown the system or say I do not mind what that you will loss something, but there are 104 other people I will stay somewhere.

What is the calculation the file system should make, can you say that something wrong with your stuff I am. So, that I do not what to do with it I will proceed with other guy that is what a file system does it. So, the thing is what is a file system doing it saying the following, remember told me what is important and unimportant.

So, I have been finding out, but I know what is good from me, the file system says, I have various point us I will keep my pointers or the auxiliary information or the metadata consistent. So, that I am seen is tough that you lost something, but as long as at least I am seeing, at least my pointers etcetera are in reasonable condition most of them.

I am still able to service rest of you right. So, that is what a file system does, that is in why unless you take special care a file system can lose certain things whereas, the database you will do the opposite, you will try to keep because usually databases use for some let us say inventory some finance packages where (Refer Time: 23:53) your money real world things are there whereas, the file system can also be has accurate as database in constable, but general design point for the file system is to make sure the metadata is complete intact, at the cost of losing if you sectors here and there.

If you lose if you sectors here and there and you lose some pointers, you do a fsck file system check and we will basically recover whatever can be recovered and then it is secured it will, and the at the end of it a file system says I am seeing some [FL] happens some bad things happened some sectors got lost, some got corrupted, and end of it I am

saying I can still use wherever information I could legitimately pick from there, I still will put in the reasonable format I will give it to other.

That's what. So, this is a different kind of models about how to handle loss, and there are many other, I am just giving you 2 examples, you can have many other styles of handling loss. So, these are all again correction with naming and storing, because start seeing now that these are all because you ask to retrieve something, something got lost, and what you have to do then all these things are connected to do that, again if we are asking for storing and retrieving stuff, it turns out there are device specific aspects that determine speed, example take certain sector motion that you have to read and write sequential, let us divide discs for example, they can be more random access.

So, this thing such as determine lots of the effective speed at which you get things, sometimes reads can be sequential it can be non sequential it can be random, what is an example of sequential thing, a good example is your watching a video, movie most likely video is sequentially encoded.

So, just keep reading and monitoring systematically read for 2 hours will be doing it mostly, it can be non sequential now what is going to example non sequential one, suppose you are let us say looking of some address book whatever right, depending on the search string you will find will it will be in different parts of their storage system, it can also be random you are updating let us say some information structure, with new information is coming in it is doing storage different places.

So, it is making seek here it is seeking there it is seeking there writing it here or there, or your information retrieval system people are asking different touch information, somewhere could be asking something on what is of let us say how does one plant grow? Some bodies asking how does you know a if a child has fewer? What has to be done let us think. So, that kind right. So, these are desperate touch information there has 2 different places, and it could be on the disc at different places. So, it can be seeking randomly, now it turns out that the most successful storage device. So, for since the last 60 years or so. So, has been the disc, the disc was introduced first in 1956, and it is till today the most important storage device.

So, this is an electromechanical contraction, it has got let us say some mechanical components and some electrical components, what are the mechanical component it is

got a head which is rotating, and it will sorry the spindle is rotating, and the head also can. So, this mechanical aspect to this, the rotation of the spindle and make of the that of the head also, in addition what can also happen is that, when for power management you may want to stop this spinning, when I stopping the spinning you have to the head also has to be put in a safe place, if you do not put in self it goes on sticks on to the surface, you might lose some bits. So, there are some aspects are very peculiar to discs, because of the electro mechanical aspect.

So, you will find that we cannot drop a disc, if you drop a disc most likely there can be some serious problems, or you cannot use the disc in tibet for example, why can not use the disc in tibet, because the terms out in tibet the air is so, thin you do not get the kind of let us say burnel effect that you get in winner here. So, do not take your laptop to tibet in expect the laptop to work, it would not work you have to do something else, if I not introduce solid state devices. So, the lot of device specific aspects like this and so, unless you are clear about this things, your system will not perform well, the other aspect is rights in place or out of place if you look at a tape for example, what is the suppose you have some piece of information you want to update it right.

So, normally what will happen is that, you will take the piece of information, read out modify it in memory, then you write it back, now the 2 possibilities I can write it back where it was, or I can write it somewhere else. Now what people will figure out in the tape consistence was that you read from one read tape, and you write on the write tape; that means, you read something from here and then write on this all. So, this is an example of where writes are not in place, that is your updating something, but you do not update in same place because writing on to a different tape, now a disc actually did it turn out the discs slightly altered a situation a bit, where in place updates be can slightly more easier. So, because disc made the writes in place a bit more easier.

Most file system started using that, but nowadays there is a new type of device, called solid state disc flash devices, where again it turns out it is more easier to write out of place then to write in place, reason being in a solid state disc, you have what is called a read size a write size and also what is called a error size. Now in this devices you will find that, my read size may be something like 4 k write size may be 32 k, and the area size might be 2 megabytes, what is the issue with this kind of devices it turns out in this devices if you write you will have to erase it, before you can complete erase it; that

means, that is a separate operation can erase has to be done before you can write to the same place.

So, in the sense, erase makes that particular area of where you writing, completely clean in some sense then you can write again. So now, it turns out that the write sizes let us 32 kilobytes. So, the write here, but the erase size is this big; that means, you want to write 32 kilobytes and you write again the same place; that means, you have erase 2 megabytes because the size is determined by the technology which is 2 megabytes. So, does not make much sense. So, what you would actually like to do is it has written here previously 32 kilobytes, you write to some other place just like the tape write some other place and then one this use of this particular area is finished completed.

You release it as free area which can used again re used again, then you do the complete erase. So, there are this kind of issues also are cropping up now, that is why you notice that if a you have a mobile phone, it will have some new types of storage devices most of them, have this flash kind of characteristic and there is some special logic. So, that it all these can be kind effectively size wanted to.


Student: (Refer Time: 32:53).

Show that this naming and storing is nontrivial, there can be unusual complications because of various aspects starting from basic aspects I told you from lambda calculus what it means. So, these are the 2 powerful operations starting from there you can see the kind of computation they required they issue such as an metadata about a device specific aspects all this things are involved.

(Refer Slide Time: 33:17).

### 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

So now, again let us you also need to think about large persistent data structures, what are this things I think, I mentioned to you earlier that you use devices like disc and storage disc and tape for example, there persistent structures, what is persistent? Means even if you remove the power it remains whereas, you look at d ram or dd r kind of memory, all those things if you remove power that data dis appears right.

Now, unfortunately for us the tape and disc are very slow, even ssds are slow compared to memory, and you cannot really do CPU cannot directly access them so; that means, that whatever data you want to manipulate it has to come to memory first; that means, there are 2 copies available, one is in the storage medium one is in memory. Now there is once you have 2 copies then there is an issue of consistency, how do you ensure that if something is there in memory, and you expected it to be the same value in the storage also, somebody has to make it happen it would not happen by itself, another due through hardware which is not usually done you want to give some flexibilities. So, that if it is done in hardware then it is no pointing having 2 copies, they will naturally work with one.

So, normally you keep that copy in memory for as long as it is safe, then you want to make it persist you put it in the disc or tape. So, there is some kind of judgment you want to exercise when you want to move across this 2 areas, when you have to go from disc to memory to disc.

So, there also issues called atomicity issues, if an there is large pieces of data, I can not write it in full one big where I cannot write a 4 gigabytes in one short automatically; that means, without any intervening operation, I write this some part in principle somebody else can also execute after I finish this part I want to write this much, but written only this much somebody else can come in and interfere with what I am going. So, I have to figure out how to avoid automaticity issues also, how to ensure that I intent to write this big and logically it should look as I wrote this big, without in spite of whatever other operation other people would do.

So, there is a big issues for automaticity and consistency this also a very fairly big subject large subject which we need to look into, and since we are talking about large scale systems right, you need to also worry about algorithmic aspects, your terabyte system for example, is very large, I thing you remember suppress to know that if you think about how much data you might be inputting yourself, what is the typical writing speed of a person, it will about 40 characters per minute, or we will make it 100 characters per minute right, or even 200 characters per minute right.

You think about it how much can you write in 1 hour 200 into 60, how much is that it is about 12,000 characters, how much is that is 12 kilobytes right, if you work for one hour consisting without taking a break your putting 12 kilobytes, if you multiply that by let us say you work for 20 hours let us say. So, how much is that, 12 into 20, 240 kilobytes you basically barely got to 1 4th of a megabyte.

Even if assume that you work continuously for one year without break, 1 4th of megabyte into 360 you basically got a 90 megabytes right. So, one full year you barely could without taking a break you made you have essentially 90 megabytes you can input, that is all nothing more than that, and even if you work for 10 years, it will be 900 megabytes about 1 kilobyte, a terabyte is how much, thousand kilobytes that is .1 percent of what you can do. So, your terabyte is about thousand times then what you can do in 10 years if you didn't taken a break. So, this things basically what I am trying to say is that, your ability to deal with this scale of systems that you we talk about at a human level is no longer visible.

What is happening is that you are generating small amounts of information, but you have programs which are generating lots of objects, in a sense what you are doing is you

setting motion certain things which creates lots of other objects, and that is basically where all this larger number is showing up and by (Refer Time: 38:36) we are not able to do it. So, what is happening now is that, the scale at which you are operating has suddenly changed. So, the size object can be in gigabytes, the number of objects can be in terms of 100s of mega millions it is possible. So, because of this size what which is the things are happening, you need to find better ways of storing and accessing things.

Last class I mentioned about mail directories, we can have thousands of messages, and I mentioned that my own directory which I keep 9 map directories it is about 18 today I check it is 18,000. So, 18,000 files sitting in a single directory, when you are searching for this things, you cannot use any order of any  $n$  square algorithms, you need to use hopefully better algorithms; that means, that you have 17,000 entries or 18,000, you need to keep some fast structures for excellent data, what is a good example of fast structure probably hashing schemes and notice that this hashing scheme has to change every time you add or delete things, every time I delete a message or every time add a message actually I have to update this hashing scheme also, there has to be also be very fast. So, and that lot of complexity in some of these things.

The reason why certain systems when you use it for the first, time it works very well, but after some period of time it tends to be becoming slower and slower and slower, I think some of you must experience this is because it may be operating very well when the scale is small, once you cross a certain scale it may long may long may no longer work very well, and it can also happen in very devious space for example, suppose I have a system which has got 16 gigabytes of memory, and somebody is keeping this metadata somewhere, right the operating system says that I have 16 gigabytes of memory, it is some piece of information somewhere sitting there. So, it has know how much number is there otherwise it can not manage the number.

Let us say by bad luck the 16 is represented in binary as 0 1 0 0 something that right, 16 gigabytes, it some it has got some 1's and 0's, I shouldnt take 16 gigabytes let us take similar a 24 gigabytes there will have some, 1 1 something right, because it is 16 plus 8. So, you will see the representation of 24 gigabytes there will be 1 and followed by 1 followed by so many 0s, now let us say by sheer bad luck, that one got flipped into 0. So, we thought you had 25 gigabytes in a memory, it may now happen that the operation system if you do not have ecc memory or error correcting codes, if you now things it has

got 16 gigabytes a memory or could be 8 gigabytes a memory depending on where it got flipped.

Suddenly the system what was working very well once upon a time, is no longer working that faster because it is now doing lot more thrashing it may be moving things around etcetera, the same thing can also happen in large scale systems, that is why when you are thinking of this kind of scale that is happening there are is lots of auxiliary structures you keep, the there is nothing which says that the auxiliary structures cannot get corrupted, and that is why the reason why often you will see some people say the database got corrupted, your using for example, fire fox it is keeping track of some information about your browsing history, and it is using a small database sql database something now what is it sql light, and that particular database can get corrupted.

When that get corrupted, you will find that if your fire fox is no longer working, basically some critical piece of information was using basically it is some metadata about how to access whatever history that you had, that sql it is essentially keeping that metadata that got corrupted, and now no longer you can accessing anything, there is you get into some messy situations and usual solution is saying that you delete that some file, the data base file and it will give re automatically re index it again and you get a newer thing. So, all this things capable. So, if you are talking about this large scale systems, you will have lot of auxiliary structures to speed up things, but you also have to be worried about what will happen to those things you they get corrupted, that is something you have to worry about.

In addition there are concurrence issues if you create a file, you need to lock the directory why is that, because you have a directory and your adding something to it right, you want to add something to it you may want to update some structures, I showed you something about hashing structures. So, you can look up fast or even just simple fact that previously I had 999 files, now have thousand files want to update it right, but if want update it there are concurrent operations, what will happen? I want to make sure that that concurrent updates goes through right.

I think some of you have done operating systems courses you will notice that, this concurrent operation has to be done properly you have take lock it. So, the way it is done is you have to lock the directory, now if you lock this directory it can create some other



interesting complications elsewhere. So, it turns out that this directory if it becomes lock bound, then other parties cannot x what is say update that same directory, they get they might see some latencies, but sometimes this can effects they can travel upwards also, what I am mean is saying is that, I want to access something this guys locked it some other operation wants to access this, and there is a substituent operation that needs to lock the directory itself.

That means the this guys getting for this directory, and this guys already locked this directory previously. And so, some other operation might require that other the directory on top. So, basically all this guys upstairs they are all waiting for this guy to finish. So, it can go all the way to the root and the whole system can hang, again some of you might have notice that you have linux system sometimes hangs sometimes, it just seems to be a idle and suddenly moves with a jerk right, some of that could be some of this locking kind of issues, in large scale system.

You have to worry about this, and that is why it is really a magic to me at least, you go to your favorite web browser you type some few keywords and the google system will give you an instantaneous answer, almost the question how does it, do it right it something truly amazing right basically it can give you it is searching something I do not know where it is keeping the information and probably keeping something in australia, in probably who knows malaysia, or japan or somewhere in here.

There I type something here it is going through this networks it is going through this storage systems it is able to figure out and it is searching and give you information in fractions of a second, the question is how can you engineer systems. So, it is in spite of whatever problems I am talking about, it can still do a proper job that is where the challenge is of course, your standard solution for all this things is try to make sure that almost everything that you are using is in memory.

What you do is you keep all the metadata in memory only, do not it a keep it in disc. So, the metadata is a thing which tells you how to look for how to look and get more detailed information about something, if it is in memory already then you are not wasting time trying to access from the disc itself. So, you already the first part of your access already is very fast, and then if you get accurate information you are able to you should do some further caching and other kinds of things, you may be able to get things very quickly say

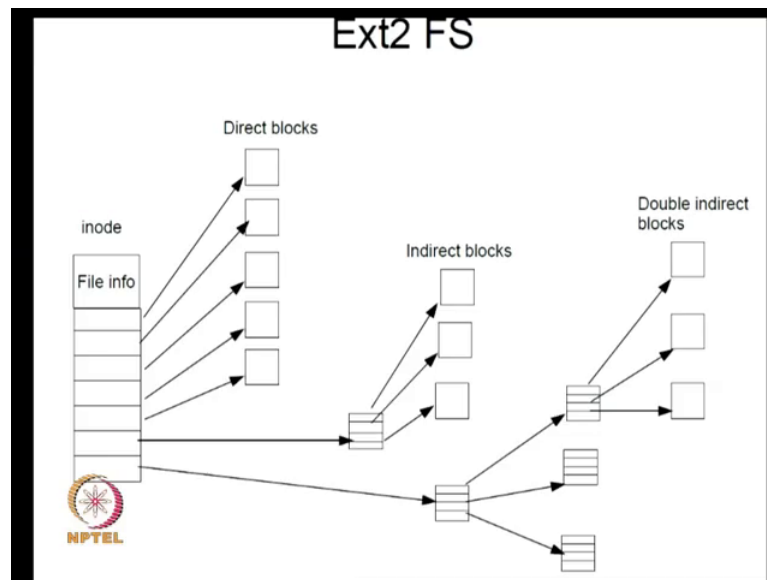
this something I want to think about because this things the reason why this systems are. So, tricky to design is because you have to think about all these things and. So, I was measuring algorithms aspects needs to be taken care of and this is critical with web level storage systems.

Basically this is on a large scale you have to figure out how to which is incorrectly, and because of this it turns out that many newer models have come up what are called key value stores, they are slightly more flat systems compared to file systems, file systems give you lot of functionality because applications use it they require lots of support application require lots of support, and they have the file systems have been developed since 1970's, the kind of file systems we use the history started from mid 1965, basically the our original unix right, the original file system that came is unix is what roughly best still user approximately, and we have kept adding certain new facilities because we program in mostly c now c and mostly the kernel code or the application code is mostly in c.

So, it has got a certain idea about how to request towards resource, and mostly all these same thing that has there is most this thing there with some additional newer facilities, but with the web kind of systems, our access patterns are different we are accessing it mostly in different types of applications for example, this search kind of applications, or the facebook kind applications, it requires certain different model and key value stores are better for it, you do not need the kind of things that we need in the case of file systems, a good example is you if you use such storage system on a particular machine, right let us say you have a pc you have file system running on it.

The file system is going to do things like caching for you, it is going to do some if I already refer do that file once before, I can the next time I refer to it I refer through a handle, I do not refer through a full file name. Now these kinds of facilities are there in file systems, now on a web on a web kind of systems such facilities may not be that critical. So, you can still refer to it using without using, you can give it is still the same name that used before you do not have to find a file handle for example,. So, it terms out that key value stores they optimize in a different direction. So, those things are become common, I can we look into this some of this next time let us just again we will go into some other details later.

(Refer Slide Time: 50:11)



But this is an example of the kind of metadata that is kept in a file system, the is the ext 2 file system that is there in linux systems.

To keep some information about a some piece of data, the metadata that is corresponding that file is called inode these are inode, it has some information about a file for example, the size of the file, when was it created when was it last modified all this stuff is here, these are file information, and it also has the data also, that is basically the blocks of information.

So, the first 4 kilobytes the pointed to it is here of the data, the second 4 kilobytes is here now typically you have about 7 8 9 of this what are called direct blocks, but it turns out that if you use direct blocks all through then basically you need if you have something like 100 megabytes right, you need so many entries here 100 megabyte it does not make any sense. So, we use something called indirect blocks, what they do is I have an indirect block here, and this itself again in turn stores the block at this is. So, this is the first level of indirect blocks. So, this is not us this is not storing the blocks directly now, the address of blocks you have to go to another auxiliary structure and this itself now actually talks to be blocks.

So, normally the essentially what is happening is that you are able to access about 7 into 4 k r 8 into 4 can directly about 32 kilobytes or 28 kilobytes directly, then this will give you access to if it is this is itself is 4 kilobytes, you are talking about thousand addresses

here. So, thousand into 4 kilo blocks 4 megabytes here, now we are able go from about 28 kilobytes we now got to about 4 megabytes, now 4 megabytes is still small somewhere still you will have one more level of single indirect block this is the double indirect blocks, you will now go to up to till gigabytes now gigabytes are still not enough.

So, you will look out to triple indirect (Refer Time: 52:36) upto triple indirect the triple are not doing anything beyond it that is the reason for it what is the problem with these kind of structure, you will find the structures straightly regular find there is direct blocks after sometime its. So, indirect block after sometime you get indirect, but you notice that to access triple in directed blocks first I have to access the first level indirect block, then the second indirect block, third indirect block, then only will have to get the block.

Why is it a problem? There could be a problem because suppose I am doing multimedia, the multimedia I need to get access to blocks between certain bounds, now basically what I am doing right now is that I am telling you right now, if this kind of structure I might be 4 times as low as when I am accessing something here then only here.

That means, your design has to worry about a variation at this level and multimedia files audio especially you can not tolerate too much jitter, every you need to get small amount finds in a audio file it turns out to just need typically about 100 sub bytes is for my speaking for example, or music typically a small amount sub bytes, but I need it every 10 milliseconds and it is very strict. So, if I go to 4 levels of indirect, you know that access to a disc block can take about 10 milliseconds. So, our 4 of those guys; that means, I can as much I can take as much as forty milliseconds.

So, I cannot handle audio kinds this design work. So, this kind of a design is the voltage design, it has stood well as long as our file storage are small or we didn't use multimedia files, you didn't have multimedia a play back like audio and to some extent video. So, that is why nowadays people want structures who are more symmetric, symmetric in sense that, whatever block your are referring to it takes some amount of time approximately, that is why you have something called b trees b plus trees this kind of things are used in more newer types of file systems, I think I will conclude to this point in next class we will look at some slight different types of a systems, slightly a bigger scale like google file system we just give a high level perspective and continue from it.

Thank you.