4.6. Scatter

MPI_SCATTER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, root, comm)

IN sendbuf	address of send buffer (choice, significant only at root)
IN sendcount	number of elements sent to each process (integer, significant only at root)
IN sendtype	data type of send buffer elements (significant only at root) (handle)
OUT recvbuf	address of receive buffer (choice)
IN recvcount	number of elements in receive buffer (integer)
IN recvtype	data type of receive buffer elements (handle)
IN root	rank of sending process (integer)
IN comm	communicator (handle)

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

```
MPI_SCATTER(SENDBUF, SENDCOUNT, SENDTYPE, RECVBUF, RECVCOUNT, RECVTYPE, ROOT,
COMM, IERROR)
<type> SENDBUF(*), RECVBUF(*)
INTEGER SENDCOUNT, SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM, IERROR
```

MPI_SCATTER is the inverse operation to MPI_GATHER.

The outcome is *as if* the root executed `n` send operations,

MPI_Send(sendbuf+i· sendcount· extent(sendtype), sendcount, sendtype, i,...),

and each process executed a receive,

MPI_Recv(recvbuf, recvcount, recvtype, i,...).

An alternative description is that the root sends a message with MPI_Send(sendbuf, sendcount·n, sendtype, ...). This message is split into `n` equal segments, the *i*th segment is sent to the *i*th process in the group, and each process receives this message as above.

The send buffer is ignored for all non-root processes.

The type signature associated with sendcount, sendtype at the root must be equal to the type signature associated with recvcount, recvtype at all processes (however, the type maps may be different). This implies that the amount of data sent must be equal to the amount of data received, pairwise between each process and the root. Distinct type maps between sender and receiver are still allowed.

All arguments to the function are significant on process root, while on other processes, only arguments recvbuf, recvcount, recvtype, root, comm are significant. The arguments root and comm must have identical values on all processes.

The specification of counts and types should not cause any location on the root to be read more than once.

Rationale.

Though not needed, the last restriction is imposed so as to achieve symmetry with MPI_GATHER, where the corresponding restriction (a multiple-write restriction) is necessary. (*End of rationale.*)

MPI_SCATTERV(sendbuf, sendcounts, displs, sendtype, recvbuf, recvcount, recvtype, root, comm)

IN sendbuf	address of send buffer (choice, significant only at root)
IN sendcounts	integer array (of length group size) specifying the number of elements to send to each processor
IN displs	integer array (of length group size). Entry `i` specifies the displacement (relative to sendbuf from which to take the outgoing data to process `i`
IN sendtype	data type of send buffer elements (handle)
OUT recvbuf	address of receive buffer (choice)
IN recvcount	number of elements in receive buffer (integer)
IN recvtype	data type of receive buffer elements (handle)
IN root	rank of sending process (integer)
IN comm	communicator (handle)

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

```
MPI_SCATTERV(SENDBUF, SENDCOUNTS, DISPLS, SENDTYPE, RECVBUF, RECVCOUNT,
RECVTYPE, ROOT, COMM, IERROR)
<type> SENDBUF(*), RECVBUF(*)
INTEGER SENDCOUNTS(*), DISPLS(*), SENDTYPE, RECVCOUNT, RECVTYPE, ROOT, COMM,
IERROR
```

MPI_SCATTERV is the inverse operation to MPI_GATHERV.

MPI_SCATTERV extends the functionality of MPI_SCATTER by allowing a varying count of data to be sent to each process, since sendcounts is now an array. It also allows more flexibility as to where the data is taken from on the root, by providing the new argument, displs.

The outcome is as if the root executed `n` send operations,

MPI_Send(sendbuf+displs[i]· extent(sendtype), sendcounts[i], sendtype, i,...),

and each process executed a receive,

MPI_Recv(recvbuf, recvcount, recvtype, i,...).

The send buffer is ignored for all non-root processes.

The type signature implied by sendcount[i], sendtype at the root must be equal to the type signature implied by recvcount, recvtype at process `i` (however, the type maps may be different). This implies that the amount of data sent must be equal to the amount of data received, pairwise between each process and the root. Distinct type maps between sender and receiver are still allowed.

All arguments to the function are significant on process root, while on other processes, only arguments recvbuf, recvcount, recvtype, root, comm are significant. The arguments root and comm must have identical values on all processes.

The specification of counts, types, and displacements should not cause any location on the root to be read more than once.
