Parallel Computing for Science & Engineering Spring 2013: MPI collectives 1

Instructors:

Victor Eijkhout, Research Scientist, TACC

Kent Milfeld, Research Associate, TACC

MPI Collective Communications

- Involves a group of processes.
- Basic Routines

  Broadcast— `MPI_Bcast()`

  Reduce— `MPI_Reduce()`

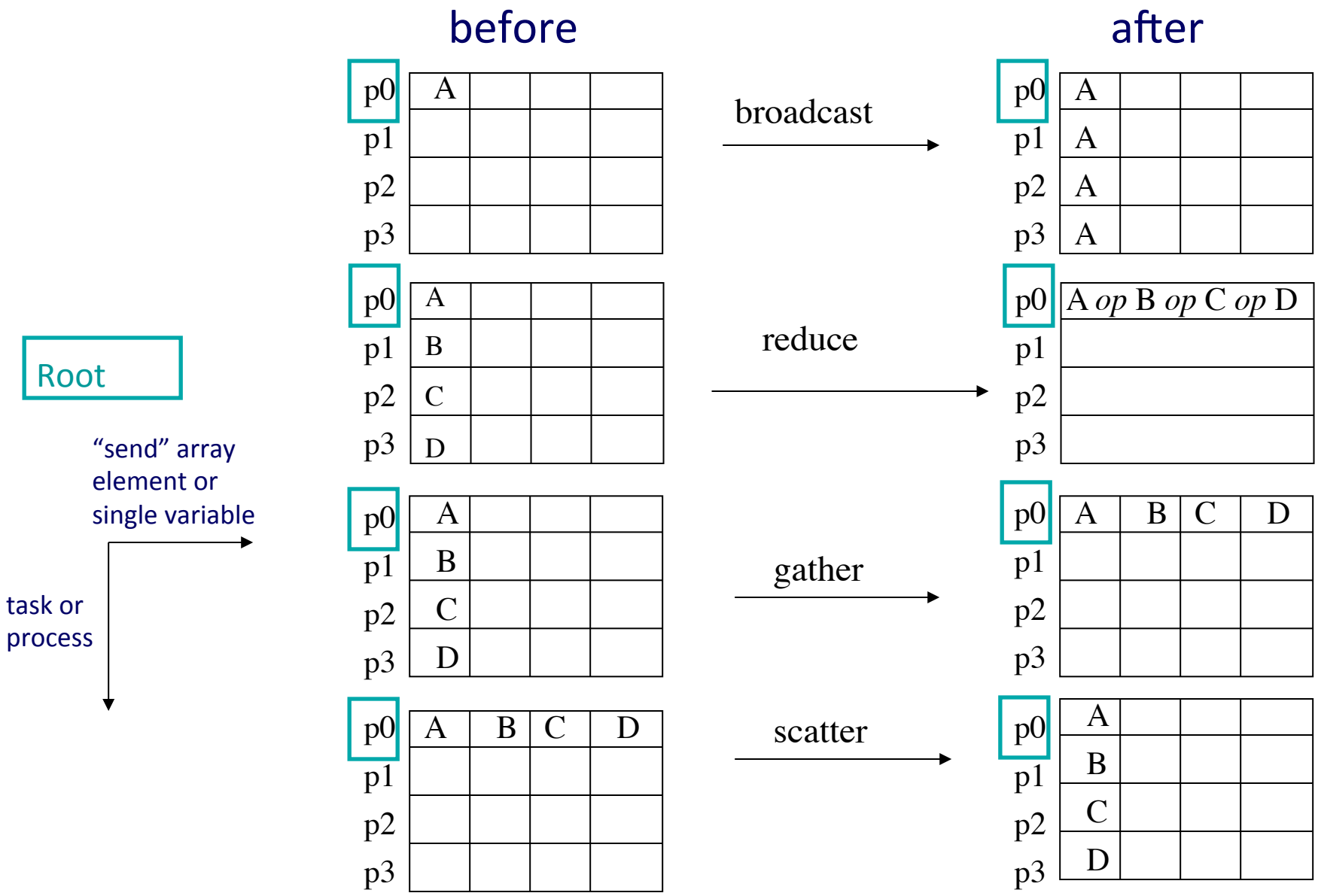
  Gather/Scatter— `MPI_Gather()/MPI_Scatter()...`
- "All" versions

  `MPI_Allreduce()`

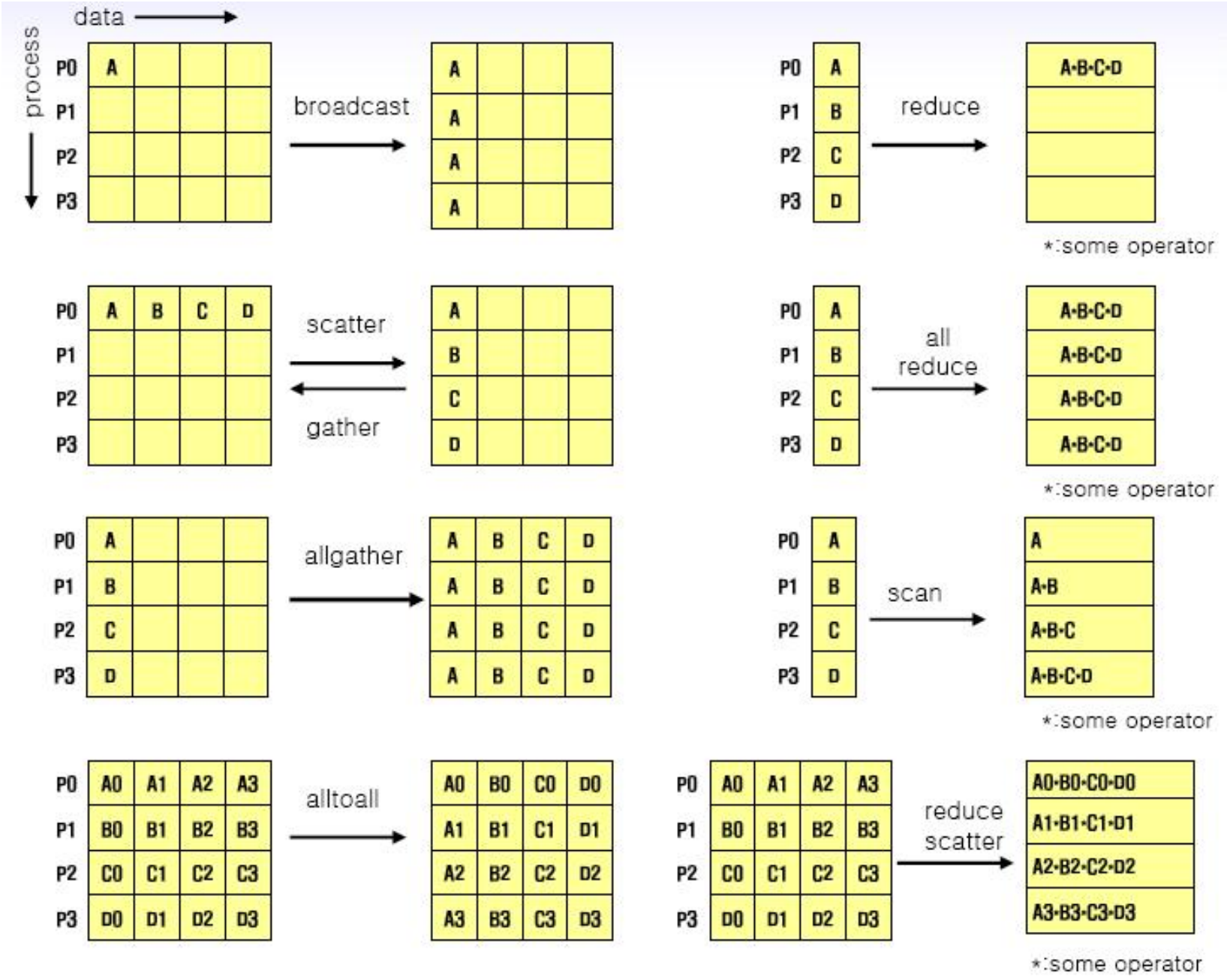
  `MPI_Allgather()`

  `MPI_Alltoall()...`
- Others

  `MPI_Barrier ...`


before
after
Root
"send" array element or single variable
task or process
broadcast
reduce
gather
scatter
p0
p1
p2
p3
A
B
C
D
A op B op C op D

Collective Communication: Summary



MPI Collective Communications

- Every process MUST call the routine
  - All calls are blocking
  - A task may return when participation is complete
  - May or may not synchronize (implementation dependent)
- Must have "matching" arguments
  - no status
  - no tags
- Send and Receive sizes must match
  - mapping may vary
- Basic calls have a *root*—"all" versions don't

Broadcast Operation: MPI_Bcast

- All nodes call MPI_Bcast
- One node (root) sends a message to all
  - all others receive the message
- C

```
ierr=MPI_Bcast(&dat[0], cnt, datatype, root, comm);
```

- Fortran

```
call MPI_Bcast( dat,    cnt, datatype, root, comm, ierr)
```

Reduction Operations

- Used to combine (reduce) partial results from all processors
- Result returned to root processor
- pre-defined or user-defined operations
  - Predefined: associative & commutative (com)
    Order may not be canonical (rank order)
  - User defined: Must be associative. com or non-com
    "Canonical" evaluation
- Works on a scalar variable or arrays (elemental)

MPI_Reduce

- C

```
ierr=MPI_Reduce(&sbuf[0], &rbuf[0], count,
        datatype, operator, root,     comm)
```

- Fortran

```
call MPI_Reduce( sbuf,     rbuf,     count,
        datatype, operator, root,     comm, ierr)
```

- Parameters
  - like MPI_Bcast, a root is specified
  - operation is a type of mathematical operation
- Applies the operator to each element globally
  - send and receive buffers are the same size
- Use MPI_Op_create for user-defined operation.

Operations for MPI_Reduce

MPI_PROD	Product
MPI_SUM	Sum
MPI_**L**AND	**L**ogical and
MPI_**L**OR	**L**ogical or
MPI_**L**XOR	**L**ogical exclusive or
MPI_**B**AND	**B**itwise and
MPI_**B**OR	**B**itwise or
MPI_**B**XOR	**B**itwise exclusive or
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_MAXLOC	Maximum value and location
MPI_MINLOC	Minimum value and location

Dot Product of Two Vectors

```
double    a[N], b[N];
double    globalSum=0.0, localSum=0.0;
...
for(i=0;i<N;++i) localSum+=a[i]*b[i];

MPI_Reduce(&localSum,&globalSum,1, MPI_DOUBLE
           MPI_SUM,root,MPI_COMM_WORLD);
```

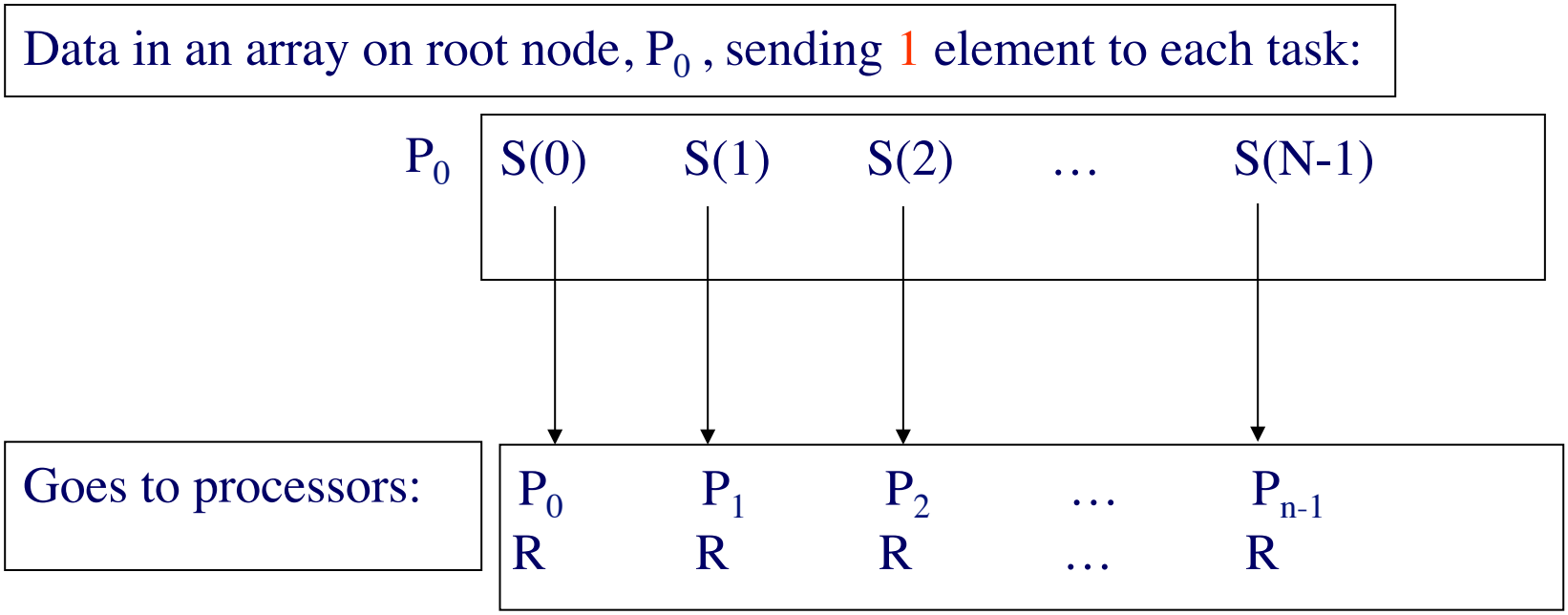
Dot Product of Two Vectors

```
real*8 :: a(N), b(N);
real*8 :: globalSum=0.0, localSum=0.0;
...
localSum = dot_product(a,b);

MPI_Reduce( localSum, globalSum,1, MPI_REAL8 &
            MPI_SUM,root,MPI_COMM_WORLD,ierr);
```

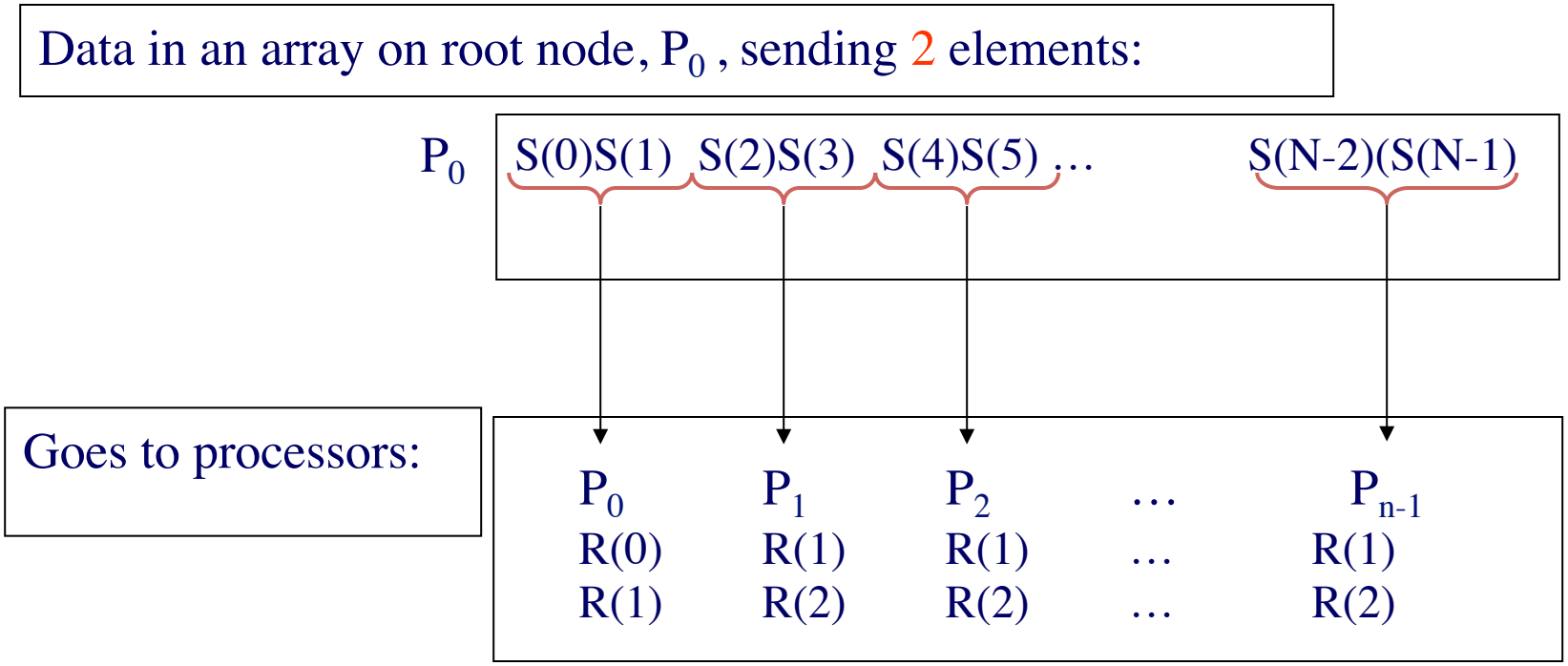
Scatter Operation using MPI_Scatter

- Similar to Broadcast but sends a section of an array to each processor



Scatter Operation using MPI_Scatter

- 2-elements per processor



MPI_Scatter Syntax

- C

  ierr = MPI_Scatter(&sbuf[0], scnt, stype, &rbuf[0], rcnt, rtype, root, comm);

- Fortran

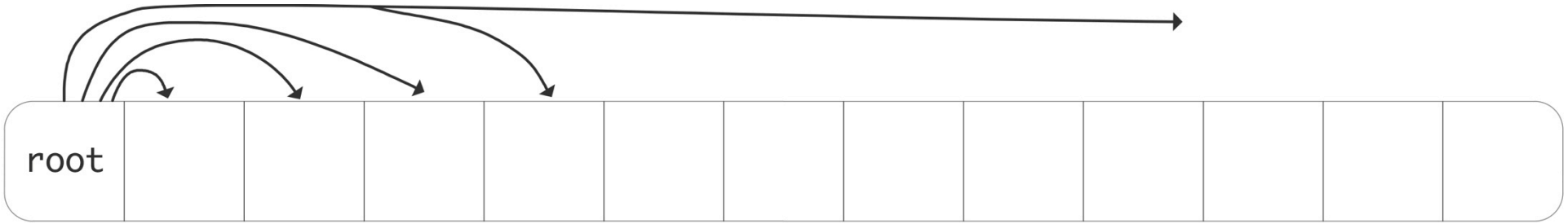
  call MPI_Scatter(sbuf, scnt, stype, rbuf, rcnt, rtype, root, comm,ierr)

- Parameters
  - sbuf = array of size np*scnt (np = # of ranks)
  - scnt = number of elements sent to each processor
  - rcnt = number of element(s) obtained from the root processor
  - rbuf = element(s) obtained from the root processor (rcnt in size)

e.g. MPI_Scatter(S, 1, stype, R, 1, rtype, root, comm)
Array Scalar

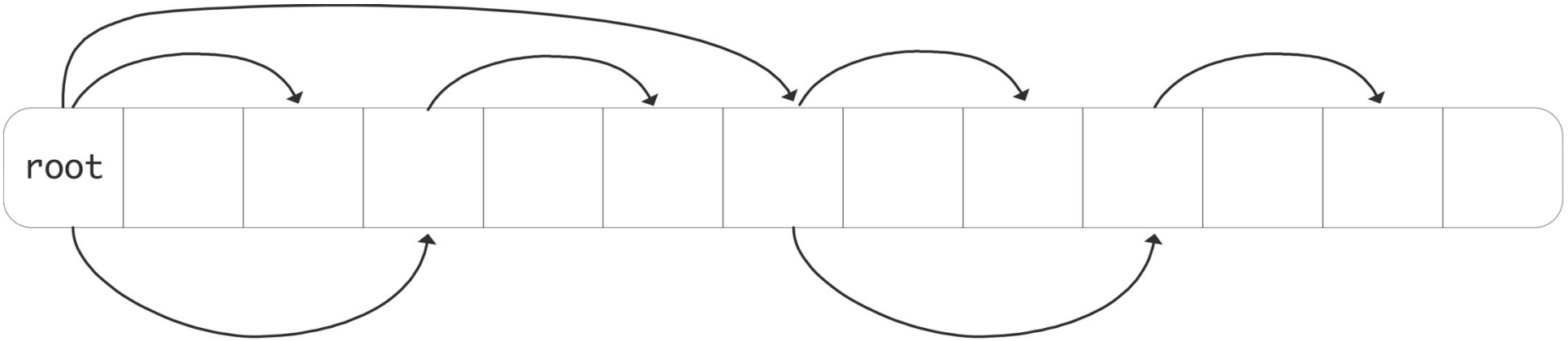
A word about implementation

How do you do a broadcast?



- Complexity?

How really to do a broadcast



- Complexity?