## Question1 : How much data transit on each link in total? What is the most loaded link?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Network Structure 1 | Network Structure 2 | Network Structure 3 |
| Algorithm 1 | Link (0-1) will have P-1 data transits  Link (1-2) will have P-2 data transits  .  .  .  Link ((n-1) – n) will have n data transits  .  .  Link ((P-2) – P) will have 1 data transit  Link (0 – 1) will be the most loaded link | In this Clique structure only the links attached to Node 0 will be used. Every link attached to node 0 will have one data transit and every other links are not used in data transmission.  Links attached to Node 0 are most loaded links (BEST) | Here,  Links which connect from level o to level 1 will have P/2 data transits  Links which connect from level 1 to level 2 will have P/4 data transits  .  .  Links in level log(P)-1 to log(P) will have 1 data transit  Link’s (0-1) and (0-2) are most loaded links |
| Algorithm 2 | Every Link will have only one data transit.   Since, every link has only one data transit every link will be most loaded link | Only the links which are adjacent to its numerical neighbors will have one data transit.  Most loaded Links are Links connecting adjacent numerical neighbors. | Right child link will have in level 0-1 will  Most loaded link will be (0-1) |
| Algorithm 3 | Only the links which starts with odd node and ends with even node will have data transits(ex: 0-1)  And will have data transits of  ((P/2)\*(P/2))-2) for first even to odd node  ((P/2)\*2)-(P/4)) for second even to odd node  .  .  Link 0-1 will be most loaded link | Only the links which starts with odd node and ends with even node will have data transits(ex: 0-1)  And will have data transits of  ((P/2)\*(P/2))-2) for first even to odd node  ((P/2)\*2)-(P/4)) for second even to odd node  .  .  Link 0-1 will be most loaded link | Incorrect algorithm |

### Question: How much data is received/sends by each node? What is the node that receives/sends the most data?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Network Structure 1 | Network Structure 2 | Network Structure 3 |
| Algorithm 1 | P-1 node  sends 1 / receives 0 data  P-2 node  sends 2 / receives 1 data  .  .  .  (P-p) node  Sends p/ receives p-1 data  .  .  0 node will receive P-1 data  Node 1 will sends most data (P-1)  Node 0 will receive most data (P-1) | Each node will send 1 data to Node 0.  Except Node 0.  Node 0 will receive P-1 data from every other node  Every node except node 0 will send 1 data  Node 0 will receive most data  (P-1) | Node 0 will receive most data (P-1)  Node 1 and 2 will send most data (P/2 each) |
| Algorithm 2 | Each node will send 1 data and receive one data except The boundary nodes (0 and P-1)  Every node will send one data and receive one data | Only the links which are adjacent to its numerical neighbors will have one receive and send to it’s neibhoring node.    There is no most loaded node, every node will perform one send and one receive operation. | Right child link will have in level 0-1 will  Most loaded link will be (0-1) |
| Algorithm 3 | Incorrect Algorithm | Incorrect Algorithm | Incorrect Algorithm |

### What is the longest chain of communication?

|  |  |  |  |
| --- | --- | --- | --- |
|  | Network Structure 1 | Network Structure 2 | Network Structure 3 |
| Algorithm 1 | P-1 to 0 node will be longest chain of communication | Since it is a clique network, every communication happens directly without any intermediate nodes | Communication from all of the bottom level nodes (Last log(P) nodes) to zero will be longest chain of communication |
| Algorithm 2 | Every communication happens without any help of intermediate nodes | Since it is a clique network, every communication happens directly without any intermediate nodes | Communication from log(P)/2 to log(P)/2 – 1 will be longest chain of communication |
| Algorithm 3 | Incorrect Algorithm | Incorrect Algorithm | Incorrect Algorithm |

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

Algorithm 1: Clique would be appropriate from the given 3 networks. But most of the links will not be used for communication (Links not connect to node 0). A network structure in which every node has only one link connected to node 0 would be best.

Algorithm 2: Chain would be best network for this algorithm

Algorithm 3: Incorrect Algorithm.

### 2. Heat Equation – 1D:

### Algorithm for Round-Robin Decomposition:

HEAT\_ROUND\_ROBIN

get RANK

get SIZE // size of 1D array

i = RANK

WHILE i < SIZE

IF i is a boundary condition //0 or n-1

IF i is 0

GET prevoius interations i+1 heat value from i+1 node

(2\*HEAT\_PREVIOUS[0]+HEAT\_PREVOIUS[1])/3

ELSE

GET prevoius interations i-1 heat value from i-1 node

(2\*HEAT\_PREVIOUS[SIZE-1] + HEAT\_PREVIOUS[SIZE-2])/SIZE-2

ELSE

GET i-1 and i+1 heat values from i-1 and i+1 nodes

(HEAT\_PREVIOUS[i-1] + HEAT\_PREVIOUS[i] + HEAT\_PREVIOUS[i+1])/3

i = i + P

### Algorithm for BLOCK Decomposition:

HEAT\_BLOCK

get RANK

get SIZE

START\_ITERATION = SIZE / RANK

END\_ITERATION = SIZE+1 / RANK

i = START\_ITERATION

WHILE i < END\_ITERATION

IF i is START\_ITERATION

IF i is 0

(2\*HEAT\_PREVIOUS[0]+HEAT\_PREVOIUS[1])/3

ELSE

get previous iterations i-1 heat value from i-1 node

(HEAT\_PREVIOUS[i-1] + HEAT\_PREVIOUS[i] + HEAT\_PREVIOUS[i+1])/3

ELSE IF i is END\_ITERATION

IF i is SIZE-1

(2\*HEAT\_PREVIOUS[SIZE-1] + HEAT\_PREVIOUS[SIZE-2])/SIZE-2

ELSE

get previous iterations i+1 heat value from i+1 node

(HEAT\_PREVIOUS[i-1] + HEAT\_PREVIOUS[i] + HEAT\_PREVIOUS[i+1])/3

ELSE

(HEAT\_PREVIOUS[i-1] + HEAT\_PREVIOUS[i] + HEAT\_PREVIOUS[i+1])/3

### Question: How much communication happen per iteration of the heat equation?

### Round Robin:

In boundary cases 1 and 0, there will be one communication between it’s adjacent rank node. In normal cases 2 communication will happen.

### Block:

For start and end-1 iterations of a node, communication will happen with it’s adjacent neibhor’s. Only exception is in boundary case in which there is no need for communication

### Question: What data partitioning would you use? How much communication does it do per iteration of

### the heat equation (total, per link, per node)?

I would use Block data portioning because, it will result in much less data communication than Round-Robin portioning

### Communication per node:

Nodes 0 and P-1 will have to do only one communication, every other node requires to do 2 communications

### Communication per Link:

Only the links who connect adjacent neighbors will be participated in communication, links connecting adjacent nodes(on basis of their rank) will have 2 data transits

### Communication Total:

There will be a total of (P-2)/2 communications, if we use Block decomposition

### Question: Write the algorithm that performs y = Ax; x = y; 10 times in a loop.

DENSE\_MATRIX\_MULTIPLICATION\_HORIZONTAL

get RANK

get SIZE

START\_ITERATION = SIZE/(RANK+1)

END\_ITERATION = SIZE/(RANK+2)

FOR i = START\_ITERATION to END\_ITERATION

FOR j = 1 to 10 //to perform 10 times in a loop

SUM = 0;

FOR k = 0 to SIZE-1

SUM = SUM + a[i][k]\*x[k]

y[i] = SUM

x[i] = SUM

DENSE\_MATRIX\_MULTIPLICATION\_VERTICAL

get RANK

get SIZE

START\_ITERATION = SIZE/(RANK+1)

END\_ITERATION = SIZE/(RANK+2)

FOR k = 1 to 10

FOR i = 0 to SIZE-1

SUM = 0;

FOR k = START\_ITERATION to END\_ITERATION

SUM = SUM + a[i][k]\*x[k]

temp[i] = SUM

IF RANK == 0 //end caluclation

FOR i = 0 to SIZE-1

SUM = 0

get all of temp[i] values from every other node in network and add it to a variable SUM

y[i] = SUM

x[i] = y[i]

DENSE\_MATRIX\_MULTIPLICATION\_BLOCKS

get RANK

get SIZE

compute HORIZONTAL\_START\_ITERATION, HORIZONTAL\_END\_ITERATION, VERTICAL\_START\_ITERATION, VERTICAL\_END\_ITERATION from rank and size

FOR k = 1 to 10

FOR i = HORIZONTAL\_START\_ITERATION to HORIZONTAL\_END\_ITERATION

SUM = 0;

FOR k = VERTICAL\_START\_ITERATION to VERTICAL\_END\_ITERATION

SUM = SUM + a[i][k]\*x[k]

temp[i] = SUM

IF P%RANK == 0 //end caluclation

FOR i = 0 to SIZE-1

SUM = 0

get all of temp[i] values from every other node in network and add it to a variable SUM

y[i] = SUM

x[i] = y[i]

### Question: How much memory does each node need?

Horizontal:

Input Matrix – n\*n/P

Vector X - n

Total: (n\*n/P) + n

Vertical:

Input Matrix – n\*n/P

Vector X - n

Temp array – n

Sum - 1

Total: (n\*n/P) + 2n + 1

Block:

Input Matrix – n\*n/P

Vector X - n

Temp array – n/sqrt(P)

Sum - 1

Total: (n\*n/P) + n + n/sqrt(P) + 1

### Question: How much communication does the algorithm do per iteration? (total, per link, per node)

Horizontal:  
No communication is needed in all cases

Vertical:

P data transit is required per iteration

Node 0 will have P-1 data transits, every other will have one data transit

Only the links which are connected to 0 Node are used, in those links each link will have n data transits

Block:

Sqrt(P) data transits are required per iteration

There will be