MPI – Message Passing Interface

Program starts as a set of parallel processes

- **Initializing MPI environment**

```
if(MPI_Init(argc,argv) != MPI_SUCCESS) crash("MPI_Init fail!\n");

if(MPI_Comm_rank(MPI_COMM_WORLD,&id) != MPI_SUCCESS)
crash("MPI_Comm_rank fail!\n");

if(MPI_Comm_size(MPI_COMM_WORLD,&np)!= MPI_SUCCESS)
crash("MPI_Comm_size fail!\n");
```

communicator – ID of the group of processes

- **Finishing MPI environment**

```
MPI_Finalize();  // normal finishing – will wait on barrier

MPI_Abort(MPI_COMM_WORLD,-1);  // crash – stops all processes
```

Point-to-point communications

- **Blocking communication**

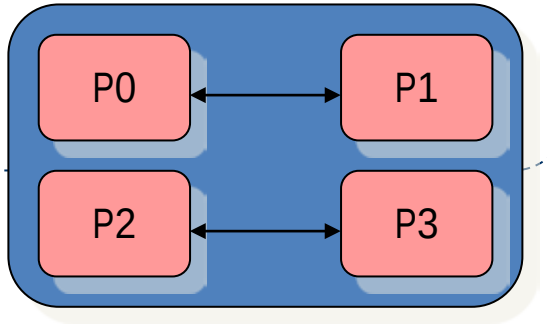
int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

buf - initial address of send buffer
count - number of elements in send buffer
datatype - datatype of each send buffer element
dest - rank of destination
tag - message tag
comm - communicator

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

Input Parameters
count - maximum number of elements in receive buffer (integer)
datatype - datatype of each receive buffer element (handle)
source - rank of source (integer)
tag - message tag (integer)
comm - communicator (handle)

Output Parameters
buf - initial address of receive buffer
status - status object



Point-to-point communications

- **Non-blocking communication**

```
int MPI_Isend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
              MPI_Comm comm, MPI_Request *request)

int MPI_Irecv(void *buf, int count, MPI_Datatype datatype, int source, int tag,
              MPI_Comm comm, MPI_Request *request)

int MPI_Wait(MPI_Request *request, MPI_Status *status)
int MPI_Waitall(int count, MPI_Request requests[], MPI_Status statuses[])
```

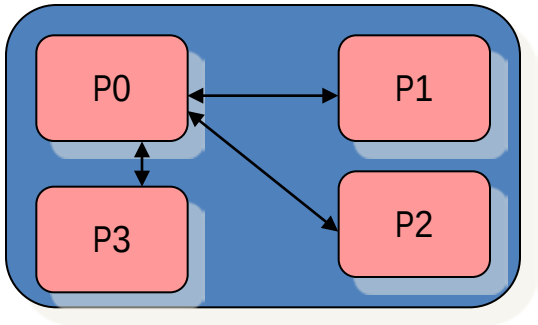
Reduction group communications

- **Operations with O(log(P)) exchanges – P data sets of size N joined into one data set of size N**

```
int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype,
               MPI_Op op, int root, MPI_Comm comm)

int MPI_Allreduce ( void *sendbuf, void *recvbuf, int count,
                    MPI_Datatype datatype, MPI_Op op, MPI_Comm comm )
```

- **Example of summation over all processes**

```
void GatherSum(MPI_Comm C, double* A, int N){
  double *Res=new double[N];

  if(MPI_Allreduce((void*)A,(void*)Res,N,MPI_DOUBLE_PRECISION,MPI_SUM,C)!= MPI_SUCCESS)
    crash("GatherMax: MPI_ALLREDUCE(sum) failed! \n");

  for(int i=0;i<N;i++)A[i]=Res[i];
  delete[] Res;
}
```



Gather and scatter communications

- P data sets of size N joined into a data set of size N*P

```
int MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype,  void *recvbuf, int recvcnt,
        MPI_Datatype recvtype, int root, MPI_Comm comm)

int MPI_Gatherv(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf,
        int *recvcnts, int *displs, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

- Inverse operation

```
int MPI_Scatter(void *sendbuf, int sendcnt, MPI_Datatype sendtype,
        void *recvbuf, int recvcnt, MPI_Datatype recvtype, int root,  MPI_Comm comm)

int MPI_Scatterv( void *sendbuf, int *sendcnts, int *displs, MPI_Datatype sendtype,
        void *recvbuf, int recvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

- Combined operation

```
int MPI_Allgather(void *sendbuf, int sendcount,  MPI_Datatype sendtype,  void *recvbuf,
        int recvcount, MPI_Datatype recvtype,  MPI_Comm comm)

int MPI_Allgatherv(void *sendbuf, int sendcount, MPI_Datatype sendtype,
        void *recvbuf, int *recvcounts, int *displs, MPI_Datatype recvtype, MPI_Comm comm)
```