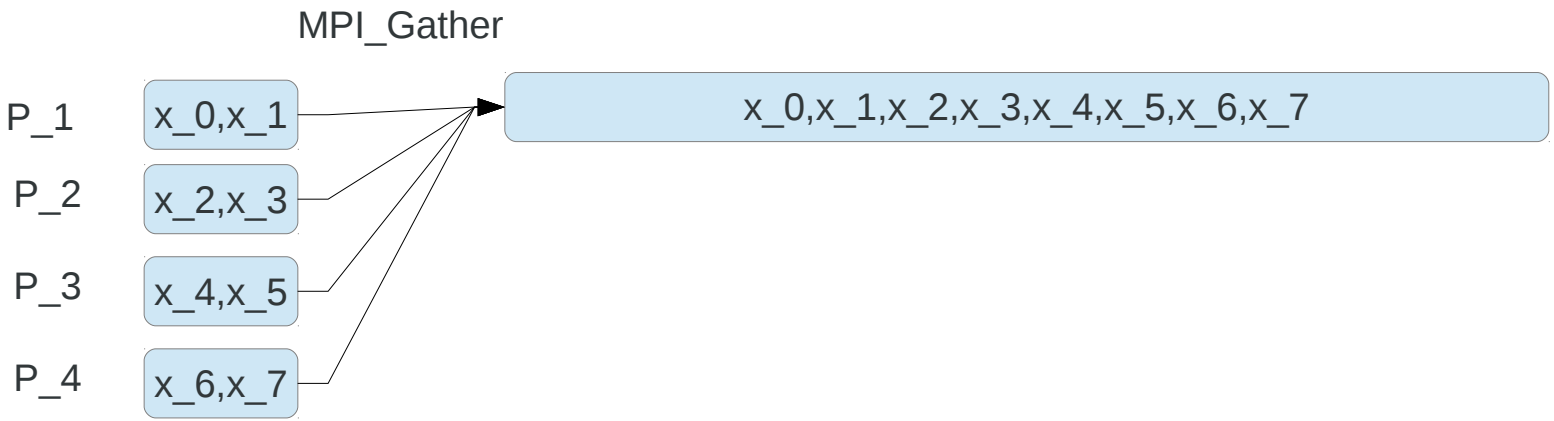
Communication Functions

A **Gather** operation consists of every node sending a consistent amount of data to a single node.

A few more MPI functions

`MPI_Gather` is a method for gathering information from multiple processes to one process.

A few more MPI functions



```
MPI_Gather(sendbuf,sendcount,sendtype,recvbuf,
 recvcount,root,comm)
```

- `void* sendbuf` – The start of the buffer to send.
- `int sendcount` – The number of items to send.
- `MPI_Datatype`– The type of data getting sent.
- `void* recvbuf`– The buffer to receive data. Only matters on the receiver.
- `int recvcount`–The number of items to receive **from each process**
- `int root`–The rank of the process that is collecting the data
- `MPI_Comm comm`–The communicator

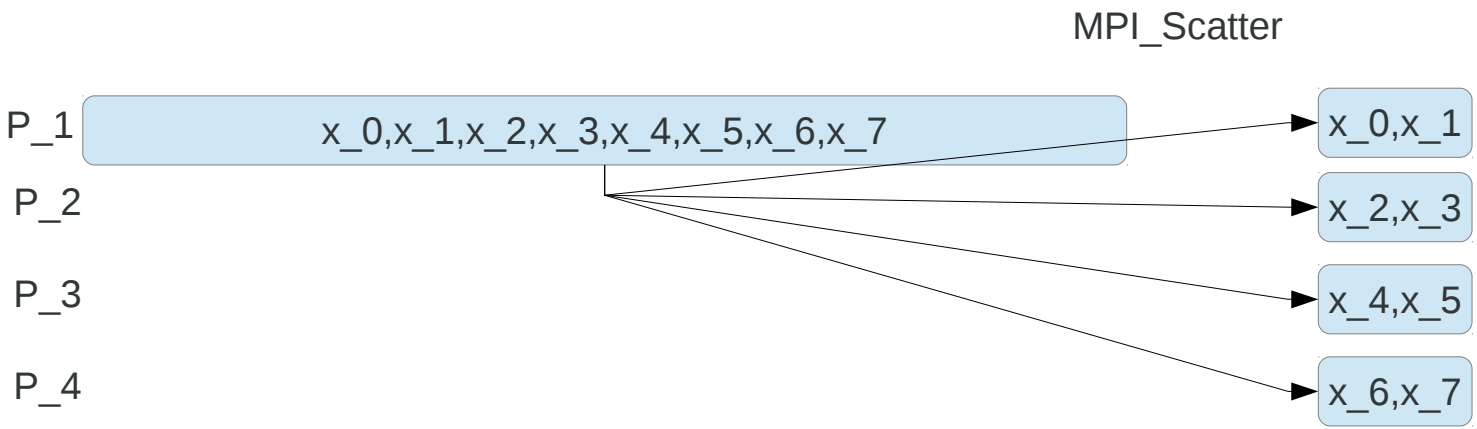
```
if (rank==0){
   int mydata=-1;//root still sends data.
   //make room to collect
   int* recvbuf=malloc(total_procs*sizeof(int));
   MPI_Gather(&mydata,1,MPI_INT,recvbuf,1,MPI_INT,0,
    MPI_COMM_WORLD);
   for (i=0;i<total_procs;i++){
     printf("%d\n",recvbuf[i]);
   }
 }
 else{
   int mydata=rank*2;
   MPI_Gather(&mydata,1,MPI_INT,NULL,0,MPI_INT,0,
    MPI_COMM_WORLD);
 }
```

```
[joe@eichholz-2 MA335]$ mpirun -np 4 a.out
-1
2
4
6
```

Scatter

A **scatter operation** takes a large amount of data on a single node and distributes the data equally among the other nodes.

A few more MPI functions



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

- `sendbuf` -The buffer to be send. Significant only to the sender.
- `sendcount` - The number of elements of sendbuf to send each processor
- `sendtype` - The datatype being sent.
- `recvbuf` - The receive buffer. Significant only on receivers.
- `recvcount` - The number of elements to receive.
- `recvtype` - The datatype to receive.
- `root` - The rank of the processor doing the sending.
- `comm` - The communicator over which this is happening.

MPI_Allgather is just like MPI_Gather, except that with Allgather the data shows up on every process instead of just one copy on the root.

```
if (rank==0){
  int mydata=-1;//root still sends data.
  int* recvbuf=new int[size]; //make room to collect
  MPI_Allgather(&mydata,1,MPI_INT,recvbuf,1,MPI_INT,
   MPI_COMM_WORLD);
}
else{
  int mydata=rank*2;
  int* recvbuf=new int[size];
  MPI_Allgather(&mydata,1,MPI_INT,recvbuf,1,MPI_INT,
   MPI_COMM_WORLD);
  for (int i=0;i<size;i++){
    cout<<"Rank: "<<rank<<" index: "<<i<<" "<<recvbuf[i]<
  }
}
```

```
[joe@eichholz-2 MA335]$ mpirun -np 3 a.out
Rank: 1index: 0 -1
Rank: 1index: 1 2
Rank: 1index: 2 4
Rank: 2index: 0 -1
Rank: 2index: 1 2
Rank: 2index: 2 4
```

```
MPI_Allgather(sendbuf,sendcount,sendtype,recvbuf,recvcount,
recvtype, comm)
```

- Exactly the same as for `MPI_Gather`, note that there is no `root` since everyone is receiving the data.