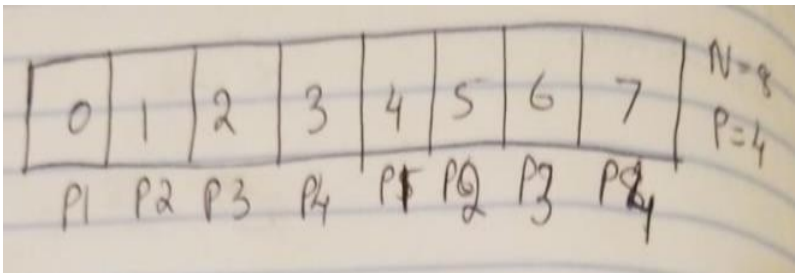
Assignment: Distributed Memory: representation and algorithm

1) Heat Equation - 1D

1.1 Round Robin Decomposition



1) Generate the local matrix using (rank+iP) where i is any number between 0 and N/P.

2) If rank=0 send only to rank+1, else if rank=P-1 send only to rank-1 else send to both rank-1 and rank+1.

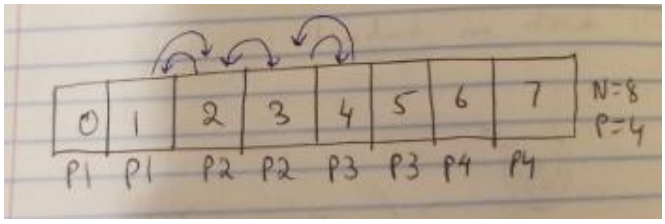
3) Similarly, if rank=0 receive only from rank+1, else if rank=P-1 receive only from rank-1 else receive from both rank-1 and rank+1.

4) Now calculate the heat equation based on the given formula.

5) Repeat the process for K number of iterations.

Communication: Assuming the array is of size N. Except the first and last node every other node sends to both the previous and next node. Hence number of communications would be 2*N/P per node. And for the first and last node number of communications is N/P. Total number of communications is 2*(N/P)*(P-2)+2*(N/P).

1.2 Block Decomposition



1) Generate the local matrix using (rank*N/P+i) where i is any number between 0 and N/P.

2) If rank=0 send only the last element to rank+1, else if rank=P-1 send only the first element to rank-1 else send first element to rank-1 and last element to rank+1.

3) Similarly, if rank=0 receive only the first element from rank+1, else if rank=P-1 receive only the last element from rank-1 else receive last element from rank-1 and first element from rank+1.

4) Now calculate the heat equation based on the given formula.

5) Repeat the process for K number of iterations.

Communication: Except the first and last node every other node sends first element to the previous node and last element to the next node. Hence number of communications would be 2 per node. And for the first and last node communication would be 1 per iteration per node. Total number of communications for all nodes would be 2*(P-2) +2.

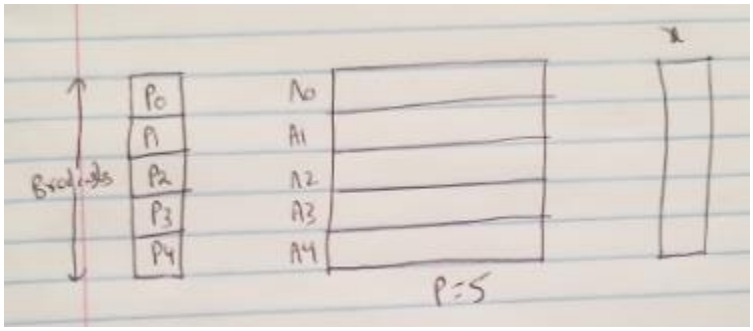
1.3 Reflection

Q) What data partitioning would you use?

I would use block decomposition because irrespective of the value of N the number of communications per iteration per node is 2 whereas for round robin it is 2*(N/P).

2 Dense Matrix Multiplication

2.1) 1D partitioning: Horizontal stripes

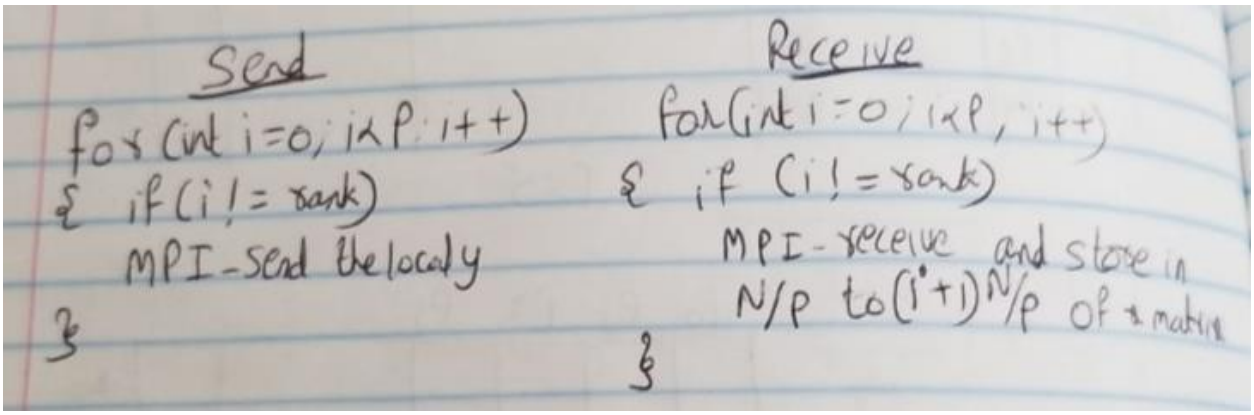


1) Initialize k=0.

2) Generate the matrices such that local A has a size of N X (N/P) and X has size of N.

3) Since each processor has its own copy of X matrix, compute the matrix multiplication between N/P rows of local matrix and column vector of x save the partial result in y.

4) Now each node sends the partial y and receives the partial y from all other nodes into appropriate positions in X matrix.

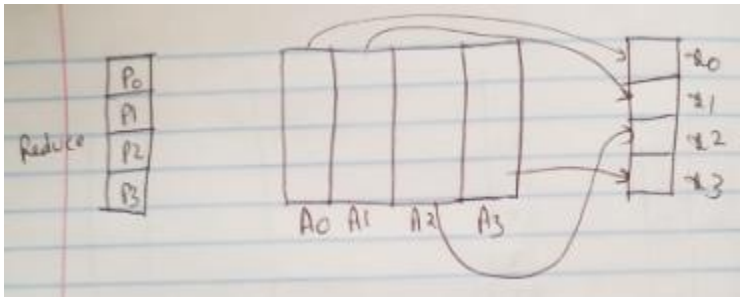


5) Go to step 3 and repeat the process for 10 iterations.

Memory: Assume each element requires O(1) memory. Therefore, memory required for local A matrix is O(N*N/P), memory required for X is O(N) and finally local y requires O(N/P). So total memory required per node is O((N^2/P) +N+N/P).

Communication: Every node sends the local y to all other nodes. So, the number of communications for 1 node is (N/P* (P-1)). Since all the P nodes have to send to every other node, total number of communications will be N*(P-1).

2.2) 1D partitioning: vertical stripes



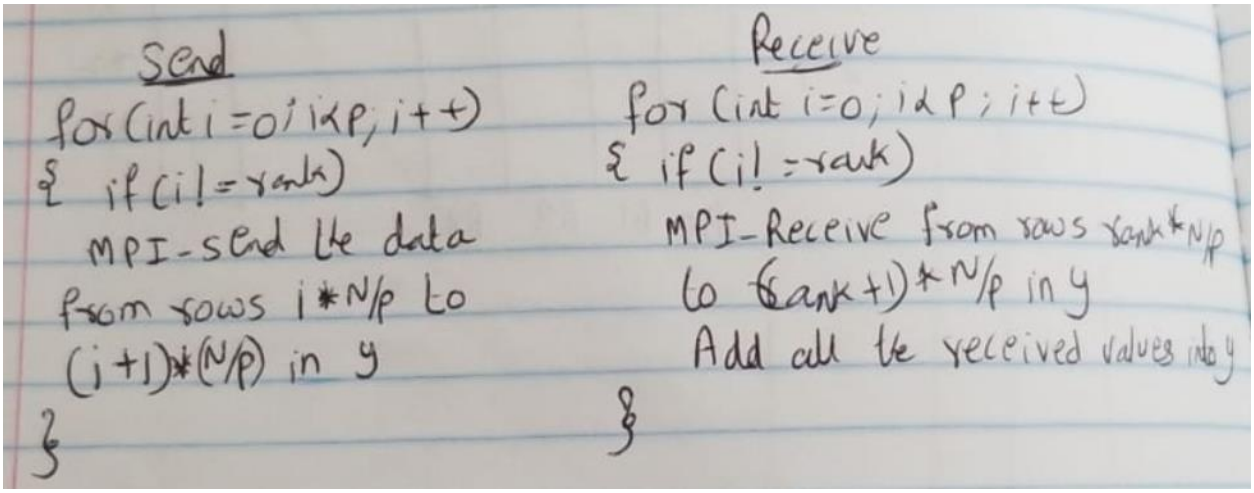
1) Initialize k=0.

2) Generate the matrices such that A is of size N * (N/P) and local X is of size N/P.

3) calculate the matrix multiplication, such that each node operates on its corresponding chunk.

i.e. A0 on X0 and A1 on X1 and so on.

4) Now send the local y matrix to all nodes other than it and receive the data from all the nodes other than it.



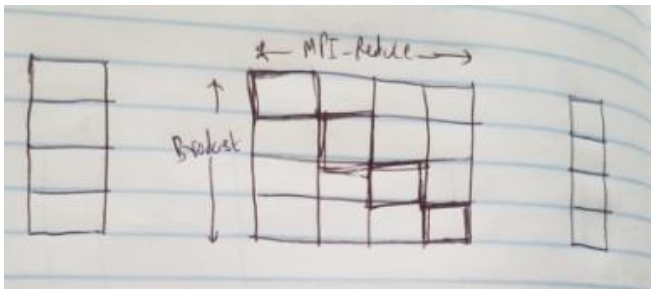
5) Swap the rank*N/P to (rank+1)*N/P elements in y matrix to local x matrix.

6) Go to step3 and repeat the process for 10 iterations.

Memory: Assume each element requires O(1) memory. Therefore, memory required for local A matrix is O(N*N/P), memory required for X is O(N/P) and finally local y requires O(N). So total memory required per node is O((N^2/P) +N+N/P).

Communication: Every node must receive on N/P elements. Hence P-1 nodes sends the N/P elements to a node. Therefore number of communications will be (P-1)*N/P. So, the number of communications for 1 node is (N/P* (P-1)). Since we have P nodes, total number of communications will be N*(P-1).

2.3) 2D partitioning: blocks



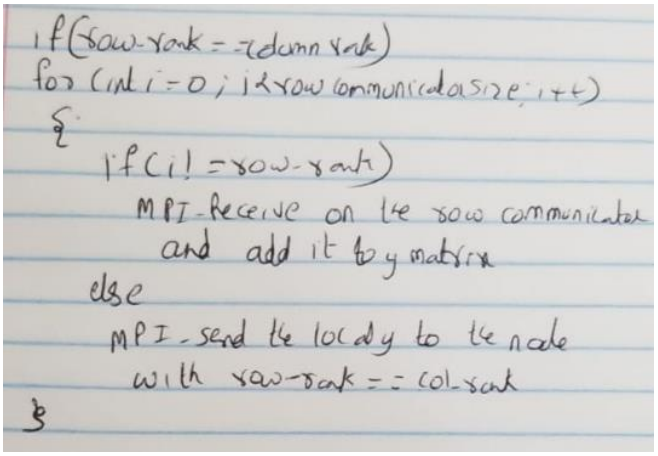
1) Initialize the K=0.

2) Generate the local A matrix of size (N/sqrt(P) * N/sqrt(P)) and local x of size N/sqrt(P).

3) create the row and column communicators.

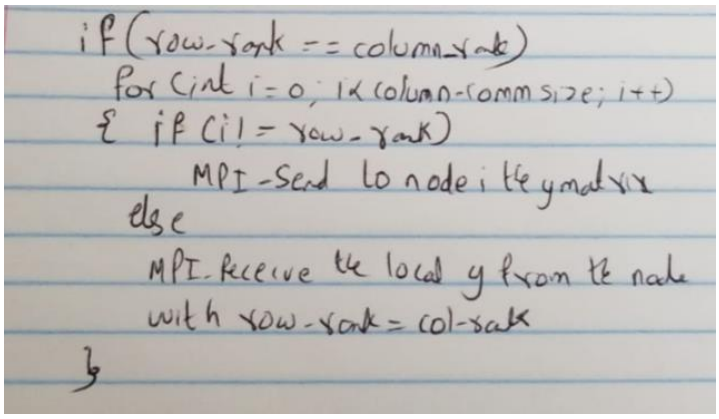
4) Now calculate the local y matrix for each node.

5) Gather all the data to the diagonal elements and add them.



6) Wait until all nodes are done.

7) Send the local y which is present on diagonal nodes to other nodes on column communicators.



8) Swap local y and local x matrices.

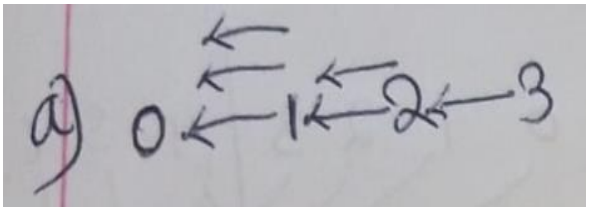
9) Go to step 4 and repeat the process for 10 iterations.

Memory: Assume each element requires O(1) memory. Therefore, memory required for local A matrix is O(N/sqrt(P) * N/sqrt(P)), memory required for X is O(N/sqrt(P)) and finally local y requires O(N/sqrt(P)). So total memory required per node is O(N^2/P+(2*N/sqrt(P))).

Communication: Every row and column communicators has sqrt(P) nodes. On all row communicators we have (sqrt(P)-1)*N) communications. Similarly on all column communicators we have (sqrt(P)-1)*N) communications. Hence total number of communications will be 2*(sqrt(P)-1)*N.

3) Reduction

A) Reduce-star on chain



Assume that the complexity is O(1) for each data transfer.

a) **Data on most loaded link:**

Most loaded link will be 1->0 and the data on it is O(P-1).

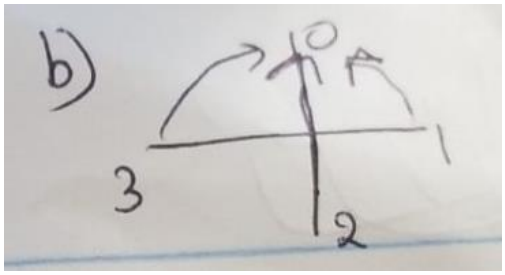
b) **Data on most loaded node:**

Most loaded node will be node 0 and data on it is O(P).

c) **Length of longest chain of communication:**

The longest chain is formed from last node to first node and its length is P-1. Assuming the length of link between the successive nodes is 1.

B) Reduce-star on clique



Assume that the complexity is O(1) for each data transfer.

a) **Data on most loaded link:**

All of the links will be equally loaded and data on them will be O(1).

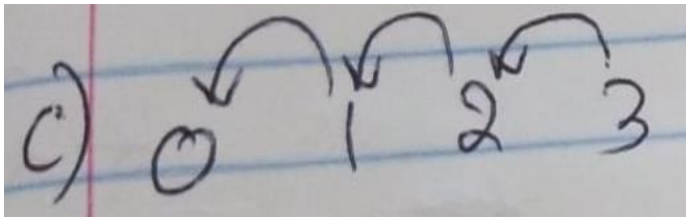
b) **Data on most loaded node:**

Most loaded node will be node 0 and data on it is O(P).

c) **Length of longest chain of communication:**

Since it is clique network, every node can directly send to every other node hence the length of longest chain of communication is the length of the link between any two nodes.

C) Reduce-chain on chain



Assume that the complexity is O(1) for each data transfer.

a) **Data on most loaded link:**

All of the links will be equally loaded and data on them will be O(1).

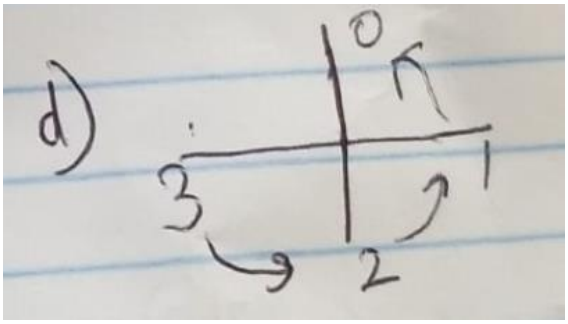
b) **Data on most loaded node:**

Except the last node every other node is equally loaded and data on them is O(2). Each node has O(1) data and receives O(1) data from the other node.

c) **Length of longest chain of communication:**

Since every node is sending to its previous node, longest chain of communication will be the link between the successive nodes.

D) Reduce-chain on clique



Assume that the complexity is O(1) for each data transfer.

a) **Data on most loaded link:**

All of the links will be equally loaded and data on them will be O(1).

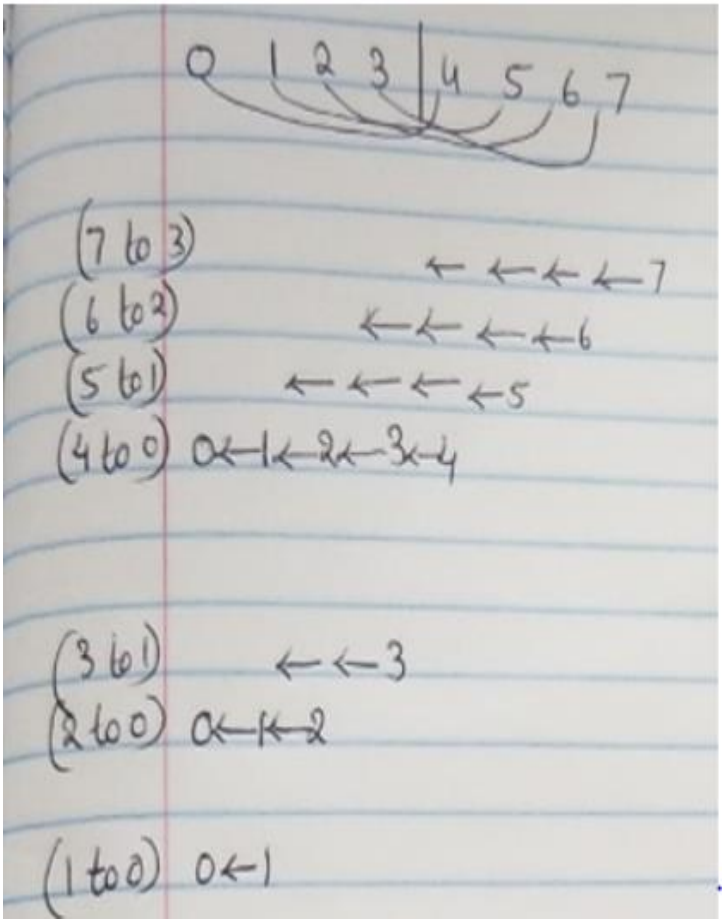
b) **Data on most loaded node:**

Except the last node every other node is equally loaded and data on them is O(2). Each node has O(1) data and receives O(1) data from the other node.

c) **Length of longest chain of communication**:

Since it is a clique network every node will directly send to the other appropriate node, hence the longest chain of communication will be the link between nodes.

E) Reduce-tree on chain



Assume that the complexity is O(1) for each data transfer.

a) **Data on most loaded link:**

Most loaded links will be each of the links 2->1, 3->2, 4->3 …… P/2-> (P/2)-1 and data on them is O(P/2).

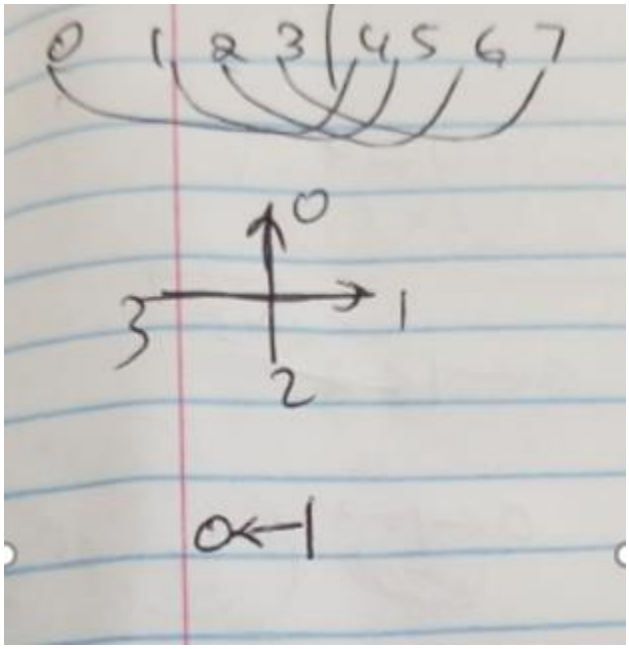
b) **Data on most loaded node:**

Most loaded node will be node 0 and data on it is O(log(P)+1).

c) **Length of longest chain of communication:**

The longest chains will be (P/2)->0, (P/2+1)->1, (P/2+2)->2, ……(P-1)->(P/2)-1 and the length will be P/2.

F) Reduce-tree on clique



a) **Data on most loaded link:**

Most loaded will be any link that is involved in communication and data on them is O(1).

b) **Data on most loaded node:**

Most loaded node will be node 0 and data on them is O(log(P)+1) including data its own data.

c) **Length of longest chain of communication:**

Longest chain of communication will be any link that is involved in the communication.

Question: What do you think is the best algorithm for each network structure? (One of the given algorithms or a different one.)

A) For chain network, reduce-chain algorithm will be the best algorithm because data on most loaded link and node is very less and no bigger chains of communications is formed other than the links between the nodes.

For Clique network, reduce-tree algorithm will be the best algorithm because data on most loaded link is O(1) and no bigger chains are formed other than the links between the nodes.