Introduction to Collective Operations in MPI

- Collective operations are called by all processes in a communicator.
- **`MPI_BCAST`** distributes data from one process (the root) to all others in a communicator.
- **`MPI_REDUCE`** combines data from all processes in the communicator and returns it to one process.
- In many numerical algorithms, **`SEND/RECV`** can be replaced by **`BCAST/REDUCE`**, improving both simplicity and efficiency.

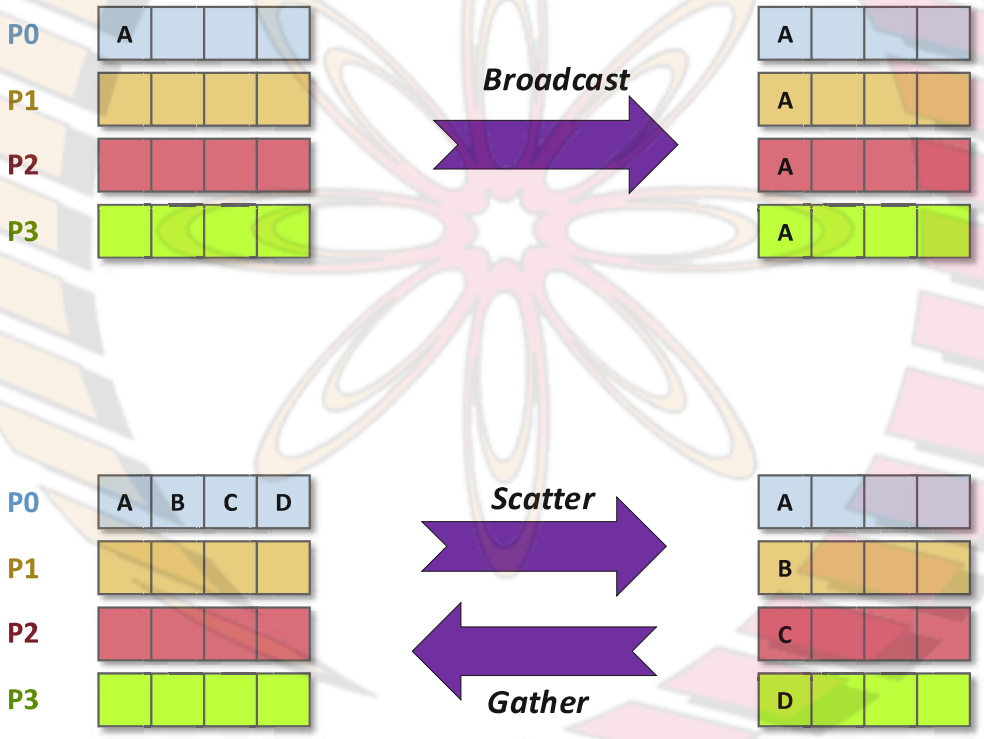
MPI Collective Communication

- Communication and computation is coordinated among a group of processes in a communicator
- Tags are not used; different communicators deliver similar functionality
- Nonblocking collective operations added in MPI-3
- Three classes of operations: synchronization, data movement, collective computation

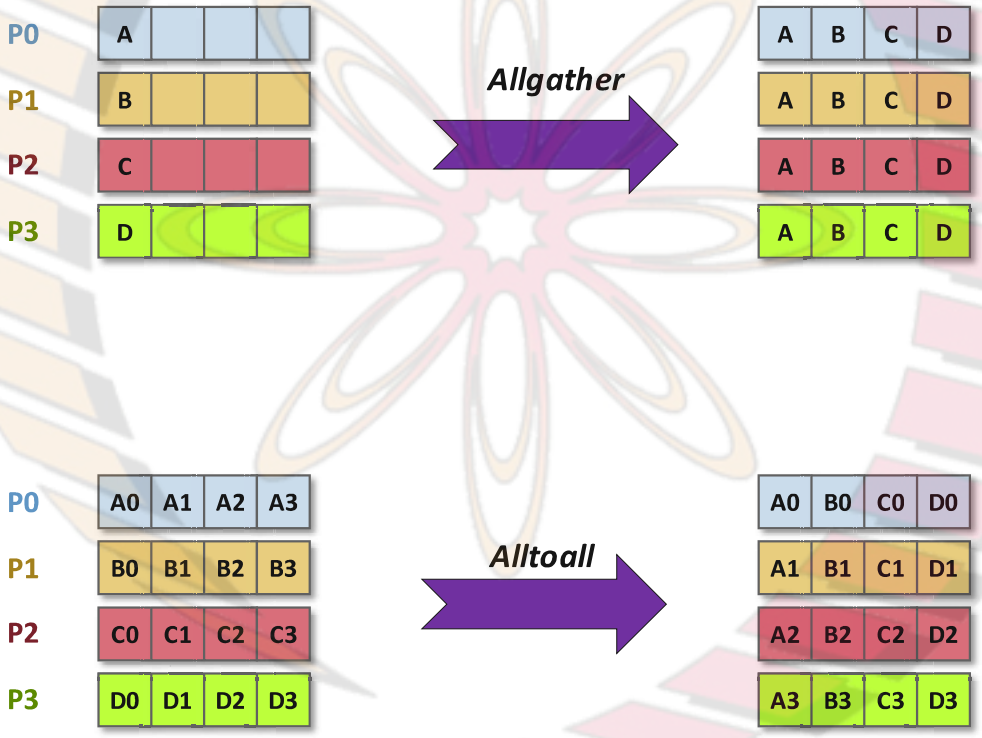
Synchronization

- **`MPI_BARRIER(comm)`**
  - Blocks until all processes in the group of the communicator **`comm`** call it
  - A process cannot get out of the barrier until all other processes have reached barrier

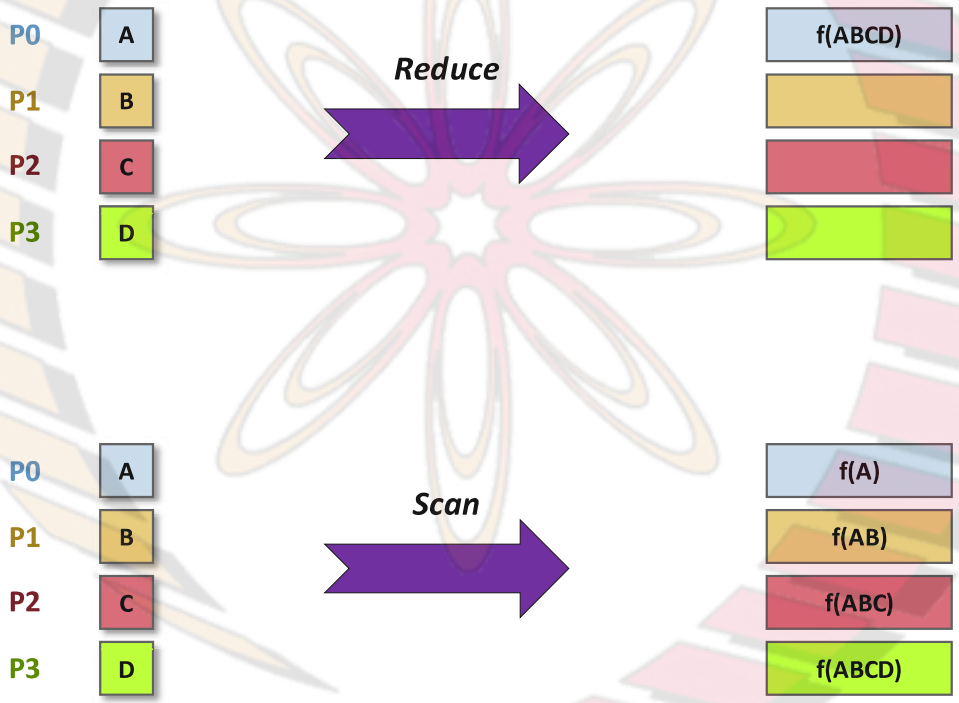
Collective Data Movement



More Collective Data Movement



Collective Computation



MPI Collective Routines

- Many Routines: **MPI_ALLGATHER**, **MPI_ALLGATHER*V***, **MPI_ALLREDUCE**, **MPI_ALLTOALL**, **MPI_ALLTOALL*V***, **MPI_BCAST**, **MPI_GATHER**, **MPI_GATHER*V***, **MPI_REDUCE**, **MPI_REDUCESCATTER**, **MPI_SCAN**, **MPI_SCATTER**, **MPI_SCATTER*V***
- "**All**" versions deliver results to all participating processes
- "***V***" versions (stands for vector) allow the chunks to have different sizes
- **MPI_ALLREDUCE**, **MPI_REDUCE**, **MPI_REDUCESCATTER**, and **MPI_SCAN** take both built-in and user-defined combiner functions

MPI Built-in Collective Computation Operations

MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_PROD	Product
MPI_SUM	Sum
MPI_LAND	Logical and
MPI_LOR	Logical or
MPI_LXOR	Logical exclusive or
MPI_BAND	Bitwise and
MPI_BOR	Bitwise or
MPI_BXOR	Bitwise exclusive or
MPI_MAXLOC	Maximum and location
MPI_MINLOC	Minimum and location

Nonblocking Collective Communication

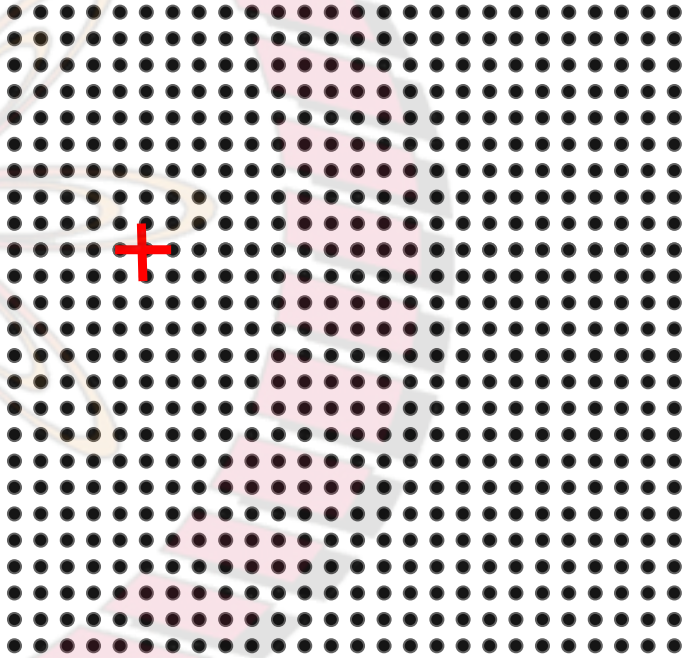
- Nonblocking (send/recv) communication
  - Deadlock avoidance
  - Overlapping communication/computation
- Collective communication
  - Collection of pre-defined optimized communication patterns
- → Nonblocking collective communication
  - Combines both techniques
  - System noise/imbalance resiliency
  - Semantic advantages

Running Example: Regular Mesh Algorithms

- Many scientific applications involve the solution of partial differential equations (PDEs)
- Many algorithms for approximating the solution of PDEs rely on forming a set of difference equations
  - Finite difference, finite elements, finite volume
- The exact form of the differential equations depends on the particular method
  - From the point of view of parallel programming for these algorithms, the operations are the same
- Five-point stencil is a popular approximation solution

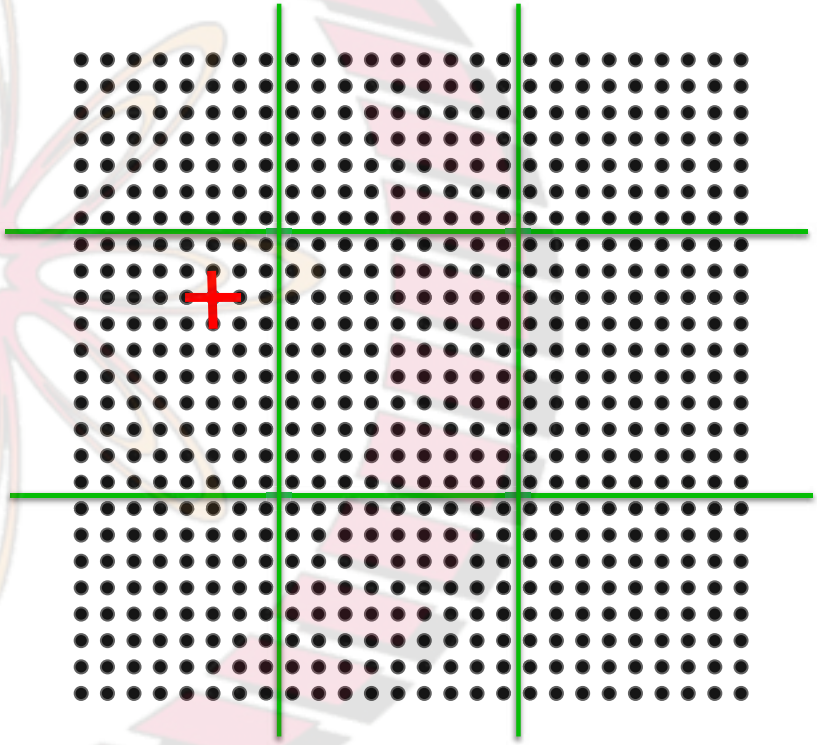
The Global Data Structure

- Each circle is a mesh point
- Difference equation evaluated at each point involves the four neighbors
- The red “plus” is called the method’s stencil
- Good numerical algorithms form a matrix equation Au=f; solving this requires computing Bv, where B is a matrix derived from A. These evaluations involve computations with the neighbors on the mesh.



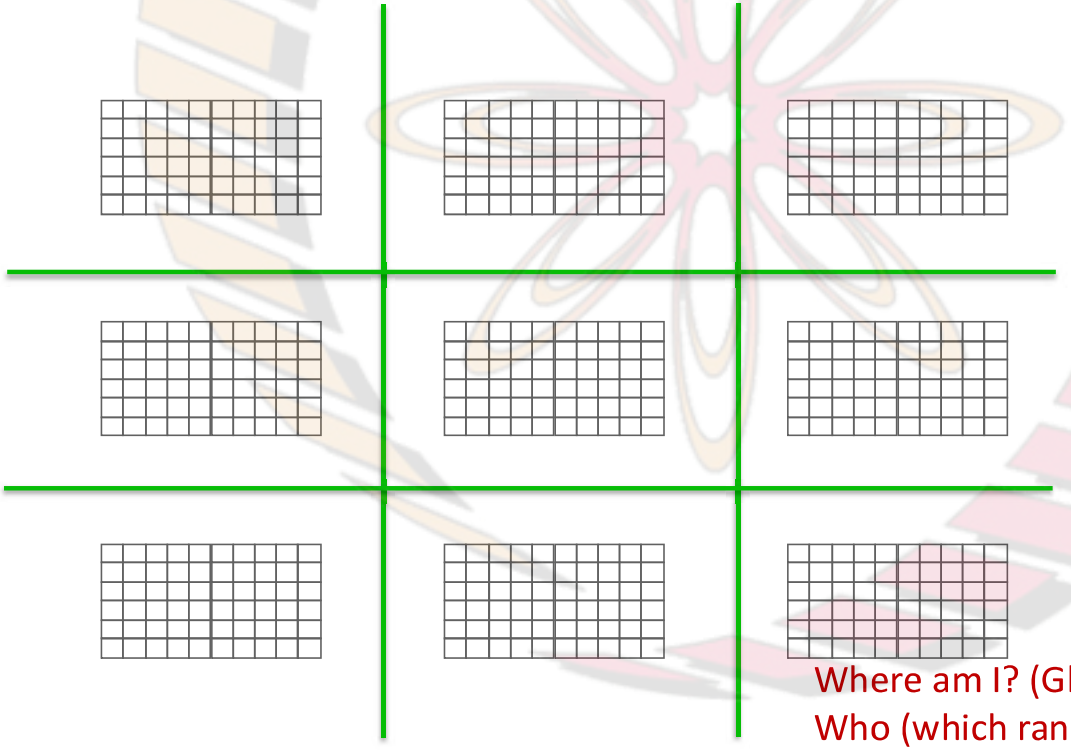
The Global Data Structure

- Each circle is a mesh point
- Difference equation evaluated at each point involves the four neighbors
- The red “plus” is called the method’s stencil
- Good numerical algorithms form a matrix equation Au=f; solving this requires computing Bv, where B is a matrix derived from A. These evaluations involve computations with the neighbors on the mesh.
- Decompose mesh into equal sized (work) pieces



Domain Decompositioin

Parameters for domain decomposition:
N = Size of the edge of the global problem domain (assuming square)
PX, PY = Number of processes in X and Y dimension
N % PX == 0, N % PY == 0



Where am I? (Global offset)
Who (which ranks) are my neigbhors?
Use MPI_PROC_NULL for boundary