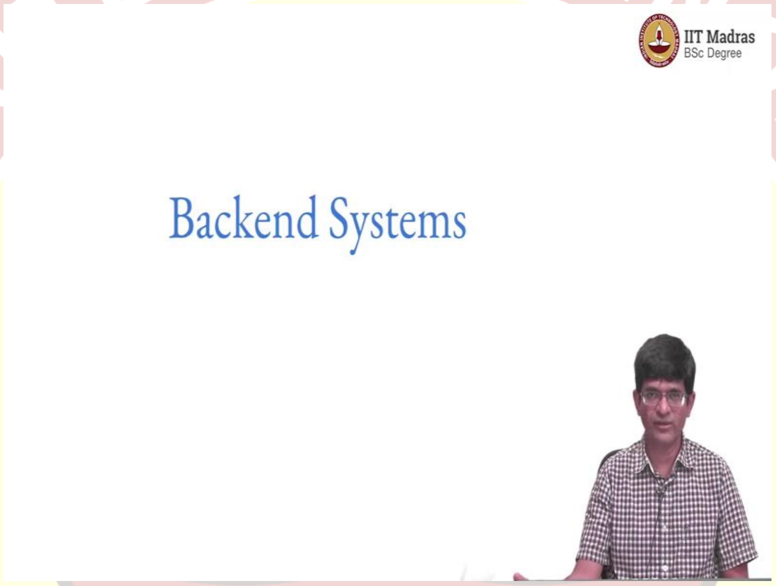
IIT Madras

ONLINE DEGREE

Modern Application Development - I
Professor Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
Memory Hierarchy

Hello, everyone, and welcome to this course on Modern Application Development.

(Refer Slide Time: 00:16)

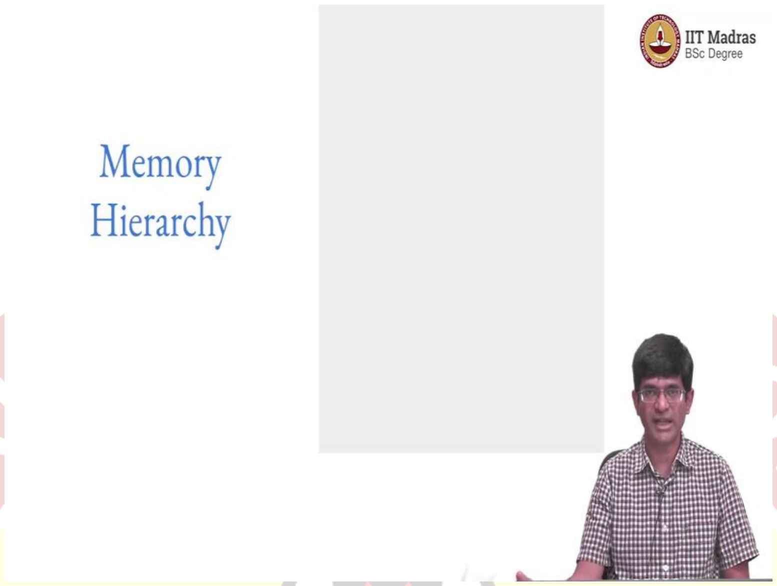


Backend Systems

Hello, everyone, and welcome to this lecture as part of the online B.Sc. course. So far, we have been looking at various aspects of, the process of developing a web app and we have got up to restful APIs. And we now have a pretty good idea of what the structure of the code should look like, what are all the components that should go into the code, what are the parts that take care of handling the data model, what are the parts that take care of presenting the user with views and the controller which basically interacts between the two.

So, now we are going to take in this part of the course we are going to sort of start going a little bit deeper into some of the aspects that have already been touched upon to some extent in the previous lectures, but are ultimately important for an app developer to know in order to make the best use of the resources that you have. So, in this video, we are going to look at backend systems. What exactly do I mean by back end systems and what are all the various variants that we need to keep in mind when you are looking at this.

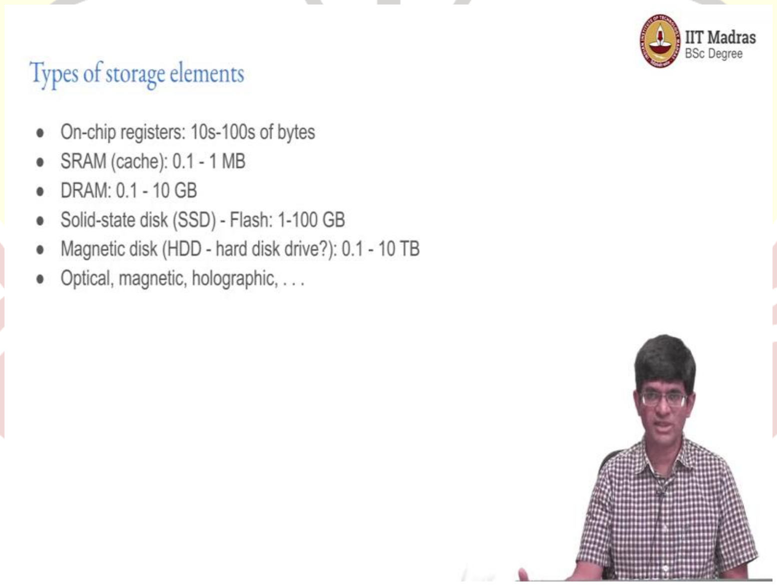
(Refer Slide Time: 01:18)



Memory Hierarchy

So, to start with, I am going to have a brief digression. This is completely unrelated to apps in general, but it is more to do with computers and how they are built and how they operate. And take a little bit of a dive into the so-called memory hierarchy.

(Refer Slide Time: 01:34)



Types of storage elements

- On-chip registers: 10s-100s of bytes
- SRAM (cache): 0.1 - 1 MB
- DRAM: 0.1 - 10 GB
- Solid-state disk (SSD) - Flash: 1-100 GB
- Magnetic disk (HDD - hard disk drive?): 0.1 - 10 TB
- Optical, magnetic, holographic, . . .

So, what exactly do I mean by the memory hierarchy? The first thing is what do we mean by memory? What we are talking about is the various kinds of storage elements that are available for storing data on a computer. The first thing that all of you would be familiar with at least as long as you have done a basic course on some kind of processor architecture or even if you have done something which involves, let us say, programming in Arduino or a microcontroller of some sort, any basic processor, a CPU that we call has one core chip. And one of the main elements of that chip is a set of registers which are used for storing temporary values.

Now, if you have written a C program and you have declared something like int a, there is a good chance that a actually gets declared just as a register, unless it is actually necessary for a to be stored somewhere in memory and you need to have pointers to it and so on. Otherwise, what will happen is the compiler will decide that, okay, you are only using a for some temporary computations, let me store it in registers.

Now, what is the catch and why that. The main thing that you need to understand about registers is they are pretty small in number. You may be able to store typically around 10 and probably at most around 30 or so values in the registers of any modern processor. Each of those values could of course be either 32 bits or 64 bits depending on the type of processor, but you can imagine that this is not really a whole lot if you are trying to write a large program.

The next thing which you may have heard of, especially when you talk about a processor, you have probably heard the term L1 cache or L2 cache that is the C-A-C-H-E pronounced cache. Now, this cache is implemented using something called static random-access memory. You do not need to exactly know why it is called static RAM, although if you are interested then you can read up more about it. The important thing that you need to understand about cache memory is that it is limited in capacity, much better than registers, because you are not talking about tens of bytes or even hundreds of bytes, you can go into several kilobytes, probably on the order of a megabyte or so.

Now, for those of you who have actually tried running programs on computers, you would know that a megabyte is not all that much memory. But those of you who have tried microcontroller programming might actually realize that quite a lot can be done within 1 megabyte. What is the purpose of a cache?

It basically serves as some kind of temporary store of data. And as you can see, it is fairly limited in quantity. But it turns out that access speeds to the cache are much faster than to the main memory, which means it is a, it performs a very nice function, sitting in between the processor and the main memory and allowing you to run parts of programs faster than they otherwise would.

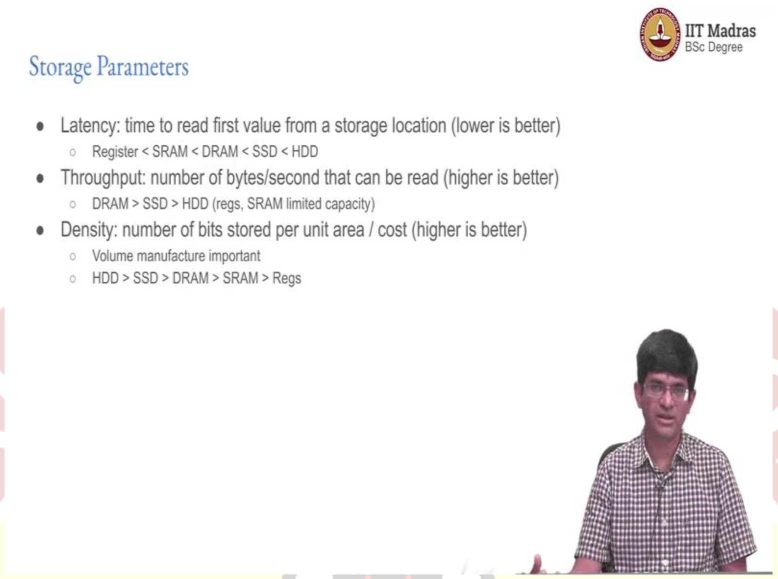
So, you have registers, you have SRAM cache, and then you have the main DRAM, the dynamic RAM in the computer. This is what is typically referred to when you talk about the RAM of your PC. Let us say you have a laptop with 4 GB of RAM. This is it. It is the DRAM that we are talking about. You would like it be as large as possible. Typical amounts these days are between 4 to maybe 16 GB or so. There are of course servers which with much more than that. But that is really fairly high-end servers.

After the DRAM comes what is popularly called these days SSD, solid state disk. Why exactly solid state? It is sort of historical terminology. The main thing is it is implemented using a certain kind of flash-based technology. It is non-volatile meaning that even if the power goes off, the data is retained on that SSD and you can have capacities up to several 100 gigabytes, so obviously, much larger than DRAM. What is the catch? We will get to that in a moment.

And then you have magnetic disks, the actual storage, the hard disk which is used in PCs. This typically these days in the order of terabytes. So, you could have even like a 10 terabyte disk, although more common probably is 2 to 4 terabytes on most systems at the moment. Now, you can go beyond this. There are memory storage technologies that are capable of storing even like hundreds of terabytes, petabytes of data.

Where do you come across petabytes of data? Let us say you are running the Google Data Center. You have to store pretty much all the data for search, all the data in every email that is used by a Gmail user. You are literally running into several petabytes, 10 to the power of 15 order of bytes storage. So, is magnetic disk enough? Are there other technologies that are needed? How do they actually manage this? Beyond the scope of this course, but very interesting topics on their own.

(Refer Slide Time: 06:47)



Storage Parameters

- Latency: time to read first value from a storage location (lower is better)
  - Register < SRAM < DRAM < SSD < HDD
- Throughput: number of bytes/second that can be read (higher is better)
  - DRAM > SSD > HDD (regs, SRAM limited capacity)
- Density: number of bits stored per unit area / cost (higher is better)
  - Volume manufacture important
  - HDD > SSD > DRAM > SRAM > Regs

So, now that we know that these are different ways in which you can store data, let us understand a few parameters that are used in order to actually understand the behavior of these technologies. The first is the so-called latency. In other words, if I want to read a value from memory, how long does it take? And lower is better, meaning that the less the latency, the faster the turnaround time. I asked for data, I get it back.

Registers, like we said earlier, are literally inside the processor core itself. So, the turnaround time, when you try to read something from a register, is on the order of nanoseconds. From SRAM, it could be tens of nanoseconds, possibly hundreds of nanoseconds, but still pretty fast. DRAM latencies can be high, meaning that they could be on the order of several microseconds even, so clearly much slower than SRAM. SSD would be even slower, hundreds of microseconds. And HDD, even the time to get the first bite out of it is in the order of milliseconds, at least, maybe tens of milliseconds, if not more.

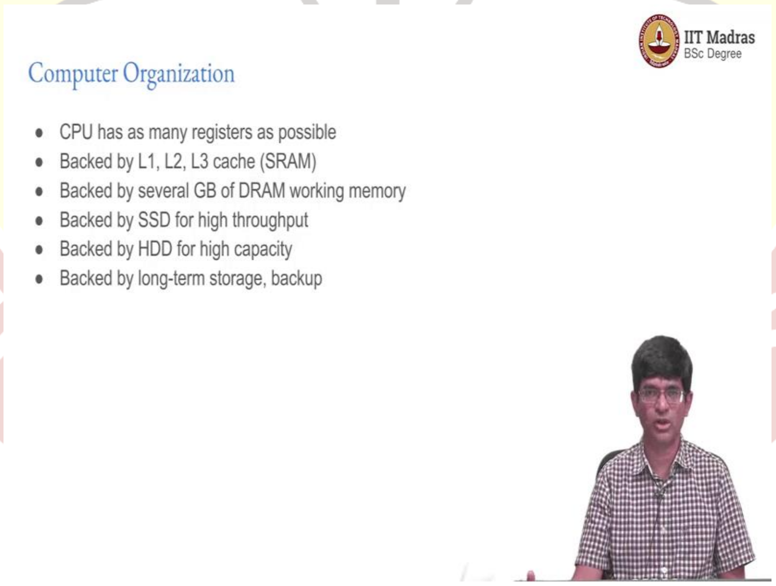
Now, a millisecond might sound fast, but when you compare a millisecond with a nanosecond, you realize that is 10 to the power of 6 order difference. And that makes a huge difference when you are actually trying to run an application fast. So, clearly, from this, we can see that registers are the fastest. But we already saw that capacity-wise there was a problem. The other thing that we have is that another parameter to consider is the so-called throughput, which is the number of bytes per second that can be read out of a system.

Now, throughput is usually not even considered for registers in SRAM because the capacity is so small that you cannot really talk about megabytes per second when you are reading only a few kilobytes. So, instead, at least for DRAM, solid state drive and hard disks, you can clearly say that there is sort of a hierarchy. DRAMs are faster than solid state disks than SSDs which are in turn faster than magnetic HDDs.

Then comes the density. How many bits can be stored per unit area or more importantly per unit cost, because area can be misleading. You might say that DRAM has extremely high density because literally it is taking a fraction of a micron to store a bit, but creating large DRAMs, which can store like hundreds of gigabytes or even terabytes of data is going to be very, very expensive.

So, from that point of view, the volume manufacture becomes very important and we see that the HDDs are the most cost efficient followed by SSDs, followed by DRAM, followed by SRAM, followed by registers, which in some ways also explains why the number of bits stored in each of these follows this trajectory.

(Refer Slide Time: 09:50)



IIT Madras
BSc Degree

Computer Organization

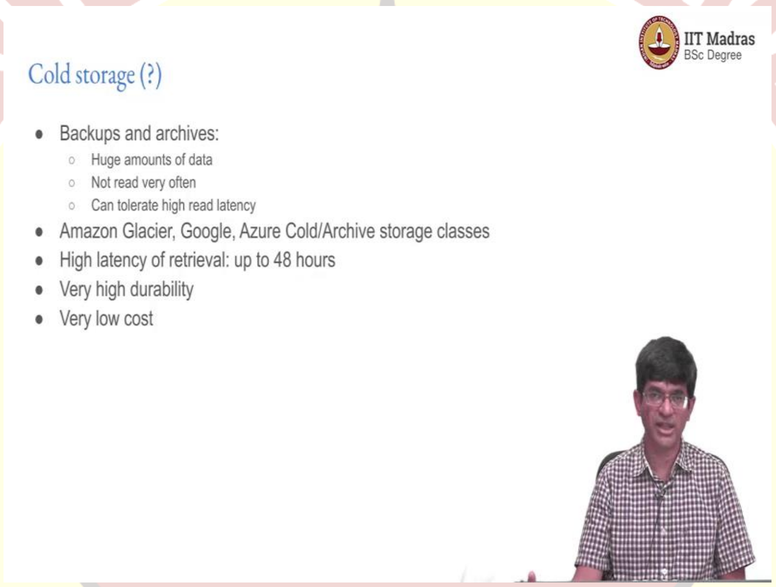
- CPU has as many registers as possible
- Backed by L1, L2, L3 cache (SRAM)
- Backed by several GB of DRAM working memory
- Backed by SSD for high throughput
- Backed by HDD for high capacity
- Backed by long-term storage, backup

So, now, why is all this important to know, because at the end of the day, you are using a CPU and from the point of view of a computer organization, the CPU, you would like it to have as many registers as possible so that your programs run fast. That, those registers are usually backed by various forms of cache memory SRAM that in turn is backed by several gigabytes of

DRAM working memory, some of that then goes to SSD drives, and finally, backed by HDD for high capacity.

Even the HDD is not the end of the story. Ideally, what you want is to have some way by which you can be sure that you do not lose data, which means that even what is stored on HDD finally it goes out to long term storage, backups or some other form of data archival, where you can be reasonably sure that you are not going to lose the data no longer, how long it is sitting over there.

(Refer Slide Time: 10:48)



Cold storage (?)

- Backups and archives:
  - Huge amounts of data
  - Not read very often
  - Can tolerate high read latency
- Amazon Glacier, Google, Azure Cold/Archive storage classes
- High latency of retrieval: up to 48 hours
- Very high durability
- Very low cost

There is such a thing as cold storage. And you might have heard of things like Amazon Glacier, or Google, Azure, Cloud, all of them have some variants of archive storage classes. And the main point over there is that we are talking about backups of data. Let us say that you have all your emails that you have sent, various files, photos that you have clicked and so on, you do not want to lose them. What if your phone crashes or your PC crashes?

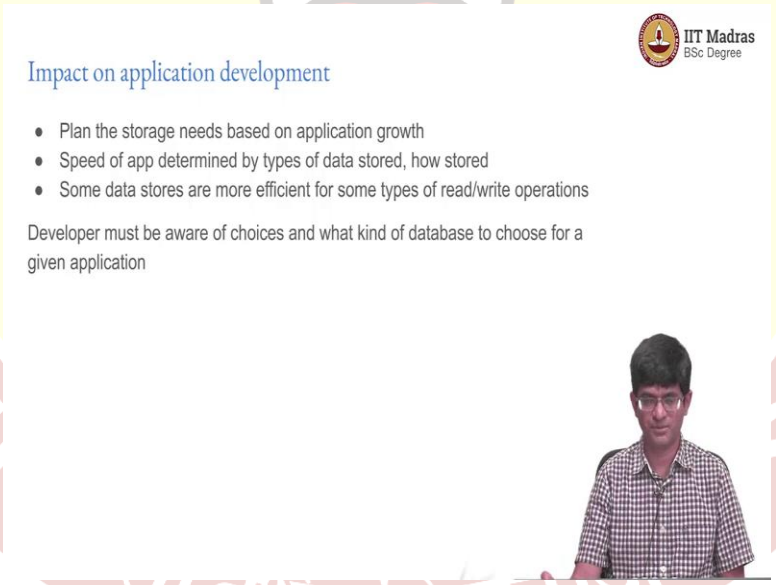
You upload them all onto Google Drive. But how does Google make sure that it does not lose the data. It cannot just put it on one drive over there and say your data safe, do not worry about it. Obviously, a hard disk can crash. So, they have to have backups, which allow them to restore data.

This is going to be huge amounts of data. The petabytes is just the beginning of the story. I mean, this is something which never gets deleted. So, it is only building up with time, which means that you need archival storage, things which can store for long periods of time, at the same time huge volumes of data.

Now, the decision that is usually taken and this is why it is usually called cold storage, is that it will be put on to forms of storage where there is no guarantee on how fast you can get the data back. You might have to wait four minutes or sometimes even hours because Amazon Glacier sometimes says that the retrieval from glacier could take 48 hours. But still the point is the data is there and it is safe.

Why is it safe, because they have put it through so many levels of redundancy and stored it literally in safekeeping wallets, which are then periodically checked, which means that if you need to pull out one particular piece of data you might have to wait for quite a while before it can actually be done. They have very high durability, very low cost for the volumes that are being stored.

(Refer Slide Time: 12:40)



Impact on application development

- Plan the storage needs based on application growth
- Speed of app determined by types of data stored, how stored
- Some data stores are more efficient for some types of read/write operations

Developer must be aware of choices and what kind of database to choose for a given application

Now, why is all this important to you as an application developer? You have an application you are building, it has a database, it stores information, let us say it is a social network kind, social media kind of application. It has information about users. It has something about what users like. It has various other connections that you have built up. And at the same time, you also want to be able to store relationships between those entities that you have created.

Or even if it is not social media, let us say, it is even the NPTEL website, we have records out here, or the online degree website, the NPTEL website already has records of several lakhs of students who have taken courses. And each of those student’s grades need to be stored, their grade cards have to be available.

What, during which semester, did they take what course, all of their information needs to be kept safe. This starts building up after a while. Which means that as an application developer, you need to think about how much is your storage, what kind of impact is it likely to, is the type of storage that you use going to have on the performance of your application.

So, let us say that you decide to just store everything in flat files on a disk, you will rapidly run into performance problems, because just reading and writing from the disk will slow down the number of requests per second that your server can handle. But let us say that you decide to put everything in main memory, either your server becomes ridiculously expensive or you need a large number of servers. And in both cases, what happens if the, if a server crashes? You lose data because it is volatile memory.

At the same time, some data stores are more efficient for certain kinds of operations. So, you might have an application like let us say log file analysis, where you are writing a lot of data, not necessarily reading it very often, some kinds of storage may be more efficient at storing information like that. Some others may be efficient at reading large amounts of data. So, as a developer you need to be aware of your choices, you need to be aware of what kind of servers your system, your application is going to run on and what kind of database to choose for a given application.

Now, this is of course, the first course on application development. So, we are not going to go into detail on all of this. What I will be doing is just explaining these at a fairly high level so that you are aware of these issues. And then we will get on to actually just sort of the different options that are available to you.