Homework 2 - Parallel Programming for Large Scale Problems - SF2568

Gabriel Carrizo

December 2017

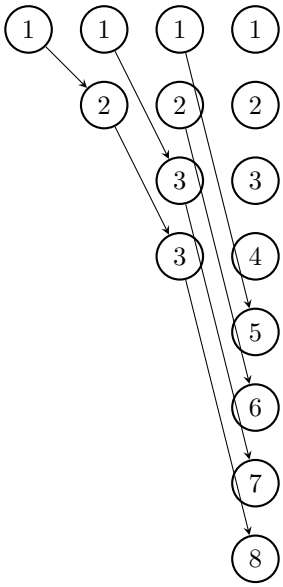

Figure 1: Graph representation of broadcasting with recursive doubling.

1 Broadcast Operation

1. Design an algorithm for the broadcast operation only using point-to-point comunication.

Solution: The brodcast operation is an operation that sends the information of one node to all other processes. Using recursive doubling one can 'spread' amongst processes, much like an infection moves through a host.

The idea is that the process with the data sends its data to another process. Once that process also possesses the data, both processes send the data to two other processes. Now we have four processes with data and all four can send data to four other processes. This procedure repeats until all processes posses the data (see figure 1).

The complexity of the solution in parallel is dependent on the number of processors used.

See Listing 1 for pseudoecode. (Made under the assumption that $P = 2^D$)

```
"""
Pseudocode for broadcast operation using only
point-to-point operations.

Note that the elif statement triggers before the
if statement for every process except for process 0.
"""

for d = 0->D-1:
    dest = bitflip(rank,d) #send or recieve destination
    if (rank < 2^d): #if rank is less than 2^d we send to dest
        send(dest)
    elif (rank < 2^(d+1)): #elif rank is less than 2^(d+1) we recieve.
        recieve(dest)
```

Listing 1: Pseudocode for broadcasting operation.

2. Do a (time-)performance analysis for your algorithm.

Solution: blabla

3. How can the scatter operation be implemented using $O(logP)$ communication steps?

Solution: Assuming we know the length N of the data and that $N = 2^K$, where $K \geq P$, such that the data is evenly distributed among the processes.

2 Transpose

Solution: Assuming that what the algorithm from class leaves us with a grid with load balanced data distribution where each process has address $P[row, col]$:

$$M = \begin{bmatrix} y_0 & y_0 & \cdots & y_0 \\ y_1 & y_1 & \cdots & y_1 \\ \cdots & \cdots & \cdots & \cdots \\ y_P & y_P & \cdots & y_P \end{bmatrix}$$

Which we want to transpoe into:

$$M^T = \begin{bmatrix} y_0 & y_1 & \cdots & y_P \\ y_0 & y_1 & \cdots & y_P \\ \cdots & \cdots & \cdots & \cdots \\ y_0 & y_1 & \cdots & y_P \end{bmatrix}$$

We observe that the diagonal of the matrix M holds values that should be broadcast to the entire column.

There are two approaches that work with the broadcast algorithm suggested in the previous section.

The first one is to send the data from the first column to the first row, such that P[0,1] sends to P[1,0], P[0,2] send to P[2,0], etc. Once the first row is filled with its initial value the broadcast be performed, just like in the previous section but for each column individually.

The second approach I will suggest is to perform a cyclic shift for the rows of each column s.t. every diagonal element is shifted to the the first row of matrix M. This method requires some adjustments so the processes send and recieve to and from the correct index (see Listing 2)

The first approach requires an extra communication step and will also require sending data to some processes on the diagonal that already posses the data. The second approach requires some extra calculation steps but should overall be quicker since it requires fewer communication steps.

```
"""
Builds on broadcast function from previous part.

Broadcasts values such as to transpose y.

Performs cyclic shift on all rows of each column such that the diagonal value
    is on the first row

Processes are distributed on a square matrix [P x P]

Each process will posses its location as location(p) = (row,col)

"""

new_row = row - column

if new_row < 0:
    new_row = new_row + P

for d 0->D-1:
    #--------- Main change from last section -------
    if new < 2^d:
        dest = col + bitflip(new_row, d)

    if dest >= P:
        dest = dest - P
    #-----------------------------------------------

    if (new_row < 2^d): #if row is less than 2^d we send to dest
        send(dest, column)
    elif (new_row < 2^(d+1)): #elif row is less than 2^(d+1) we recieve.
        recieve(dest, column)
```

Listing 2: Pseudocode for transpose with the broadcast operation operation.