Lab 11 Mini-Lecture – MPI Collectives

Prof. Brendan Kochunas

NERS/ENGR 570 - Methods and Practice of Scientific Computing (F20)

COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Outline

- Collective communication

- Deadlock Errors

Learning Objectives: By the end of Today's Lecture you should be able to

- (*Knowledge*) describe which operations are collective operations
- (*Value*) explain why it is better to use the MPI collectives than write your own
- (Skill) identify when a programming error with MPI will result in a deadlock and diagnose a deadlock in an MPI program

Collective Communication

MPI Collectives (1)

- These involve all MPI processes in a *communicator*
- Collectives can always be implemented with point-to-point routines
  - But it is often better to use the routines provided by MPI
- Common collective operations include:
  - Broadcast
  - Reduce
  - Scatter
  - Gather
  - Scan
  - Alltoall

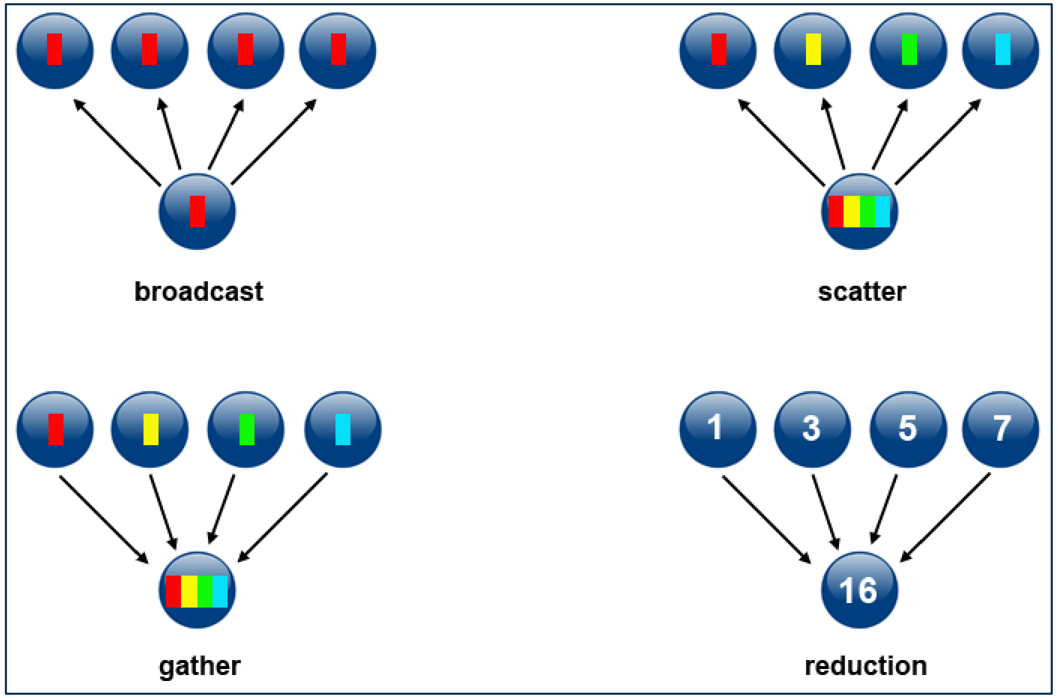


Figure from: https://computing.llnl.gov/tutorials/parallel_comp/

MPI Collectives (2)

Notable Variations

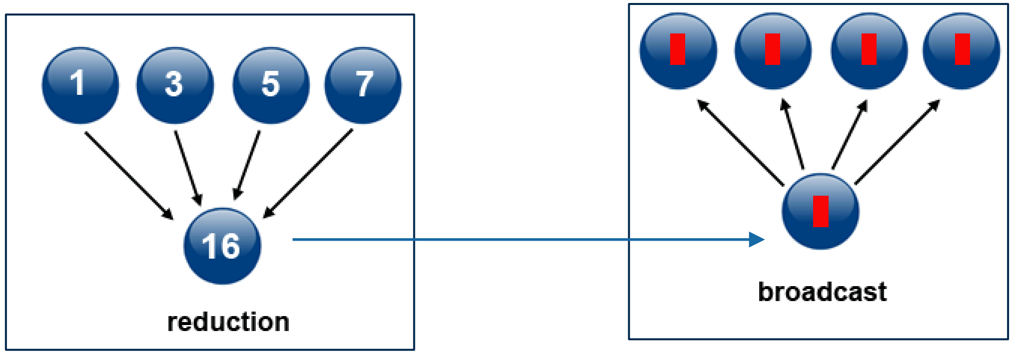
- The "`v`" suffix
  - Stands for vector
  - Means the *<u>size of data may be different</u>* for different processors
  - Gatherv & Scatterv, Alltoallv
- The "`All`" prefix
  - Means the *<u>result of the operation is the same for all processors</u>* in communicator
  - Allreduce & Allgather

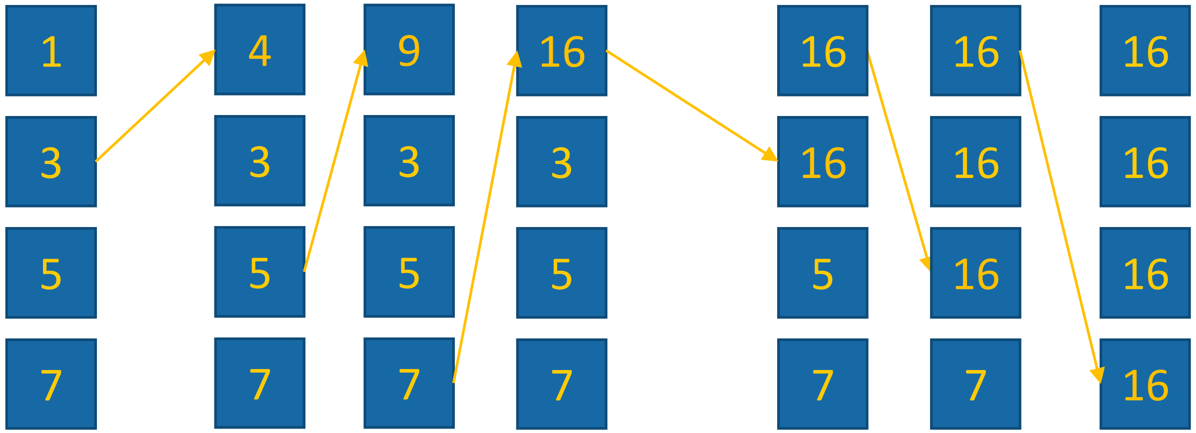
Types of reduction operations

- Arithmetic
  - MPI_SUM
  - MPI_PROD
- Relation Operators (Mins & Maxes)
  - MPI_MAX
  - MPI_MIN
  - MPI_MAXLOC
  - MPI_MINLOC
- Logical Operators
  - MPI_LAND
  - MPI_LOR
  - MPI_LXOR
- Bit-wise operators also supported

Example: MPI_Allreduce Algorithm

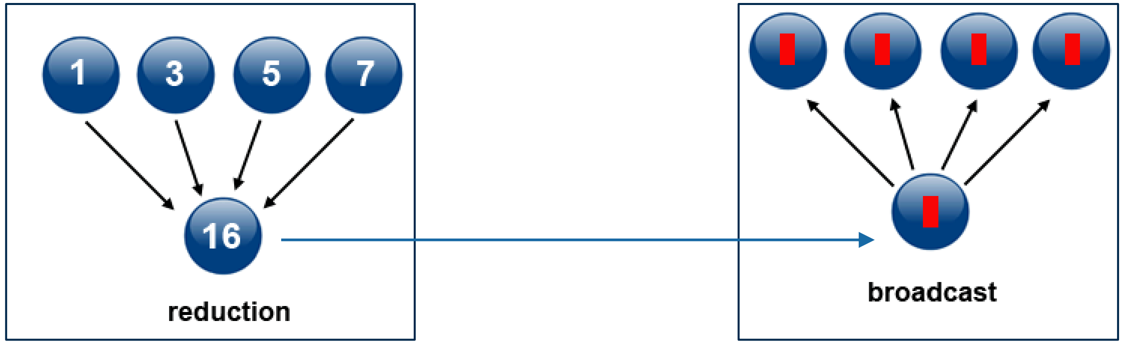
- Reduce + broadcast
- Reduce performed sequentially
  - P-1 steps
- Broadcast performed sequentially
  - Also P-1 steps
- Total of 6 steps

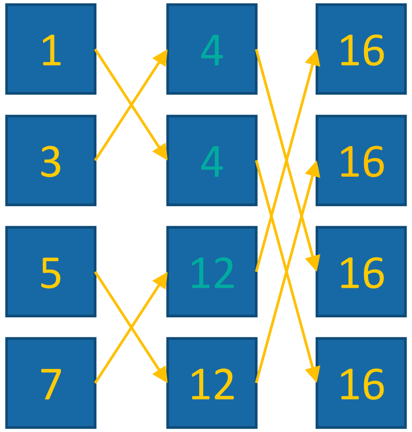




Example:Better Allreduce

- Use a binomial tree
  - Completed in $\lceil \log p \rceil$ steps
- Scales much better to higher number of processors





Even More Advanced Allreduce

- What about long messages?
  - Reduce_scatter + Allgather
- Different algorithms perform better under certain conditions

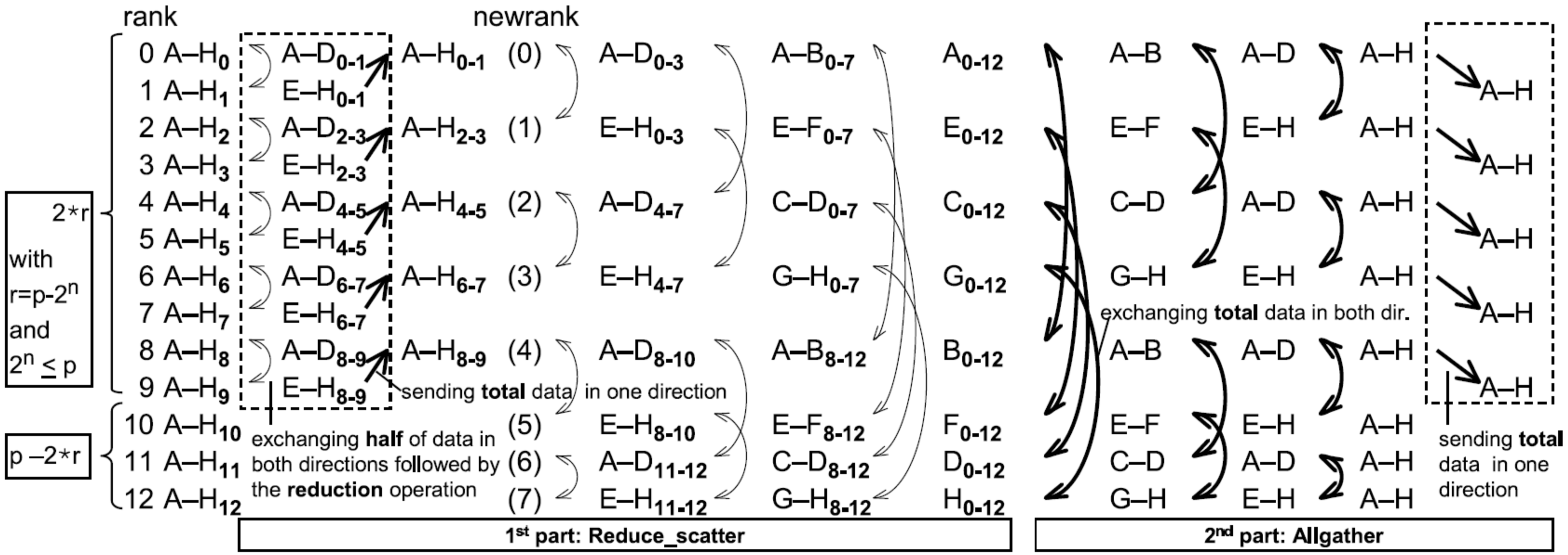


Figure 12: Allreduce using the recursive halving and doubling algorithm. The intermediate results after each communication step, including the reduction operation in the reduce-scatter phase, are shown. The dotted frames show the additional overhead caused by a non-power-of-two number of processes.

Source: http://www.mcs.anl.gov/~thakur/papers/ijhpca-coll.pdf

Summary of Collectives

- Provided as a convenience to the programmer
  - Collectives perform "common" operations that arise in programming
  - Often implemented with more complex and higher performing algorithms
    - Than what a beginner would implement.
- They represent a synchronization point in the program
- Always, always, always involves all processors *within communicator*
  - Otherwise, it causes a deadlock

Deadlock

Problem

- Symptoms
  - Code will run for a while
  - Then code will "hang".
  - Code just sits... and sits... and sits.

```
IF(MOD(myRank,2) == 0) THEN
  CALL MPI_Send(sbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank+1, 0, MPI_COMM_WORLD, mpierr)
  CALL MPI_Recv(rbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank+1, 0, MPI_COMM_WORLD, MPI_STATUS_NULL, mpierr)
ELSE
  CALL MPI_Send(sbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank-1, 0, MPI_COMM_WORLD, mpierr)
  CALL MPI_Recv(rbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank-1, 0, MPI_COMM_WORLD, MPI_STATUS_NULL, mpierr)
ENDIF
IF(MOD(myRank,2) == 0) &
  CALL MPI_Reduce(sbuf,rbuf,n,MPI_DOUBLE_PRECISION, MPI_SUM, &
    0, MPI_COMM_WORLD, mpierr)
```

Solution

- Investigate where your calls to communication are made.
  - Usually will happen around branching constructs.
- Think about how it would execute with 2 processors.

```
IF(MOD(myRank,2) == 0) THEN
  CALL MPI_Send(sbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank+1, 0, MPI_COMM_WORLD, mpierr)
  CALL MPI_Recv(rbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank+1, 0, MPI_COMM_WORLD, MPI_STATUS_NULL, mpierr)
ELSE
  CALL MPI_Recv(rbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank-1, 0, MPI_COMM_WORLD, MPI_STATUS_NULL, mpierr)
  CALL MPI_Send(sbuffer, n, MPI_DOUBLE_PRECISION, &
    myRank-1, 0, MPI_COMM_WORLD, mpierr)
ENDIF
```