High Performance Computing for Scientists and Engineers

Prof. Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Module - 01
Fundamentals of Parallel Computing
Lecture - 04
Architecture for Parallel Computing (continued)

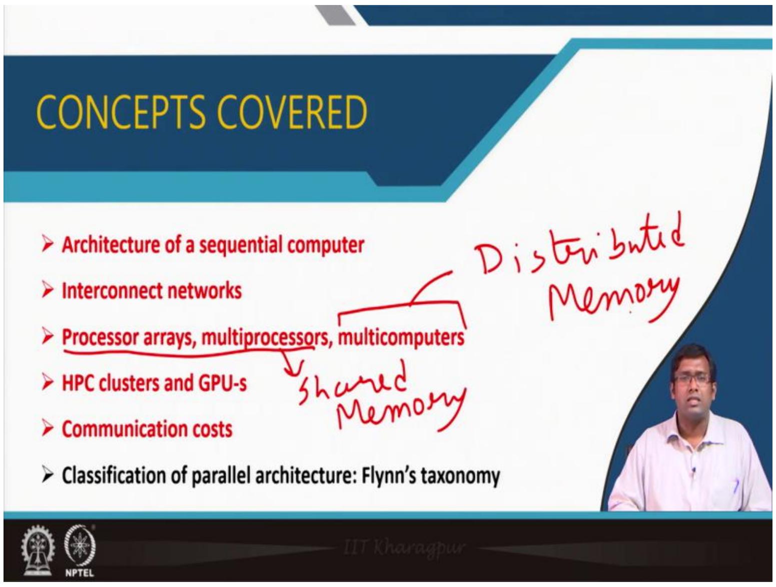
Welcome, we are in the class of High-Performance Computing for Scientists and Engineers. This is the 1st module Fundamentals of Parallel Computing on which discussions are going on. And we are continuing with the lecture on Architecture for Parallel Computing.

Well this course, as it is titled high performance computing for scientists and engineers is meant for a wide class of audience with backgrounds from different fields, which is not only strictly computer science, but also students from physics, chemistry, mathematics, biological science, different fields of engineering and other science application domains.

Therefore, we can very well assume that all the students do not have background in computer science especially, in computer architecture. So, while discussing computer architecture, in these few classes, I will try to give some fundamental ideas on parallel computer architecture. In keeping the fact in mind, that many of the students do not have this background and the concepts are very new to them, we will repeat some of these concepts later in this class. Maybe in the next class, we will see some of the concepts, at least some pointers to the concepts, which are discussed in today’s class will be touched upon. And later also when we will do HPC programming, we will sometime get back to these concepts again.

Again for the wider audience, specially who are not from computer science background, we are discussing the concepts on parallel computing architecture because when one will try to implement HPC for their own scientific or engineering purpose, or when somebody will try to apply some of the high performance computing techniques, well developed techniques, commercially available or community developed software’s which use high performance computing resources some time they need to have some idea on the architecture on which they are working. That is the key to get best performance using high performance computing.

(Refer Slide Time: 03:17)



CONCEPTS COVERED

- Architecture of a sequential computer
- Interconnect networks
- Processor arrays, multiprocessors, multicomputers
- HPC clusters and GPU-s
- Communication costs
- Classification of parallel architecture: Flynn's taxonomy

Distributed Memory
Shared Memory
IIT Kharagpur
NPTEL

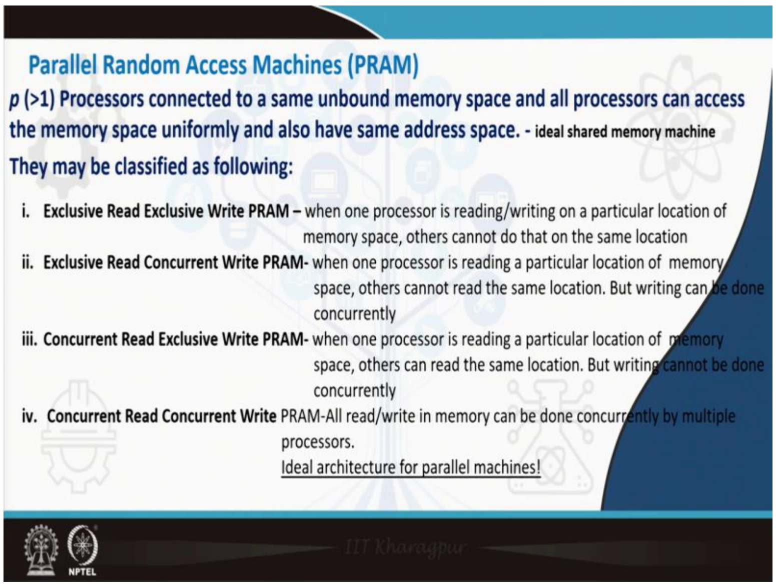
So, you need to discuss the concepts and as I said we have already discussed few of the concepts here, like we started from architecture of a sequential computer. We discussed about interconnect networks how different processors can be connected via bus or how different processors and their memories can be connected via inter connect network switch. We have discussed about processor arrays, multi processors and multi computers and one might note down, that the processor arrays and multi processors they give a shared memory platform for high performance computing or parallel computing while multi computers give distributed memory platforms. In the subsequent class, we will discuss in detail about shared memory and distributed memory.

Then we have some discussion on how HPC clusters look like from the architecture point of view and how are GPU-s or graphics processing units. Then we discussed about communication costs because, when multiple computers are working together, there is a need to communicate in between them in terms of exchanging information or data. And apart from the computers processing cycles, apart from the number crunching that the computer is doing, how much time is allotted for the communication across processors.

So, today we will be looking into classification of parallel architecture and we will see something named Flynn's taxonomy. As we have discussed in one of the previous lectures also, that computer architecture is an evolving area, what we see today's GPU was not there 20 years back. And what was the fastest supercomputer 20 years back, probably if we consider its performance, it will probably not be among top 5000 supercomputers today. So, computer architecture is an evolving area in many aspects.

However, Flynn’s taxonomy is something which classified the parallel computing architecture, and even today any of the high-performance computing platform can be classified in one of the classifications given by professor Flynn and that is why it is important to look into Flynn’s taxonomy too.

(Refer Slide Time: 05:57)



Parallel Random Access Machines (PRAM)

p (>1) Processors connected to a same unbound memory space and all processors can access the memory space uniformly and also have same address space. - ideal shared memory machine

They may be classified as following:

i. **Exclusive Read Exclusive Write PRAM** – when one processor is reading/writing on a particular location of memory space, others cannot do that on the same location

ii. **Exclusive Read Concurrent Write PRAM**- when one processor is reading a particular location of memory space, others cannot read the same location. But writing can be done concurrently

iii. **Concurrent Read Exclusive Write PRAM**- when one processor is reading a particular location of memory space, others can read the same location. But writing cannot be done concurrently

iv. **Concurrent Read Concurrent Write** PRAM-All read/write in memory can be done concurrently by multiple processors.
Ideal architecture for parallel machines!

So, one important concept is Parallel Random-Access Machine or PRAM. See there are more than 1 processors, p number of processors, p is greater than 1 connected to a same unbound memory space. It is a huge memory space in which many computers are connected, and all the processors can access the memory space uniformly so, it is a large UMA memory space (it is a same memory space where all the computers are connected). And there a large number of processors, they are connected to the same memory space and they have the same address space.

So, any variable a for one processor is seen by all other variables, it is a same variable which is shared by all the processors. So, we call it to be ideal shared memory machine. It’s a shared memory machine, where the memory has no bound. It is a large memory, and there is no restriction regarding memory and any number of processors, (processor number greater than 1) can be considered for this.

And this parallel random-access machine can be classified as following: One can be exclusive read, exclusive write PRAM, that is when one when processor is reading or writing a particular location of the memory space, other cannot do that on the same location. So, it is a same address space which is being shared by a number of processors. But if one processor reads or writing one of the locations in the address space, that location the other computers cannot access.

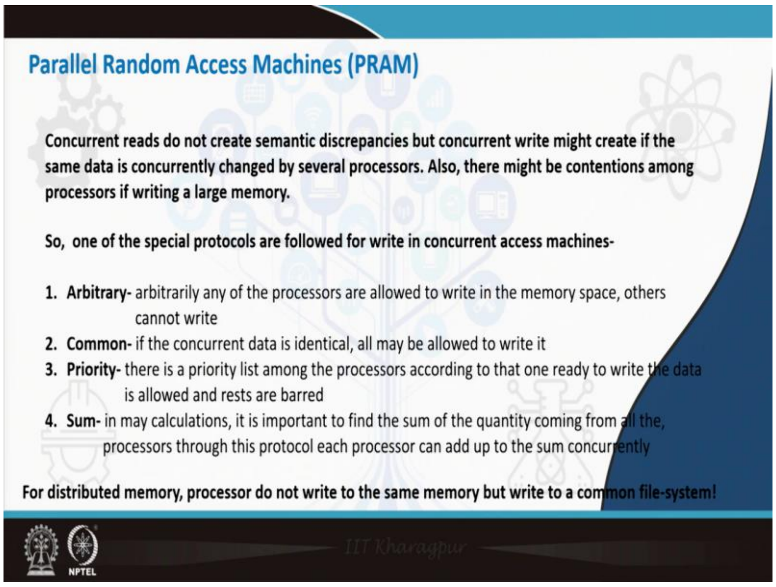
So, if somebody is reading some computer is reading a memory or writing a memory, other computers cannot do that and they therefore, there has to be some sequentially if one computer is doing that, others have to wait. Parallelly a large number of processors cannot or primarily more than one processor cannot go to the single a particular memory location.

The next is exclusive read and concurrent write. When one processor is reading a particular location, others cannot read the location, but when one is writing to that locations, other processors can do write concurrently. So, at the same instance, many processors can write the same memory location.

Concurrent read exclusive write which is the reverse case of this, that one processor is reading a particular location then, other processors also can read that location. But multiple processors cannot write in the same location concurrently. Writing in the same location concurrently has an issue, that if all the processors try to change the variable at the same address space, there can be certain conflict or contention which we will see in the next slide.

And the 4th one is called concurrent read or concurrent write. All read write in the memory location can be done concurrently by multiple processors. That means, if one processor accesses to one memory location and reads or write there, the other processors can do the same operation on the same, involving the same memory location and there will be no wait time. There will not be any wait time for any of the processors because some other processor is working on the same memory location. And therefore, all the processors can parallelly operate on the same memory location and this will be the ideal architecture for parallel machines.

(Refer Slide Time: 09:41)



Parallel Random Access Machines (PRAM)

Concurrent reads do not create semantic discrepancies but concurrent write might create if the same data is concurrently changed by several processors. Also, there might be contentions among processors if writing a large memory.

So, one of the special protocols are followed for write in concurrent access machines-

1. **Arbitrary-** arbitrarily any of the processors are allowed to write in the memory space, others cannot write
2. **Common-** if the concurrent data is identical, all may be allowed to write it
3. **Priority-** there is a priority list among the processors according to that one ready to write the data is allowed and rests are barred
4. **Sum-** in may calculations, it is important to find the sum of the quantity coming from all the, processors through this protocol each processor can add up to the sum concurrently

For distributed memory, processor do not write to the same memory but write to a common file-system!

Now, let us see what are the issues, if we think of a concurrent read, concurrent write module. Concurrent reads do not create semantic discrepancies because it is the same memory location everybody is reading; nobody is going to change anything there. There will can be some contention because due to bandwidth etcetera, all the processors are trying to jump into the same memory location. Like you have a newspaper and you are going in a in a public train and 2 people from your right and 2 people from your left wants to see the headlines of the newspaper. So, they can be little contention; however, everybody can read the same newspaper.

But concurrent write, you have a piece of paper and you were writing something and 4 people from your both your left and right are jumping together to write at the same location, that is difficult. Concurrent write means the same data is concurrently changed by several processors. So, one processor is trying to write a is equal to 1, the other jumps on there and tries to write a is equal to 2, the next one jumps there in and tries to write a is equal to 3. There might be contention among the processors, if they are writing a large memory, if they are writing lot of data in a particular space and writing together there might be contention, there might be conflicts also. Therefore, some of the spatial protocols are to be followed if we try to access concurrent write.

If we try to say that processors can concurrently write on the same location, we have to use some protocols and what are they? Arbitrarily any of the processors are allowed to write in the memory space and other cannot write that. So, though we say concurrent writing access given to all the processor, but when say more than one processors are trying to write in the same memory location to avoid any contention, we will allow one to write there once his writing is finished, the next one will be allowed to write there. Though it is a concurrent write, but there will be some sequentially in the write and who will write, and who will not write, that will be decided arbitrarily.

Common. If the concurrent data is identical, all may be allowed to write it. If everybody is writing a is equal to 2, then everybody can write that. Then, no other processor has to wait for writing it and these protocols are followed so that there is no conflict among writing in the same location.

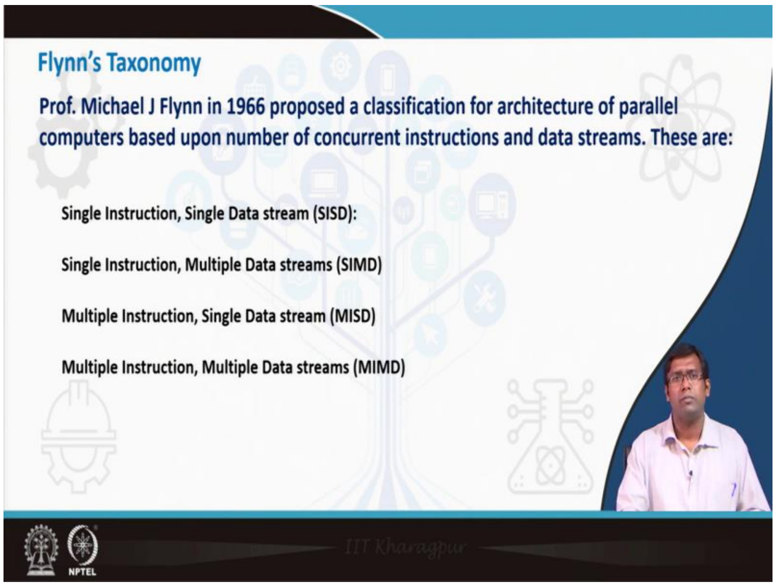
Priority. There is a priority among the processor say processor number 5 is given priority. So, if you try to write a it will write it, others cannot do. Once its writing is finished, the next processor in the priority list will be allowed to write it and the rest will be barred. And the 4th one is important which is very important in many of the scientific computations, that in many cases the processors work in parallel, find out some value and finally, you need a sum of the values obtained by the processors.

Say you have to compute an infinite series of course, you cannot compute infinite terms in the series, but you think to compute first 1 million terms of that series. And in infinite series is basically sum of first 1 million terms in the series, it is near infinite series I will say.

So, you have 1000 processor, each processor is summing up 1000 terms and after the calculations are done, all the sums coming from this 1000 processors will come together and you will write the final sum. So, when the final sum is coming, in order to get the final sum everybody is adding to the memory location. So, there will not be any conflict, they can concurrently add up to this their values to the memory location and concurrent write can be allowed. This is used in many cases, but this is for shared memory machine, that even if the processors are working in parallel and in order to access the memory in terms of read and write, some of the models are usually followed.

But for distributed memory machines, they are not writing to the same memory location; all the processors have different memory unit and they are writing at different memory locations. Therefore, these issues are not there; however, they are writing to a common file system ,when all they are connected to a common file system and when they are writing their results, they are writing to the common file system and again if concurrent writes are allowed ,certain protocols are followed such that the files are appended etcetera so that, no data is overwritten.

(Refer Slide Time: 14:34)



Flynn's Taxonomy

Prof. Michael J Flynn in 1966 proposed a classification for architecture of parallel computers based upon number of concurrent instructions and data streams. These are:

Single Instruction, Single Data stream (SISD):

Single Instruction, Multiple Data streams (SIMD)

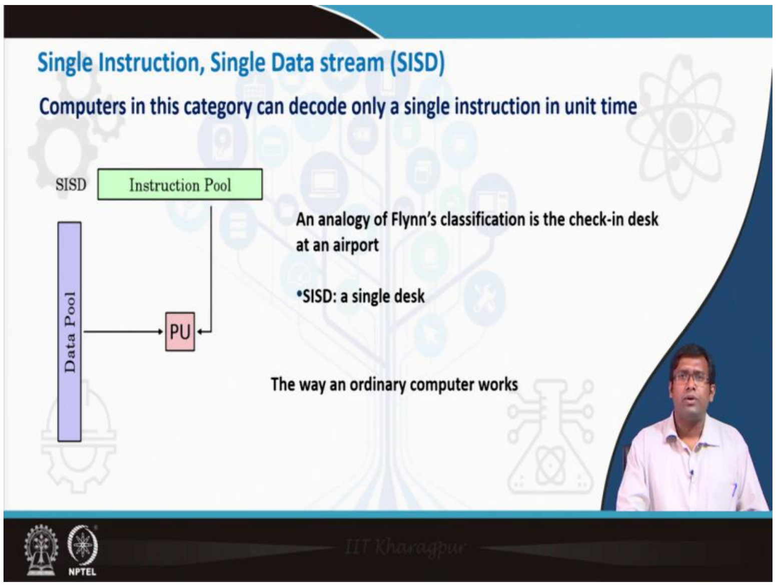
Multiple Instruction, Single Data stream (MISD)

Multiple Instruction, Multiple Data streams (MIMD)

Well, now we come to a much broader classification of architecture of parallel computers and this was given by professor Michael Flynn in 1966, which is based on the number of concurrent instructions and data streams that the parallel computer is followed and this is called Flynn’s taxonomy. This is 1966 today in 2021, the Flynn’s taxonomy is very much valid and quite well utilized when we discuss about parallel programs.

The first one is Single Instruction Single Data stream or SISD. The next one is Single Instruction Multiple Data streams SIMD, the third one is Multiple Instruction Multiple Data Single Data streams MISD and the fourth one is Multiple Instruction Multiple Data stream. So, any of the parallel programs must follow one of this architecture and should be written accordingly.

(Refer Slide Time: 15:37)



Single Instruction, Single Data stream (SISD)

Computers in this category can decode only a single instruction in unit time

An analogy of Flynn's classification is the check-in desk at an airport

•SISD: a single desk

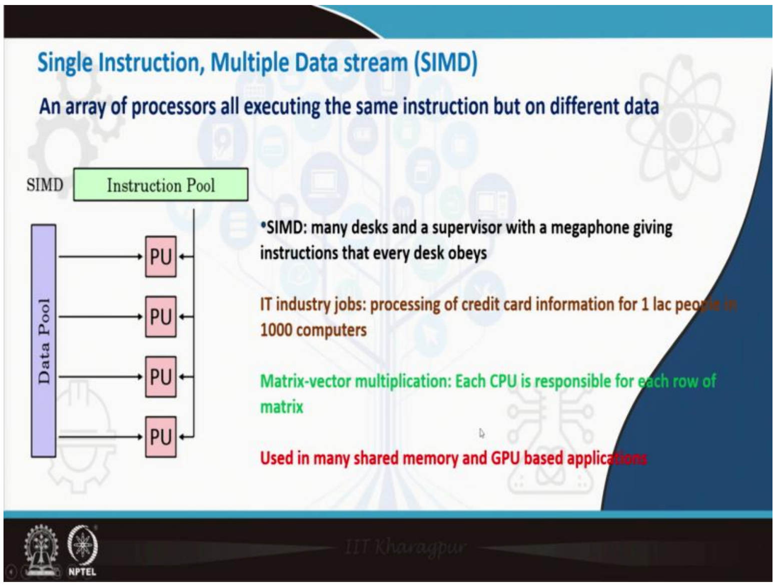
The way an ordinary computer works

So, first is Single Instruction Single Data stream or SISD, this is very much like a sequential computer. Computer in this category can decode a single instruction in unit time. So, there is a processing unit, there is a data pool, there is some instruction given to it, once an instruction comes the processing unit, it takes data from the data pool and processes the interest information.

So, it is exactly how a sequential computer should work or how a single computer should work, it is a non-parallel computing method ,that there is one data pool, CPU is taking data from the ram, one particular location data ,one CPU is accessing one data and it is getting one instruction and it is processing it.

An analogy is given; classical analogy is given as check in desk in the airport. You go to the check in desk, you present your ticket, the person sitting at the check in desk looks into the data pools, sees whether what is your ticketing number PNR, what should be the seat number prints boarding pass and gives you back. The next person comes he has to wait behind you once you are done the next person takes the ticket looks into the data, prints the boarding person and checks in him. It is a SISD means a single disk in the single check in desk in the airport counter and this is the way an ordinary computer or a sequential computer works.

(Refer Slide Time: 17:09)



Single Instruction, Multiple Data stream (SIMD)

An array of processors all executing the same instruction but on different data

SIMD
Instruction Pool
Data Pool
PU
PU
PU
PU

•SIMD: many desks and a supervisor with a megaphone giving instructions that every desk obeys

IT industry jobs: processing of credit card information for 1 lac people in 1000 computers

Matrix-vector multiplication: Each CPU is responsible for each row of matrix

Used in many shared memory and GPU based applications

The next one is single instruction, multiple data stream. There is an array of processors, it is like a vector processor or a processor array, which is executing same instruction, but on different data. So, a single instruction is given to all the processors, let say there have to add two numbers, but each processor is getting two different numbers and adding them.

In a classical example is many desks are there in a airport check in counter, and a supervisor in megaphone giving instruction that you take everybody's ticket, look into their check in data, (there are say 4 counters in parallel),so, at a go 4 person can stand in 4 counters, give their tickets he will check their data, then the supervisor is giving instruction that print their boarding pass so, 4 boarding passes will be printed and 4 people will be done.

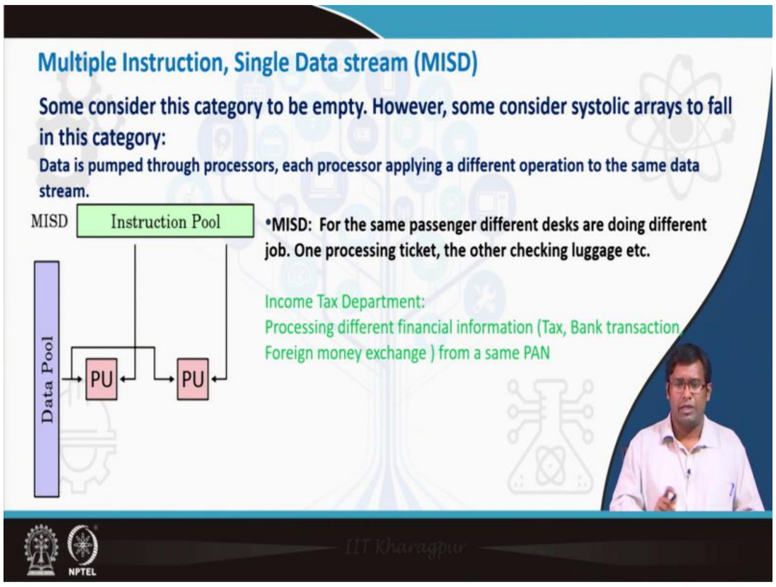
So, instead of processing one at a time, 4 at a time can be processed and this is also extremely heavily used, especially in scientific computing aspect. We can think of IT industry jobs, that, credit card information is being processed for 1 lakh people in 1000 computers. So, each computer is looking into 1000 people and doing the same work that in last month, how much they have spent and how much they have put into their credit card so, what should be their balance.

So, instead of 1 computer processing, 1 lakh which would have been the SISD architecture, this is working in SIMD architecture that same operation is being processed by all the computers; however, on different data.

For matrix vector multiplication, we will see some of the examples in next class. Say we ask a parallel computer that you have to do a matrix vector multiplication and each computer will take care of one of the rows of the matrix. So, it is one row per computer so, the entire matrix can be computed at a go, because each row instead of looking into running a loop over all the rows each computer is responsible for one row. And many computers are doing same work, that taking the first element of the row multiplying with the first element of the vector, adding with the second element, multiplied with the second element of the vector and so on and so, finishing for each row.

So, this is used in many shared memory and GPU based applications, the shared single instruction multiple data stream calculations.

(Refer Slide Time: 19:45)



Multiple Instruction, Single Data stream (MISD)

Some consider this category to be empty. However, some consider systolic arrays to fall in this category:

Data is pumped through processors, each processor applying a different operation to the same data stream.

•MISD: For the same passenger different desks are doing different job. One processing ticket, the other checking luggage etc.

Income Tax Department:
Processing different financial information (Tax, Bank transaction, Foreign money exchange) from a same PAN

The next one is Multiple Instruction Single Data stream which is called MISD. Some people also consider this category to be empty; that means, that, this is not much used in parallel computing. That you have a same data pool, but there are multiple instructions which are operated over the same data stream.

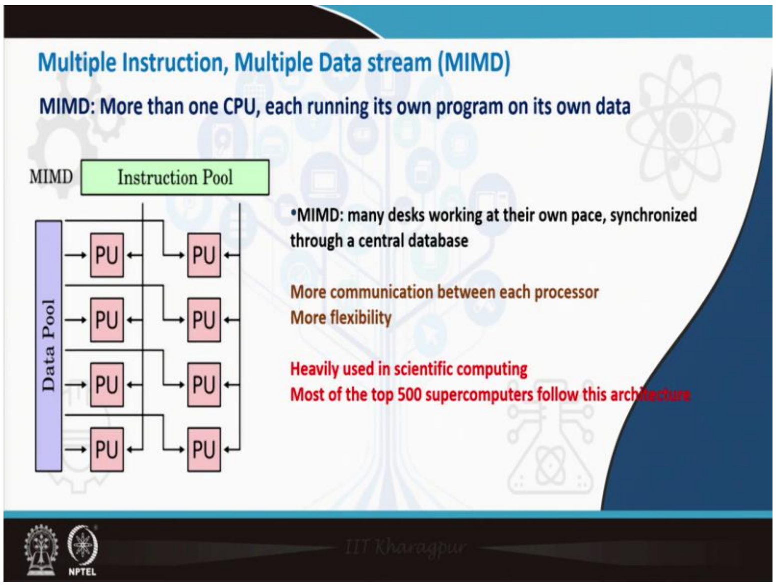
So, we can think of something like, same for same passenger different desks are doing different job. Like once the processor goes, one desk gives his ticket, so, the PNR number and all the information are obtained, one desk is processing his ticket, other desk is taking his luggage, other desk is printing his ticket etcetera. So, instead of one person working on the same data and doing different operations, different persons are doing different operations on the same data.

You can think of income tax department work and other analogy that a PAN is given and different financial information is required how much tax he has paid, what are his bank transactions, what is his foreign money exchange etcetera. So, it is the same data, it is the same pan against to which all this different information can be found out and differently processing the data.

So, if one needs to operate especially for scientific computing for post processing aspects, when one is to get several parameters out of the same data set. Say you know a vector field and you need to find out the convective acceleration, you need to find out the vorticity, you need to find out some skewness in the velocity vectors etcetera.

So, on the same data field, you ask one of the computers to read the data field and the data resides in the same shared address space. And different computers in parallel is looking into the same data in concurrent read mode and processing different information on that on the same data well.

(Refer Slide Time: 21:55)



Multiple Instruction, Multiple Data stream (MIMD)

MIMD: More than one CPU, each running its own program on its own data

MIMD
Instruction Pool
Data Pool
PU

•MIMD: many desks working at their own pace, synchronized through a central database

More communication between each processor
More flexibility

Heavily used in scientific computing
Most of the top 500 supercomputers follow this architecture

IIT Kharagpur

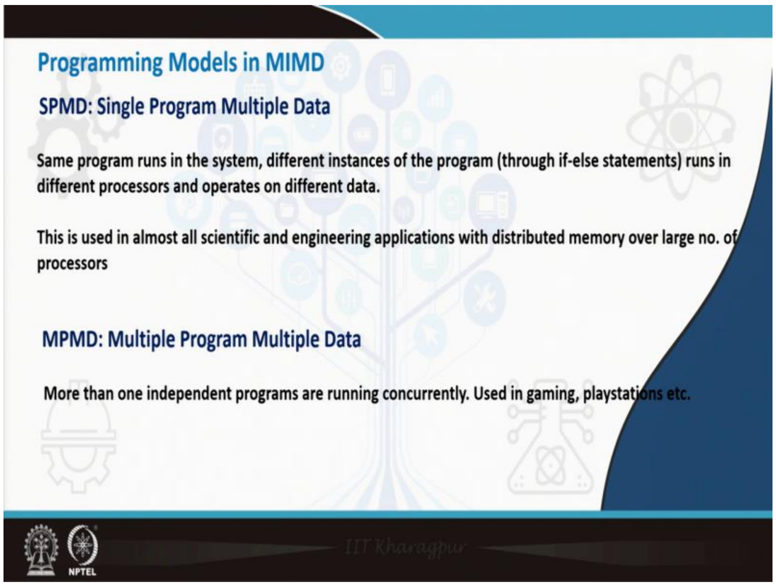
The next one is the most parallelized version Multiple Instruction Multiple Data stream or MIMD. There are many CPU’s, each CPU can access different memory, each CPU can work on different memory location and each of them are in parallel running their own data. So, there is a data pool which is accessed by different computers, it can be anyway, it can be multi computers with different data, because it is a multiple data stream and there are number of processors which are processing different operation. It is not the same instructions pool, it is different instruction pool, which is operated by different computers in parallel.

However, we can see these computers are some way connected to each other. There is some connection in between the computers and they can exchange information, they can pass some message from one to other, they can have some synchronization. So, many desks working at their own pace with different data synchronized, but they have a central database. So, there is some synchronization in between them, both in terms of data as well in terms of the operations.

It needs more communication between each processor; because they have much flexibility and in order to have synchronization they need to communicate more. When ants are moving in a line, its only communication with the previous or the following is through some chemical exchange and less amount of communication is required. But when you give more flexibility that say 15 students from 15 different backgrounds are coming to an examination hall and after the exam is over, they will go out and catch their buses. However, they will reach the same destination, then they need more communication if they at all need to do that within the stipulated time.

So, more flexibility means if you need to have synchronization, if you need to use same data you need more communication. All that is that, more flexibility means more communication and this is heavily used in scientific computing. Many of the scientific computer problems, will use MIMD because it is it is more flexible, you have a much complex algorithm, you need much more flexibility to parallelize it and get the efficient parallel performance. Most of the top 500 supercomputers follow this architecture. They provide a platform where you can run programs which are following MIMD architecture.

(Refer Slide Time: 24:49)



Programming Models in MIMD

SPMD: Single Program Multiple Data

Same program runs in the system, different instances of the program (through if-else statements) runs in different processors and operates on different data.

This is used in almost all scientific and engineering applications with distributed memory over large no. of processors

MPMD: Multiple Program Multiple Data

More than one independent programs are running concurrently. Used in gaming, playstations etc.

So, we will look into programming models in MIMD because MIMD is one of the very important architecture or paradigms for high performance scientific computing.

The first one is single program multiple data. The same program runs in the system. It is a same executable file which runs in the system. And the same executable file goes to different processors, and it works there.

However, different instances of the program through if-else statements runs in different processors and operates on different data. It is a same program which has different instances and depending on the processor number, depending on the part of the data that processor is connected to, this is operating. So, when you write a scientific program or say where we talked about matrix multiplication, matrix inter solutions etcetera and you ask many computers to work in parallel.

We do not write different programs on different computer. We write one program which is executable over different computers, but let us think of a matrix which has a varying bandwidth, the number of nonzero numbers in each row is different. So, when it goes to a part of the matrix where there are say 80 nonzero numbers, a loop over 80 runs for that particular rows 80 elements.

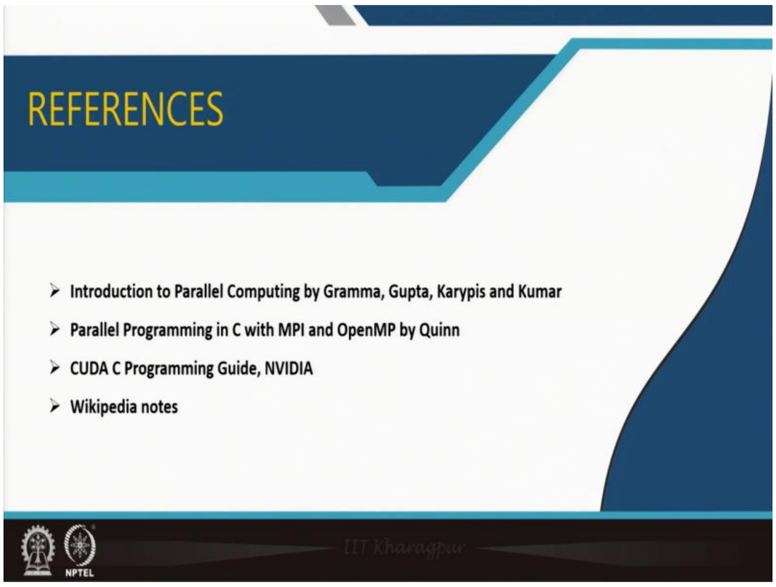
When it goes to another row where there are 13 nonzero numbers in an a loop over 13th.,so, how do you do it, we have some idea of the about the matrix so, in the program, we write if it is in these rows the computer which is getting the processor then for these rows it will operate for a loop 1 to 80. The next computer which is getting the rows where there are only 13 nonzero numbers so, we will run a loop 1 to 13. So, that way using equal statements, we can run different instances of the program and different computer and it actually runs as a multiple instruction, multiple data model.

However, it is a single program it is not different executable. Again, you use some of the commercial software say (Refer time: 27:05) or GROMACS or Ansys fluent and run in multiple computers, it is the same executable which runs in different computers. If you see what are the programs being run in different computers, you will see the same program is running in each of the computers.

This is used in almost all scientific in engineering applications with distributed memory, over large number of processors, that you do not write many programs, you write a same program you use some if else statement to provide instance different instances which will run at different computer.

The next one is multiple program multiple data; that means, different executables are now running at different computers and more than one independent programs are running concurrently, and this is used in gaming, play stations more on entertainment industry, more on visualization industry, but scientific and engineering applications mostly rely on single program multiple data model.

(Refer Slide Time: 28:08)



REFERENCES

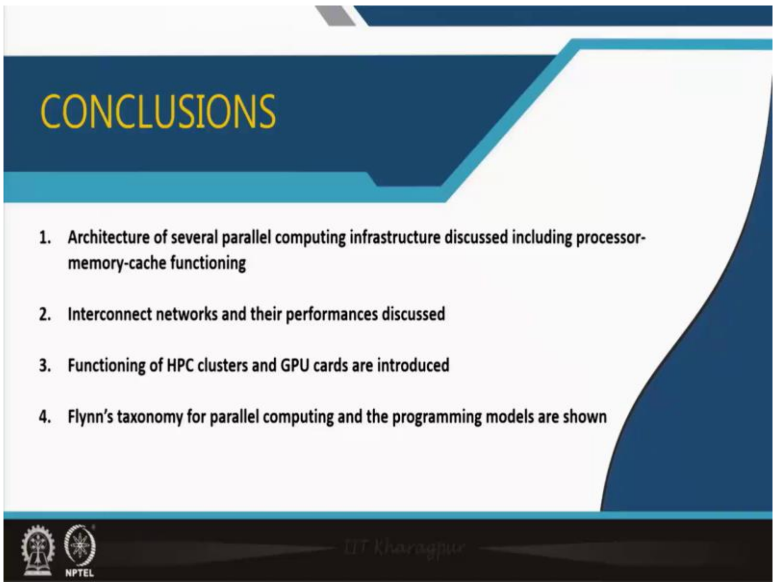
- Introduction to Parallel Computing by Gramma, Gupta, Karypis and Kumar
- Parallel Programming in C with MPI and OpenMP by Quinn
- CUDA C Programming Guide, NVIDIA
- Wikipedia notes

NPTEL
IIT Kharagpur

So, these are the few textbooks which we have followed over this class. The Gramma, Gupta, Karypis, Kumar, Parallel computing book Quinns MPI and OpenMP parallel programming book, NVIDIA’s CUDA C Programming guide. And some of the nodes and pages from Wikipedia you if you, Wikipedia has a very good repository of these discussions if you need to find out some of the details of some of the concepts with nice animations.

You can search do a Google search, go to the Wikipedia page look into their it is text and narration. And most importantly we can look into that references given by the Wikipedia and go to the reference papers there are very good papers in parallel computing which is reference by Wikipedia and.

(Refer Slide Time: 29:00)



CONCLUSIONS

1. Architecture of several parallel computing infrastructure discussed including processor-memory-cache functioning
2. Interconnect networks and their performances discussed
3. Functioning of HPC clusters and GPU cards are introduced
4. Flynn’s taxonomy for parallel computing and the programming models are shown

So, in the discussion on parallel computing architecture, our conclusions are we have discussed an architecture of several parallel computing infrastructure. And also looked into the processor memory cache functioning. We will again look into processor memory cache functioning, when we will discuss about shared memory platforms in more detail. As I said that some of the concepts we will over learn, it means we will discuss this concepts number of times.

We looked into interconnect networks and the performances including the cost of communication. We have seen how HPC clusters and GPU cards functions; that means, some introduction is given to that, and Flynn’s taxonomy for parallel computing and the programming models, a single program multiple data and multiple program multiple data and different PRAM access models are also shown.

Thank you.