ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...
- optimised communication for performance

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...
- optimised communication for performance
- code readability and maintainability

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...
- optimised communication for performance
- code readability and maintainability
- data synchronisation or temporal synchronisation if barriers are used

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...
- optimised communication for performance
- code readability and maintainability
- data synchronisation or temporal synchronisation if barriers are used
- up to MPI 3.0 they were only blocking

ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...
- optimised communication for performance
- code readability and maintainability
- data synchronisation or temporal synchronisation if barriers are used
- up to MPI 3.0 they were only blocking
- receive buffers must have the same size as send buffers

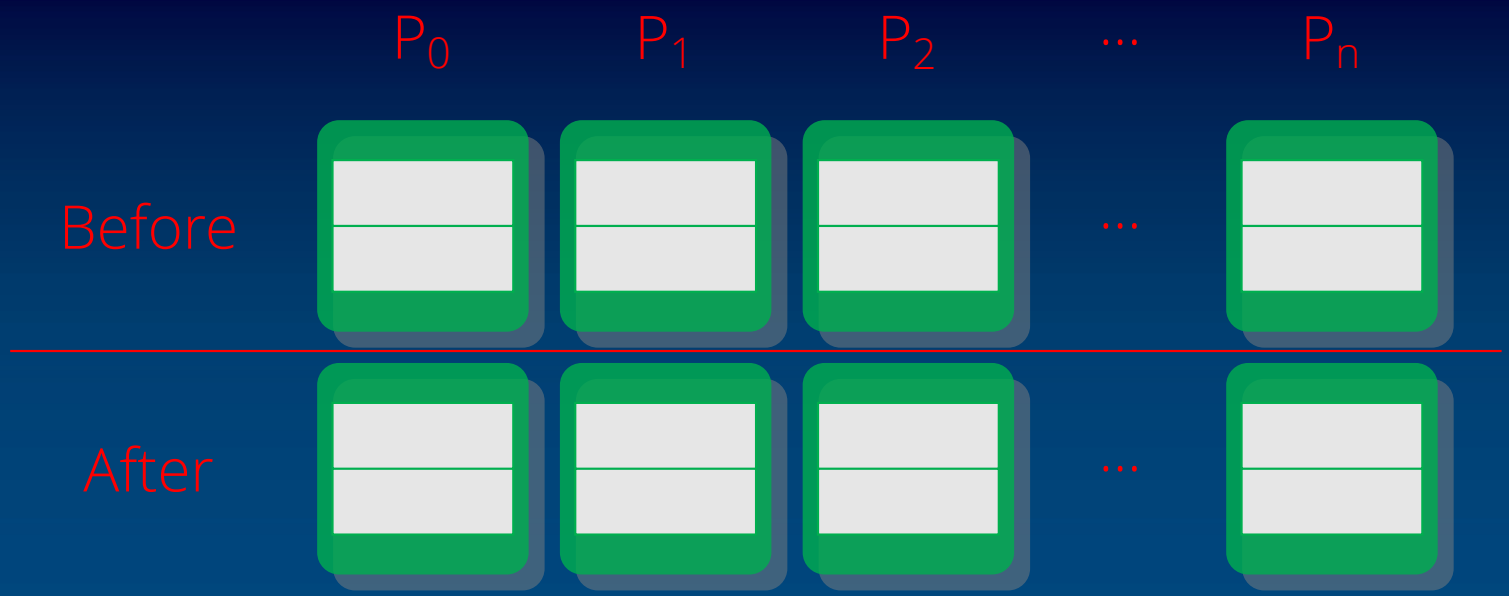
ICHEC
Irish Centre for High-End Computing

Characteristics of collective operations

- involve all the processes grouped in a communicator
- must be called by all the processes in the communicator
- can be done with point-to-point operations but...
- optimised communication for performance
- code readability and maintainability
- data synchronisation or temporal synchronisation if barriers are used
- up to MPI 3.0 they were only blocking
- receive buffers must have the same size as send buffers
- **Examples:** broadcast, scatter, gather, global sum, global maximum, barrier synchronisation...

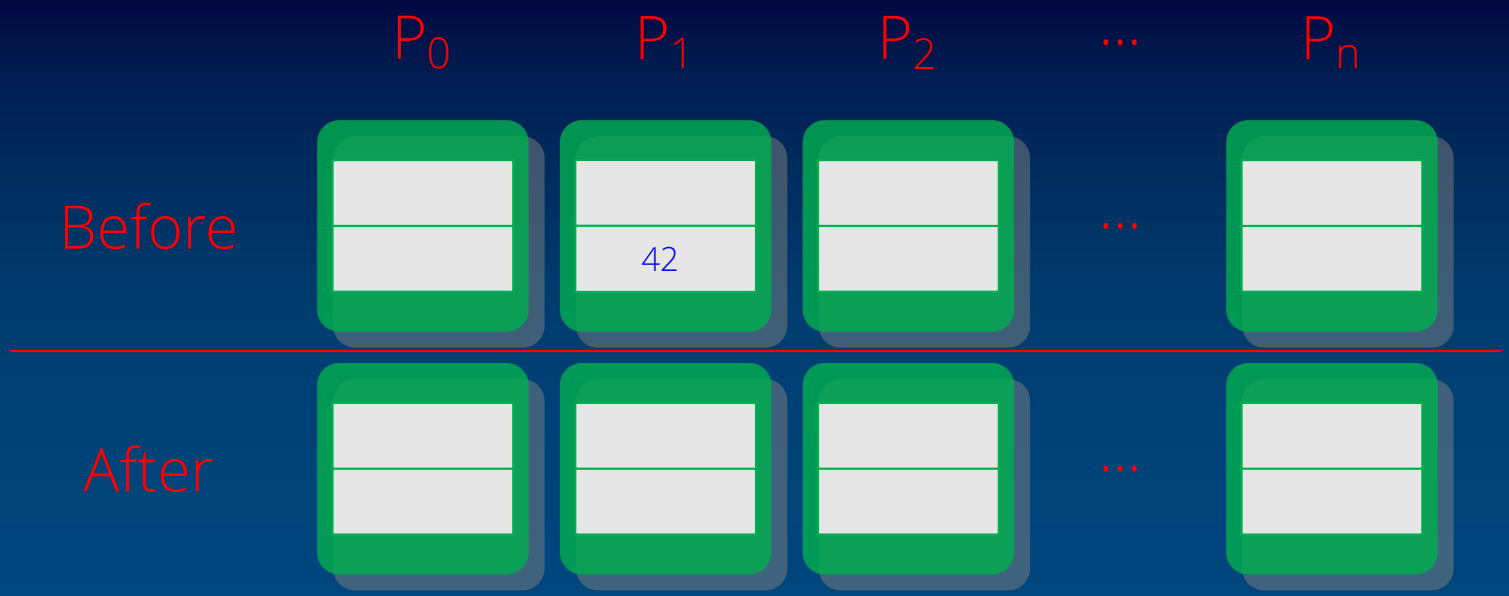
ICHEC
Irish Centre for High-End Computing

broadcast



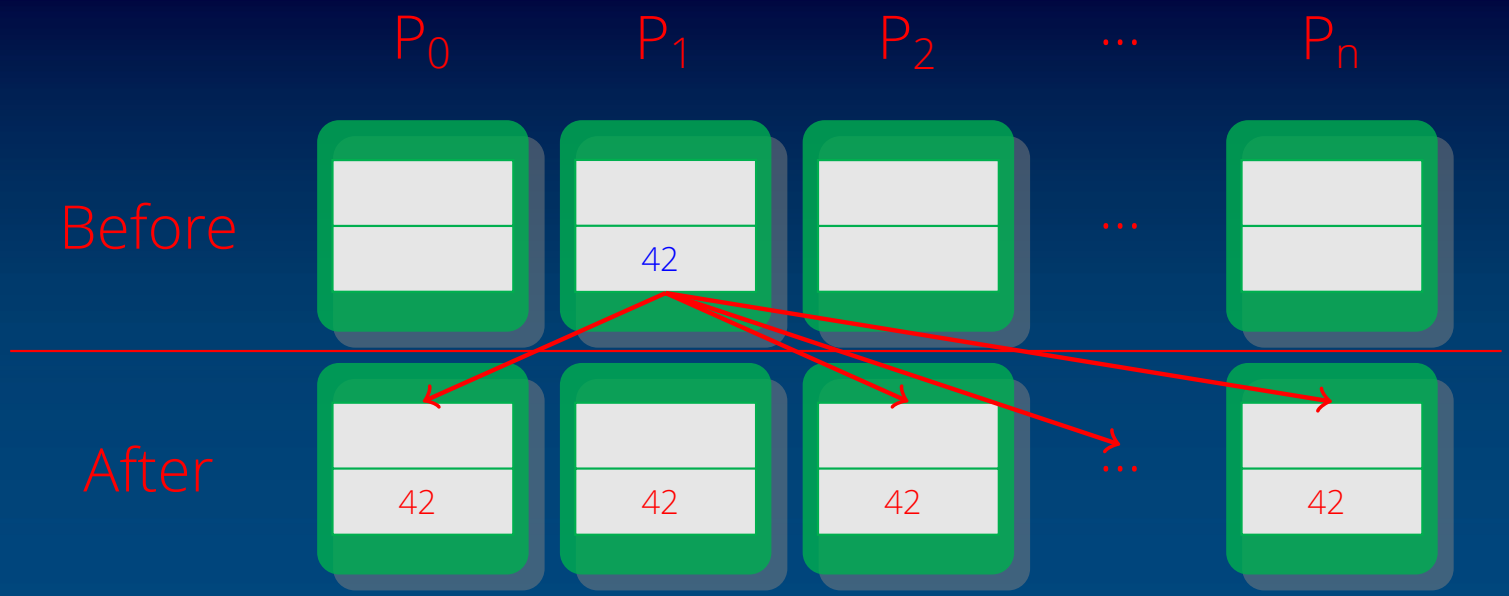
ICHEC
Irish Centre for High-End Computing

broadcast



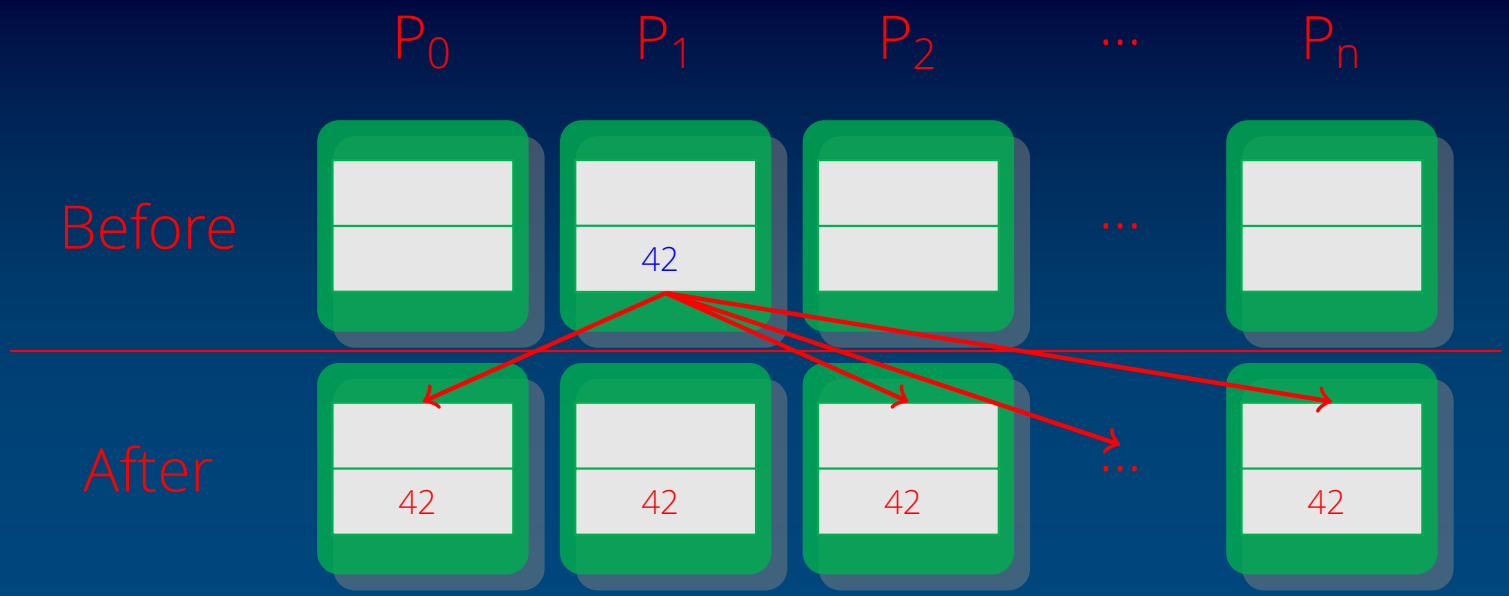
ICHEC
Irish Centre for High-End Computing

broadcast



ICHEC
Irish Centre for High-End Computing

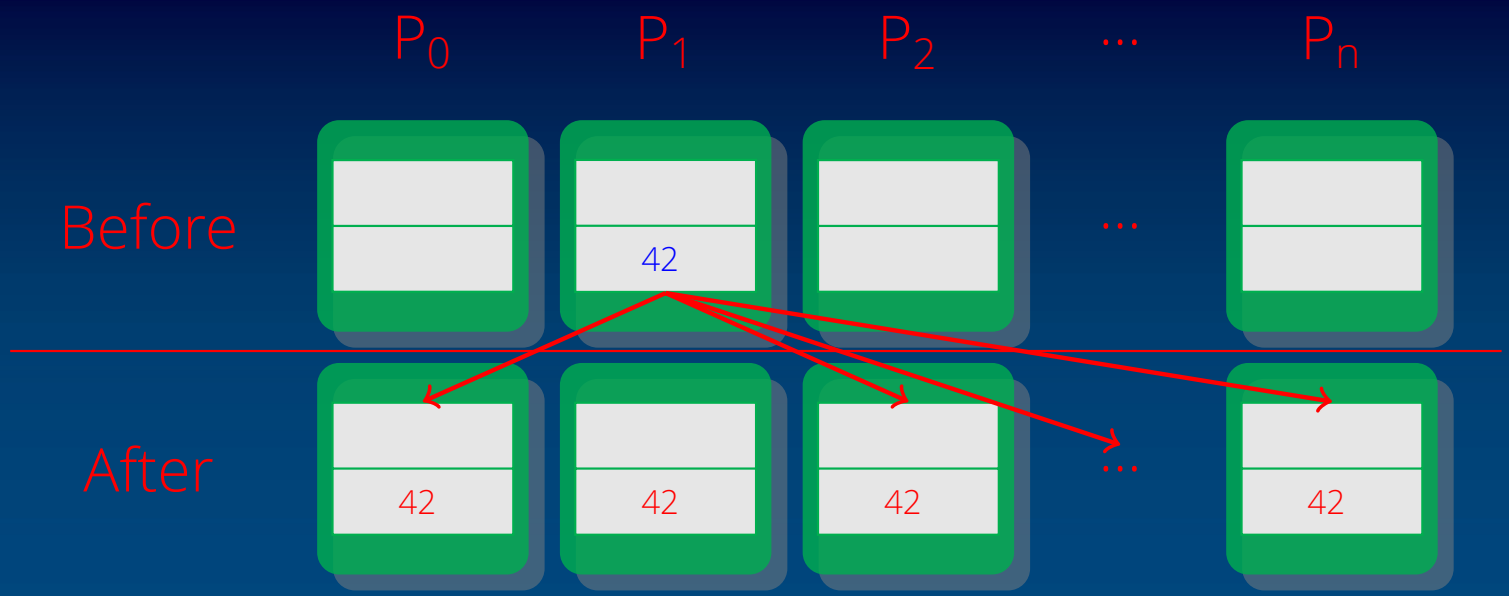
broadcast




```
int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype,
    int root, MPI_Comm comm)
```

ICHEC
Irish Centre for High-End Computing

broadcast

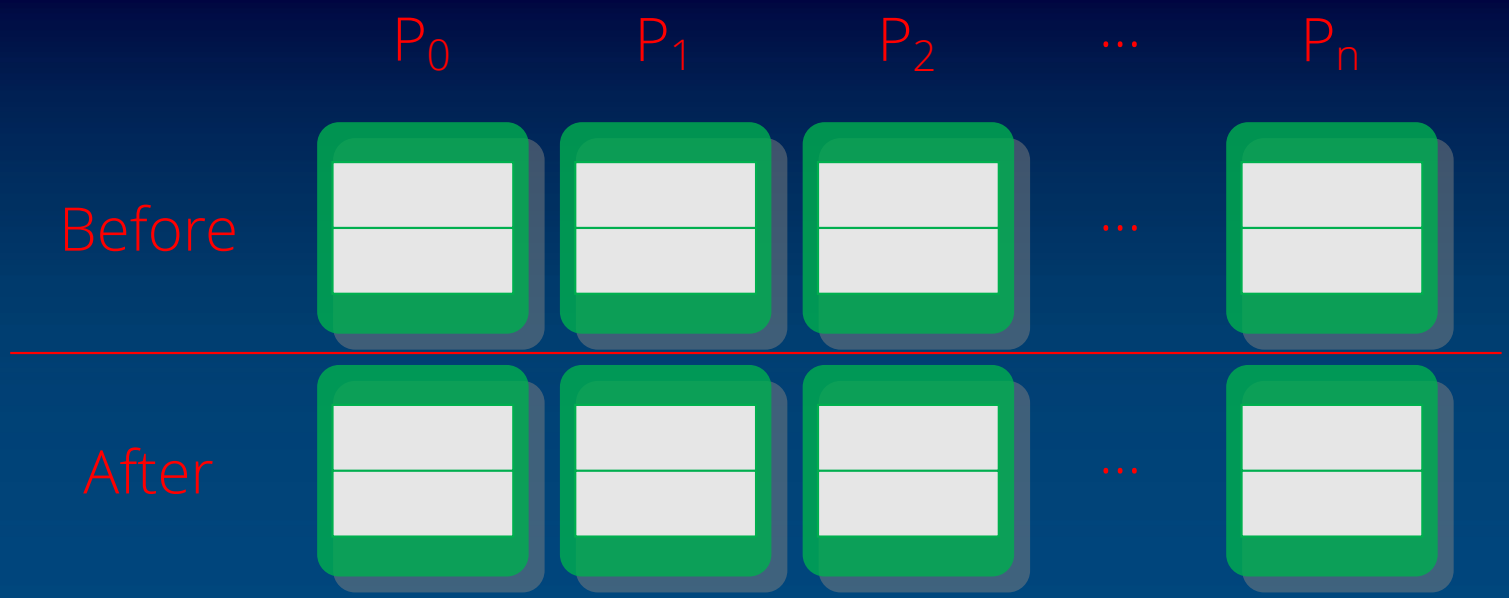



```
int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype,
    int root, MPI_Comm comm)
```

```
MPI_Bcast(buffer, count, datatype, root, comm, ierror)
    <type>  :: buffer(:)
    integer :: count, datatype, root, comm, ierror
```

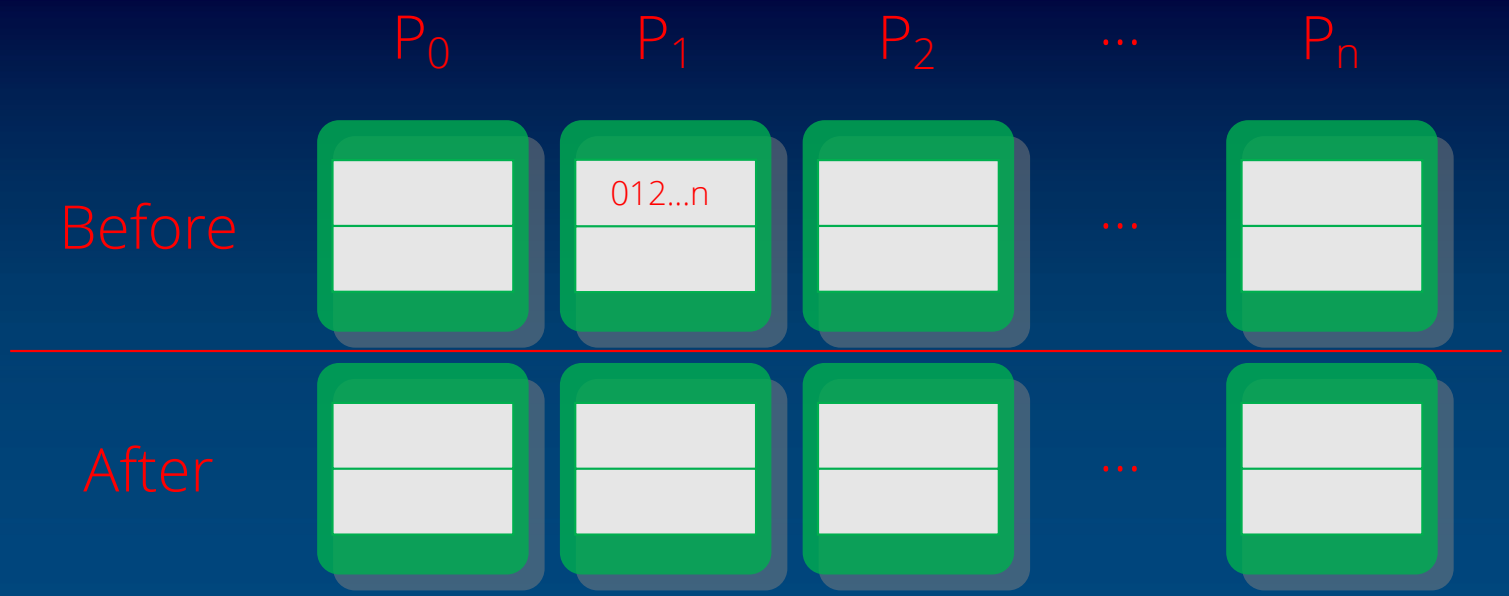
ICHEC
Irish Centre for High-End Computing

scatter



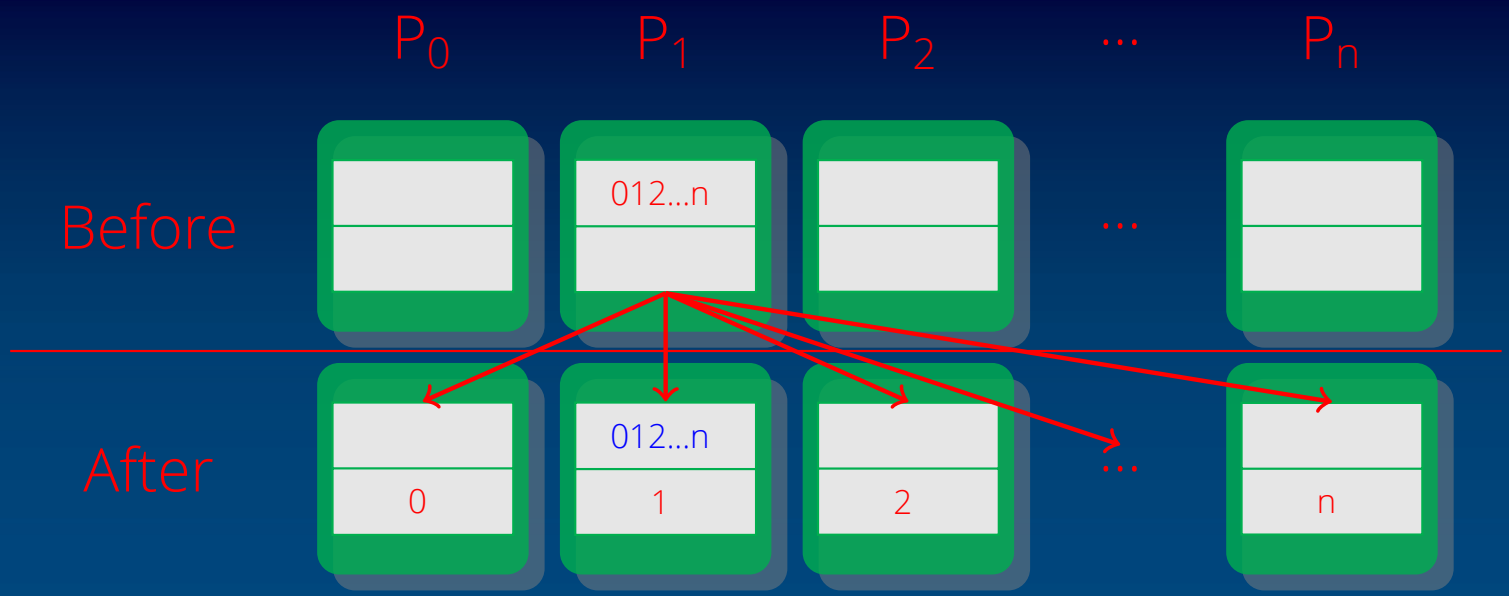
ICHEC
Irish Centre for High-End Computing

scatter



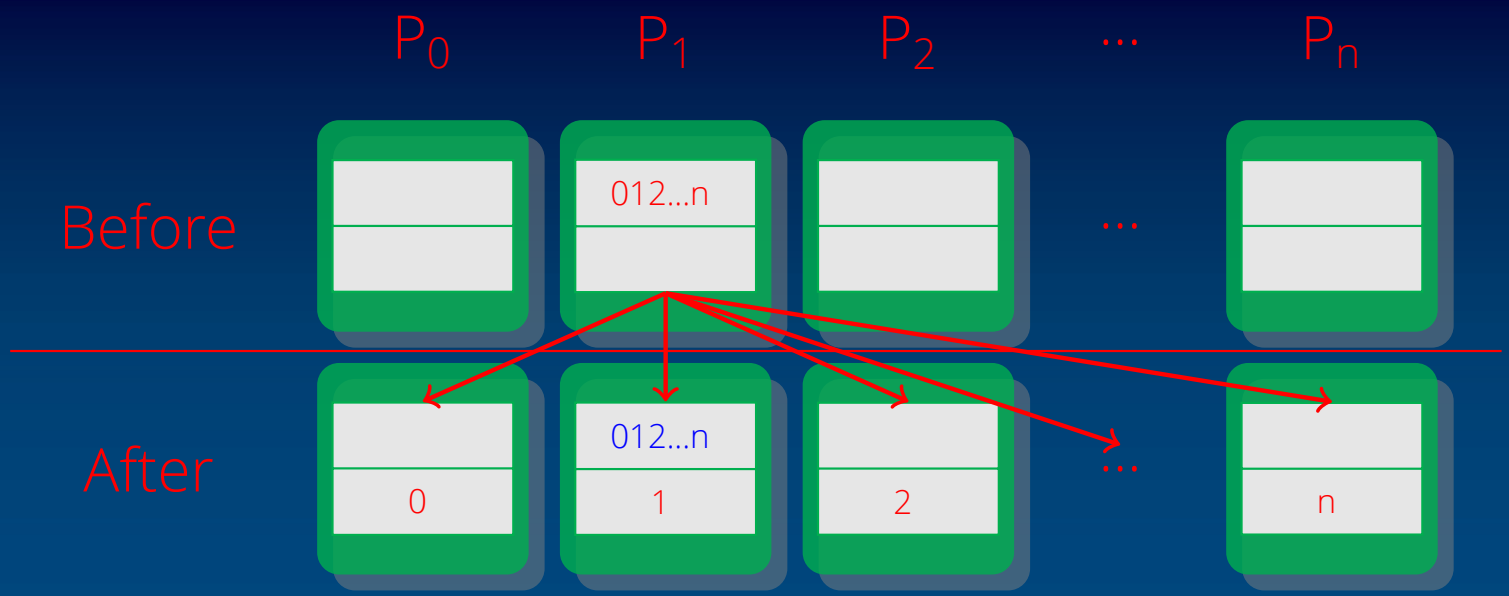
ICHEC
Irish Centre for High-End Computing

scatter



ICHEC
Irish Centre for High-End Computing

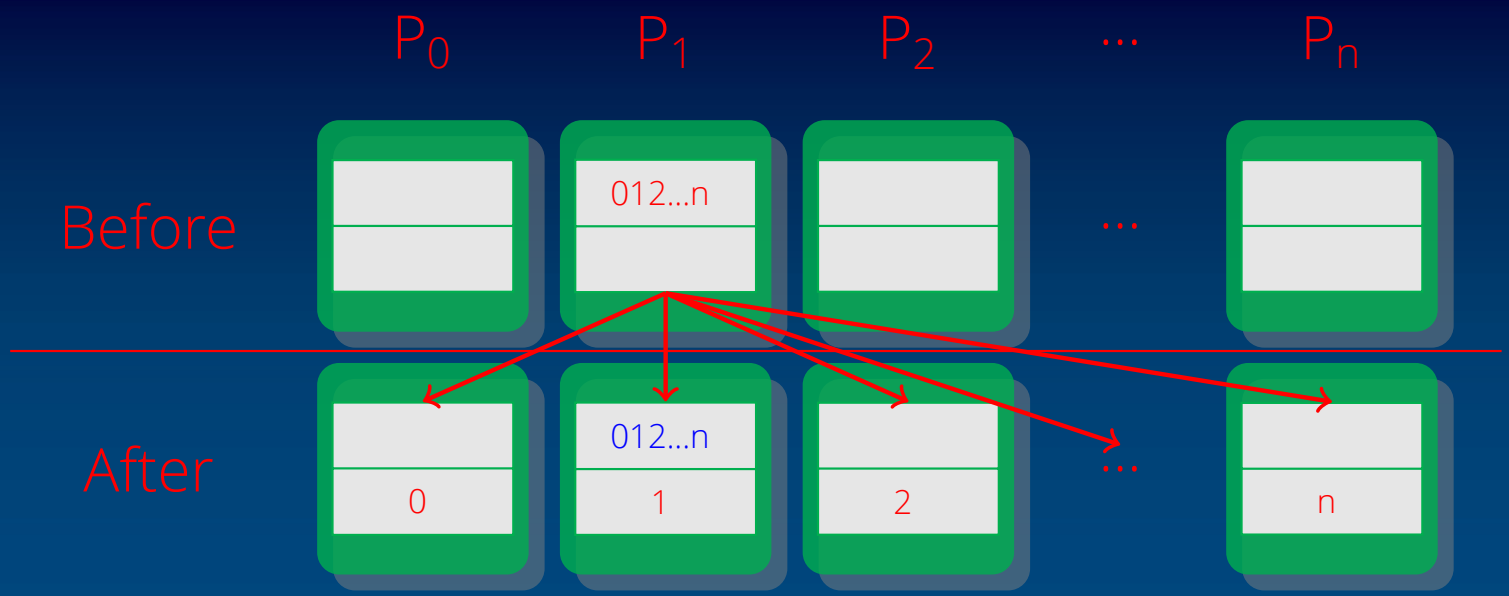
scatter




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

ICHEC
Irish Centre for High-End Computing

scatter

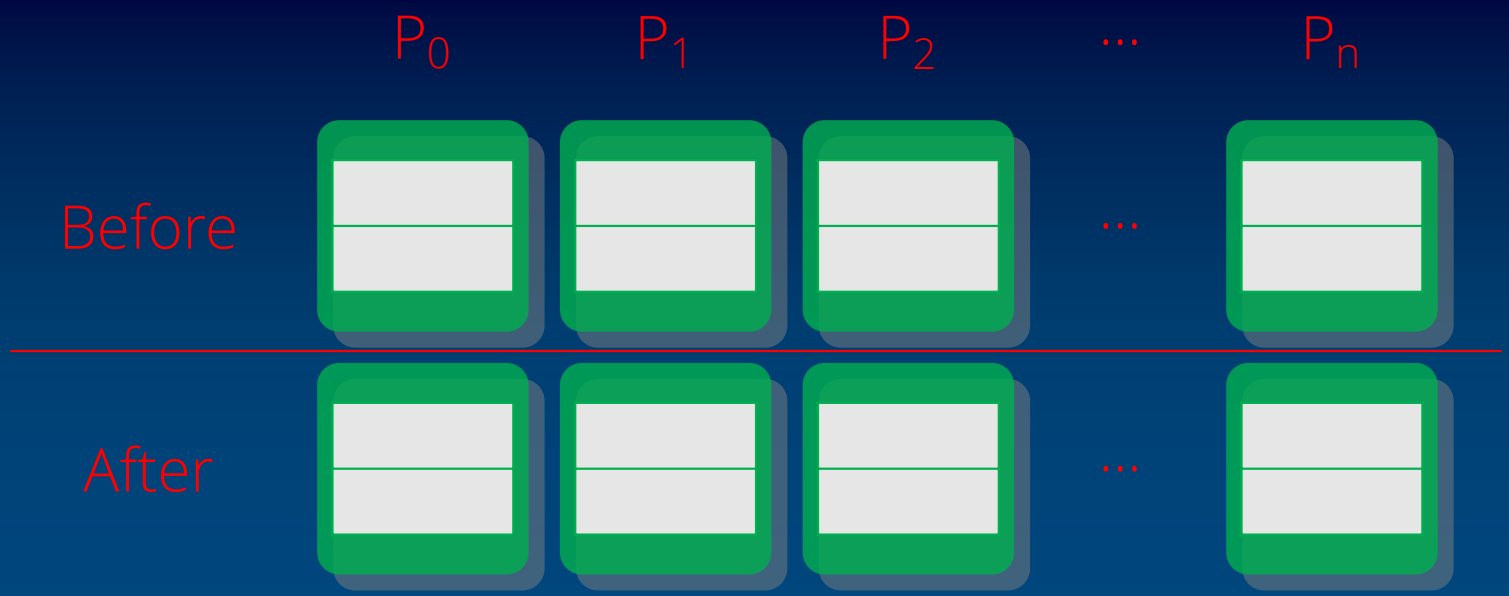



```
int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype
    sendtype, void *recvbuf, int recvcount, MPI_Datatype
    recvtype, int root, MPI_Comm comm)
```

```
MPI_Scatter(sendbuf, sendcount, sendtype, recvbuf, recvcount,
    recvtype, root, comm, ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: sendcount, sendtype, recvcount, recvtype, root
    integer :: comm, ierror
```

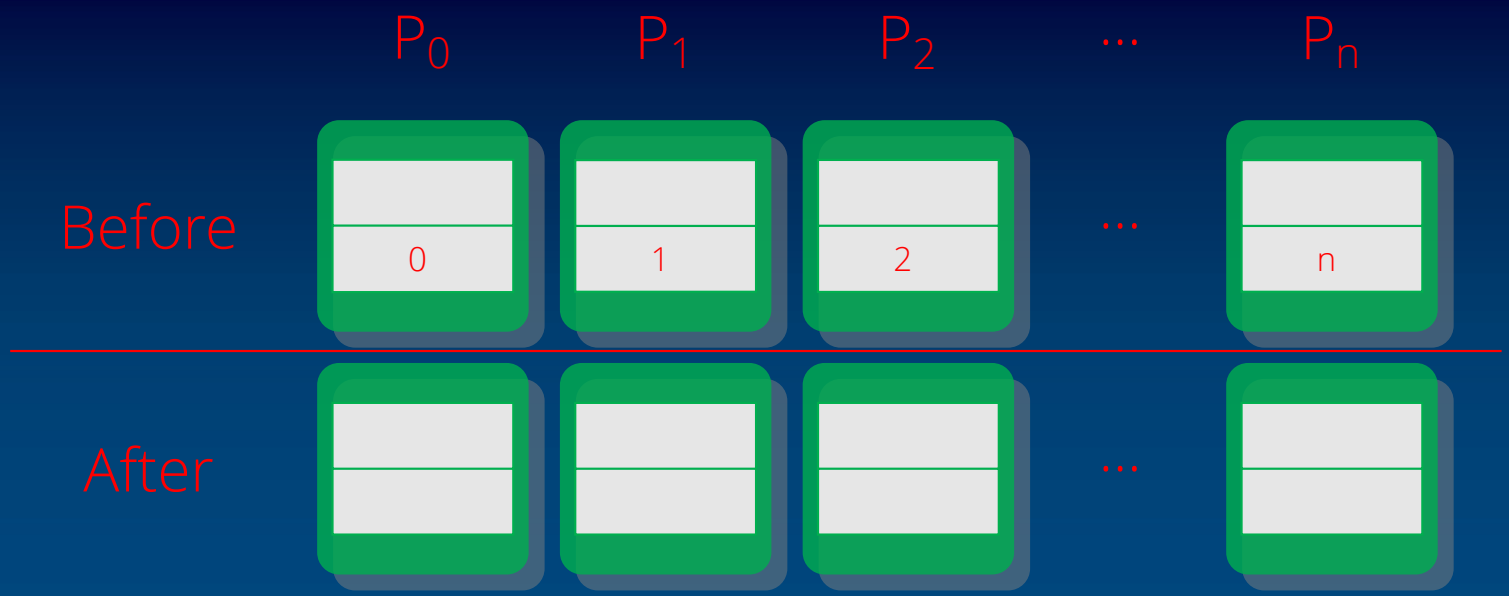
ICHEC
Irish Centre for High-End Computing

gather



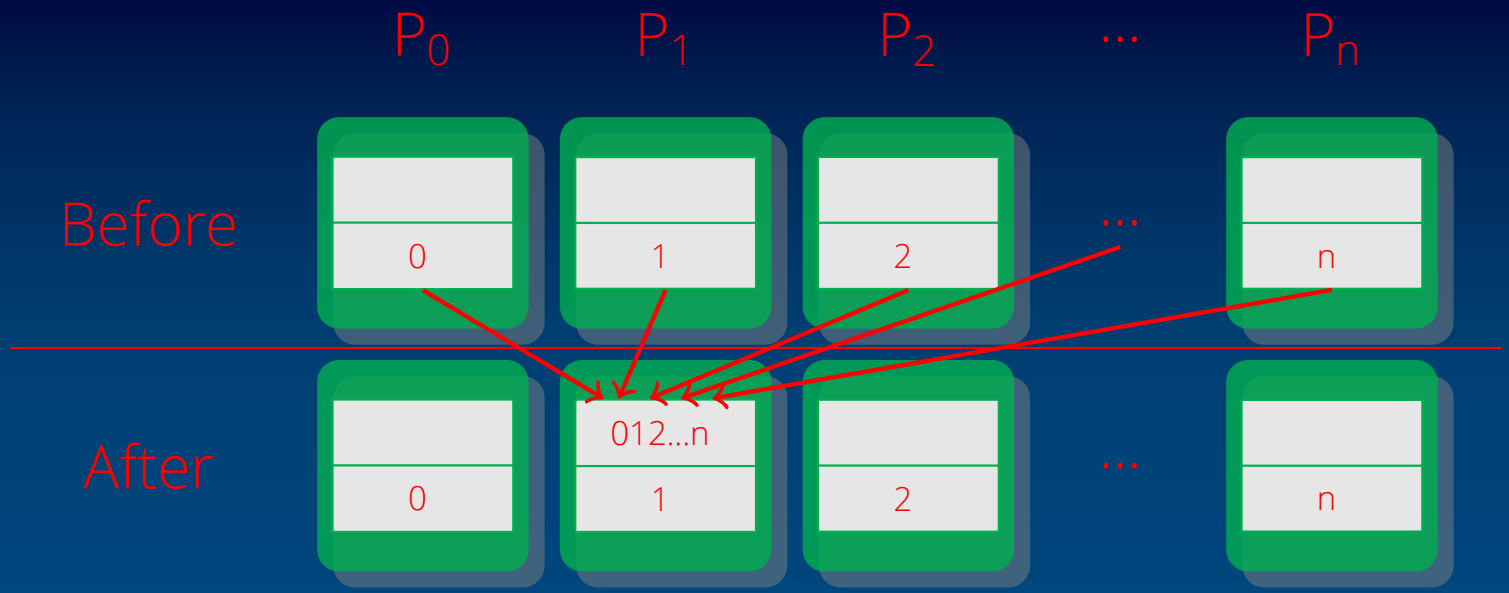
ICHEC
Irish Centre for High-End Computing

gather



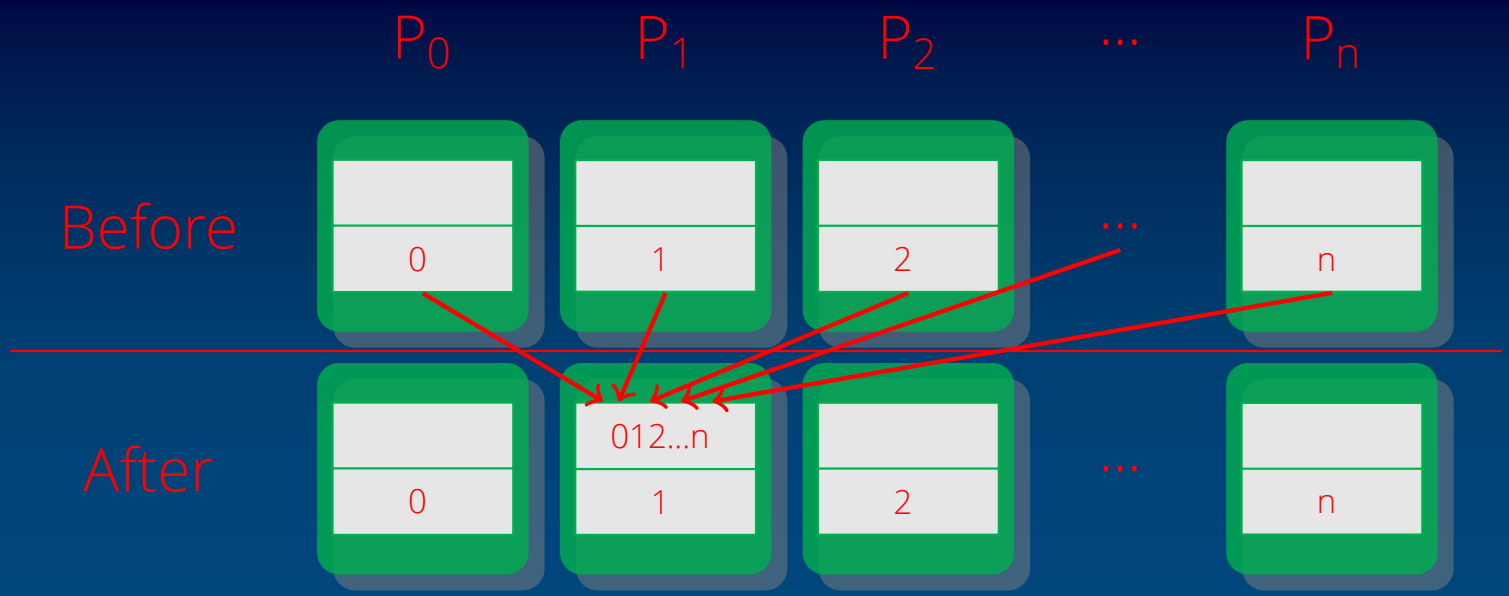
ICHEC
Irish Centre for High-End Computing

gather



ICHEC
Irish Centre for High-End Computing

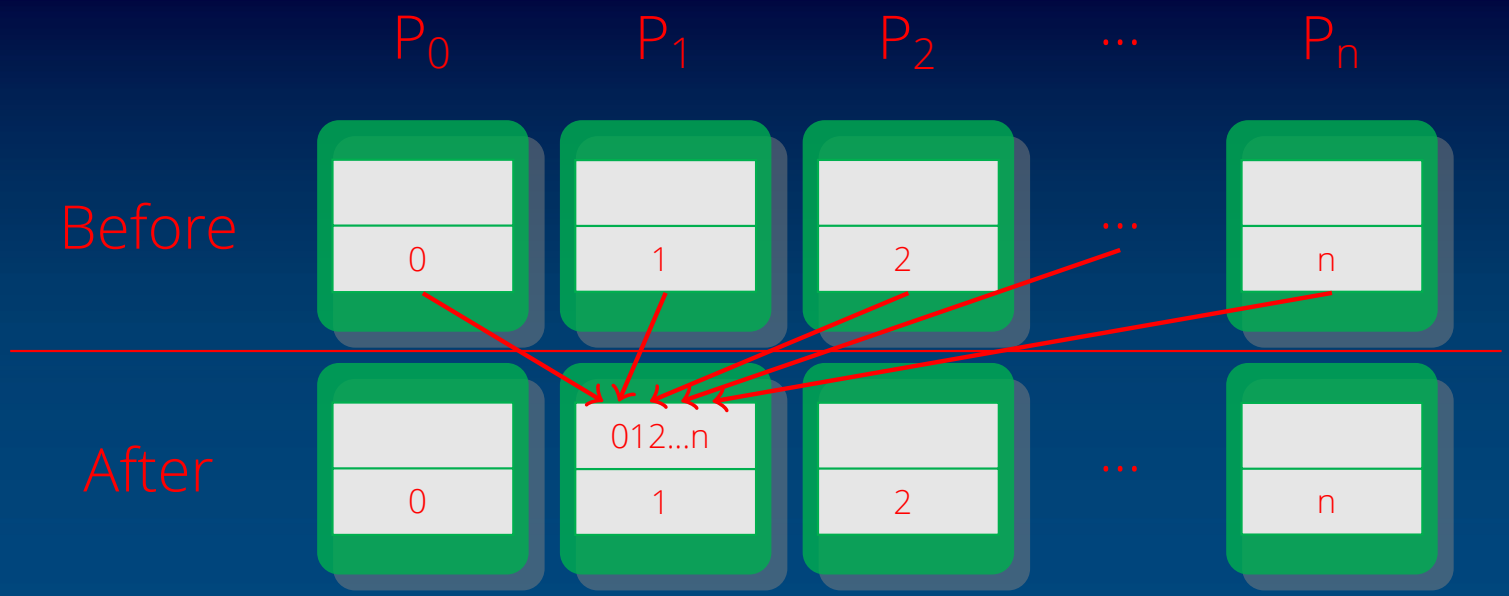
gather




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

ICHEC
Irish Centre for High-End Computing

gather

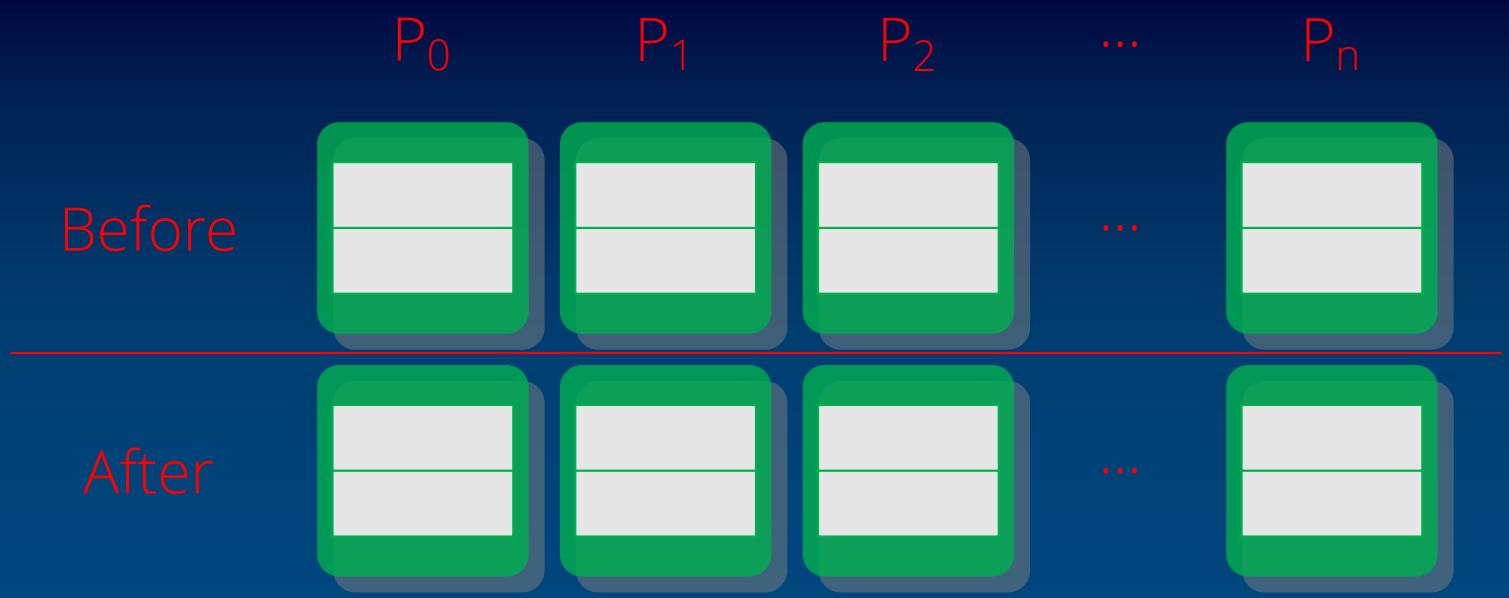



```
int MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype
    sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype
    , int root, MPI_Comm comm)
```

```
MPI_Gather(sendbuf, sendcount, sendtype, recvbuf, recvcount,
    recvtype, root, comm, ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: sendcount, sendtype, recvcount, recvtype, root
    integer :: comm, ierror
```

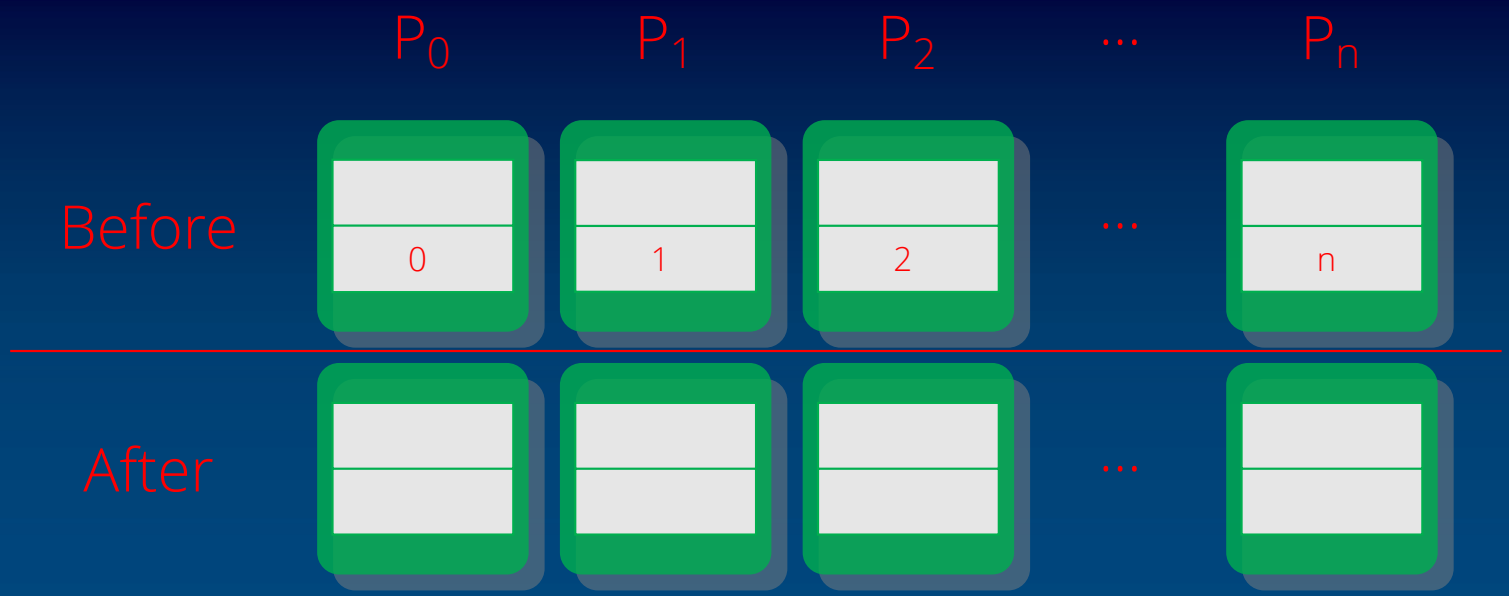
ICHEC
Irish Centre for High-End Computing

AllGather



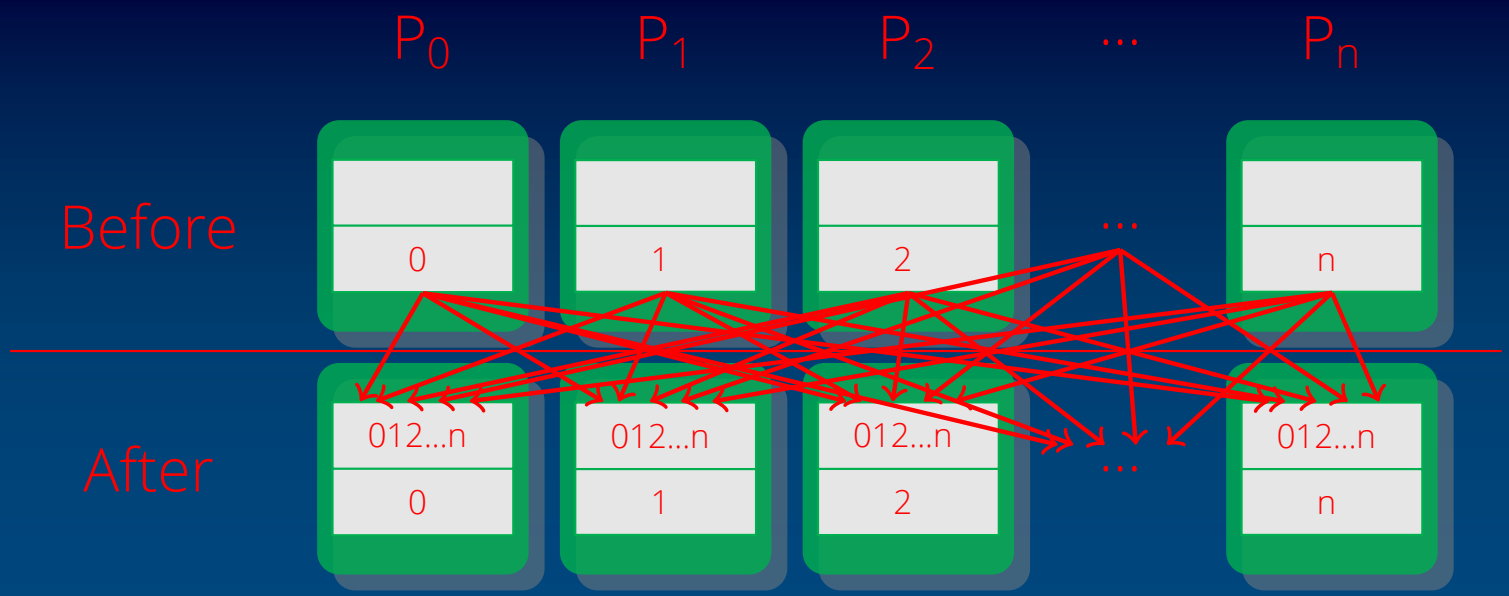
ICHEC
Irish Centre for High-End Computing

AllGather



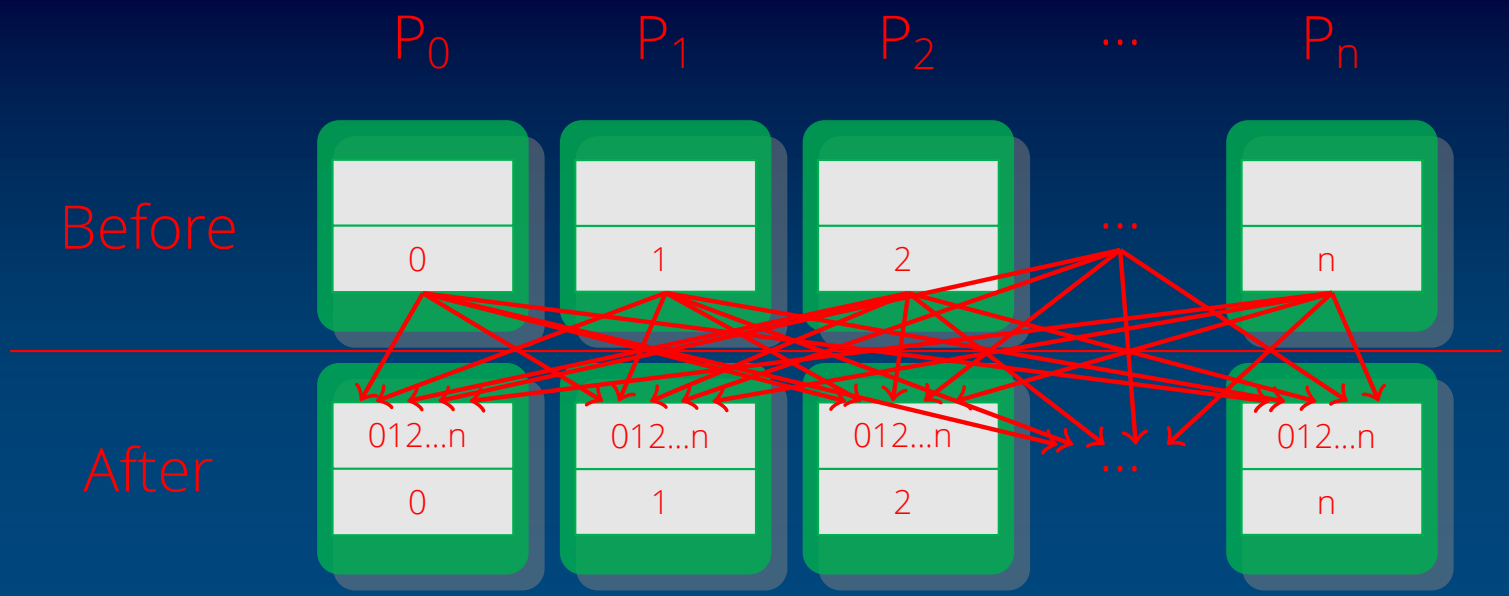
ICHEC
Irish Centre for High-End Computing

AllGather



ICHEC
Irish Centre for High-End Computing

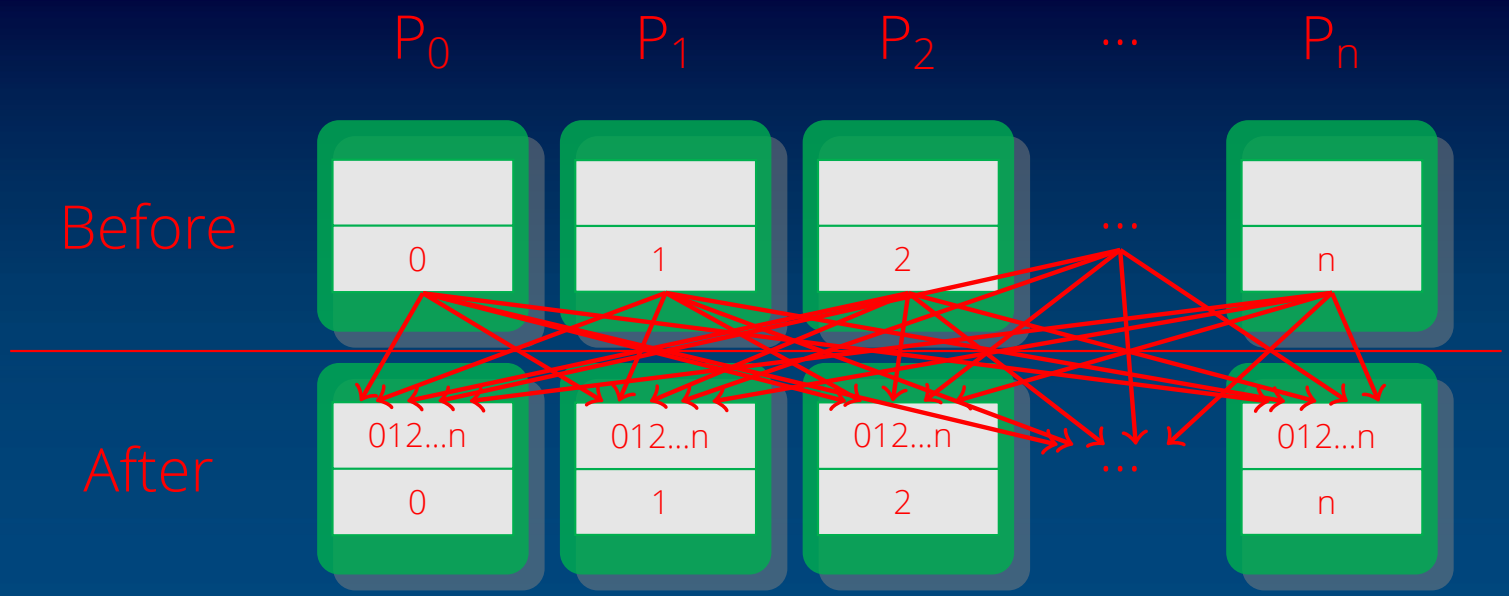
AllGather




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

ICHEC
Irish Centre for High-End Computing

AllGather




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

```
MPI_Allgather(sendbuf, sendcount, sendtype, recvbuf, recvcount,
     recvtype, comm, ierror)
    <type>  :: sendbuf (:), recvbuf (:)
    integer :: sendcount, sendtype, recvcount, recvtype, comm,
        ierror
```

ICHEC
Irish Centre for High-End Computing

AlltoAll

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

1	2
3	4
5	6
7	8

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

1	2	9	10
3	4	11	12
5	6	13	14
7	8	15	16

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

1	2	9	10	17	18
3	4	11	12	19	20
5	6	13	14	21	22
7	8	15	16	23	24

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

1	2	9	10	17	18	25	26
3	4	11	12	19	20	27	28
5	6	13	14	21	22	29	30
7	8	15	16	23	24	31	32

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

1	2	9	10	17	18	25	26
3	4	11	12	19	20	27	28
5	6	13	14	21	22	29	30
7	8	15	16	23	24	31	32

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

ICHEC
Irish Centre for High-End Computing

AlltoAll

P_0	1	2	3	4	5	6	7	8
P_1	9	10	11	12	13	14	15	16
P_2	17	18	19	20	21	22	23	24
P_3	25	26	27	28	29	30	31	32

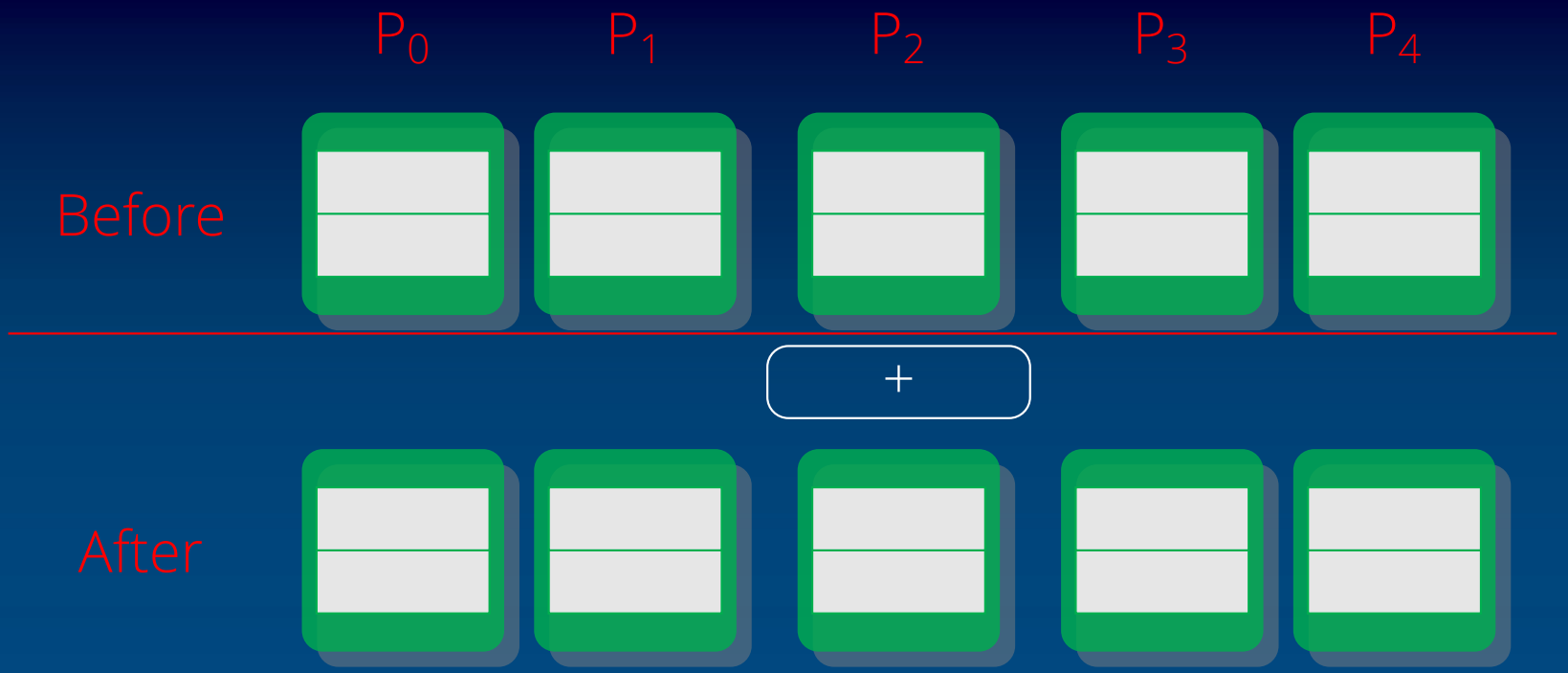
1	2	9	10	17	18	25	26
3	4	11	12	19	20	27	28
5	6	13	14	21	22	29	30
7	8	15	16	23	24	31	32

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

```
MPI_Alltoall(sendbuf, sendcount, sendtype, recvbuf, recvcount,
    recvtype, comm, ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: sendcount, sendtype, recvcount, recvtype
    integer :: comm, ierror
```

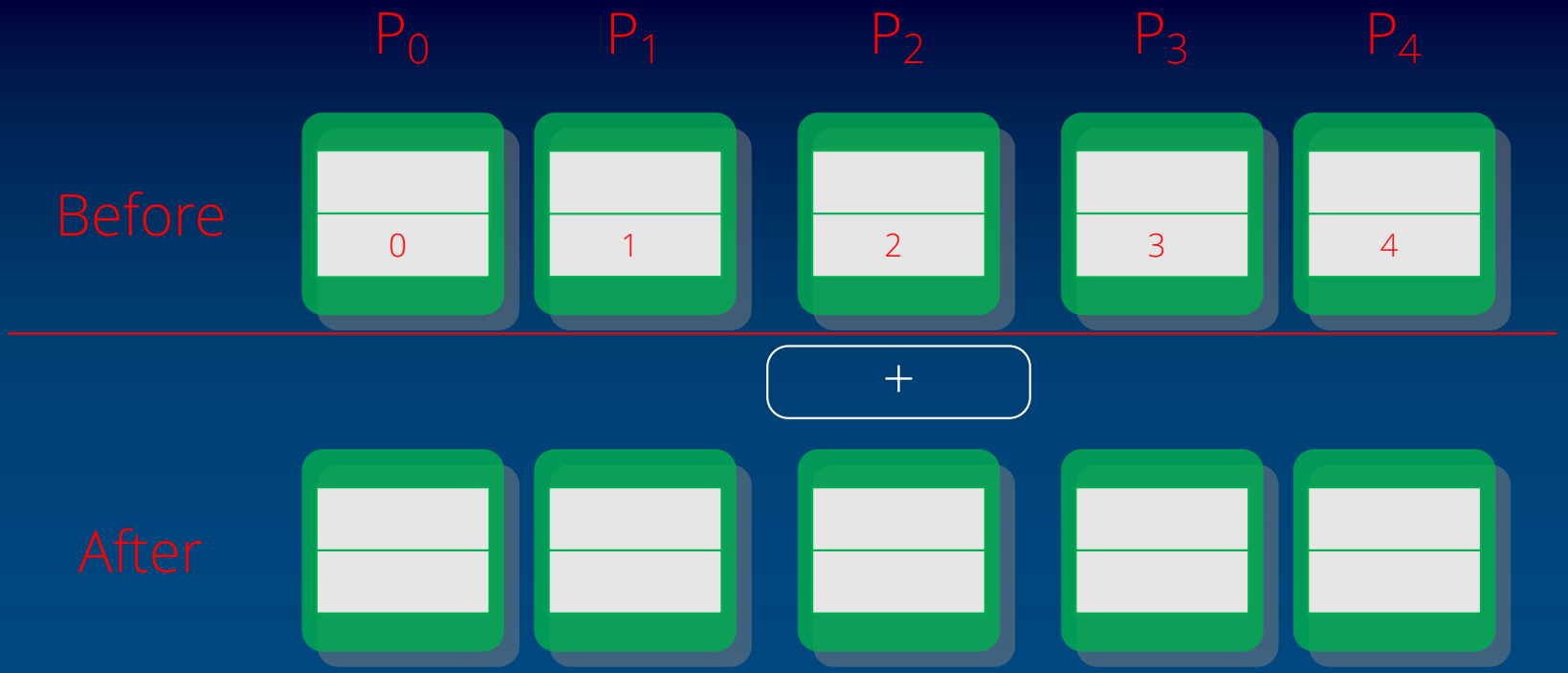
ICHEC
Irish Centre for High-End Computing

reduce



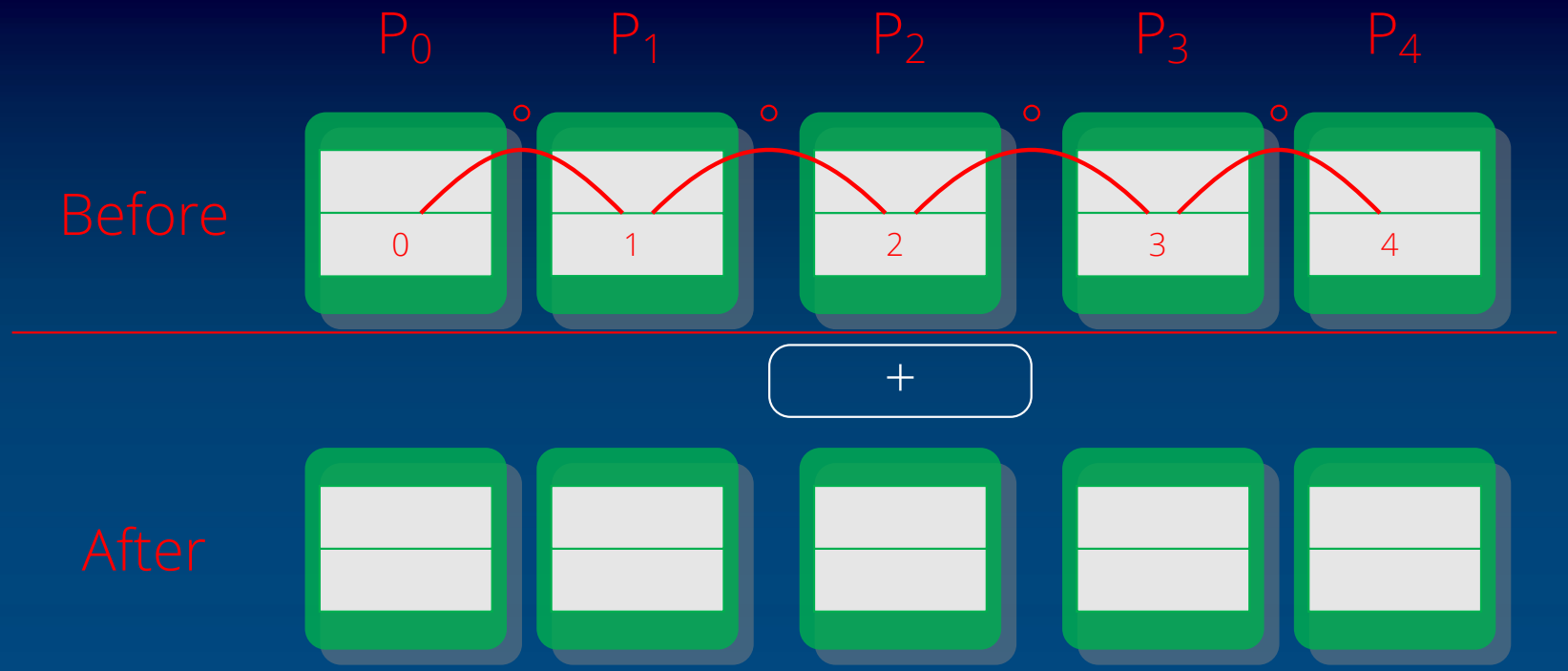
ICHEC
Irish Centre for High-End Computing

reduce



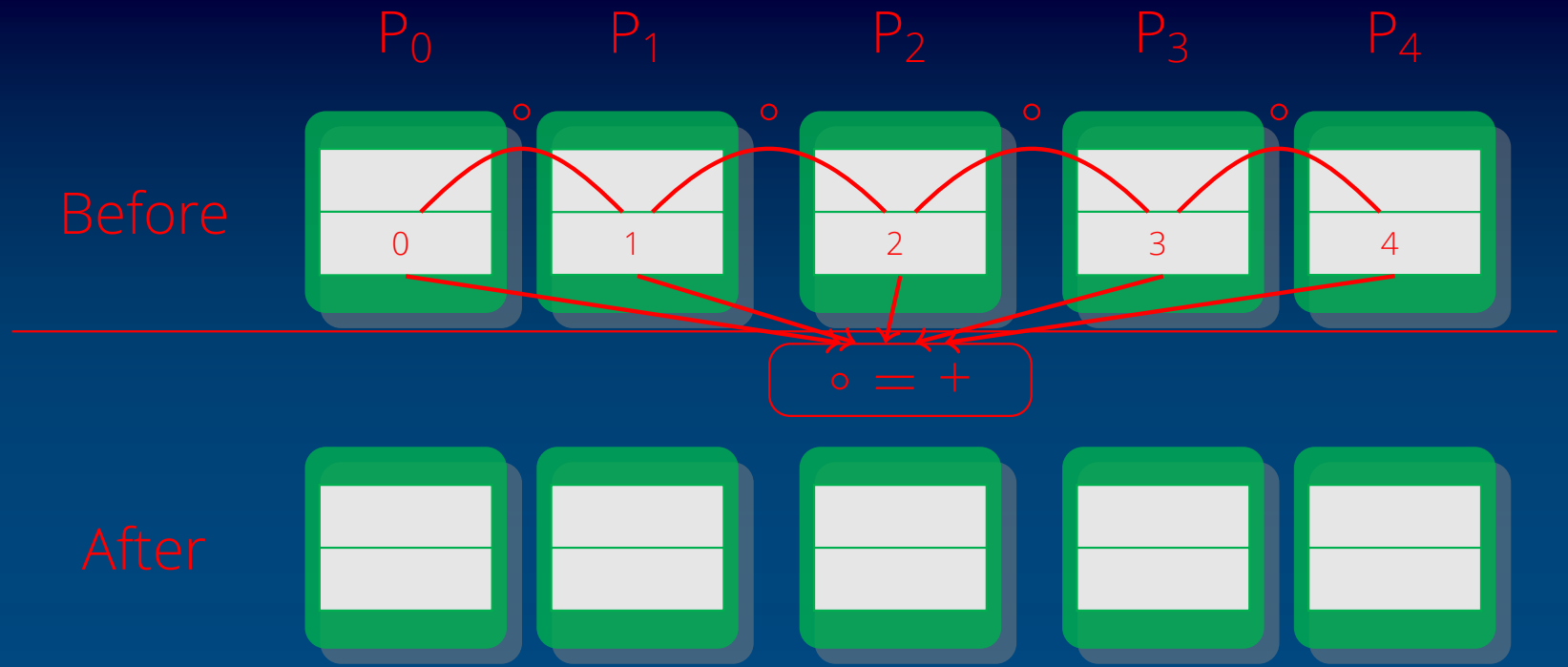
ICHEC
Irish Centre for High-End Computing

reduce



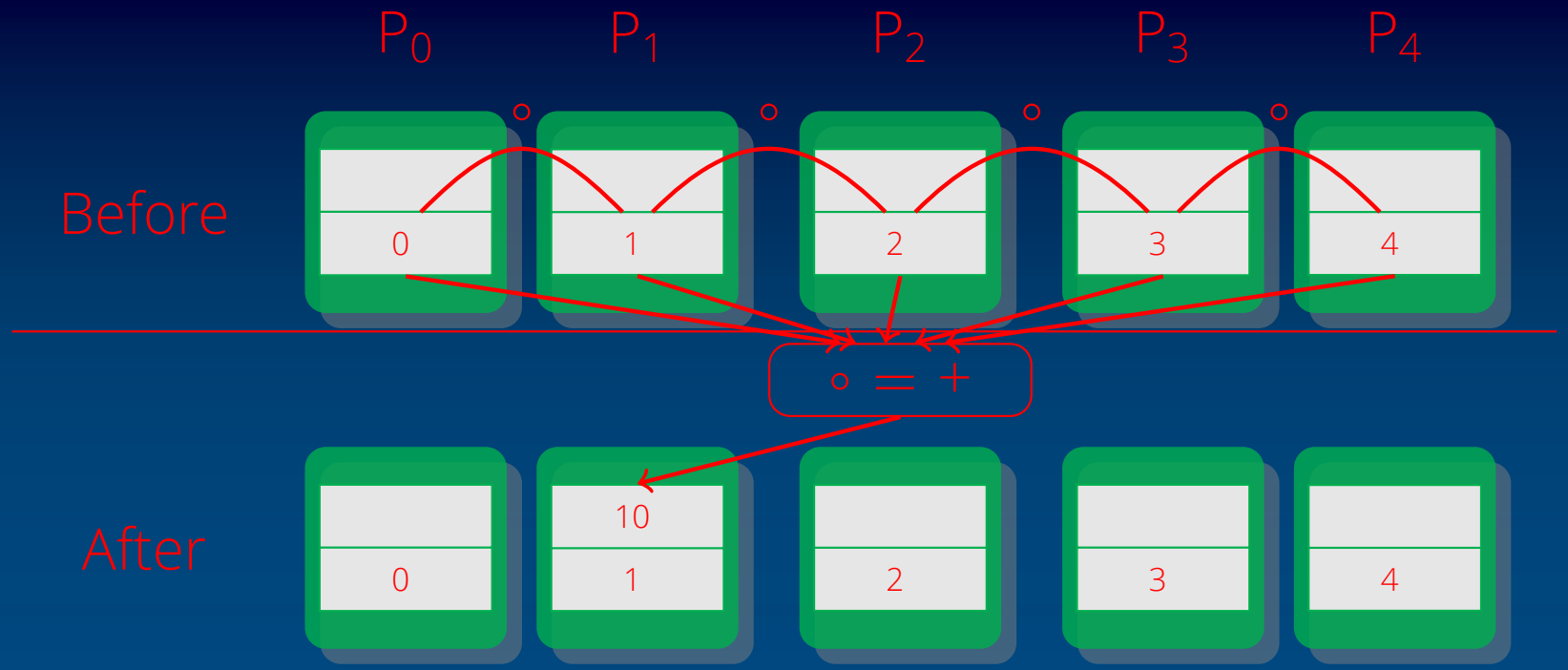
ICHEC
Irish Centre for High-End Computing

reduce



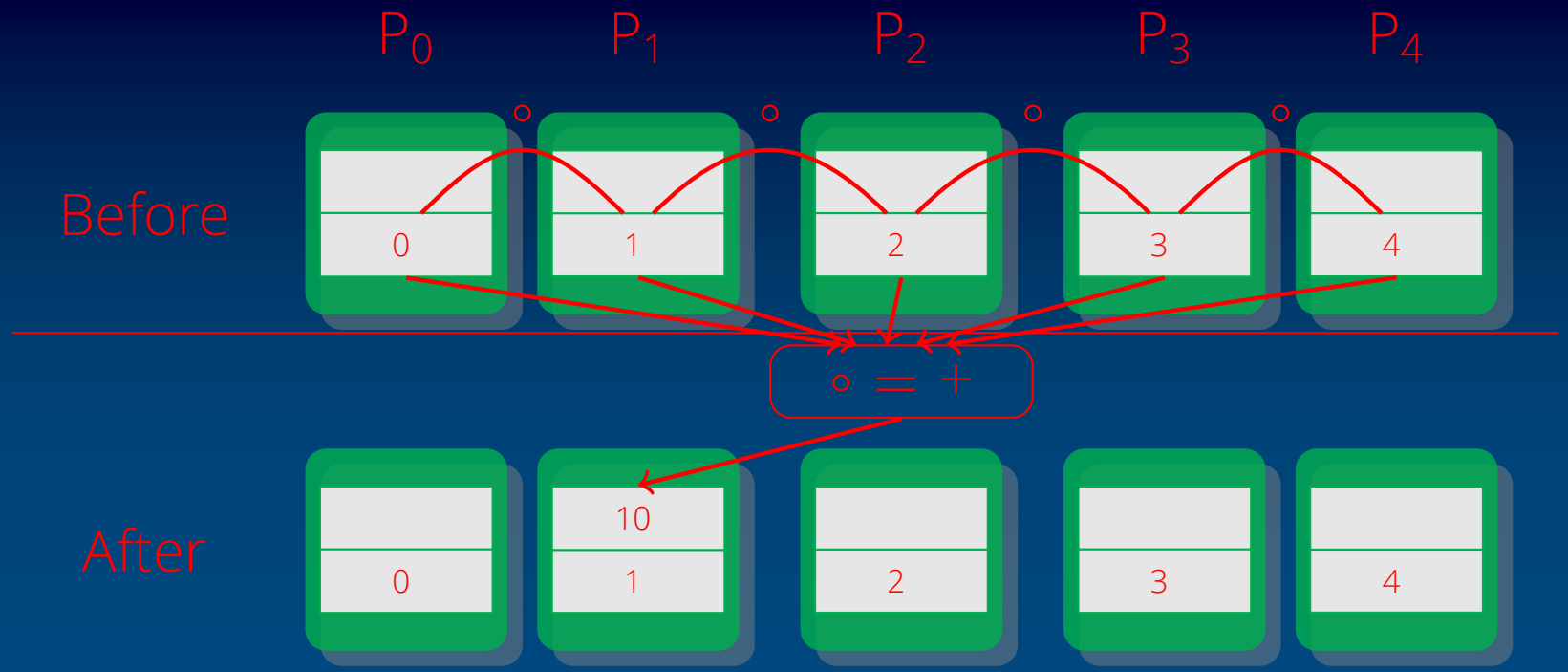
ICHEC
Irish Centre for High-End Computing

reduce



ICHEC
Irish Centre for High-End Computing

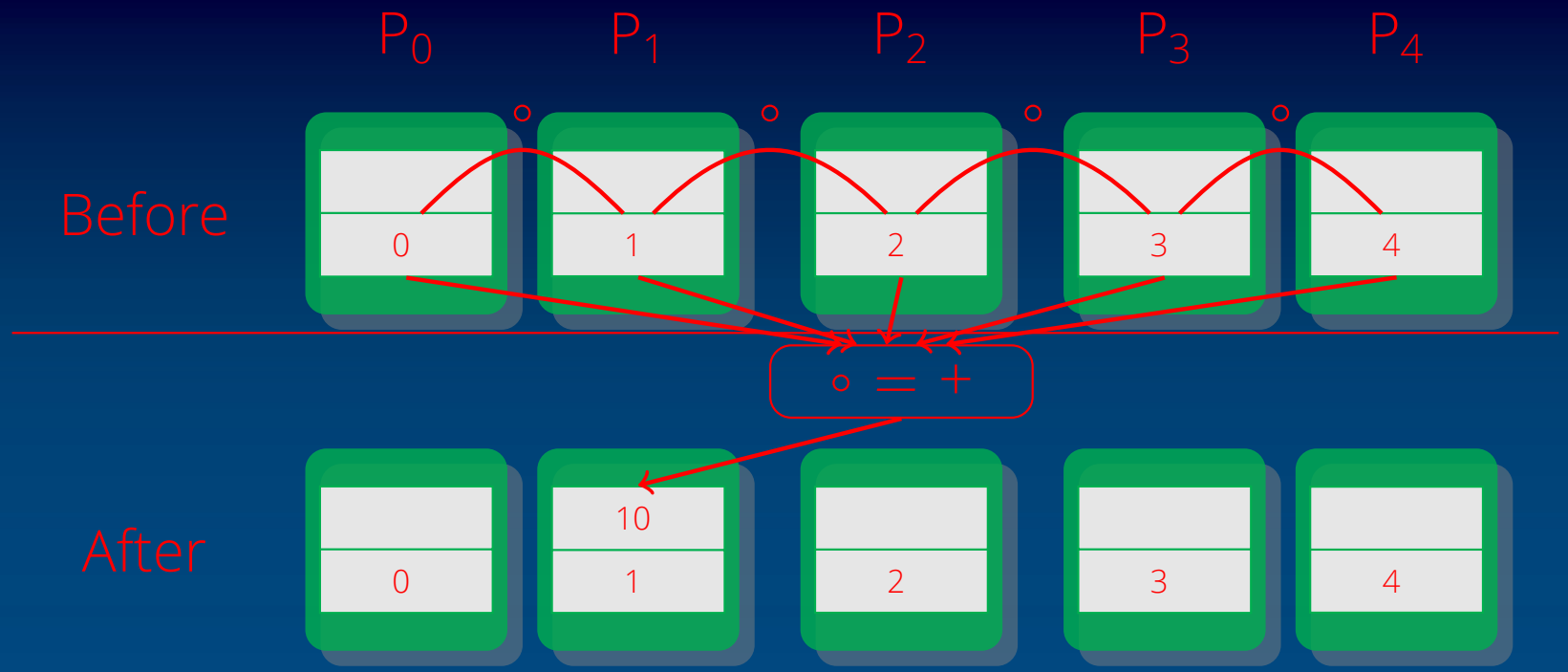
reduce




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

ICHEC
Irish Centre for High-End Computing

reduce

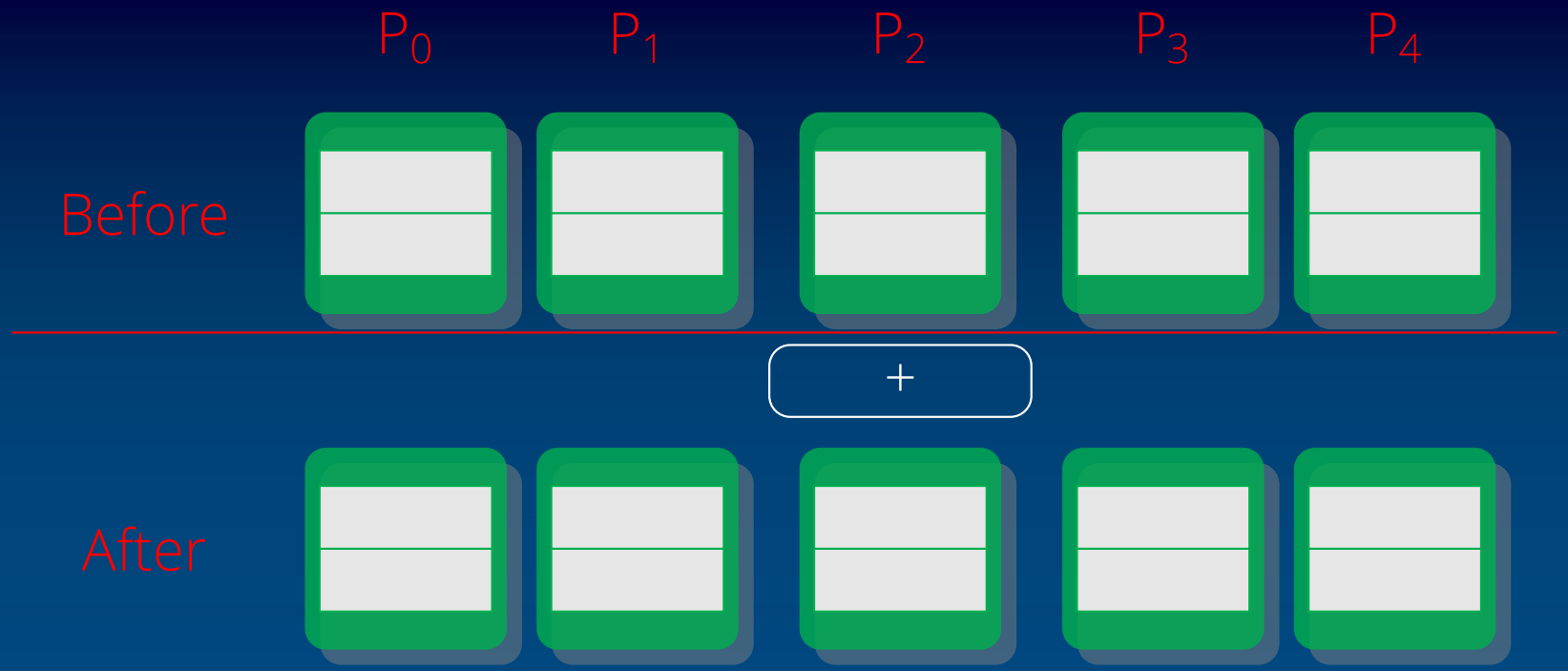



```
int MPI_Reduce(void *sendbuf, void *recvbuf, int count,
    MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
```

```
MPI_Reduce(sendbuf, recvbuf, count, datatype, op, root, comm,
    ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: count, datatype, op, root, comm, ierror
```

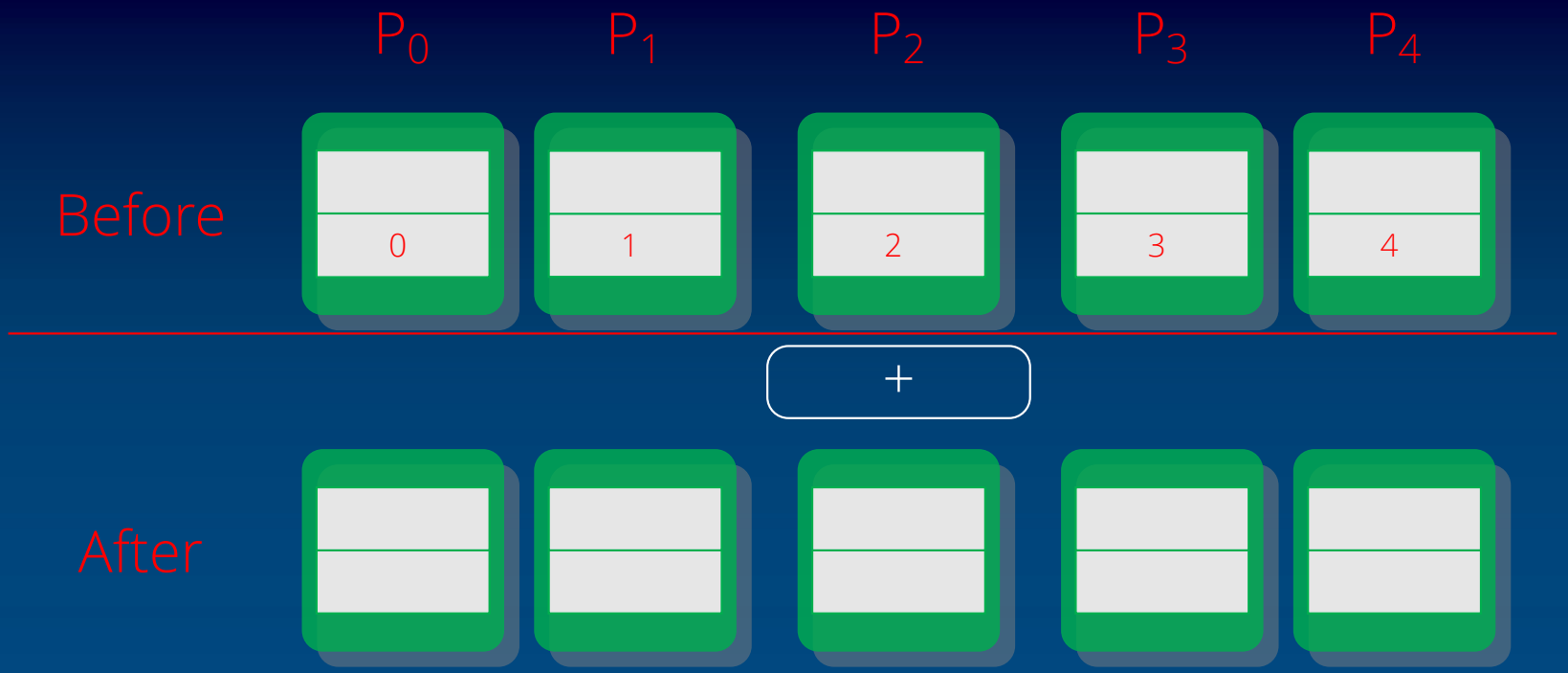
ICHEC
Irish Centre for High-End Computing

allreduce



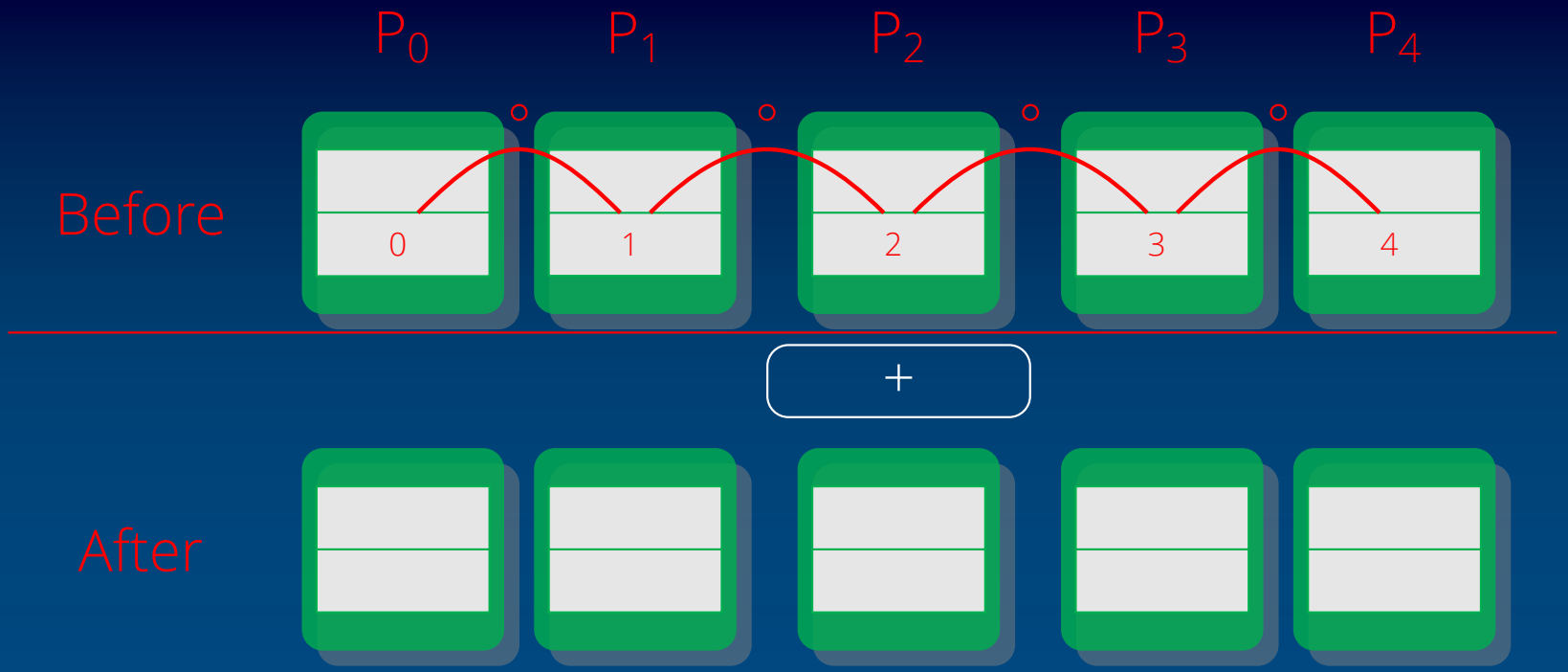
ICHEC
Irish Centre for High-End Computing

allreduce



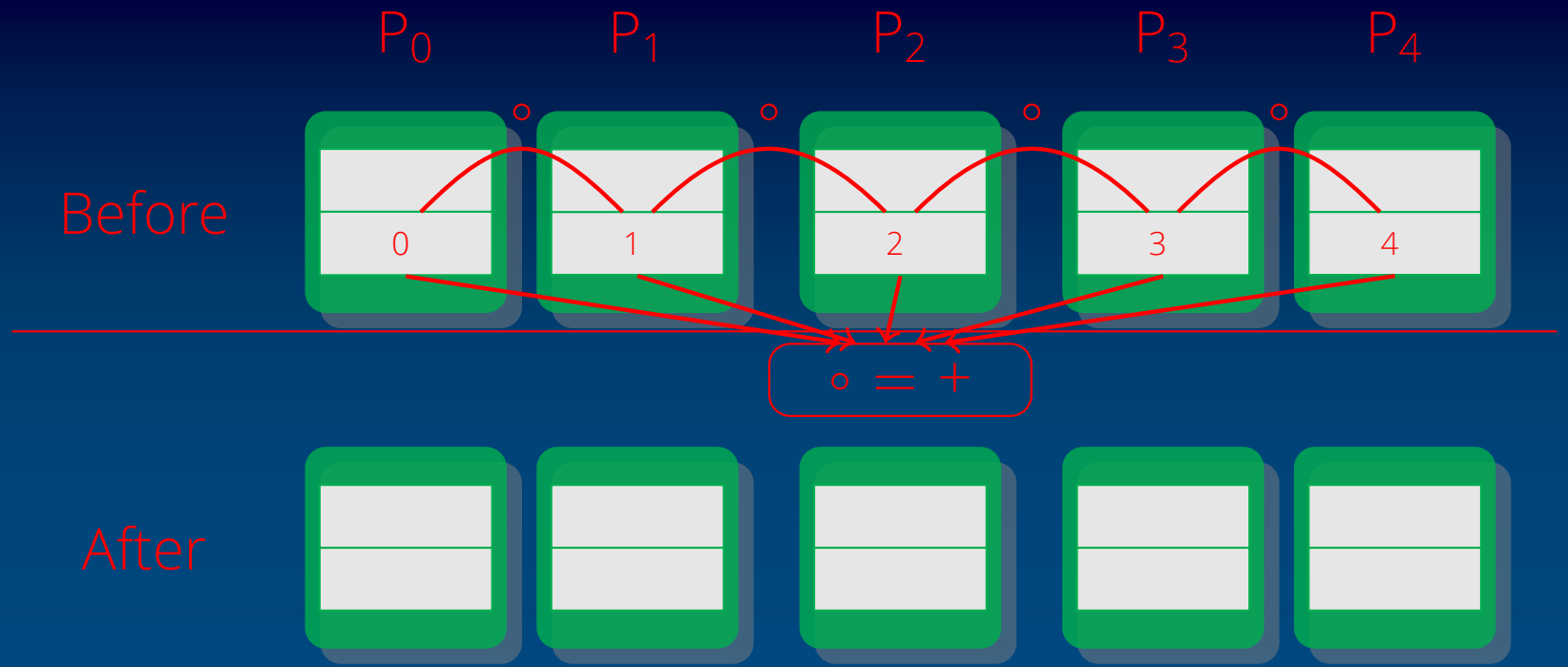
ICHEC
Irish Centre for High-End Computing

allreduce



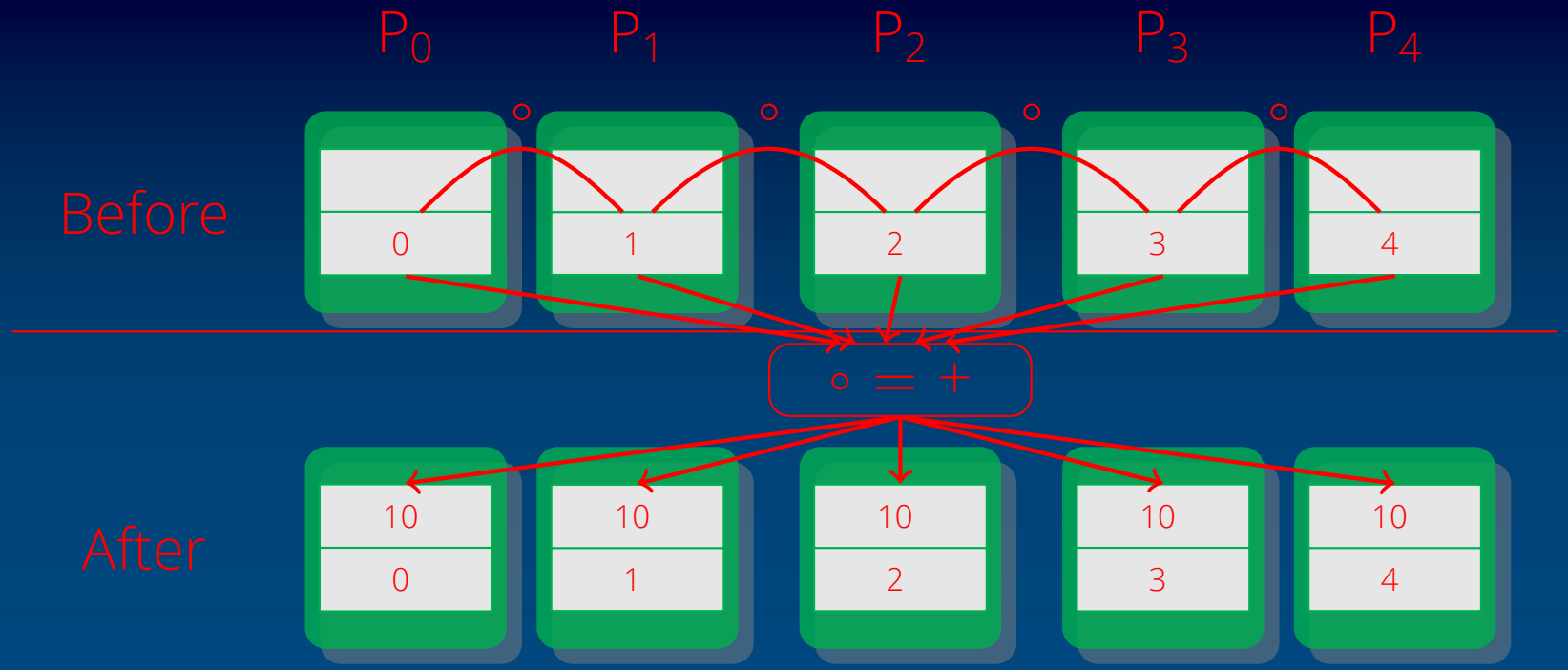
ICHEC
Irish Centre for High-End Computing

allreduce



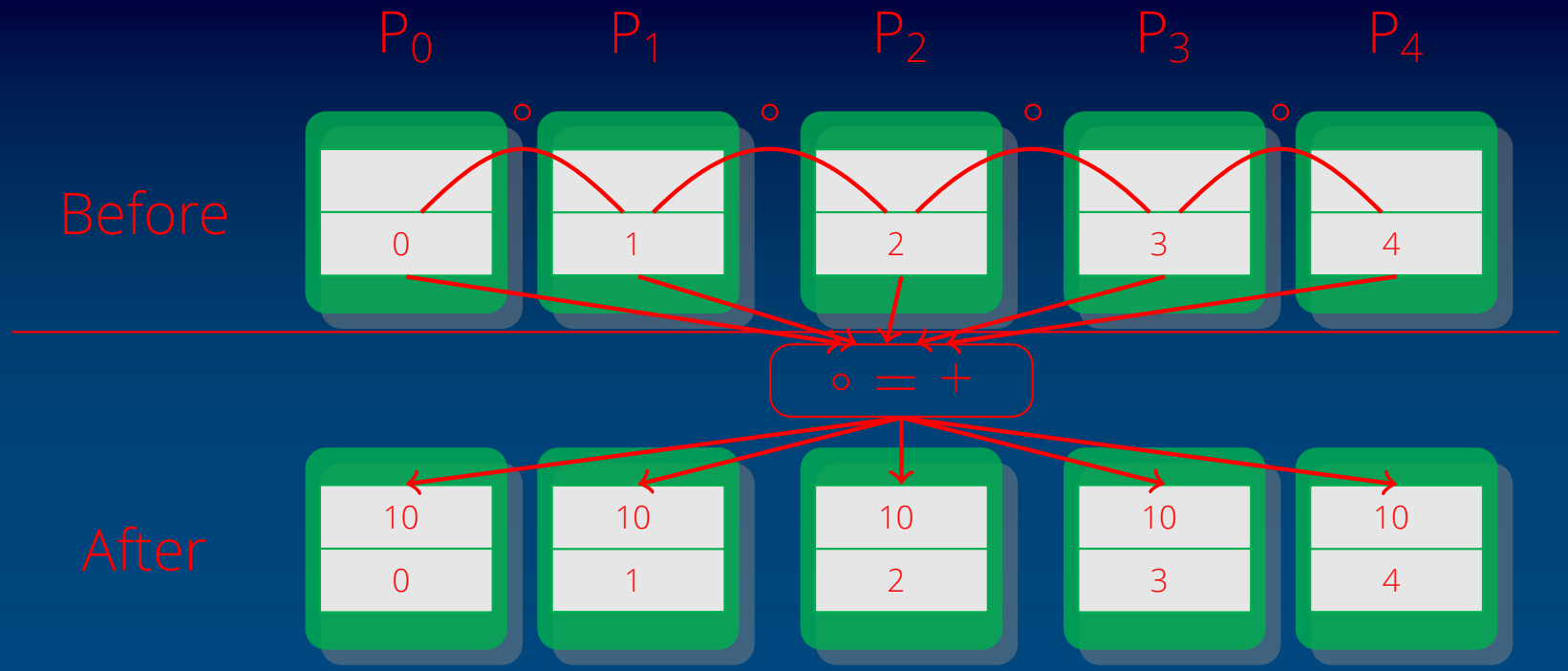
ICHEC
Irish Centre for High-End Computing

allreduce



ICHEC
Irish Centre for High-End Computing

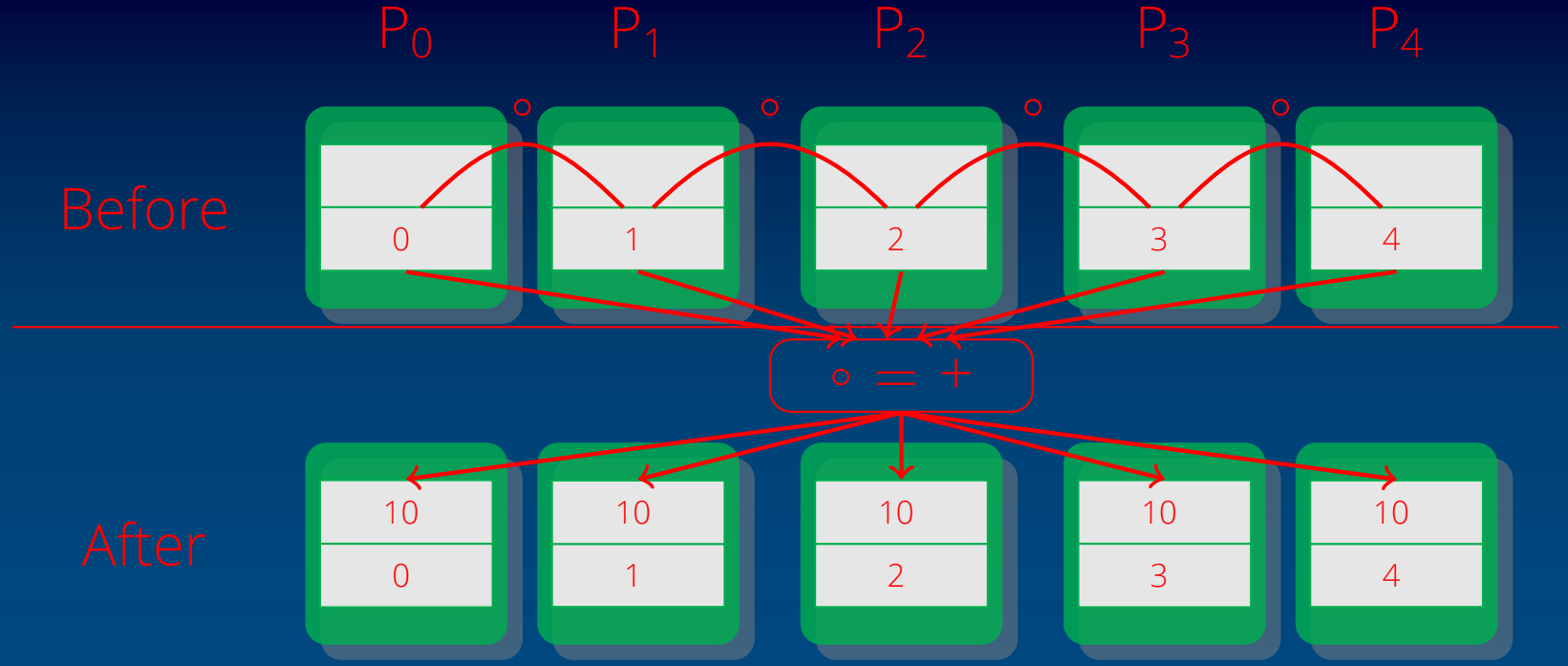
allreduce




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

ICHEC
Irish Centre for High-End Computing

allreduce

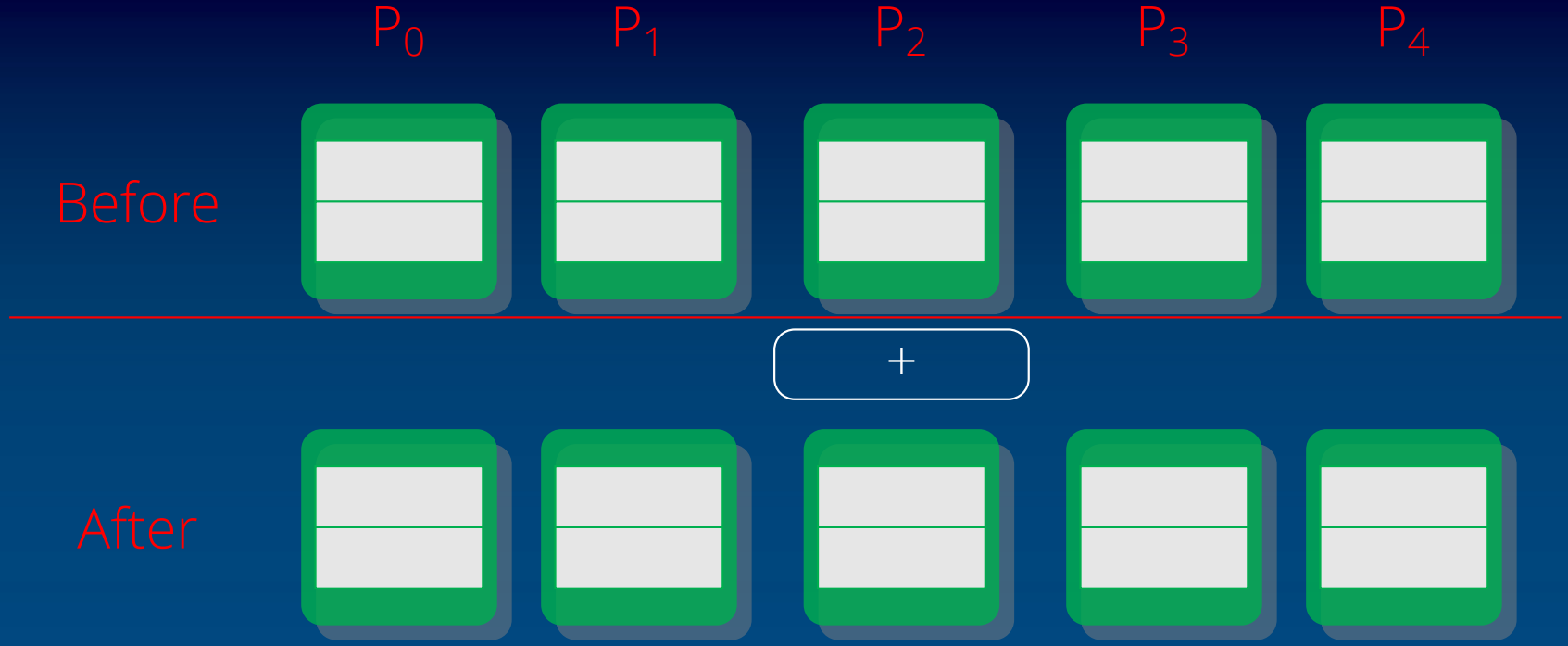



```
int MPI_Allreduce(void *sendbuf, void *recvbuf, int count,
    MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

```
MPI_Allreduce(sendbuf, recvbuf, count, datatype, op, comm,
    ierror)
    <type>  :: sendbuf(*), recvbuf(*)
    integer :: count, datatype, op, comm, ierror
```

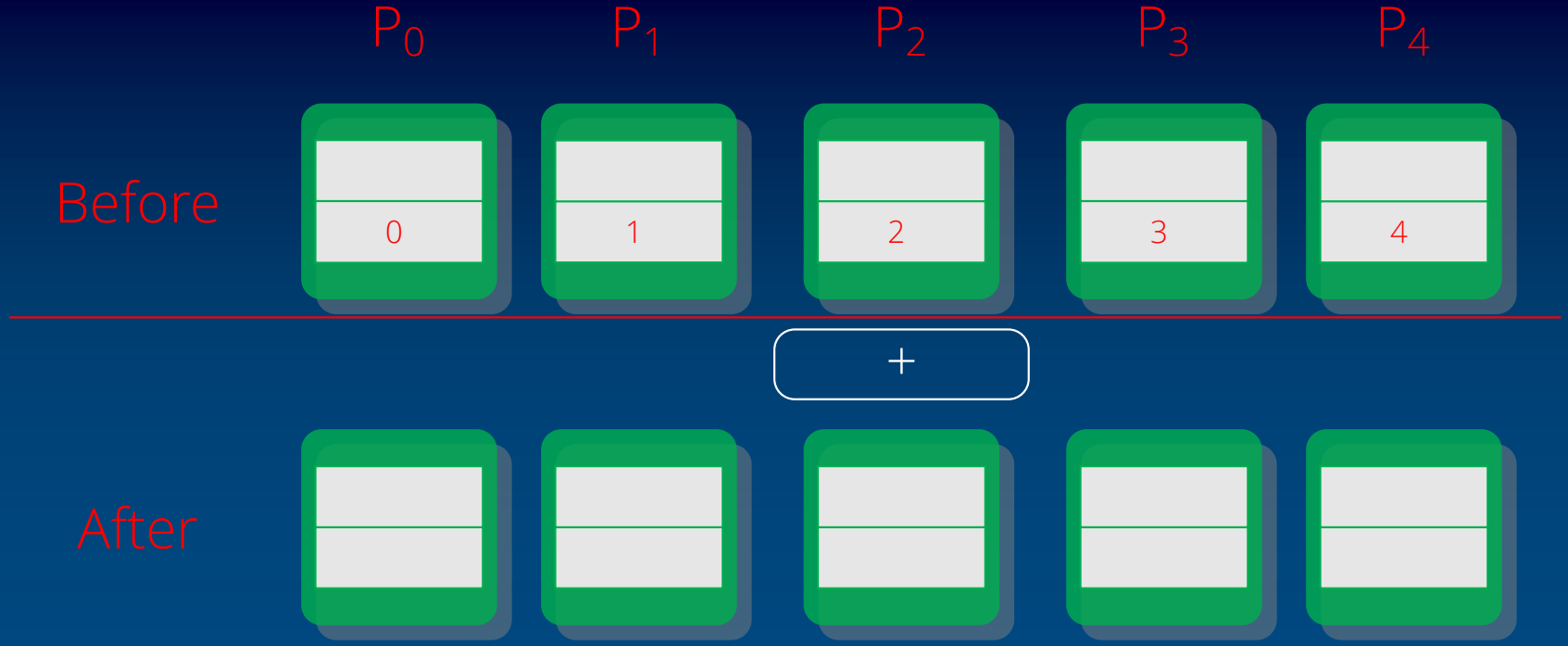
ICHEC
Irish Centre for High-End Computing

scan



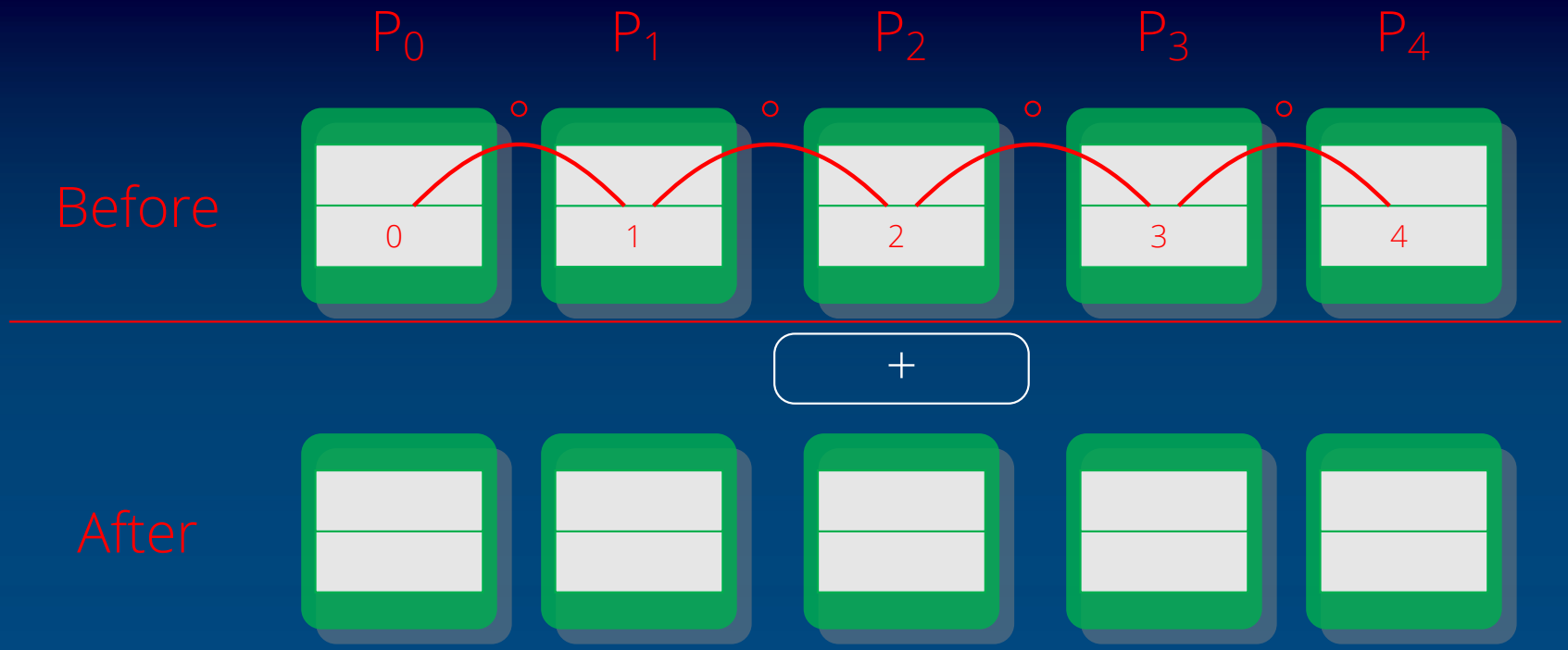
ICHEC
Irish Centre for High-End Computing

scan



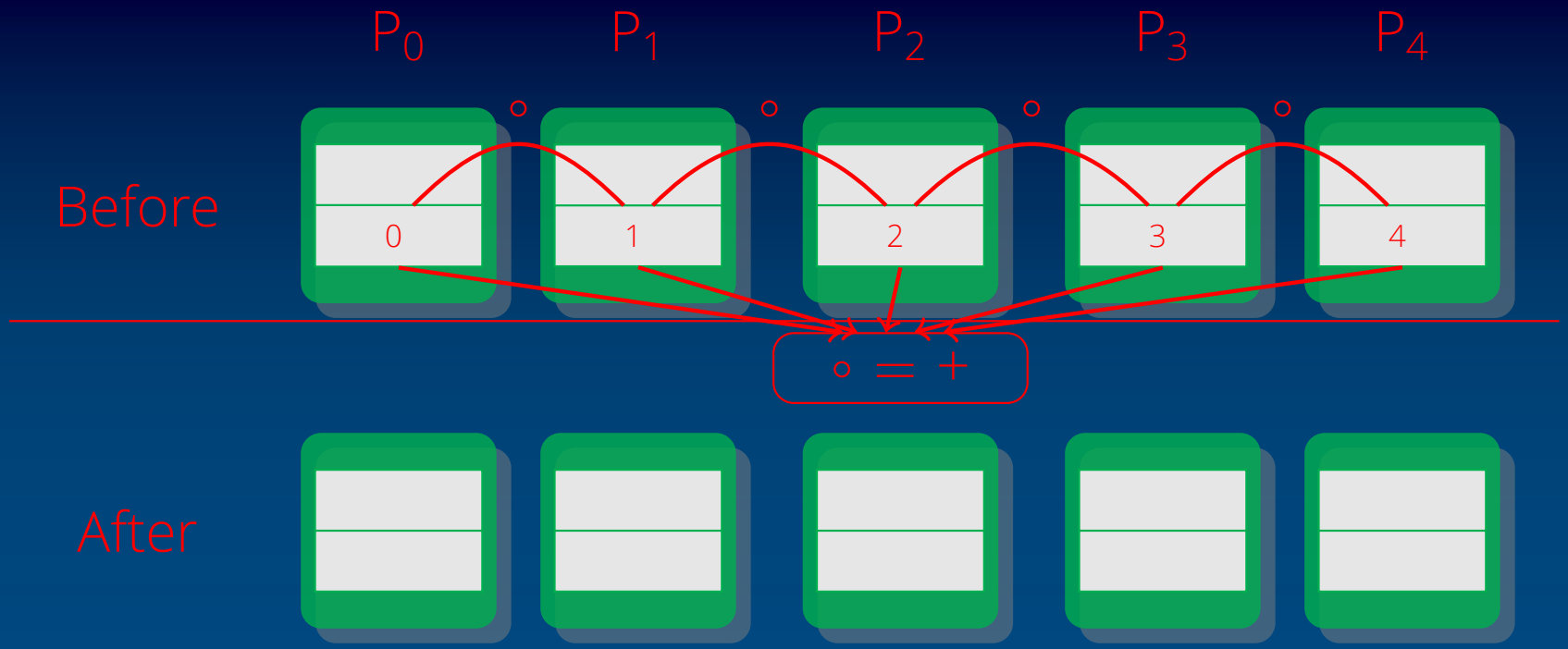
ICHEC
Irish Centre for High-End Computing

scan



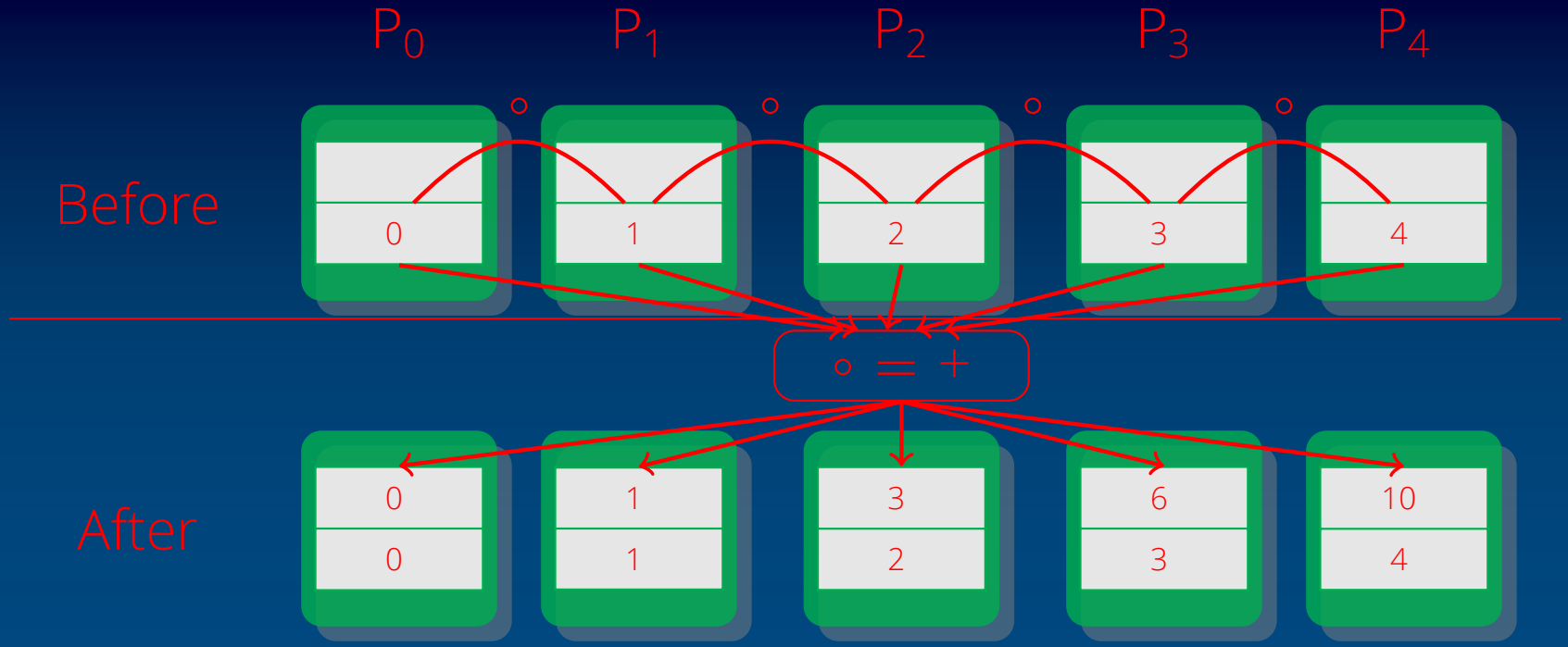
ICHEC
Irish Centre for High-End Computing

scan



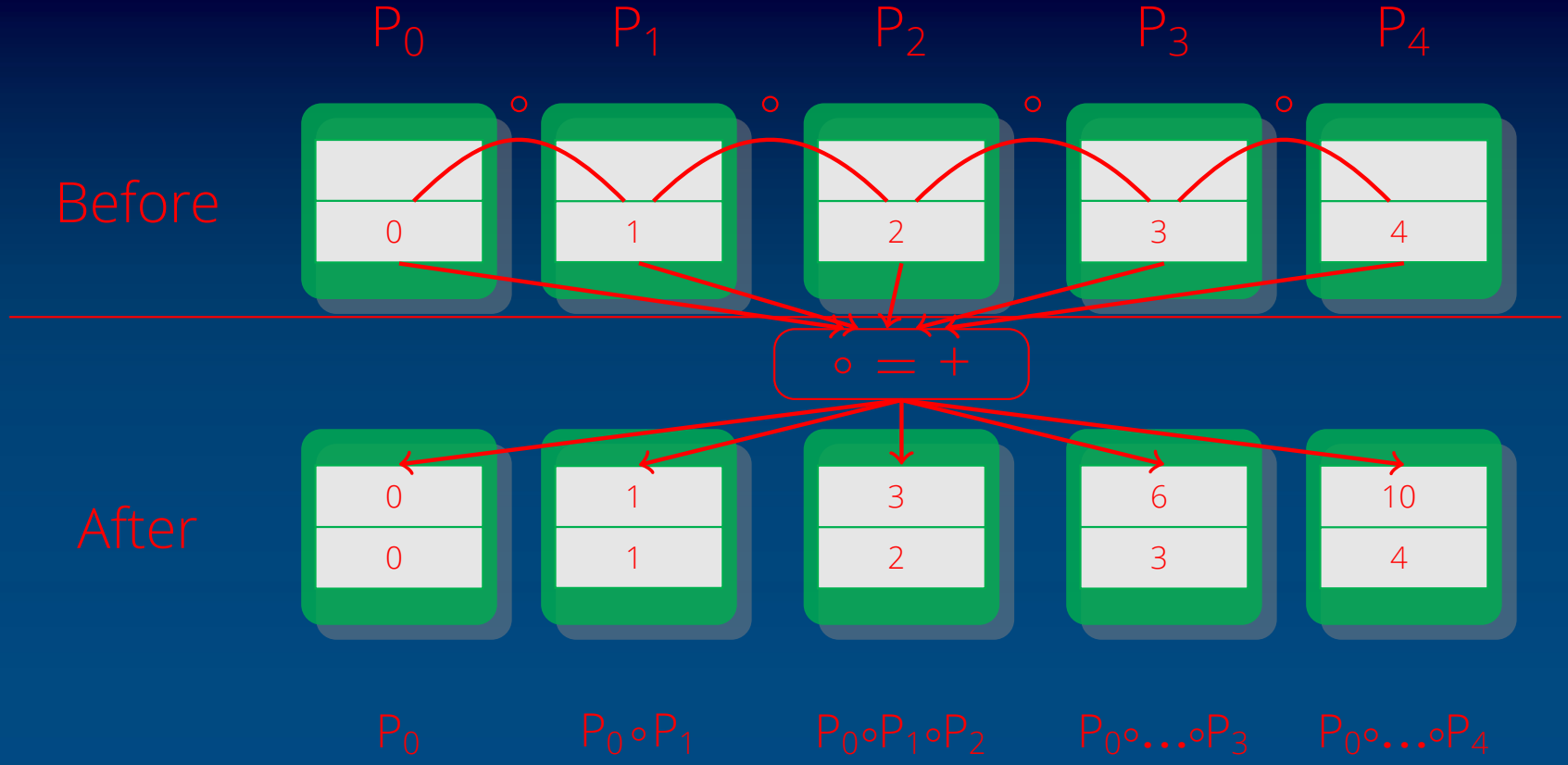
ICHEC
Irish Centre for High-End Computing

scan



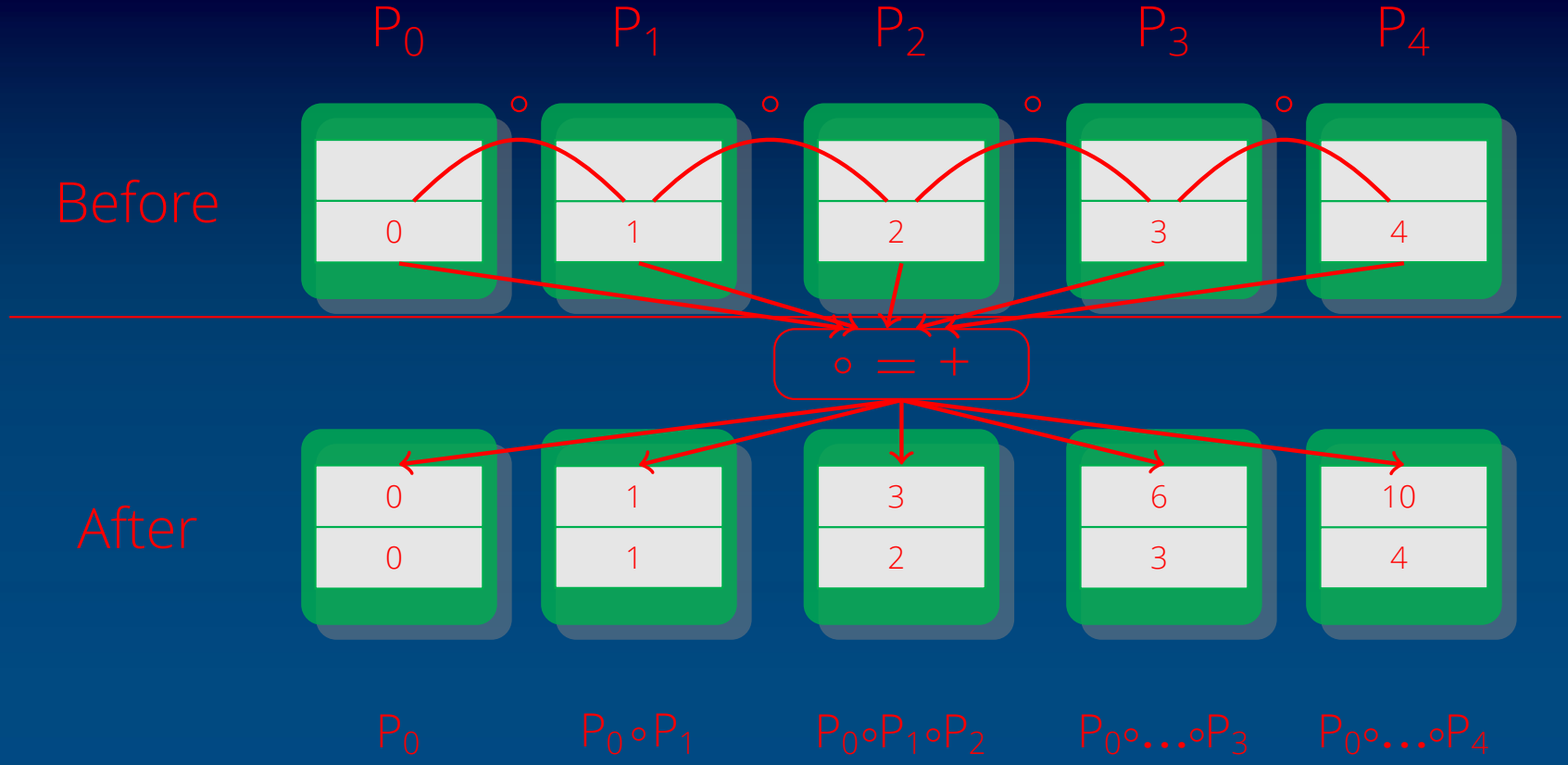
ICHEC
Irish Centre for High-End Computing

scan



ICHEC
Irish Centre for High-End Computing

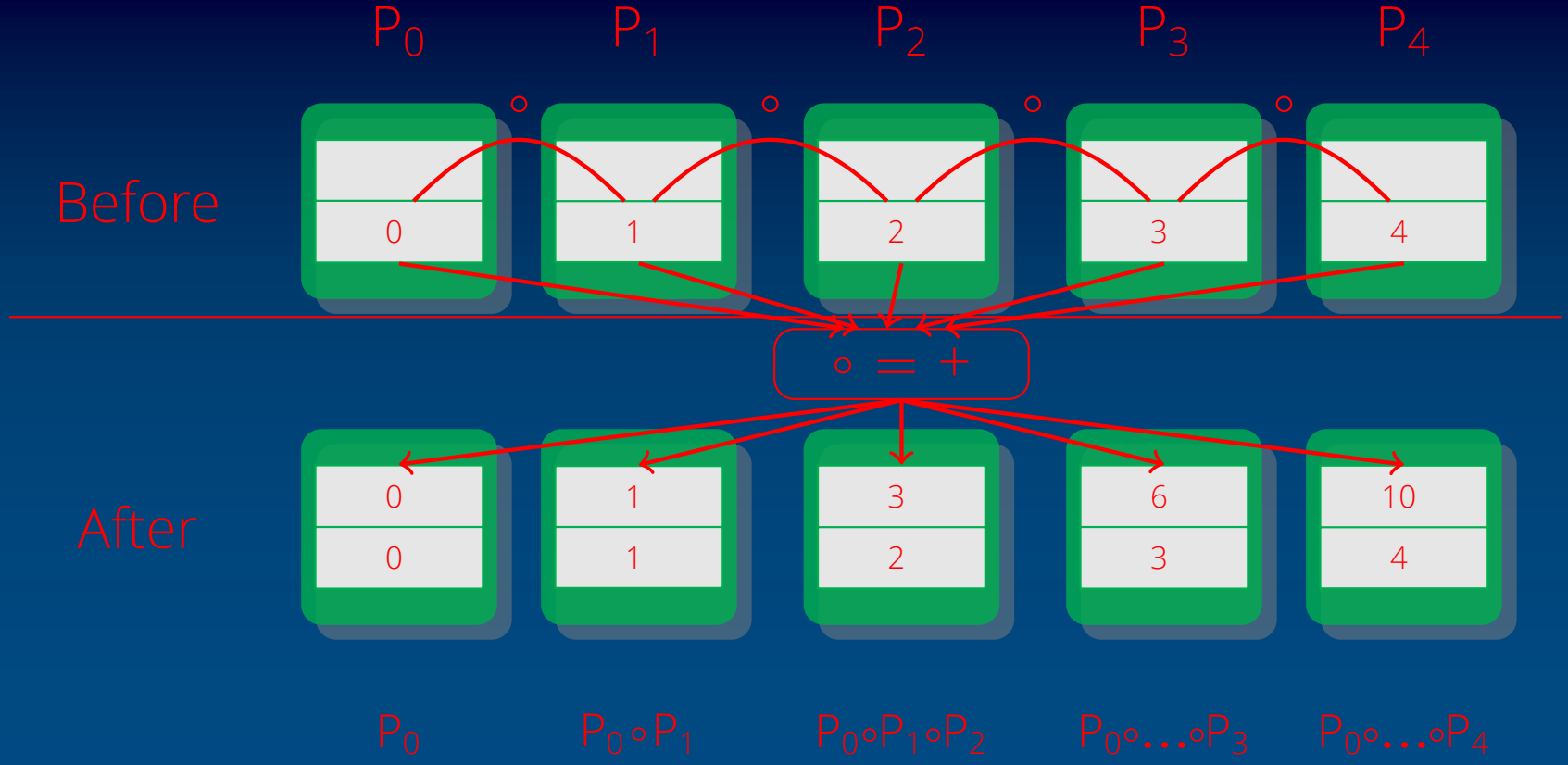
scan




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

ICHEC
Irish Centre for High-End Computing

scan

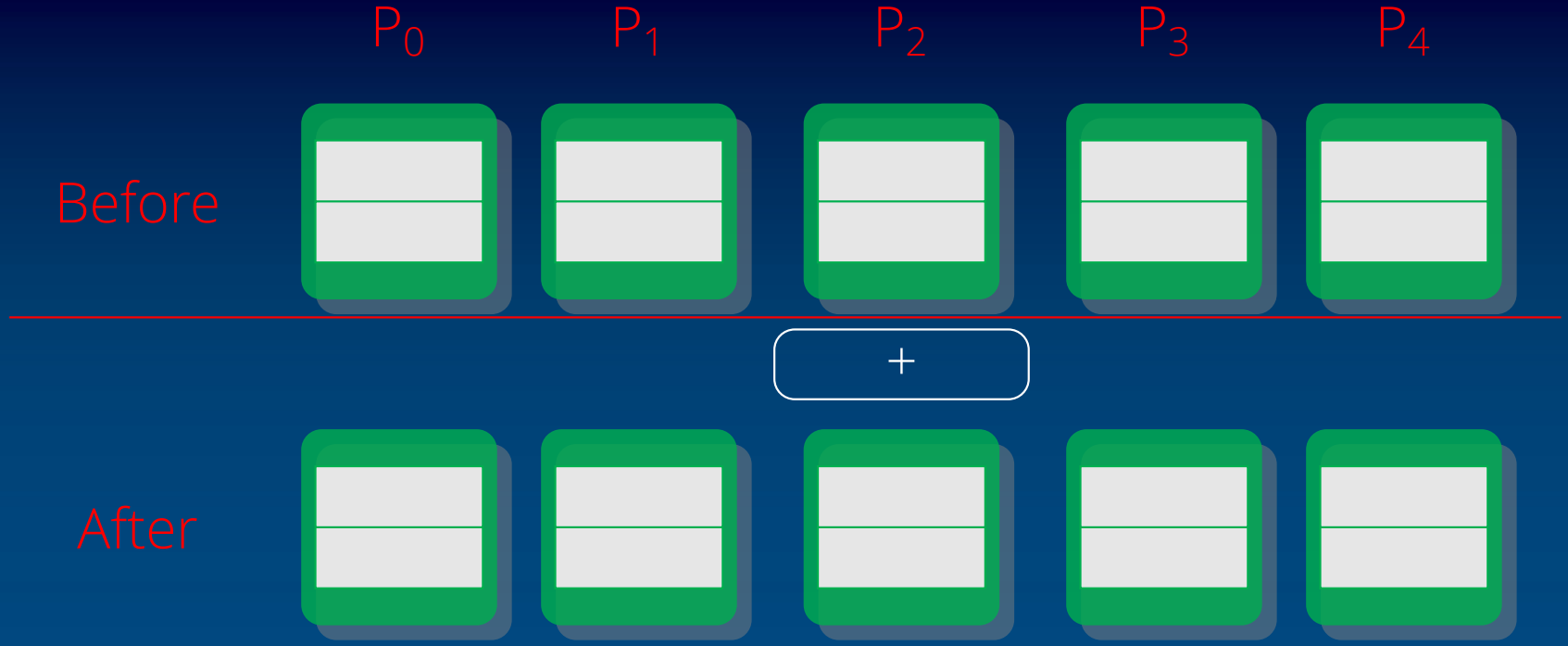



```
int MPI_Scan(void *sendbuf, void *recvbuf, int count,
    MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)

MPI_Scan(sendbuf, recvbuf, count, datatype, op, comm, ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: count, datatype, op, comm, ierror
```

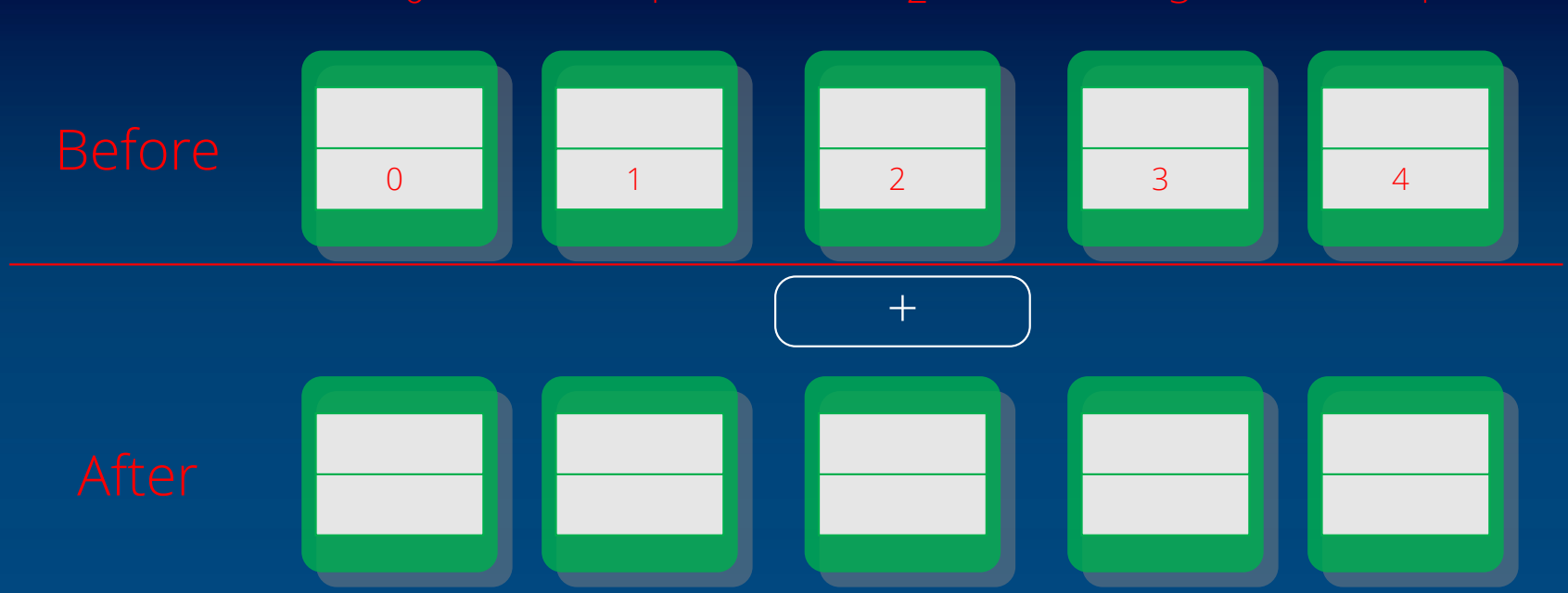
ICHEC
Irish Centre for High-End Computing

ExScan



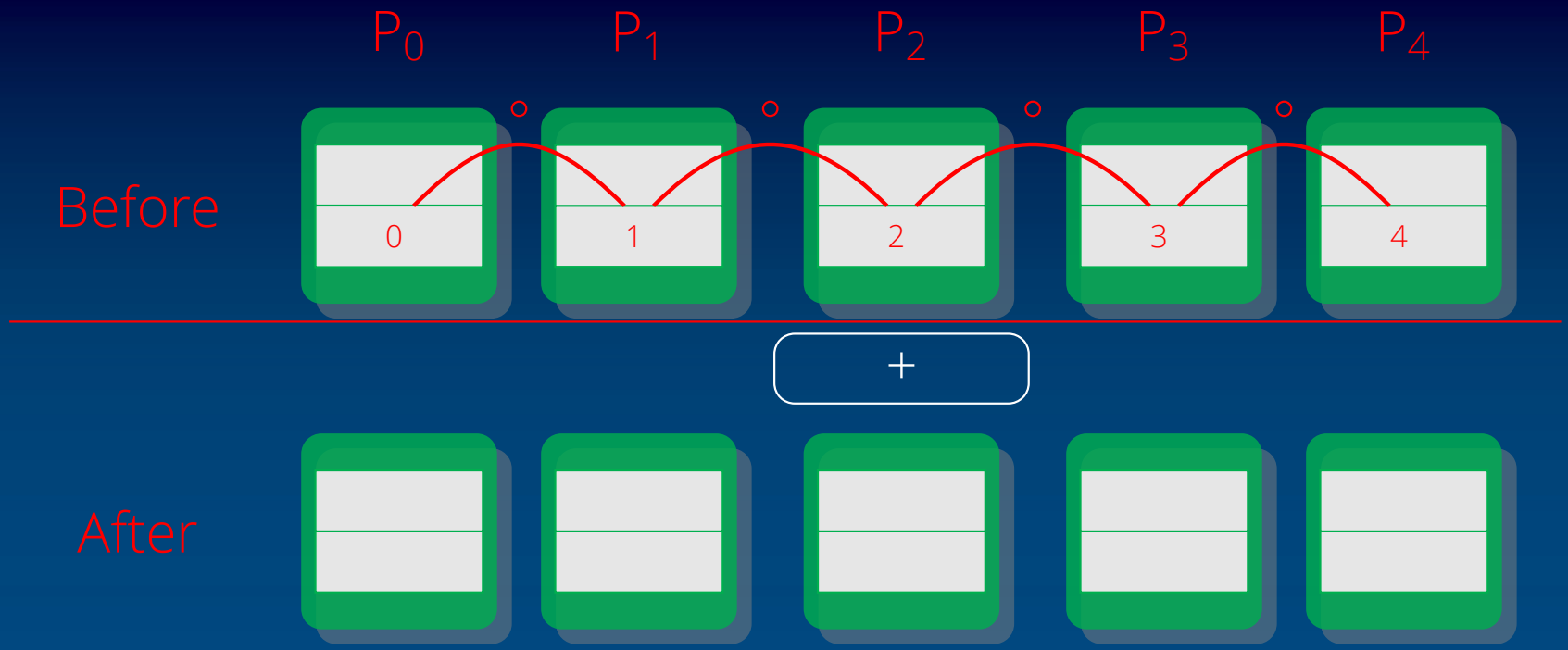
ICHEC
Irish Centre for High-End Computing

ExScan



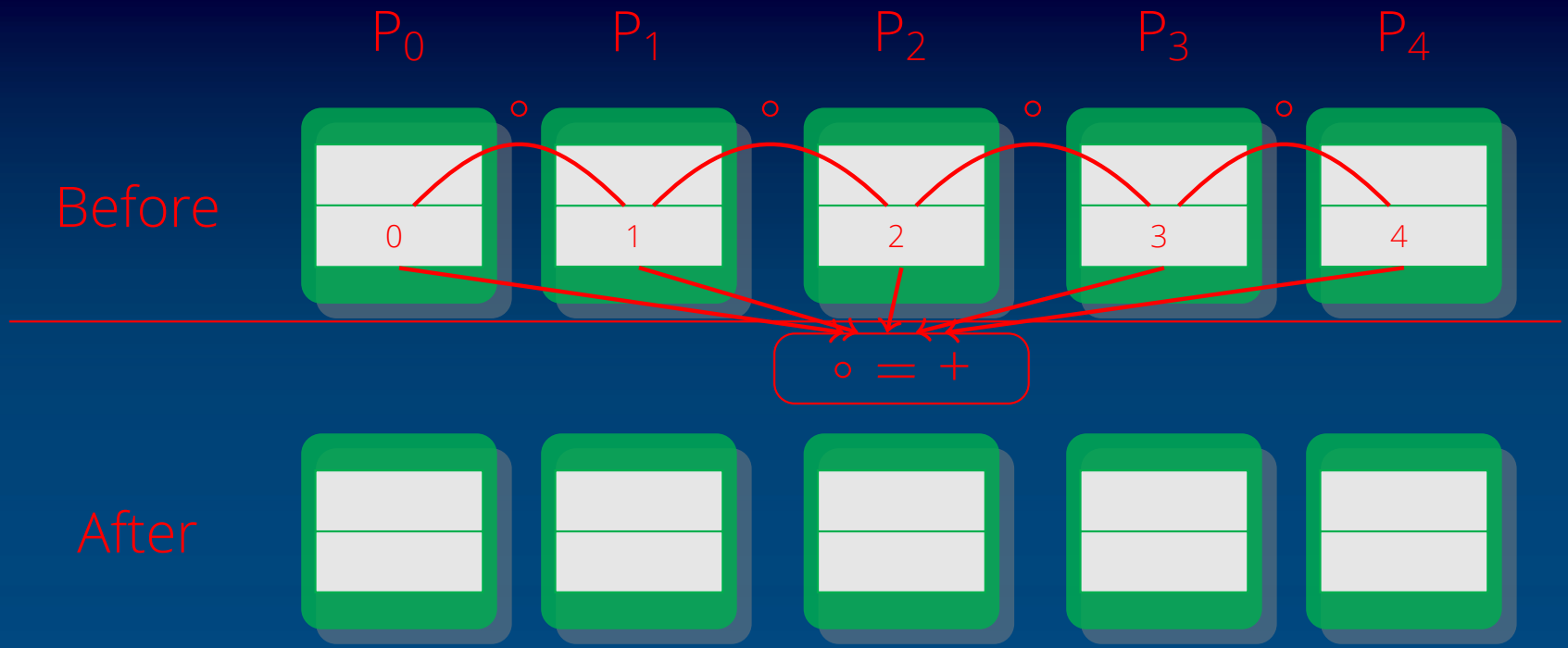
ICHEC
Irish Centre for High-End Computing

ExScan



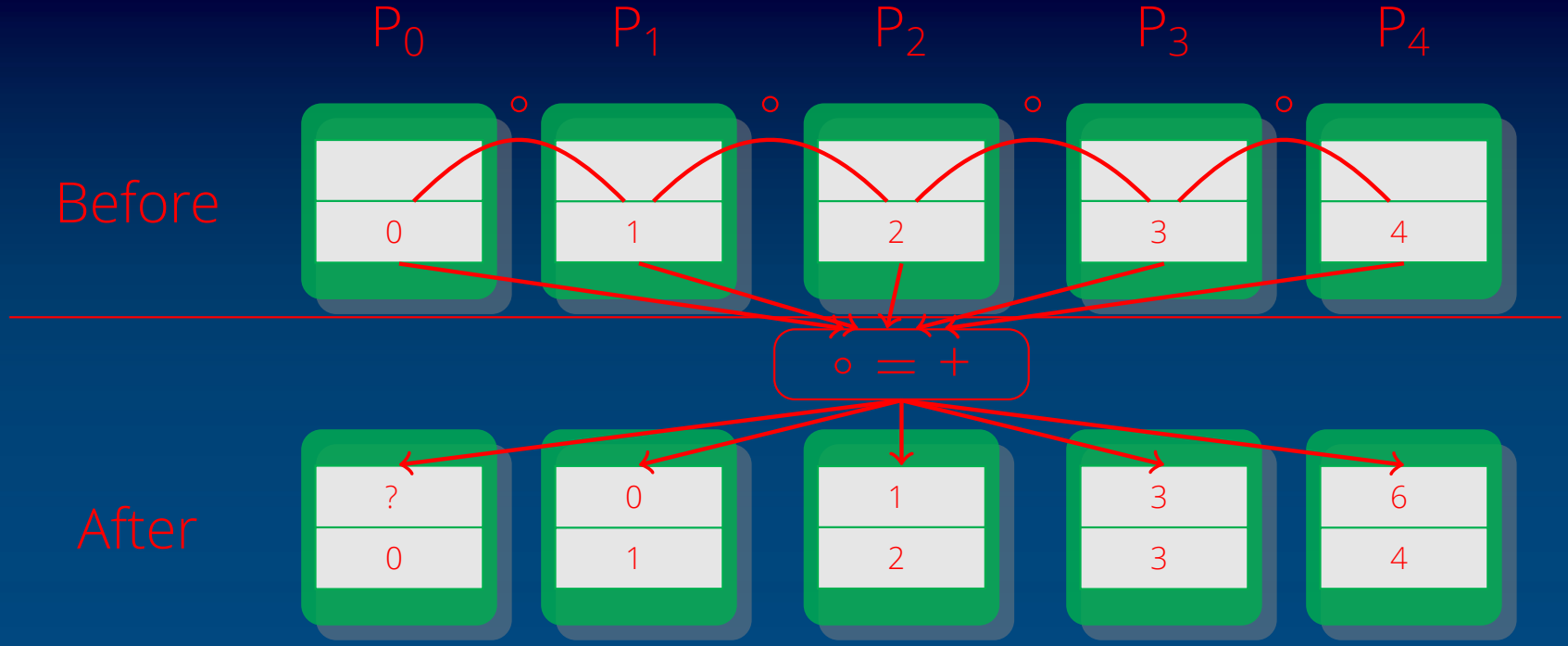
ICHEC
Irish Centre for High-End Computing

ExScan



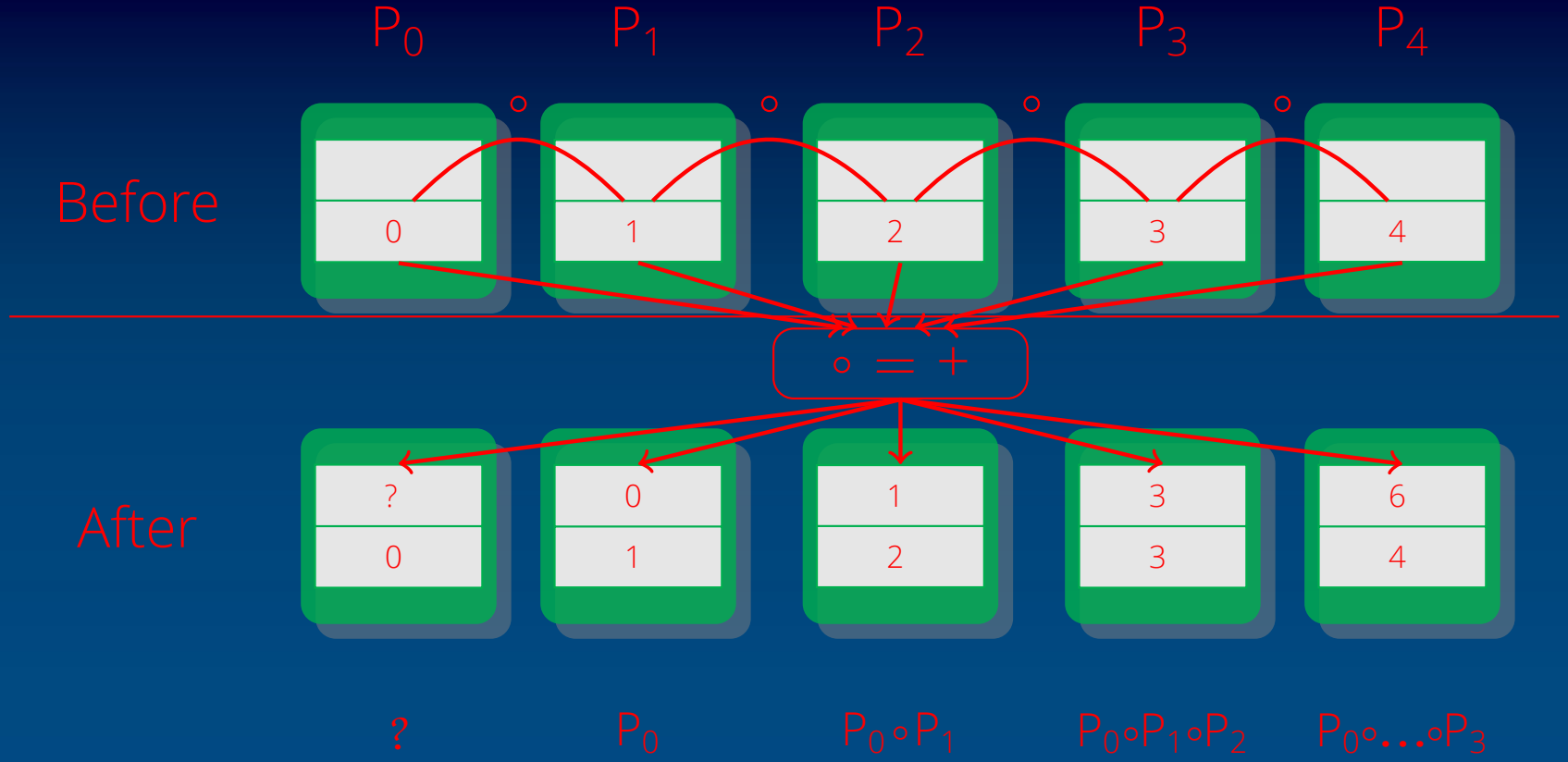
ICHEC
Irish Centre for High-End Computing

ExScan



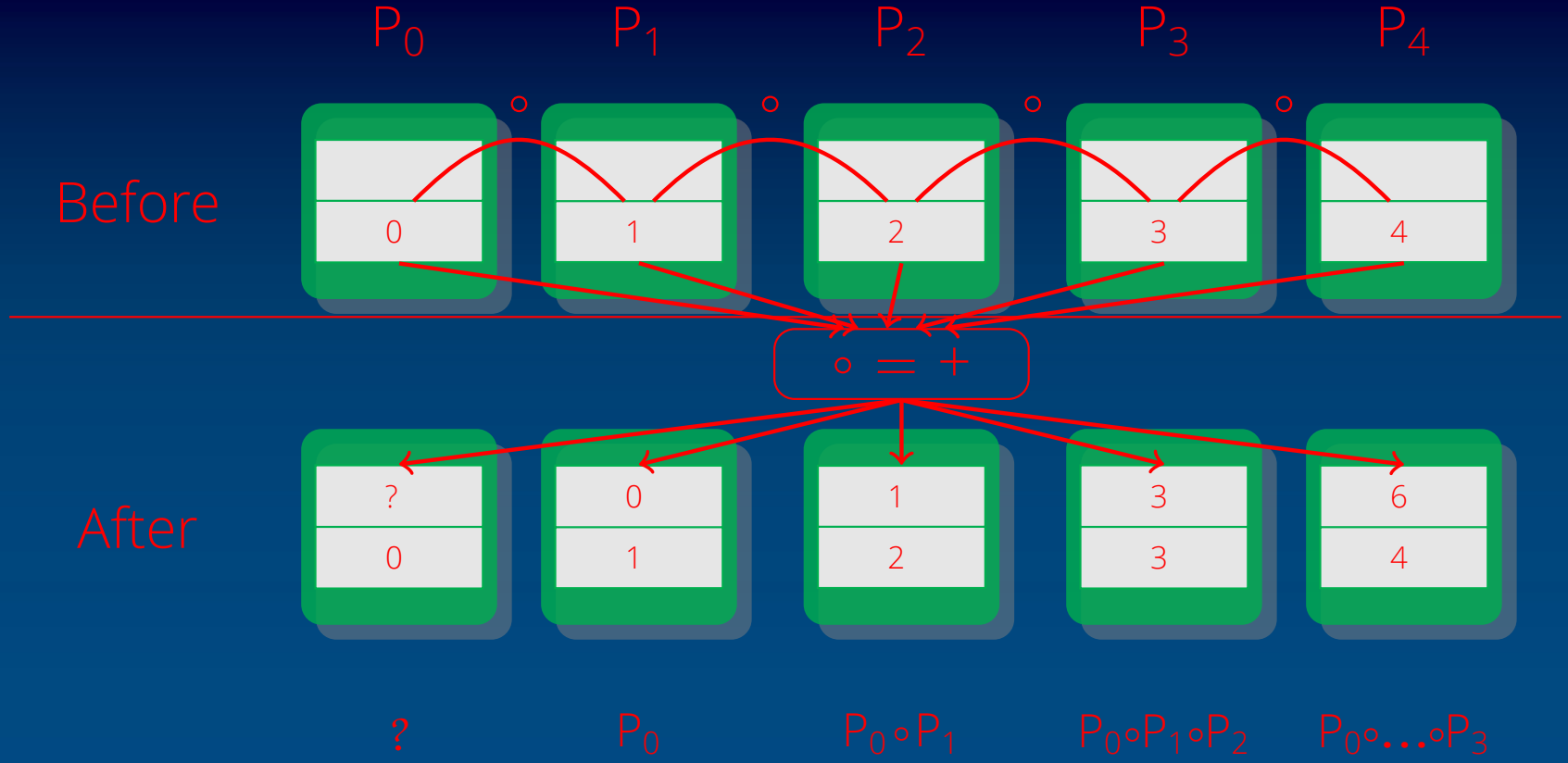
ICHEC
Irish Centre for High-End Computing

ExScan



ICHEC
Irish Centre for High-End Computing

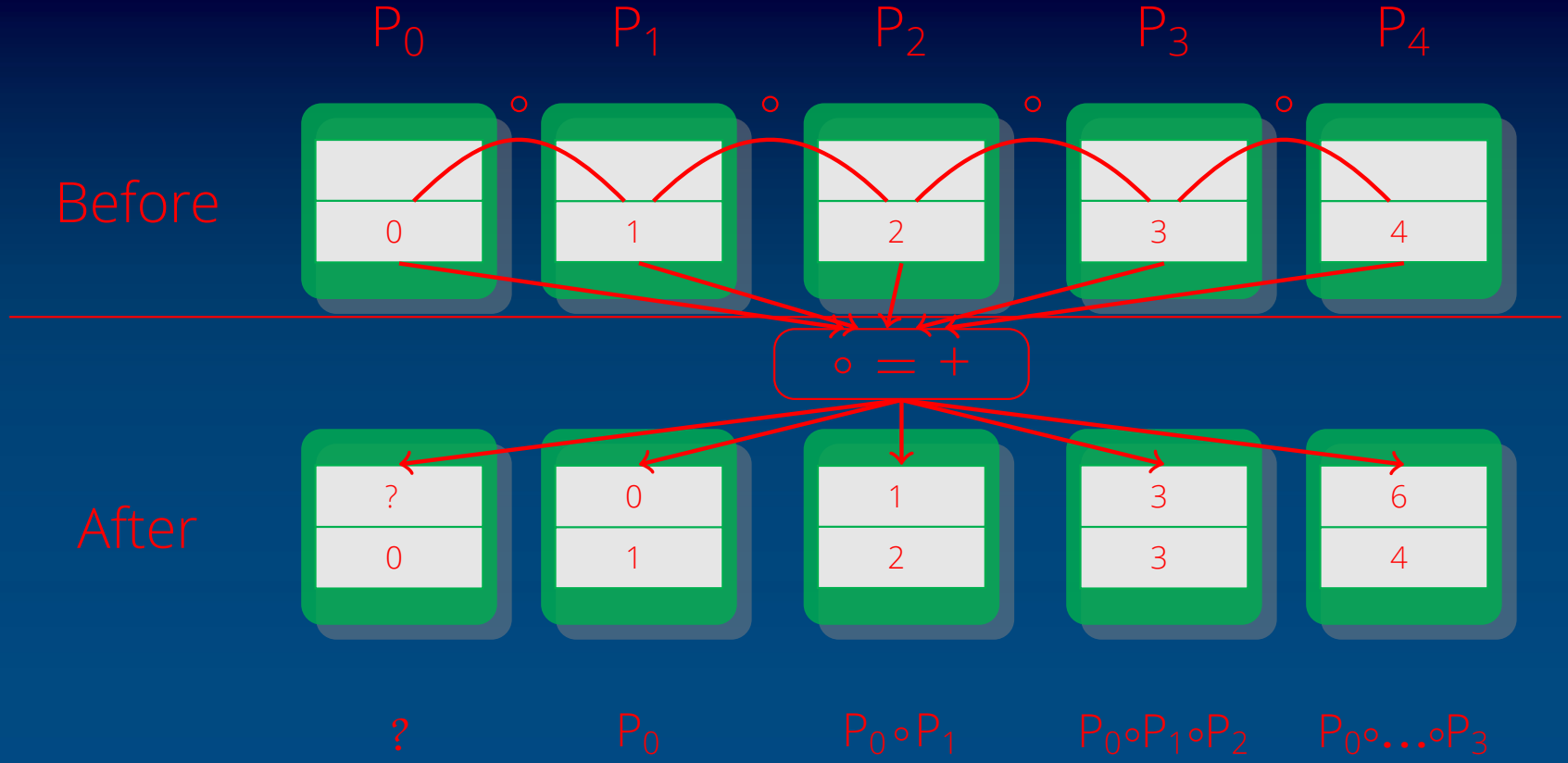
ExScan




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

ICHEC
Irish Centre for High-End Computing

ExScan




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

MPI_Exscan(sendbuf, recvbuf, count, datatype, op, comm, ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: count, datatype, op, comm, ierror
```

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

- associative

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

- associative
- operates on single variables or vector elements

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

- associative
- operates on single variables or vector elements
- pre-defined or user-defined

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

- associative
- operates on single variables or vector elements
- pre-defined or user-defined
- results may differ when reduction is applied to floats.

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

- associative
- operates on single variables or vector elements
- pre-defined or user-defined
- results may differ when reduction is applied to floats.

ICHEC
Irish Centre for High-End Computing

MPI reduction operations

- associative
- operates on single variables or vector elements
- pre-defined or user-defined
- results may differ when reduction is applied to floats.

Handle Name	Operation
MPI_MAX	maximum
MPI_MIN	minimum
MPI_SUM	sum
MPI_PROD	product
MPI_LAND	logical and
MPI_BAND	bit-wise and
MPI_LOR	logical or
MPI_BOR	bit-wise or
MPI_LXOR	logical xor
MPI_BXOR	bit-wise xor
MPI_MAXLOC	max value and location
MPI_MINLOC	min value and location

ICHEC
Irish Centre for High-End Computing

user-defined MPI operation

ICHEC
Irish Centre for High-End Computing

user-defined MPI operation

associative

ICHEC
Irish Centre for High-End Computing

user-defined MPI operation

- associative
- acts on vectors

ICHEC
Irish Centre for High-End Computing

user-defined MPI operation

- associative
- acts on vectors
- prototype of the function specified in standard

ICHEC
Irish Centre for High-End Computing

user-defined MPI operation

- associative
- acts on vectors
- prototype of the function specified in standard

```
void MPI_User_function(void *invec, void *inoutvec, int *len,
    MPI_Datatype *datatype);
```

```
subroutine user_function(invec, inoutvec, length, MPItype)
<type>  :: invec(length), inoutvec(length)
integer :: length, MPItype
```

ICHEC
Irish Centre for High-End Computing

user-defined MPI operation

- associative
- acts on vectors
- prototype of the function specified in standard

```
void MPI_User_function(void *invec, void *inoutvec, int *len,
    MPI_Datatype *datatype);
```

```
subroutine user_function(invec, inoutvec, length, MPItype)
<type>  :: invec(length), inoutvec(length)
integer :: length, MPItype
```

```
int MPI_Op_create(MPI_User_function *function, int commute, MPI_Op *op)
int MPI_op_free(MPI_Op *op)
```

```
MPI_Op_create(function, commute, op, ierror)
external function
logical :: commute
integer :: op, ierror
MPI_Op_free(op, ierror)
integer :: op, ierror
```

ICHEC
Irish Centre for High-End Computing

Generalized collectives

ICHEC
Irish Centre for High-End Computing

Generalized collectives

- MPI_Allgatherv
- MPI_Alltoallv
- MPI_Alltoallw
- MPI_Gatherv
- MPI_Scatterv
- MPI_Reduce_scatter

ICHEC
Irish Centre for High-End Computing

Generalized collectives

- MPI_Allgatherv
- MPI_Alltoallv
- MPI_Alltoallw
- MPI_Gatherv
- MPI_Scatterv
- MPI_Reduce_scatter

```
int MPI_Alltoallv(void *sendbuf, int *sendcounts, int *sdispls,
      MPI_Datatype sendtype, void *recvbuf, int *recvcounts, int
      *rdispls, MPI_Datatype recvtype, MPI_Comm comm)
```

ICHEC
Irish Centre for High-End Computing

Generalized collectives

- MPI_Allgatherv
- MPI_Alltoallv
- MPI_Alltoallw
- MPI_Gatherv
- MPI_Scatterv
- MPI_Reduce_scatter

```
int MPI_Alltoallv(void *sendbuf, int *sendcounts, int *sdispls,
      MPI_Datatype sendtype, void *recvbuf, int *recvcounts, int
      *rdispls, MPI_Datatype recvtype, MPI_Comm comm)
```

```
MPI_Alltoallv(sendbuf, sendcounts, sdispls, sendtype, recvbuf,
    recvcounts, rdispls, recvtype, comm, ierror)
    <type>  :: sendbuf(:), recvbuf(:)
    integer :: sendcounts(:), sdispls(:), sendtype
    integer :: recvcounts(:), rdispls(:), recvtype
    integer :: comm, ierror
```

ICHEC
Irish Centre for High-End Computing

barrier

ICHEC
Irish Centre for High-End Computing

barrier

ICHEC
Irish Centre for High-End Computing

barrier

ICHEC
Irish Centre for High-End Computing

barrier

ICHEC
Irish Centre for High-End Computing

barrier

ICHEC
Irish Centre for High-End Computing

barrier

ICHEC
Irish Centre for High-End Computing

barrier


```
int MPI_Barrier(MPI_Comm comm)
```

ICHEC
Irish Centre for High-End Computing

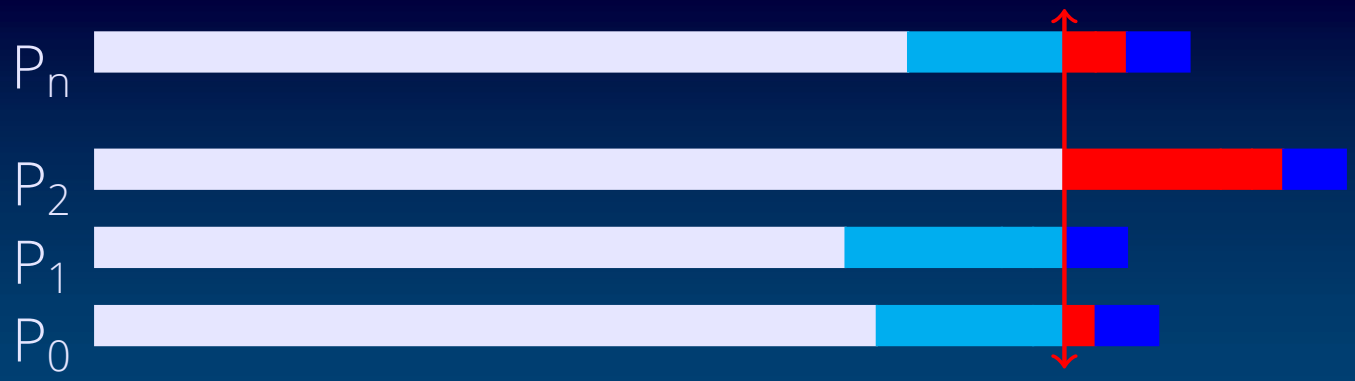
barrier


```
int MPI_Barrier(MPI_Comm comm)
```

```
MPI_Barrier(comm, ierror)
    integer :: comm, ierror
```

ICHEC
Irish Centre for High-End Computing

barrier




```
int MPI_Barrier(MPI_Comm comm)
```

```
MPI_Barrier(comm, ierror)
    integer :: comm, ierror
```

- **never use in production** useful only in debugging

ICHEC
Irish Centre for High-End Computing

barrier


```
int MPI_Barrier(MPI_Comm comm)
```

```
MPI_Barrier(comm, ierror)
    integer :: comm, ierror
```

- **never use in production** useful only in debugging
- synchronisation, in general, does not need to be done by hand, data communication does it for you