

**Parallel Computing**  
**Prof. Subodh Kumar**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology – Delhi**

**Module No # 03**  
**Lecture No # 14**  
**Shared Memory & Message Passing**

(Refer Slide Time: 00:45)

**What's the strictest memory consistency model being supported?**

Initial values are U

thread 1	thread 2	thread 3
read x [a]	x = a	read x [a]
..	..	..
read x [c]	x = b	read y [0]
..	..	..
read x [b]	..	x = c
..	..	..
y = 0	..	read x [c]
		..

Let us talk about the memory consistency okay h so we have got three threads alight and each thread is only required to see the causely related operation in a consistent order in a essentially sequentially consistent order okay. So thread one must see all the orderly related events in the same order as the thread two does which must be as same as thread three does okay.

And so what are the causly related events here  $X = A$  is an event that happens before  $X = B$  to write a write is ordered after everything that happens before h then there is a right to X of the value C which is thread 3 that also is casly related to the two reads before it okay the 2 reads that happens before it and there are other the reads that are cuasly related to thread once first read is casely related to the  $X = A$  right it is second read is causely related to  $X = C$  right and its third read is cause related with  $X = B$  right okay.

What else and similarly on thread 3 the read of X related to A and read of Y is related to  $Y = 0$  and the read of X is related to the just the previous right okay. These are all the causes orders

which must be seen the all the threads in the same order now each thread really does not see other threads read in directly it might but directly does not see other threads reads which means that if you can ensure that other thread writes or seen by the same order by every thread then you can ensured that there reads will also be causely consisted alright.

Which means that we need to make sure that for thread 1 the writes there are 3 writes  $X = A$ ,  $X = B$ ,  $X = C$  and they are crossly related with respect to each other through other reads okay the first  $2X = A$  then  $X = B$  are definitely directly costly related but  $X = C$  is causly related through the read that is happening h and so  $X = A$   $X = B$  and  $X = C$  must be seen in the same order where all the three thread so what should the order B come up with that order okay.

So if it is so it is not just ACB order is also for a given thread we have to place  $X = A$  somewhere  $X = B$  somewhere  $X = C$  somewhere so you are saying ACB right. So A as to come before C but we have to place it somewhere and where does A  $X = A$  come in the beginning read X.  $X = C$  comes before that and  $X = B$  comes before that okay what about the Y? Y is cause related to all the X reads and its happening after right.

That means because the first thread says that all the three X writes have to happen before the last read you need the all the three X rights have to happen before the Y write no  $Y = 0$  is the right way we are talking about and we are saying that all the other reads ensure that writes to X must happen before the Y is right okay. So A thread 1 says that the order should be X return to X value of A written to X of value of C return to X of value of B and written to Y value of 0 that is the order is that consistent everywhere.

Thread three cannot be plane with that order right so there is some caution relationship that is been related for whatever reason alright okay any other question. What does processor consistency says that writes of the given processor have to be in the same order everywhere right meaning that  $X = A$ ,  $X = B$  must be related consistently okay. By itself that means processor consistency of top it says all the writes to X even if there happening on the different thread must be sync consistency.

So that means the  $X = A$ ,  $B$  and  $C$  must be seen consistently everywhere and that order first  $A$  then  $C$  and  $B$ . We will explain all right writes to  $Y$  is only in thread one.  $ACB$  is not necessarily required here I might be  $ABC$  because then only from one processor should be consistent. If it is  $ABC$  then how will you explain what thread wants are if it wrote  $X = B$  then  $Y$  is it reading  $X = 3$ . According to the processor consistency wrote that the right bios in one individual processor would be seen in the sequence.

That is the 5, 4 part of it there is two condition right one says rights by processor or let me say rights of  $A$  processor have to be seen consistently everywhere okay in the same order. And then it additionally says that there is some write to a given variable and that must also be seen variable consistently and clear okay. And so the variable thing ensures that  $X = A$ ,  $B$  and  $C$  must be seen consistently you have given.

But  $Y = 0$  appears only in thread one which means it does not have to be ordered effect to any other right of thread one. It does not have to be ordered one any other write at all because there is no other write to  $Y$  anywhere else so  $Y = 0$  can be seen wherever by other thread 1, 2 and 3 or thread 2 and 3 okay anything else. So we will talk about message passing interface next because primarily one reason is that is what the next assignment is going to be .

And I was thinking if you want to do this assignment in group then that may not be such a bad idea. But group as to be no more than size of two okay. In the other in the feedback they were not that much of the feedback then one person said that let us not discuss MPI right now in class learn that separately which is true you should learn that separately that I am not teaching you in class because you should not rely on all I said but at the same time the reason.

I have also spending a lecture or so on kind of the programming interfaces it is not just learned specific program interface and we got to understand what programming interfaces in fact if you are designing an interface what are the types of decisions you would be making or you would have to make okay. And so although you would learn how it works in manual looking at it at a slight level I hope you are getting some sense of in the class is useful okay.

That precisely why we are talking about MPI also so open MP was shared memory model and as far as can give or some software layer can give that impression that i have got shared memory that all processor actually physically share a memory or not that is the excellent programming model there are other. There is INTEL as a programming model that it has recently started advertising so this week there is other things call self which is bit more language like than open MP.

Open MP is basically macro level thing but once you have seen open MP you will understand all of them because they have very similar ah design decisions very similar way of thinking about auto program shared memory and so just the details may be as long as seen one you can pick up the others the relatively easy none of those are well designed for non-share. There is software that will sit on top of computer that do not have shared memory and give you an interface to a shared memory.

So you can still declare variables which may not reside on this machines computer but on other machines to this computer and accept. Typically these have not been successful primarily because of performance okay. When the program it does not know so far when you in the context let us say open MP when you accessed a variable if it was I cache you get cache behavior it was in main memory you will get main memory behavior.

The difference between to access to the main memory would be insignificant let us also because it was the uniform memory architecture right. We also talked about non-uniform memory architecture and although INTEL of the date are moving towards non-uniform memory architecture. But as long as that non-uniform structure is still tightly bound say cross about or within single computer the different I am that but as soon as say this is across the network on another machine that latency to your main memory and to a different computers main memory becomes significantly different.

As soon as that happens program is typically wants to involve about program would want to ensure that if your variable can sit on place where it is access then that is where it should. If a variable is on a remote machine from where it will take a long time to fetch it to access it to write

to it then you do that only for other reasons because your program structure (()) (14:01) or at least you want to know that concuss you want to do that concuss the program you want to control and that been h one of the reason that MPI has become much more successful than other competing methods on multiple computer actions okay.

There is another interface called PVN which is N in fact MPI adop style programming which is map reduce and there all in some ways a competitor to style interface except that is somewhat higher level. So you do not get full control over everything but sometime that is a good thing because you can program on the more abstract a way so we will to some extent what PVN and may reduce look like PVN seems to be like not popular any more came out it was quite popular.

A primarily because h its performance with respect to MPI there is not kept that h but map reduced seems to be upcoming quite mature yet still developing. So I do not want base assignment on that h but it is something that it was knowing about so we will probably at the end of the cause try to spend sometime okay. So that all is leading to the fact that now we are not talking any more about a distributed programming model person okay.

Because MPI is not same here is a function and am going to write for  $I = 0$  to 10 or 0 to some other number and like open MP somebody run it on ten computer  $I = 0$  on this  $I = 0$  and that  $I = 0$  you would like something of that's what the programming interface from a parallel programming model. In MPI you are responsible for that so MPI as the name says message passing interface it says am going to abstract the details of the underlying network and MPI works on a Ethernet as well as on other thing and sitting across the internet can be sitting on other kind of fabric h and so all of that is hidden from you.

And you see a generic way of communicating to other computers that you co-operating with in solving a problem.

**(Refer Slide Time: 17:09)**

## Message Passing Model

- **Process (program counter, address space)**
  - Multiple threads (pc, stacks)
- **MPI is for inter-process communication**
  - Process creation
  - Data communication
  - Synchronization
- **Allows**
  - Synchronous communication
  - Asynchronous communication
    - compare to shared memory

What would you have to do so that you only want to know that I have got these ten people somewhere potentially remote trying to do this tempings and I know what everybody is trying to do and then I either give intermediate piece of data and somebody or accept piece of data from somebody okay. The intermediate can be input or output my output can be given to somebody.

Somebody output may be coming over here you can do it you have to do it in a processor because they are different machines so we will be running at least one processor machine one process each machine it is possible to have in this context to have multiple processes run one machine also just like in open MP context we said we can run multiple thread for core. You can run multiple processors for computer but typically that would not give you the best performance.

Why is it different from threads I am asking I had earlier said that you would want to run many threads for core right. So that you can hide latencies fill the ideal periods however for a single machine however many cores they may have I am suggesting you to run one process one MPI process per machine. Processors do not share so they are independent address spaces right so you can run multiple process but they will be like doing two independent things and you would be communication them through the MPI or some similar interface.

We would not be communicating with shared memory so you would rather use an open MP style inside each of the process is now the process is made of many threads such that they among

themselves keep that process or that setup process set of course busy individual process may have set of threads right so you can imagine for example having running one process one program let us say on each of the machines that program is in open MP style program that you have written and they communicate with other processes on other machines.

That is visible to the programmer know that these threads are not the same group of threads that thread set A thread set B okay h okay so that about the general structure of how you may have your computation units tasks divided what do you mean by run like array? Open MPI is library we will come to that part also we will so that you will get started I will give you some of the prime side of things also.

Talking about design side of things we have been discussing essentially over the last lecture and half processor design right I said many times when you communicate it synchronize as when I am sending you data and you are receiving the data and I have to know that you receive that data then one of us is probably waiting right. This is very unlikely that we are going to get to that receive and send point at precisely the same time do quick exchange and move on right.

And so you are designing communication you got to be very careful about do you actually want to say connect to that fellow do some data exchange and then move on or have some data exchange happen in a background hide the latency again okay. So you have to got to think about whether it should be synchronize or it probably means to support both methods sometime you do need to know that fellow as received that data only then I can move forward.

At other time you say sometime in the future that receiver will receive may be I want to know when that receive is completed I do not need to wait for it to complete right now and stop everything else okay. And so that is the synchronize style I say go ahead send this and later I will come back and say is that done yet so that if something depends on it which probably is true h I will at that time determine what to do okay.

So I think this is probably a point where we need to stop just a few seconds and the continue on that shortly. So I said this is really about message passing and MPI has many implementations in

reality MPI only a standard bunch of folks got together and say this is what type of communication is required should be done by cooperating processes which are trying to cooperate and solve the problem to compute something and then there are various implementations of which the most active right now is this open MPI implementation okay which is what is going to be available on the machine that you are going to run.

It is going to be able to not only say here is one process and it is sending data to another process okay it is also going to be able to do some more group communication and group synchronization will allow you to barrier across multiple processes it will allow you to broadcast data it will allow you to gather a piece from here a piece of data and collect all of it since the opposite of which is scatter.

It will allow you to reduce all the different processes to produce some result and you reduce that into single value the kind of reduction we have seen in open MP also so all those things also attached to this interface because it is not only message passing interfacing as the name suggests that is what MPI is also but it is more far the context of parallel programming okay. Which means that there is notion of task you there is notion of starting means stopping task there is no notion of killing task there is notion of saying how many tasks to run on different computers.

There is notion of saying that this program is going to start on these computers you run it at one place and it automatically itself is copied at multiple different computers okay those kinds of facilities come with it.

**(Refer Slide Time: 26:25)**



# MPI Overview

- MPI, by itself is a library specification
  - You need a library that implements it, like openMPI
- Performs message passing and more
  - High-level constructs
    - broadcast, reduce, scatter/gather message
  - Packaging, buffering etc. automatically handled
  - Also
    - Starting and ending Tasks
      - Remotely
    - Task identification
- Portable
  - Hides architecture details

And above all is portable right so it is windows or Linux or Mac OS or some other so open MPI is ported on lots of different platforms it is going to work similarly so you will see the same interface you program the same way but one machine that you may be talking to windows another machine you may be talking to is Linux and it is all going to work point. Because it is little more than network protocol right that is what I am trying to elaborate on that it is primarily a message passing things.

But other also synchronization primitive it also has a higher level structure on the data that is you are passing across so it is you can think of it message passing when it is really message passing plus okay.

**(Refer Slide Time: 27:36)**

## Running MPI Programs

- **Compile:**
  - `mpic++ -O -o exec code.cpp`
  - Or, `mpicc ...`
  - script to compile and link
  - Automatically add flags
- **Run:**
  - `mpirun -host host1,host2 exec args`
  - Or, may use `hostfile`

So let us start with how you would run it. You can compile your MPI code with GCC or C++ or whatever but there is a wrapper called MPI C++ or MPI CC which then does not need to be provided with path just like in open MP there was `dash open MP` to that these program is going to use open MP and use all the flags paths etc., MPI CC is basically that knows where you have been stopped MPI CC is setup with all the location information when it is compiled in that know what is where on your platform.

You basically have to see MPI is C++ my code dot CPP okay and it is the code `(())` (28:36) things and is going to link in with right set of library to produce executable. To run you cannot run simply that code as you would on a single machine. That code have been possible because in you are a dot out it is possible library would have been set something to make sure that it automatically starts in multiple places.

It is less clean of an method so there is an execution program that MPI comes with and you say instead of you are just typing the name of the program pressing return you say MPI run program return. MPI run is actually an executable which is going to create some Demon which is going to run this program at multiple places and a set up the connection so that they can talk to each other you.

If I am running MPI program on multiple machine of this particular or process is getting executive first on all of these some MPI run or similar execute will. No you execute run on one machine but MPI run on execute something yes it will then it will do an exact for your program exactly after a while I just see my slash or running on all them machines.

It takes arguments right for example in this example you are saying its dash post some name 1 name 2 it says run this program on 2 host. Host 1 and Host 2 okay in this case you are not even running on the local machine necessary it was to be run one machine host 2 may be third machine and you just say run this program on those two machines okay. It different process will be no conflict yeah this independent official that is inside your code.

The code that is in this exit right so it is exit you can open MP style program yes MPI is not directly only if you run multiple process is on one machine right. You can say run this program twice on host 1 you can do that. I will just say if I host N then it will run twice on different code or that is OSS business OS will assuming that there is nothing else happening will actually schedule it on two different codes but this is not directly part of it.

The binary needs to get accessible on all the machine so basically MPI log you in remotely on all the machines right which is a bit of problem over there h which will talk about shortly. Because log in you need to provide a password you need to asses it so MPI knows that can SSH to remote self but you need to provide the password which means you are running on five machines you need to provide five sets of password.

Because it is going to run once on each machine okay and it gets very confused it is not really h designed to keep that consistent and so in order to run it on our department and our requirement you are going to use a public key and private key there okay.

# Remote Execution

- Allow remote shell command execution
  - Using ssh
    - Without password
- Set up public-private key pair
  - Store in subdirectory `.ssh` in you home directory
- Use `ssh-keygen` to create the pair
  - Leaves public key in `id_rsa.pub`
- Put in file `.ssh/authorized_keys`
- Test: `ssh <remotehostname> ls`
  - Should list home directory
- But must have kerberos token
  - ssh once to all remote hosts, then run the program

Which means you SSH (refer time: 33:21) will not ask you for the password it will do authentication by looking at your SSH private files on the other machines and just execute that program on the other machine. That is not full story because we run Kerberos which does not let SSH run if you do not have Kerberos kit okay which means if you are going to run it on our department machines.

You are actually going to run it to log in to that machine in different shell to obtain the token and then run because now you have the token it is going to be able to finish their public key private key authentication and we will be able to run your program. Yes so in fact you cannot do it on other machines that was sitting on the desktops there is an overhead if you do not want to do through this.

There is a matter of fact if other the remote locations read something from input they are not going to get any more because all the remote I/O inputs are channel through develop. So when you run your program with than some file and the entire file is going to be the main program if you are running it on that host okay. The outputs are produced but that cause synchronization issues mean that output will have to be sent it here merge with other output that are happening here and all the other outputs are being sent.

And so all the output will appear they will appear but they will appear in some arbitrary order unless you have caused it. All these messages will be through that SSH program which is made all that right because that you can be if you are not using SSH directly or not getting SSH we are not getting right that is true. Open MPI is that is not open MPI business right open MPI that is your local how you run the remote self-business.

If you are allowing people to simply our research something then it is going to do RSH okay h on the machine that you are going to be running which is called (()) (36:18) I think earlier pointed out h it does not allow RSH as of now and look at that h but department security rules say that probably RSH will not be used but it is not using Kerberos authentication for SSH meaning that if you set up your key properly then you should not actually have to log in to ten different machines before you run your program.

But ten let is not allowed and it is off except 25 that is true but I do not believe MPI has that control to change the port either going to research RSH what are the remote execution methods possibilities are only that it can do. So it does not have its own remote shell routines okay so let us get to MPI itself okay h first of all because that we know that we are N things co-operation to do such things you are going have the logical addressing right.

It is going to simply say that you are 1 you are 2 just like an open MPI you use to say thread ID you just need a similar thing so we do not called it thread ID we do not called it process ID you just call it ID so if I am generating N processes may be on N different machines may on one machine may be on two three whatever sum number of machine.

**(Refer Slide Time: 38:48)**

# Process Organization

- Context
  - “communication universe”
  - Messages across context have no ‘interference’
- Groups
  - collection of processes
  - Creates hierarchy
- Communicator
  - Groups of processes that share a context
  - Notion of inter-communicator
  - Default: **MPI\_COMM\_WORLD**
- Rank
  - In the group associated with a communicator

Then all those processes together form a group you can create the sub groups and all that. So that additional set of tools are provided right and then you get a rank in this group whatever the rank may be you can query my thread number 3 so here you can say what is my rank? And you can query how many members are there in group so you can say I am 5 or 7 okay.

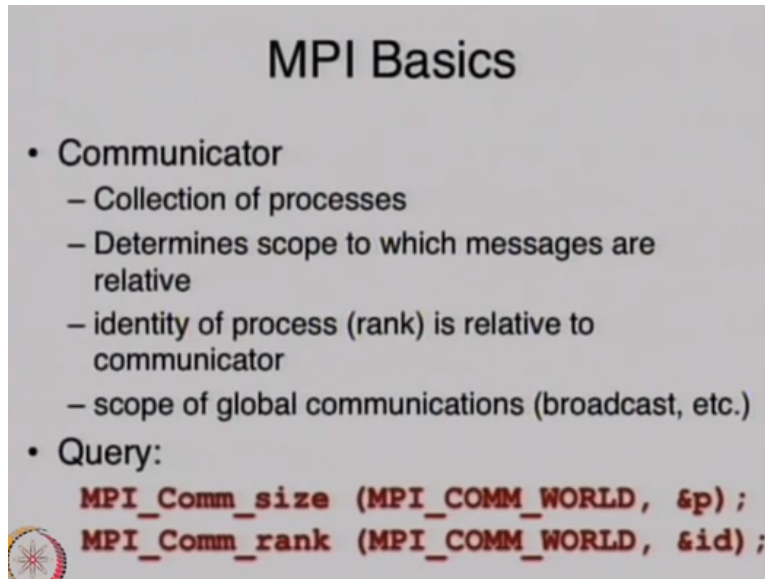
So there is notion of groups there is also notion of context which we will not get too much into context can have multiple groups in all likelihood you are going to have single context groups can have sub groups and so on there is also notion of the communicator right it is essentially an abstraction that is really the abstraction of the network okay in fact it is still on top of network because you do not really necessarily always say communicate to somebody there are higher level primitives also.

But communicators is this is the channel so to speak the channel on which we talk that channel may allow you to connect or top to 1 % the channel may allow you to talk to all the people at the same time okay so there is the notion of the channel that you build up call it has communicator. And there is a default communicator you can create new communicators is that default communicator which is which goes by this constant MPI underscore comm underscore world which starts with the channel on which all the member of the group say.

So that is the default channels of this you can your sub channels just like you create channels and

for each channel there is a group and that group becomes the context now of this communicator. So for every communicator that I am this channel and number 3 and number 4 and number whatever.

**(Refer Slide Time: 41:30)**



**MPI Basics**

- Communicator
  - Collection of processes
  - Determines scope to which messages are relative
  - identity of process (rank) is relative to communicator
  - scope of global communications (broadcast, etc.)
- Query:

```
MPI_Comm_size (MPI_COMM_WORLD, &p);  
MPI_Comm_rank (MPI_COMM_WORLD, &id);
```

So communicator can be queried the examples are listed over here or you can say ask for what the size of the communicator is you can ask for what the rank is and in this case the first parameter is the communicator that we are talking and this case this is the channel is the global communicator looks everybody of the group started with and then you can center in the pointer and then it will return to you the number of people in this communicator in this group as well as your rank okay.

**(Refer Slide Time: 42:09)**

## Starting and Ending

```
MPI_Init(&argc, &argv);
```

– Needed before any other MPI call

```
MPI_Finalize();
```

– Required

Some book keeping you have to call this function MPI\_Init if you do not MPI\_Init gets (0) (42:20) okay. If you do not call MPI\_Finalize after you before you quit again MPI\_Finalize gets signed okay it is going to give you unpredictable and usually (0) (42:35). And in fact there is because you can be creating processes online and you do not know whether you have earlier called MPI\_Init in it.

There is query function that says MPI\_Init has been called so do not call it again okay so where the query it has been united.

(Refer Slide Time: 42:57)

## Send/Receive

```
int MPI_Send(void* buf, int count,  
             MPI_Datatype datatype, int dest,  
             int tag, MPI_Comm comm)
```

```
void MPI::Comm::Send(const void* buf,  
                    int count, const MPI::Datatype&  
                    datatype, int dest, int tag) const
```

```
int MPI_Recv(void* buf, int count,  
             MPI_Datatype datatype, int source,  
             int tag, MPI_Comm comm, MPI_Status  
             *status)
```

Okay so this is in some sense MPI okay there is function called MPI send similarly there is another function called MPI receive and those two together are the message passing side of MPI



there are variance there are several variance against to it but those two are the main now am not going to repeat this is open MPI am talking about okay. The function names are open MPI but all variance of the MPI under using the same function name so that it is consistent or whichever you are using.

There is also C++ binding so there is a class MPI which has a sub class called Comm which has a method called send okay. So if you are interested in C++ then you can use the class so we will take essentially the same parameters I am this is only time am going to show you the C++ variant of the function you just going to look at the C variant through the rest of the MPI later.

And this is MPI receipt so obviously we can send to send something right you there is a buffer going to fill data in that you will be sending you will send some number of elements in this buffer. So count says how many I am sending and this type MPI data type says what is the time of am sending you can send (()) (44:44) you can send 1000 bytes you can send 1000 floats you can send 1000 some other strut you may have h so all of that is in terms of elements units elements okay.

And there is the rank on the destination who you wanted to get to and with every message you get to attach with the tag right so that you can create several streams on this channel so if you say this is essentially a type of messages say I am sending you the message and this belongs to message type 379 and so the receiver can say I am only reading messages for type 379 alternatively the receiver may say I want to receive all messages I want to know what time I received somewhere it is 379 somewhere it is 378.

So instead of embedding message type in the message itself you can send an addition piece of data that is this is command am sending you this is a request I am sending you this is control data that I am sending telling you something about stealing your load or something of that right. So generic elements you send the data you want to send and one tag is you can use to say this is the type of data I am trying to send so it can be interpret properly.

If you are sending many types of data receive is just the opposite of it receive will also provide a buffer pointer it will say that I am providing so much space count it will say that I wanting to receive so many elements of this type and I am trying to receive it from the source and rank of the source okay. It can say that only want to receive control messages of the time at this moment so the tag is provided by the receiver the communicator which channel you are talking on and MPI status is the written value.

It says something about how many so it does not necessarily have to be that you are reading twenty you wanted to read up to twenty things is what we sent. Sometimes you know precisely I am sending twenty you are receiving twenty sometime that varies and so status says something about it. Status as detail about where the message came from what was it is tag? How many elements came and so on and hence so forth okay. Why does it have to have tag? Or the source? Not to authenticate not identical did not status at all send does not have status yes because when says it was I have to read from this source it can get data from that source.

But it can says source is any source whoever wants to send to me I can read your tag can be any tag whatever type of message whoever wants to send I can read and I am going to deal with I am going to see what type it is do it from and then I take a necessary action okay. So when you see wild cards in your source and tag then you need this information what actually came okay. So far okay these things are essentially compared to well in a in some ways you can think of it even the once we are going to talk about the synchronization mechanisms you can think about it our C++ and some ways is RPC does not have them but it might have it okay.

And all of these variants are blocking meaning that when you send you are block until some guarantee has been made. It is not necessarily that you are blocked until receive has happened we are blocking until may be your data has been copied by into some other (()) (49:56) it is essentially is an indication if you making a blocking call and you return that your command has been taken it is going to get a executed.

You can go change your buffer if you like you can go change your variable if you like but your command is guaranteed to be executed as you wanted okay. We will look at those various types and then it will become a little bit more clear.

**(Refer Slide Time: 50:32)**

## Receive

- message contents      memory buffer to store received message
- count                      space in buffer, overflow error if too small
- message type            type of each item
- source                     sender's rank (can be wild card)
- tag                         type (can be wild card)
- communicator
- status                     information about message received

```
int MPI_Recv(void* buf, int count, MPI_Datatype
datatype, int source, int tag, MPI_Comm comm,
MPI_Status *status)
```

This is just a repetition of all the parameter we talk about in which is replicated for h received let us go through very quick example which we run out of time let me stop here.