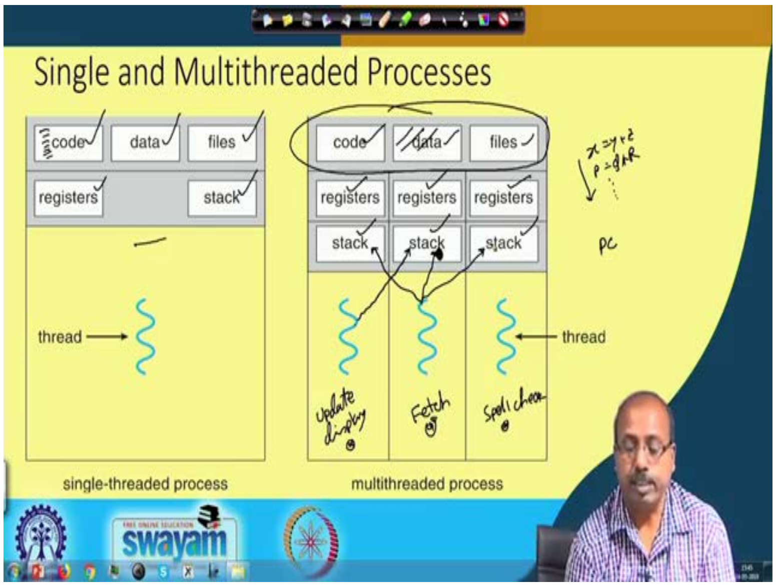
Operating System Fundamentals

Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Threads (Contd.)

(Refer Slide Time: 00:28)



So, next will be looking into the single and multi threaded processes their structure. A process if it has got only one thread of execution, then we will call it a single threaded process a process may have multiple threads of execution. So, simultaneously number of computations are going on, so that situation will call a multithreaded process.

So, as we know any process that we have, so it has got its code segment; it has got its code segment, data segment and stack segment plus it will be it will have the CPU register, so they will have their content plus it will open a few files. So, these are the parts of this; parts of this process and then within the code, so it follows when the program is under execution it is executing line by line. So, this code that we have, so it is executing line by line.

So, that we represent as if there is a single thread of execution through this program maybe the first line is x equal to y plus z second line is P equal to Q into

R, so like that we have got several lines. So, these lines are executed one after the other, so that is a single threaded kernel single threaded process.

On the other hand there may be a situation where if there are multiple threads of execution in the program like previously we have seen an example where maybe in one application we are displaying, we are doing some display, updating some display we are doing something to fetch data over a network and we are doing a spell check on the data that is fetched.

So, there may be I have got three different routine. So, this maybe this thread may be doing an update of display. So, this is update display now this one is say some fetch data and this one maybe that spell check. So, in my program I have got three different functions which are doing this operations, but in previous case what I have to do. So, these three functions I have to call one after the other and as a result execution will be proceeding in a sequence only.

But in this second design the multithreaded process what we are assuming is that I have got say three different threads of execution. So, within the same program we have got three different executions. So, as if this program counter that we have. So, in case of a single threaded process the program counter has got a single value, in case of multithreaded process for each thread there is a value of the program counter.

So, how this can be done? At the same time what do you want is that since this threads are all part of the same application, so they are this data part can be shared. So, whatever data this fetch modulus fetched. So, that has to be updated by this update display module and on that data the spell checker will run to check the spelling. So, this data segment that we have; so this fetch thread, so it will be writing on to this segment whereas, this update display spell check. So, they will also do a read write on those data segment, so this is data has to be common.

So, if they are made three different processes, then the difficulties that does data segment for each of them will be different. So, as a result we cannot do sharing very easily. Whereas, in this particular design style you see that data is same for all the thread. So, whatever modification one thread will be doing, so it is visible to others also, so therefore, them also the data is modified.

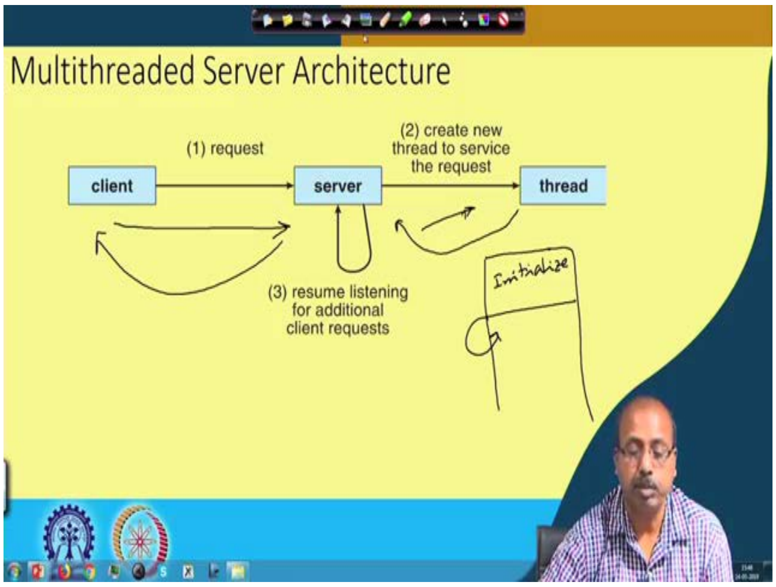
And the code is same, so for all of them the code segment part is same only thing is that the program counter is different and the CPU registers they are different. So, for each thread we have got this registers CPU registers and stack, so these are separate. Similarly for the second thread we have got the content of CPU registers and stack, third thread we have got this thing, but this code data and files, so they are common across all the thread.

So, any modification done to the global data variable, so they are seen by all the threads there, so it is one fellow modifying it is there. And also this threads they have some property like this thread it can see the content of the stack of the other thread though it will not be able to modify it, but it can see the content of that. So, that way if there is some local variable of this thread that is created. So, this is visible to other threads also.

Similarly, this thread can see the local variables that is that are created in the stack or local variables that are created in this stack. So, that can be seen, but the they cannot modify them. So, it can modify only the local variable that are here. So, the only these values can be modified these are these two cannot be modified by this fetch thread. But that is good enough like many a times we are happy to see like what is happening with my fellow thread and then we can just do some operation based on that value. And if there is something to be shared across all the thread, so they will be put onto the data segment.

So, we have got this multithreaded; we have got this multithreaded processes where there can be multiple threads of execution which are going on parallelly across all of them ok; across all the threads.

(Refer Slide Time: 06:01)


Multithreaded Server Architecture

client
(1) request
server
(2) create new thread to service the request
thread
(3) resume listening for additional client requests
Initialize

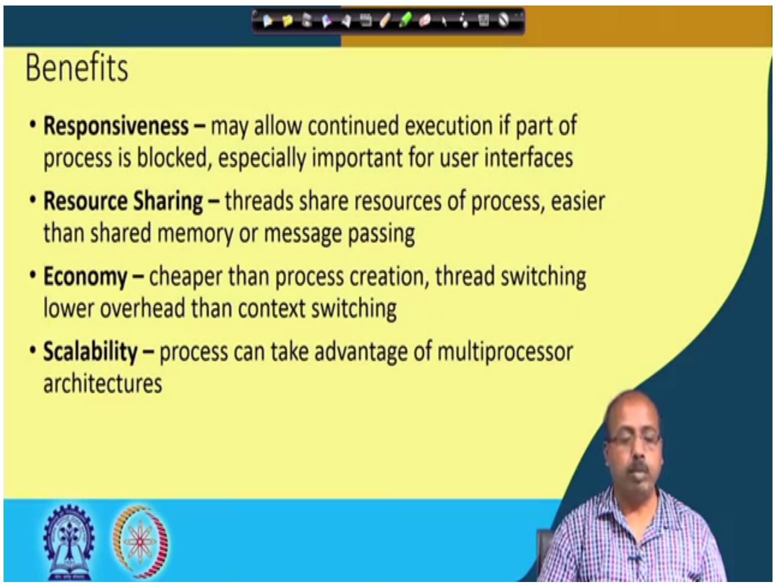
So, this is a typical example of a multithreaded server architecture. So, we have got a server; so how this server is designed? So, if you look into the code for this server you will see that it first does some initialization. So, there is a portion of the server code which performs initialization and then after initialization; after initialization it waits in a loop for some client request to arrive. So, this is the first part it does some initialization global initialization and then it waits for the requests to come for clients.

So, now, suppose one client process, so has sent a request on this. So, what the server does is that it creates a new thread to service that request. So, it will create a thread that will serve this request and the server will continue listening to some additional client request. So, every client request comes, so this is a one the new thread is created and that way that thread takes care of the client request. So, when this thread is over, so it has got this information then this thread will be telling the; telling the server that now I am over. So, server will know where is the computed value or the written value that has to be passed to the client and accordingly it will send it back to the client.

So, for example, suppose assume that this is a database server and these are different people who are putting some queries to the database server. So, the first query comes; so this database server creates a thread which will be taking care of that. So, it will look into the appropriate records of the database and then maybe calculate some value and the written value is stored at some place and then it involve informs the server that my job is over. So, this thread is; this thread is over. So, then this server will take the value and may give it back to the client program the client that has requested.

So, this way this client server mode of operation works and since this requests that are coming from clients are all similar in nature, so I do not have to create different processes for them because if I want to create process. Then the amount of memory that will be needed will be more because the data and stack segment have to be copied, then this we have to create separate data segment like that data segment stack segment like that. So, that makes it difficult. So, that is actually taken care of by in this particular module.

(Refer Slide Time: 08:41)



Benefits

- **Responsiveness –** may allow continued execution if part of process is blocked, especially important for user interfaces
- **Resource Sharing –** threads share resources of process, easier than shared memory or message passing
- **Economy –** cheaper than process creation, thread switching lower overhead than context switching
- **Scalability –** process can take advantage of multiprocessor architectures

Now, what is the benefit that we get? The first benefit that we have is the responsiveness. So, it may allow continued execution if part of process is blocked, say suppose one thread due to some reason or the other gets blocked like if there is a networking problem maybe the module the or the thread that was fetching data over the network, so that is blocked.

But somebody; so some other thread which was updating the display. So, that can go on, so that display update can continue or the spell check module can continue ok. So, that way this can be done or if there is in case of database query processing, if one query has got some difficulty in its execution. So, other queries are not stalled because of that ok.

So, that way this responsiveness is very important specially for user interfaces like we have got say. So, one window we have got multiple windows created on the screen, now each window can be one thread. Now, if one window due to some problem or the other becomes unresponsive. So, we go we switch over to other window and possibly that window continues. So, that is one good thing that we can have.

On the other hand if this was not there, then all the windows they are serviced one after the other. So, one of them getting blocked, so others will also get blocked because they will not get turn for service. Then the resource sharing threads shared resources of process easier than the shared memory or message passing.

So, previously we have seen at the process level if you want to share information or say we have to either have some shared memory or we have to have some sort of message passing between the processes, but both of them are costly because the shared memory is limited in size and this message that you want to pass. So, message queues are also used also implemented by using some system level memory, so, that is also limited.

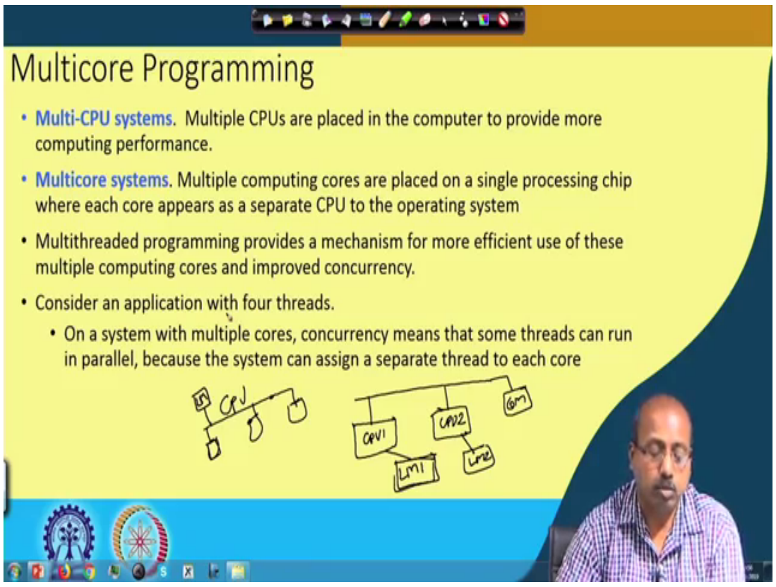
So, if we; so that way if you have got large number of processes and you want to associate one mailbox with each such process, so it becomes resource it is not very friendly to the resource that we have in the system. Whereas, in case of threads I do not create separate resources like the same data segment is common to all the threads. So, that way the amount of space required may be much less compared to the process.

So, it as far as the economy is concerned, so it is cheaper than process creation. So, cheaper than process creation because you have to you do not need to create the data segment stack segment like that. So, you can just take it as it is, so that is the data segment is common. Then stack segment of course, we have to create because each thread will have its own stack, but this many other things like that the, so they are all shared.

And threads switching is lower overhead because in case of thread switching many of the context switching jobs will not be required like this memory limit register loading and all. So, they will not be required because all of them. So, all the threads they have got unrestricted access to the entire address space of the program or of the process.

So, we do not have to modify this limit registers and all. So, that way it is much more economical to have this thread level switching rather than this process level switching. And scalability the process can take advantage of multiprocessor architecture. So, if you have got multiprocessor architectures then of course, you can take help of that and you can utilize this threads effectively for exploiting them.

(Refer Slide Time: 12:31)



Multicore Programming

- Multi-CPU systems. Multiple CPUs are placed in the computer to provide more computing performance.
- Multicore systems. Multiple computing cores are placed on a single processing chip where each core appears as a separate CPU to the operating system
- Multithreaded programming provides a mechanism for more efficient use of these multiple computing cores and improved concurrency.
- Consider an application with four threads.
  - On a system with multiple cores, concurrency means that some threads can run in parallel, because the system can assign a separate thread to each core

So, next we will look into multicore programming. So, multicore programming as such it; so there are two terms which are often confused one is multi CPU systems and another is multi core system. So, multiple the multi CPU system, so they means that multiple CPUs are placed in the computer to provide more computing performance. So, we have got multiple CPUs altogether. So, they are totally independent and they are, so if you are if you think in terms of this like we have got say CPU 1, CPU 2 like that.

So, we give different jobs to do them. So, then we have got the they may have their own memory. So, maybe I have got these local memory 1, local memory 2 ok. So, this is for this CPU 1, this is for CPU 2 and there may be a global memory somewhere here over a bus. So, this maybe some global memory and if you want to access the global memory maybe it has to be done like this. So, which is a processor has got a local memory as well as overall system as a global memory.

So, if you look into any computer architecture book, so we will find this type of architectures where the CPUs are placed in the computer to provide more computing performance. On the other hand the multi core systems, so multiple computing cores are placed in a single processing chip. So, I do not have separate CPU, so within a CPU I have got different cores so, and where each core appears as a separate CPU to the operating system.

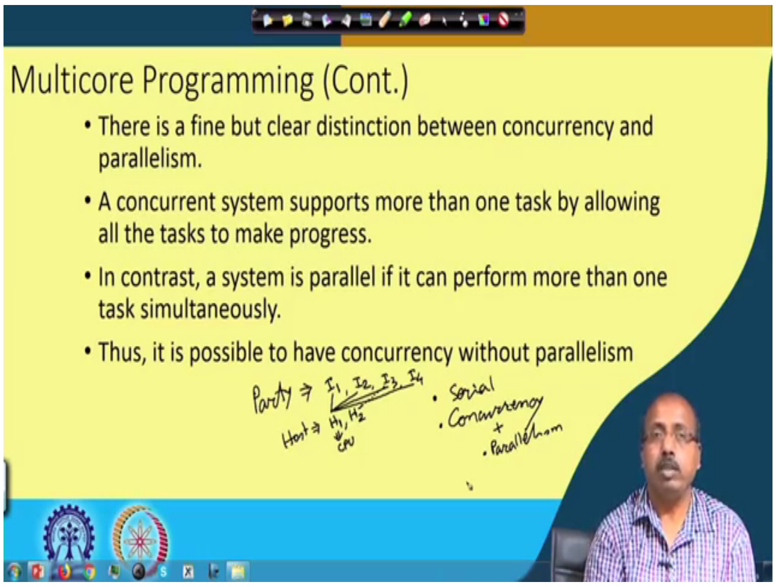
So, we have go; so within the single CPU we, so this is a CPU. So, within that CPU itself I will have a different cores there. So, now, how this operating system will utilize this cores, so that is an issue of course, but if you can exploit that, so in this case this each processor having local memory, so that concept is not there. So, this CPU itself has got a local memory. So, that is visible to all the processors; however, in this case you see that the local memory of CPU 2 is not visible to CPU 1, but here all the cores they can have a; they can have this access to the local memory. So, that way the operation becomes easy.

So, this type of situation is a multi core system we have got multiple computing platforms available within the same CPU compared to a separate CPU. So, this multithreaded programming, it provides a mechanism for more efficient use of this multiple computing cores and improving the concurrency. So, that is the advantage of multithreaded programming. So, the programs that are written to exploit this multicore architecture. So, I have to create separate threads and each thread will go to one such core and that way there will be parallelism.

So, multithreading; so suppose we have got an application with four threads on a system with multiple cores. So, concurrency means that some threads can run in parallel because the system can assign a separate thread to each core. So, if I have got four cores; if I have got four threads, now if I have got say two cores in my system, then two of the threads can be given to one core and two of the threads can be given to other core.

Similarly, if I have got four cores then the one thread can be given to each core. So, that way the distribution has to be done. So, these distribution though it is set very easily, but it is not that easy to do and the operating system must take special care for this distribution, so the two of this threads to cores. So, that is important, but if this threading concept was not there then we cannot think about exploiting this core architecture. So, that is why this threading and this cores, so they come hand in hand in providing the flexibility to the programmer.

(Refer Slide Time: 16:34)



Multicore Programming (Cont.)

- There is a fine but clear distinction between concurrency and parallelism.
- A concurrent system supports more than one task by allowing all the tasks to make progress.
- In contrast, a system is parallel if it can perform more than one task simultaneously.
- Thus, it is possible to have concurrency without parallelism

So, there is a fine, but clear distinction between concurrency and parallelism, so this thing we try to understand. Like suppose I have got say, I will take an example of some say suppose we have got a party in our home and in that party there are several invited guests. So, I have got the invited guests I 1, I 2, I 3 and I 4 ok. So, they are all invited guests and they are definitely we have a host. So, suppose there is a single person H 1, so who is the host and the host has to entertain each of these invited members invited members to the party. Now, what can happen is that this host may talk to I 1 for some time and then after finishing with I 1, so the host goes to I 2 ok. So, they will talks to him or her.

So, like that so, but the; so what happens is that this I 1 when the host is talking to I 1, so the host is not listening to others I 2, I 3, I 4, so like that. So, this is basically I have got a this host is basically the CPU or the server that you have ok. So, or they are there is a; so they it is a purely sequential process that is going on ok. So, this individual invitees, so they are attended one by one.

On the other hand it may so happen that, so that is a serial type of attendance attention that is given. Now, suppose we think about a different situation in which this host talks to I 1 for some time and then goes to I 2, then after sometime goes to I 3 then, after sometime I 4 and then again goes to say I 1 then to I 2. So, the host does not finish the full conversation with I 1 before going to I 2 only part of it is done.

So, as a result this all this invitees they will feel that the host is interacting with me; host is interacting with us. So, this is basically the concurrency. So, all this invitees, so they are feeling that I am being sobbed I am being a attended by the host. So, this is the there is concurrency in the attention. So, all the hosts all the invitees, so they are attended to in some sense concurrently. However, you see, but host is only one, so at as a result at one point of time the host is not talking to two different invitees ok, so it is not a parallel one.

So, it is concurrent, but it is not parallel now suppose I have got in the host instead of one hosts, so there are two hosts H 1 and H 2, then what can happen? I can have concurrency plus parallelism also because when H 1 is talking to say I 1, so H 2 may be talking to I 3. So, when H 1 is talking to I 1 simultaneously H 2 maybe talking to I 3. So, that way you have got parallelism also.

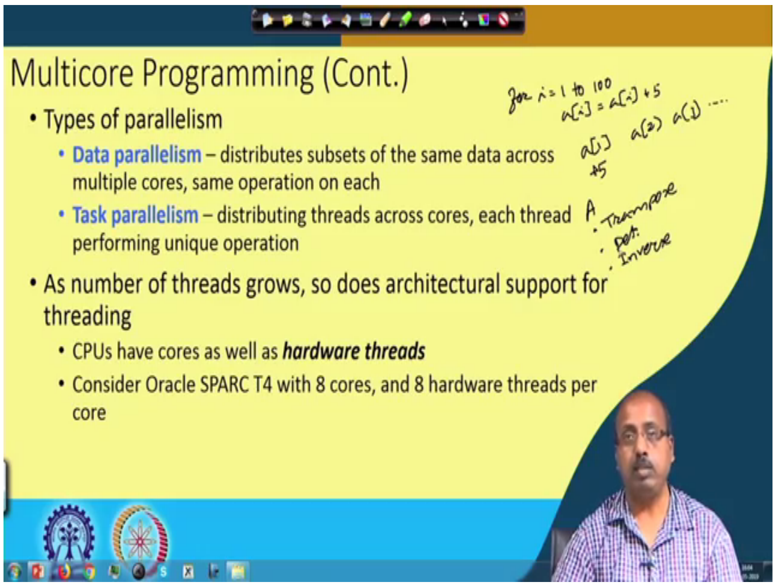
So, we have got pure serial type of operation, we can have concurrent operation, we have got concurrent and parallel operation. So, there is a slight difference between these concurrency and parallelism that we must understand. So, in terms of this process execution by the CPU, so I can have a single CPU which is executing the processes.

So, it may take up each process one after the other, ones the processes taken up, so it completes the entire process and then goes to another process. So, that is its pure sequential type of computation. Or it may so happen that the CPU is computing for first process for sometime, then it goes to second process for again for some small amount of time and after finish after sometime it comes back to process one again.

So, that way there is a concurrency because all the processes they are being handle concurrently, but that is not parallel because at one point of time the processor is given to only a single process. But if I have got multiple cores or multiple CPUs, then I can do this parallel processing also. So, we have got concurrency plus parallelism. So, a concurrent system it supports more than one task at a more than one task by allowing all the task to make progress, so that is the. So, all the tasks are progressing, so that is a concurrent system.

In contrast a system is parallel if it can perform more than one task simultaneously. So, concurrent means all the tasks are progressing, parallel means all of them are taking place simultaneously. So, concurrency does not mean parallelism because they may not be progressing simultaneously, but they may they may be done one after in some intervals which is not parallel. So, it is possible to have concurrency, so you can have concurrency without parallelism also.

(Refer Slide Time: 22:01)



Multicore Programming (Cont.)

- Types of parallelism
  - Data parallelism – distributes subsets of the same data across multiple cores, same operation on each
  - Task parallelism – distributing threads across cores, each thread performing unique operation
- As number of threads grows, so does architectural support for threading
  - CPUs have cores as well as ***hardware threads***
  - Consider Oracle SPARC T4 with 8 cores, and 8 hardware threads per core

for i = 1 to 100
a[i] = a[i] + 5

a[1] a(2) a(3) .....
+5

A
· Transpose
· Det.
· Inverse

So, if you try to look into parallelism that is supported in multicore programming. So, we have got data parallelism and we can have task parallelism. So, data parallelism it says it distributes the subsets of the same data across multiple cores and some with same operation on each. So, what we are doing is that the whole data set I am distributing to a number of cores and each core is doing the same operation.

For example, if I have to say I have got a simple operation like for each array element I want to increment it by 2. So, it may be that I want to have this operation for i equal to 1 to 100 a i equal to a i plus 5 ok. Now, if I have got say; if I have got say 100 different cores, then what we can do? I can give you a 1 to one core, a 2 to another core, a 3 to another core, so like that I can give one to each of them and then I can tell each of this cores to increment by plus s. So, that is the data level parallelism.

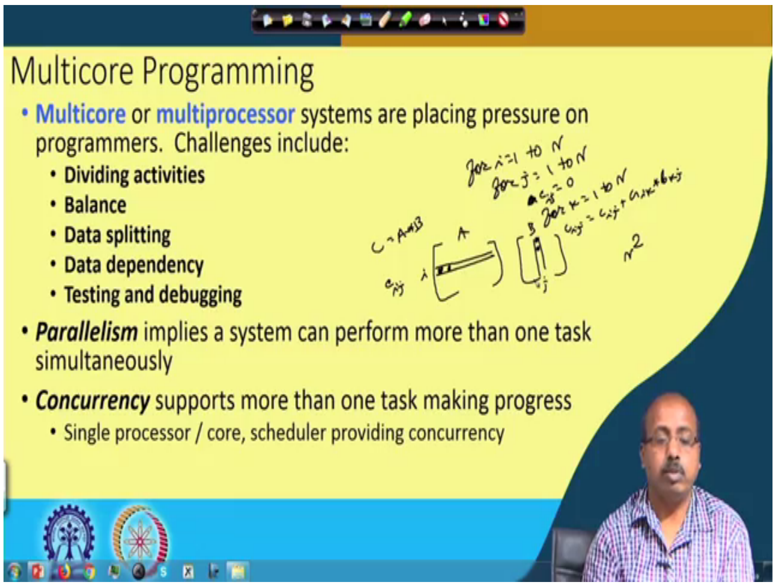
So, instruction is same for all the cores. So, all of them are just incrementing by 5. So, operation is same, but the data on which they operate they are, so a 1, a 2, a 3 up to a 100. So, distributes in incase of data parallelism, so, it distribute subsets of the same data across multiple cores, but same operation is done on each. The other hand we can have task level parallelism, so it distributes threads across cores and each thread performing unique operation.

As I was telling like suppose I have got a matrix and I have to do several operations on that, so one they have. So, I have got a matrix A and maybe one job is to do A transpose, one job is to find say the determinant of the matrix another job maybe to find the inverse of the matrix, so like that. So, for the same data I want to do three different jobs. So, I can create three different thread for doing each of them.

So, here that data is same, but the thread the task is tasks are difference, so each core is doing one, so one operation. So, distributing threads across cores each thread performing some unique operation. Now, as the number of threads grows, so does the architectural support for threading. So, this is if you have got large number of threads to be supported, then the basic underlying hardware should support the this threading, like CPUs have cores as well as hardware threads.

For example, Oracle SPARC T4 that has got 8 cores and 8 hardware threads per core. So, we have got 8 core, so they can do 8 different operations parallelly and we can also have 8 each core can do 8 different operations together. So, that way we have got this hardware threads and hardware cores.

(Refer Slide Time: 25:17)



Multicore Programming

- **Multicore** or **multiprocessor** systems are placing pressure on programmers. Challenges include:
  - Dividing activities
  - Balance
  - Data splitting
  - Data dependency
  - Testing and debugging
- ***Parallelism*** implies a system can perform more than one task simultaneously
- ***Concurrency*** supports more than one task making progress
  - Single processor / core, scheduler providing concurrency

Now, so this multicore programming; so multi core or multiprocessor systems are it play it places the pressure on the programmers because now it is not only writing a correct program that is important. So, what is more required is that I should be able to exploit the underlying thread threading facility or the architectural level support that is there, so I want to I have to exploit that. So, I have to be able to divide the activities between the threads, I should be able to divide the activities among the cores. Then there should be balancing, so maybe I do a division, but the division is not balanced.

So, as a result the thread which takes maximum amount of time will determine the application run time. So, that becomes a bottleneck sort of thing, so I have to be do a balancing. Now, how do you split the data? So, it is not very easy to split the data. For example, the previous previously I said that for i equal to 1 to 100 a i equal to a i plus 5.

So, they are the splitting was very easy, but if so happens that this a i computation depends on some other a elements of this a array as well, then taking the appropriate data elements together. So, that this decision can be taken by individual cores without bothering the data in available in other cores, so, that is that becomes a problems. So, how do you split data across this thing, so that becomes a problem.

A typical example can be like say for matrix multiplication you know that we have got three nested loops like for i equal to 1 to, for J equal to 1 to N we have got a c i J if I am doing like the C matrix is equal to A matrix into B matrix, then I have to do like c i equal to sum 0 and then for k equal to 1 to N c i J equal to 0. So, c i J equal to c i J plus a i k into b k j.

So, essentially what I am doing is that if this is the A matrix and this is the B matrix; this is the B matrix, now to get the element c i J I have to take the ith row from this a matrix and I have to take the Jth column from the B matrix and then I have to do element by element multiplication first element will be multiplied by the second element multiplied by second element. So, like that I have to do element by element multiplication and do this.

So, if you are asked to do a data distribution, then ideally I should have N square number of processors and for a particular processor I should give one row of a matrix and one column of J B matrix for doing the multiplication. So, that way it becomes, now if you have got less than N square number of core, then how do you do this distribution? So, that is an important question to be answer.

So, this data splitting is not very easy, so there will be data dependency also. And finally, the testing and debugging becomes difficult because you need to you need to test the system. So, sequential program testing is easy, but something is going on parallel, then what is the problem how, if there is some erroneous result coming the what is the cause of that, so debugging becomes difficult.

So, we can have parallelism that implies that system can perform more than one task simultaneously whereas, concurrency supports more than one task making progress. So, it may be single processor per code or in that case the scheduler will provide concurrency because scheduler will put one job for sometime, then take it off put the second job for sometimes it goes that way. So, concurrency is provided by means of the scheduler.

Whereas, if you have got multiple processors then of course or multi core system, then this parallelism and concurrency that is automatically ensured once you have distributed tasks among the cores. So, this way this multicore programming becomes a challenge and many of the programming environments that we are having now. So, they are supporting this multicore programming as a special thing.