If comm is an intracommunicator, MPI_ALLREDUCE behaves the same as MPI_REDUCE except that the result appears in the receive buffer of all the group members.

Advice to implementors. The all-reduce operations can be implemented as a reduce, followed by a broadcast. However, a direct implementation can lead to better performance. (*End of advice to implementors.*)

The "in place" option for intracommunicators is specified by passing the value MPI_IN_PLACE to the argument sendbuf at all processes. In this case, the input data is taken at each process from the receive buffer, where it will be replaced by the output data.

If comm is an intercommunicator, then the result of the reduction of the data provided by processes in group A is stored at each process in group B, and vice versa. Both groups should provide count and datatype arguments that specify the same type signature.

The following example uses an intracommunicator.

Example 5.21 A routine that computes the product of a vector and an array that are distributed across a group of processes and returns the answer at all nodes (see also Example 5.16).

```
SUBROUTINE PAR_BLAS2(m, n, a, b, c, comm)
REAL a(m), b(m,n)    ! local slice of array
REAL c(n)            ! result
REAL sum(n)
INTEGER n, comm, i, j, ierr

! local sum
DO j= 1, n
  sum(j) = 0.0
  DO i = 1, m
    sum(j) = sum(j) + a(i)*b(i,j)
  END DO
END DO

! global sum
CALL MPI_ALLREDUCE(sum, c, n, MPI_REAL, MPI_SUM, comm, ierr)

! return result at all nodes
RETURN
```

ticket24.

5.9.7 Process-local reduction

ticket24.

The functions in this section are of importance to library implementors who may want to implement special reduction patterns that are otherwise not easily covered by the standard MPI operations.

The following function applies a reduction operator to local arguments.

MPI_REDUCE_LOCAL(inbuf, inoutbuf, count, datatype, op)

IN	inbuf	input buffer (choice)
INOUT	inoutbuf	combined input and output buffer (choice)
IN	count	number of elements in inbuf and inoutbuf buffers (non-negative integer)
IN	datatype	data type of elements of inbuf and inoutbuf buffers (handle)
IN	op	operation (handle)

```
int MPI_Reduce_local(void* inbuf, void* inoutbuf, int count,
             MPI_Datatype datatype, MPI_Op op)

MPI_REDUCE_LOCAL(INBUF, INOUBUF, COUNT, DATATYPE, OP, IERROR)
    <type> INBUF(*), INOUTBUF(*)
    INTEGER COUNT, DATATYPE, OP, IERROR

{void MPI::Op::Reduce_local(const void* inbuf, void* inoutbuf, int count,
             const MPI::Datatype& datatype) const (binding deprecated, see
             Section 15.2) }
```

ticket150.

ticket150.

The function applies the operation given by op element-wise to the elements of inbuf and inoutbuf with the result stored element-wise in inoutbuf, as explained for user-defined operations in Section 5.9.5. Both inbuf and inoutbuf (input as well as result) have the same number of elements given by count and the same datatype given by datatype. The MPI_IN_PLACE option is not allowed.

Reduction operations can be queried for their commutativity.

MPI_OP_COMMUTATIVE(op, commute)

IN	op	operation (handle)
OUT	commute	true if op is commutative, false otherwise (logical)

```
int MPI_Op_commutative(MPI_Op op, int *commute)

MPI_OP_COMMUTATIVE(OP, COMMUTE, IERROR)
    LOGICAL COMMUTE
    INTEGER OP, IERROR

{bool MPI::Op::Is_commutative() const (binding deprecated, see Section 15.2) }
```

ticket150.

ticket150.

5.10 Reduce-Scatter

ticket27.

[MPI includes a variant of the reduce operations where the result is scattered to all processes in a group on return.]MPI includes variants of the reduce operations where the result is scattered to all processes in a group on return. One variant scatters equal-sized blocks to all processes, while another variant scatters blocks that may vary in size for each process.

ticket27.

Annex B

Change-Log

This annex summarizes changes from the previous version of the MPI standard to the version presented by this document. [Only changes (i.e., clarifications and new features) are presented that may cause implementation effort in the MPI libraries.] Only significant changes (i.e., clarifications and new features) that might either require implementation effort in the MPI libraries or change the understanding of MPI from a user's perspective are presented. Editorial modifications, formatting, typo corrections and minor clarifications are not shown.

B.1 Changes from Version 2.1 to Version 2.2

1. Section 2.5.4 on page 14.
   It is now guaranteed that predefined named constant handles (as other constants) can be used in initialization expressions or assignments, i.e., also before the call to MPI_INIT.

2. Section 2.6 on page 16, Section 2.6.4 on page 19, and Section 16.1 on page 469.
   The C++ language bindings have been deprecated and will be removed in a future version of the MPI specification.

3. Section 3.2.2 on page 29.
   MPI_CHAR for printable characters is now defined for C type char (instead of signed char). This change should not have any impact on applications nor on MPI libraries (except some comment lines), because printable characters could and can be stored in any of the C types char, signed char, and unsigned char, and MPI_CHAR is not allowed for predefined reduction operations.

4. Section 3.2.2 on page 29.
   MPI_(U)INT{8,16,32,64}_T, MPI_AINT, MPI_OFFSET, MPI_C_BOOL, MPI_C_COMPLEX, MPI_C_FLOAT_COMPLEX, MPI_C_DOUBLE_COMPLEX, and MPI_C_LONG_DOUBLE_COMPLEX are now valid predefined MPI datatypes.

5. Section 3.4 on page 40, Section 3.7.2 on page 51, Section 3.9 on page 71, and Section 5.1 on page 133.
   The read access restriction on the send buffer for blocking, non blocking and collective API has been lifted. It is permitted to access for read the send buffer while the operation is in progress.

6. Section 3.7 on page 50.
   The Advice to users for IBSEND and IRSEND was slightly changed. ticket143.

7. Section 3.7.3 on page 54.
   The advice to free an active request was removed in the Advice to users for MPI_REQUEST_FREE. ticket137.

8. Section 3.7.6 on page 66.
   MPI_REQUEST_GET_STATUS changed to permit inactive or null requests as input. ticket31.

9. Section 5.8 on page 159.
   "In place" option is added to MPI_ALLTOALL, MPI_ALLTOALLV, and MPI_ALLTOALLW for intracommunicators. ticket64.

10. Section 5.9.2 on page 167.
    Predefined parameterized datatypes (e.g., returned by MPI_TYPE_CREATE_F90_REAL) and optional named predefined datatypes (e.g. MPI_REAL8) have been added to the list of valid datatypes in reduction operations. ticket18.

11. Section 5.9.2 on page 167.
    MPI_(U)INT{8,16,32,64}_T are all considered C integer types for the purposes of the predefined reduction operators. MPI_AINT and MPI_OFFSET are considered Fortran integer types. MPI_C_BOOL is considered a Logical type.
    MPI_C_COMPLEX, MPI_C_FLOAT_COMPLEX, MPI_C_DOUBLE_COMPLEX, and MPI_C_LONG_DOUBLE_COMPLEX are considered Complex types. ticket24.

12. Section 5.9.7 on page 178.
    The local routines MPI_REDUCE_LOCAL and MPI_OP_COMMUTATIVE have been added. ticket27.

13. Section 5.10.1 on page 180.
    The collective function MPI_REDUCE_SCATTER_BLOCK is added to the MPI standard. ticket94.

14. Section 5.11.2 on page 183.
    Added in place argument to MPI_EXSCAN. ticket19.

15. Section 6.4.2 on page 202, and Section 6.6 on page 222.
    Implementations that did not implement MPI_COMM_CREATE on intercommunicators will need to add that functionality. As the standard described the behavior of this operation on intercommunicators, it is believed that most implementations already provide this functionality. Note also that the C++ binding for both MPI_COMM_CREATE and MPI_COMM_SPLIT explicitly allow Intercomms. ticket66.

16. Section 6.4.2 on page 202.
    MPI_COMM_CREATE is extended to allow several disjoint subgroups as input if comm is an intracommunicator. If comm is an intercommunicator it was clarified that all processes in the same local group of comm must specify the same value for group. ticket33.

17. Section ?? on page ??.
    New functions for a scalable distributed graph topology interface has been added. In this section, the functions MPI_DIST_GRAPH_CREATE_ADJACENT and