Version 2013-1
as of 31 May 2013

Parallel Computing for Science and Engineering

MPI Collectives 1

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

MPI Collective Communications

- Involves a group of processes.
- Basic Routines

  Broadcast— `MPI_Bcast()`

  Reduce— `MPI_Reduce()`

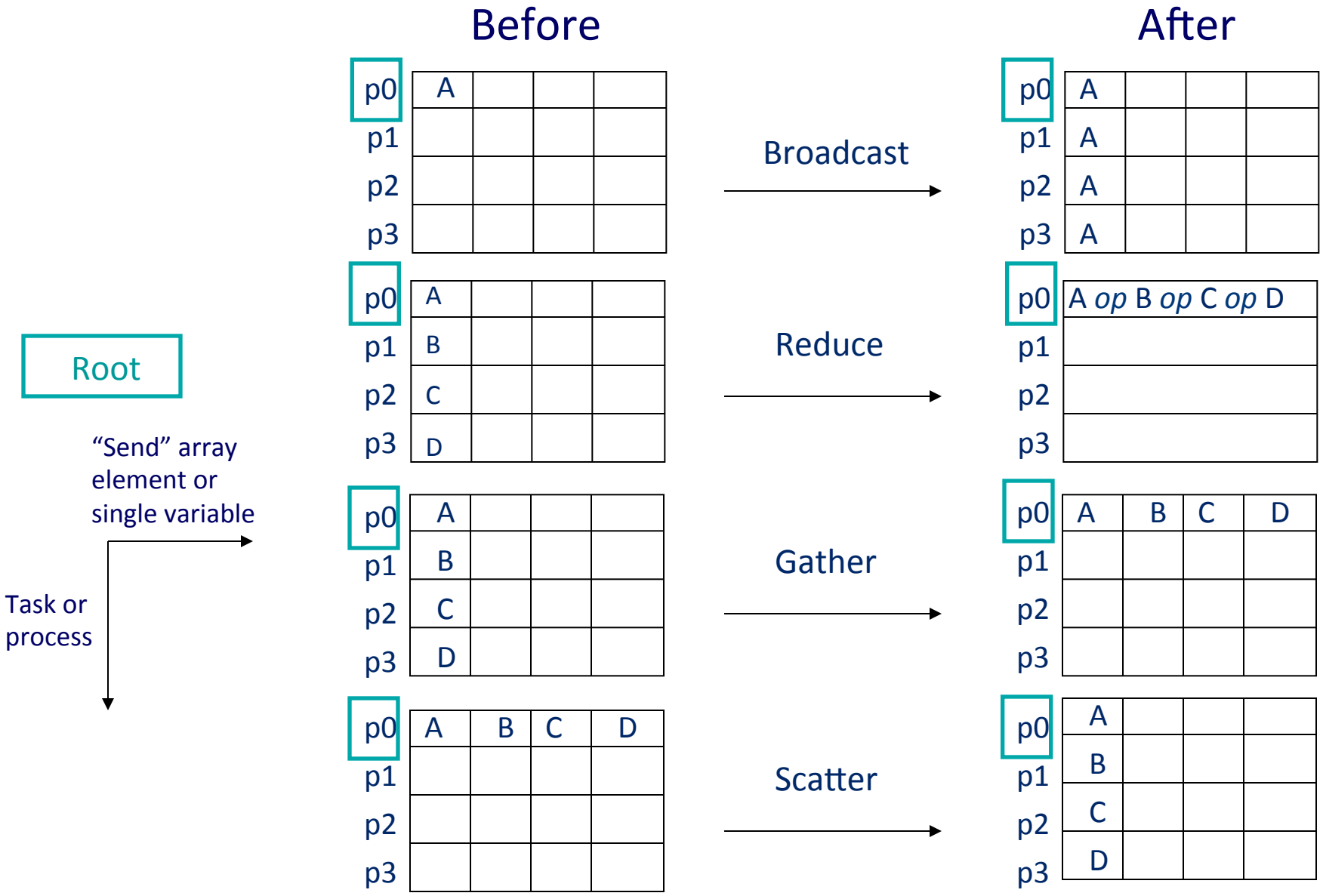
  Gather/Scatter— `MPI_Gather()MPI_Scatter()...`
- "All" versions

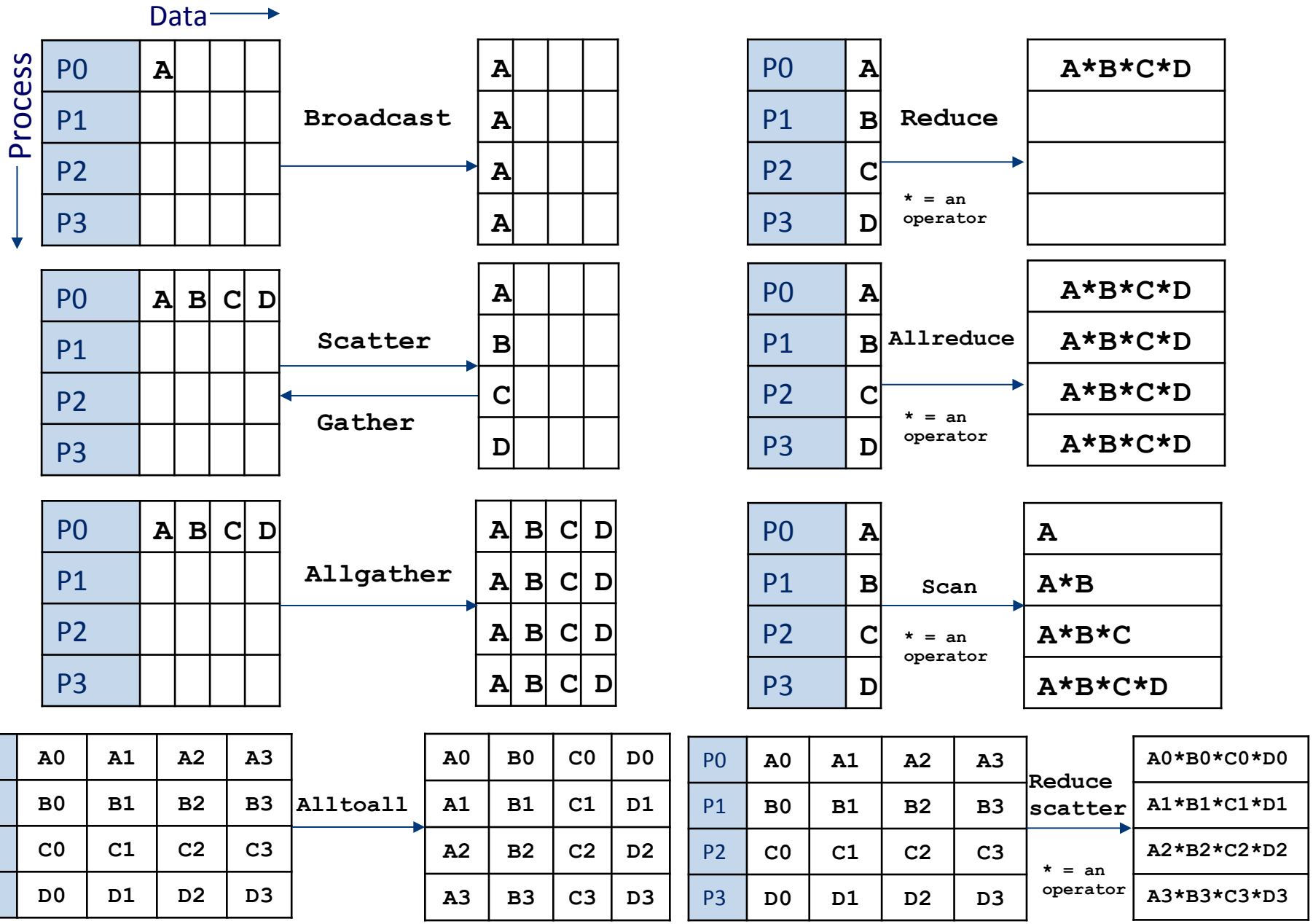
  `MPI_Allreduce()`

  `MPI_Allgather()`

  `MPI_Alltoall()...`
- Others

  `MPI_Barrier ...`


Before
After
Root
"Send" array element or single variable
Task or process
p0
p1
p2
p3
A
B
C
D
Broadcast
Reduce
A op B op C op D
Gather
Scatter


Data
Process
P0
P1
P2
P3
A
Broadcast
A
A
A
A
P0
P1
P2
P3
A
B
C
D
Reduce
* = an operator
A*B*C*D
P0
P1
P2
P3
A
B
C
D
Scatter
Gather
A
B
C
D
P0
P1
P2
P3
A
B
C
D
Allreduce
* = an operator
A*B*C*D
A*B*C*D
A*B*C*D
A*B*C*D
P0
P1
P2
P3
A
B
C
D
Allgather
A B C D
A B C D
A B C D
A B C D
P0
P1
P2
P3
A
B
C
D
Scan
* = an operator
A
A*B
A*B*C
A*B*C*D
P0
P1
P2
P3
A0
A1
A2
A3
B0
B1
B2
B3
C0
C1
C2
C3
D0
D1
D2
D3
Alltoall
A0
B0
C0
D0
A1
B1
C1
D1
A2
B2
C2
D2
A3
B3
C3
D3
P0
P1
P2
P3
A0
A1
A2
A3
B0
B1
B2
B3
C0
C1
C2
C3
D0
D1
D2
D3
Reduce scatter
* = an operator
A0*B0*C0*D0
A1*B1*C1*D1
A2*B2*C2*D2
A3*B3*C3*D3

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_LAND	**L**ogical and
MPI_LOR	**L**ogical or
MPI_LXOR	**L**ogical exclusive or
MPI_BAND	**B**itwise and
MPI_BOR	**B**itwise or
MPI_BXOR	**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

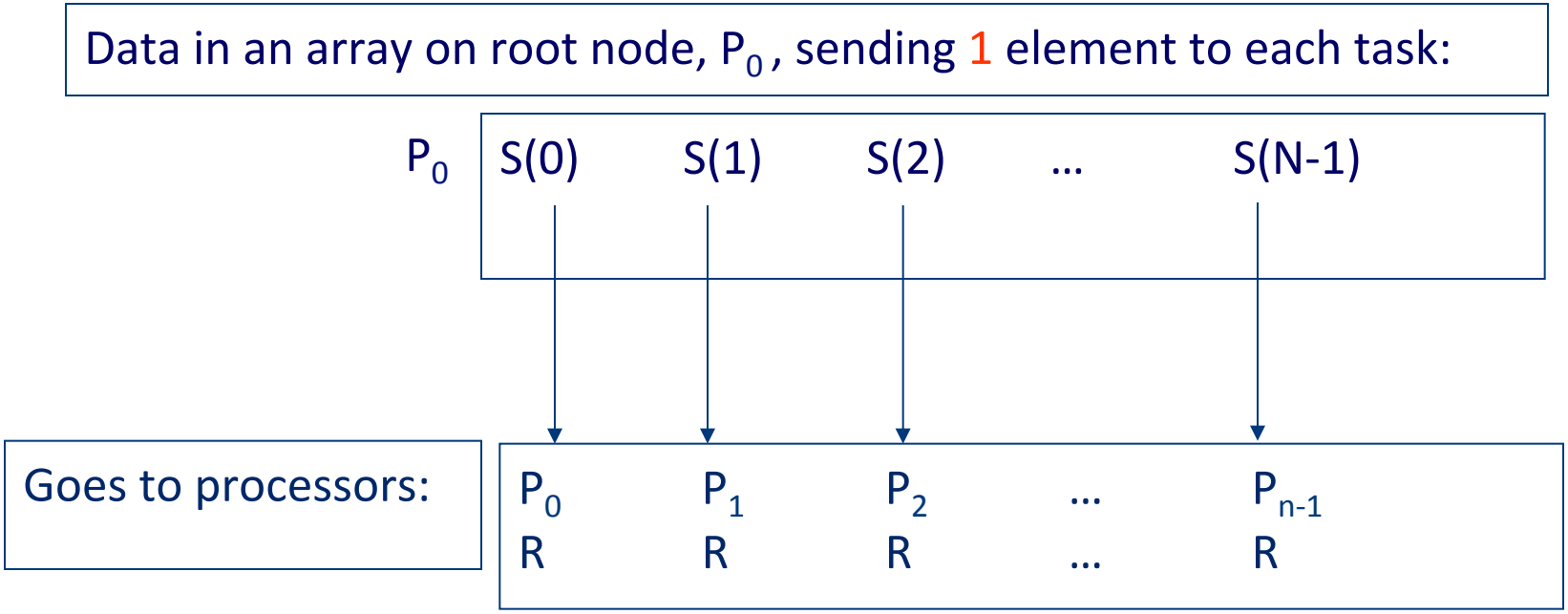
```
real*8 :: a(N), b(N);
real*8 :: globalSum=0.0, localSum=0.0;
...
localSum = dot_product(a,b);

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


11

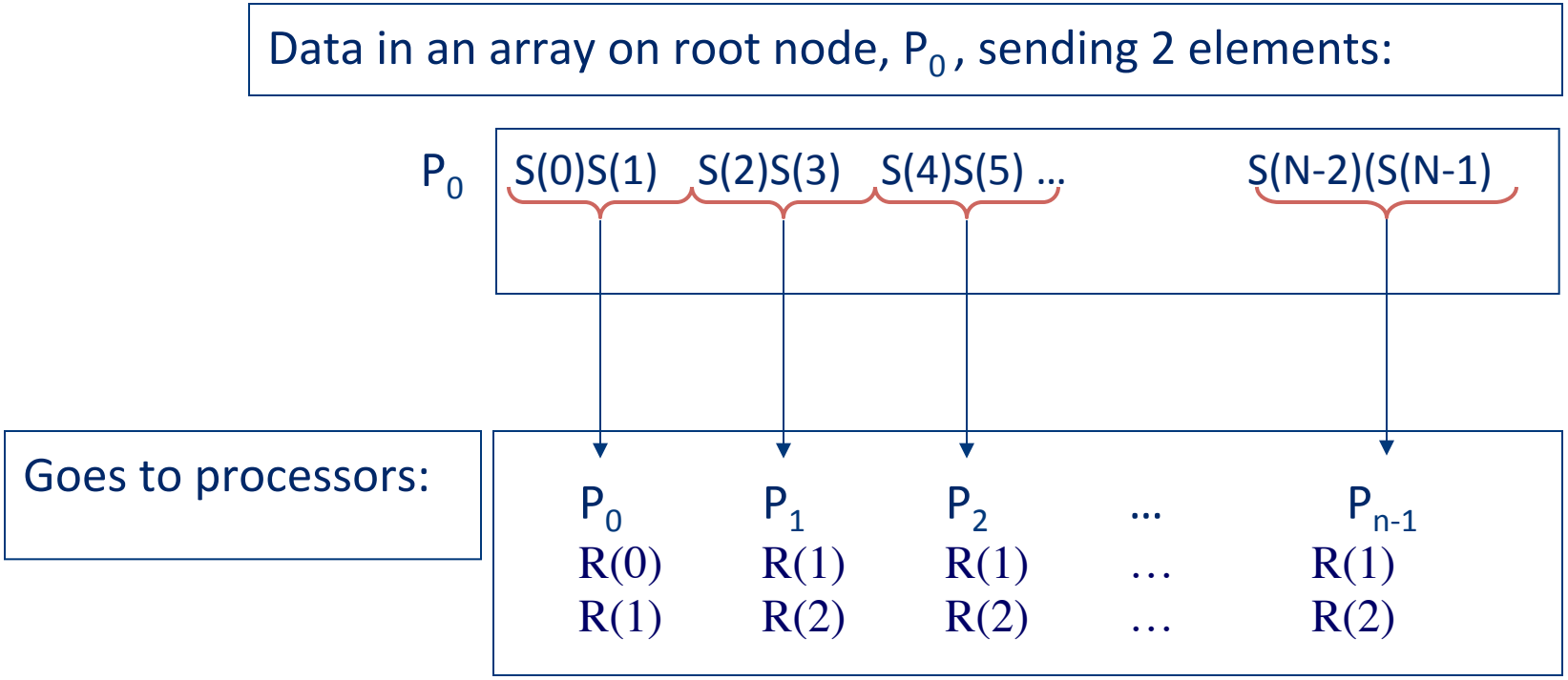
Scatter Operation using `MPI_Scatter`

- 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

We gratefully acknowledge the sponsorship of Chevron Corporation, whose generous support of TACC has made possible this Scientific Computing Curriculum and other student-focused initiatives.

License